UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO - UFERSA
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Nícholas André Pinho de Oliveira

Aplicação de *Zoom* em Imagens Digitais Utilizando Regressão Linear

Mossoró-RN

2016

Nícholas André Pinho de Oliveira

Aplicação de *Zoom* em Imagens Digitais Utilizando Regressão Linear

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal Rural do Semi-Árido como requisito parcial para a obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Leandro Carlos de Souza

Mossoró-RN

2016

O48a Oliveira, Nícholas André Pinho.
Aplicação de zoom em imagens digitais utilizando regressão linear / Nícholas André Pinho Oliveira. - 2016.
57 f. : il.

Orientador: Leandro Carlos Souza.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência da Computação, 2016.

1. Algoritmos de ampliação. 2. regressão linear. 3. interpolação de imagens. 4. Transformação Box-Cox. I. Carlos Souza, Leandro, orient. II. Título.

Nícholas André Pinho de Oliveira

Aplicação de *Zoom* em Imagens Digitais Utilizando Regressão Linear

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade Federal Rural do Semi-Árido como requisito parcial para a obtenção do grau de bacharel em Ciência da Computação.

Trabalho aprovado. Mossoró-RN, 10 de novembro de 2016:

Prof. Dr. Leandro Carlos de Souza - UFERSA
Orientador e Presidente

Prof. Dr. Angélica Félix de Castro - UFERSA
Primeiro membro

Prof. Dr. Odacir Almeida Neves - UFERSA
Segundo membro

Mossoró-RN
2016

Uma imagem vale mais que mil palavras
(Confúcio)

Resumo

Ampliação de imagens é o processo de aumentar a resolução espacial de uma dada imagem digital. Este processo envolve a estimativa de cores dos pixels introduzidos na imagem ampliada. Essa estimativa é realizada por algoritmos de interpolação, existindo diversas abordagens como interpolação bilinear e bicúbica. A interpolação bilinear tende a borrar a imagem resultante na medida que o nível de ampliação cresce. Já a bicúbica introduz menos borramento, mas não funciona muito bem para altos fatores de ampliação. Este trabalho apresenta um novo método de interpolação de imagens que utiliza regressão linear e a transformada de Box-Cox para a estimação das cores dos novos pixels na imagem ampliada. A transformada é usada para linearizar os pontos utilizados nessa estimativa. Esse método de interpolação foi utilizado para a criação de um novo algoritmo de ampliação de imagens com o objetivo de apresentar resultados melhores que a interpolação bilinear e preservar as bordas da imagem original. O algoritmo de ampliação proposto é comparado com três algoritmos amplamente conhecidos através de uma análise visual e de uma análise numérica por meio do cálculo do PSNR.

Palavras-chave: Algoritmos de ampliação. regressão linear. interpolação de imagens. Transformação Box-Cox.

Abstract

Image zooming is the process of enlarging the spatial resolution of a given digital image. This process involves the color estimation of the pixels that are introduced in the enlarged image. The estimation is performed by interpolation algorithms and there are several approaches which includes bilinear and bicubic interpolation. The bilinear interpolation tends to blur the enlarged image as the zoom level increases, while the bicubic interpolation introduces a smaller blur effect but it doesn't work very well for high zoom levels. This work describes a new interpolation method that uses linear regression and Box-Cox transformation to estimate the new pixels color in the enlarged image. Box-Cox is used to obtain data normality in the set of data used in the estimation. This new interpolation method was used to create a new zoom technique with the goal of getting better results when compared to bilinear interpolation and also to preserve sharpness of the image. The proposed algorithm is compared to three well-known algorithms through a visual analysis and a numerical analysis using PSNR.

Keywords: Image zooming. linear regression. image interpolation. Box-Cox transformation.

Lista de ilustrações

Lista de tabelas

Lista de Algoritmos

Lista de abreviaturas e siglas

PDI	Processamento Digital de Imagens
MSE	*Mean Squared Error*
PSNR	*Peak signal-to-noise ratio*
LSE	*Least Square Estimation*
WLS	*Weighted Least Square*

Sumário

1 Introdução

O processamento digital de imagens (PDI) refere-se ao processamento de imagens por meio de um computador digital. Esta área da computação inclui algoritmos, que, aplicados em imagens, geram resultados diversos como a remoção de ruído, a rotação, a deformação, a ampliação, a extração do conteúdo e a segmentação de imagens digitais. As aplicações do PDI se estendem por diversas áreas do conhecimento, tais como medicina, astronomia e meteorologia. (GONZALEZ; WOODS, 2006).

Aplicações do PDI incluem o processamento de imagens formadas por raios gama, utilizadas para escaneamento ósseo e tomografias; imagens formadas por raios X ou pela luz ultravioleta, utilizadas para inspeção industrial e microscopia e imagens formadas pela captura de ondas rádio, que são utilizadas em ressonâncias magnéticas (PITAS, 2000).

As operações de ampliação de imagens (*Zoom In*), em particular, são utilizadas para os mais diversos fins. As aplicações vão desde a ampliação de imagens astronômicas e médicas à ampliação de imagens para detecção de placas veiculares e detecção de faces em circuitos internos de segurança. De fato, a área de ampliações de imagens tem sido uma área de pesquisa bastante ativa e diversos métodos já foram propostos. Chadda, Kaur e Thakur (2012) reafirmam isso ao apresentar um estudo sobre um vasto número de algoritmos de ampliação de imagens.

Em uma imagem ampliada, as cores das novas regiões criadas devem ser estimadas. Um método de ampliação deve, portanto, buscar manter as características das cores da imagem original. Entretanto, métodos de ampliação incluem efeitos de serrilhamento e de borramento nas imagens geradas. Minimizar o efeito serrilhado e o efeito de borramento se torna cada vez mais difícil na medida que o fator de ampliação é incrementado. Quanto menor a geração destes efeitos na imagem ampliada, melhor é o método de ampliação aplicado.

Esse processo de estimar novas cores em uma imagem é denominado interpolação de imagens, e estes algoritmos são utilizados não somente nas operações de ampliação mas também em outras operações como rotação e operações de reconstrução de imagens em geral (GONZALEZ; WOODS, 2006). Dentre os algoritmos de interpolação de imagens mais conhecidos, é possível citar a interpolação por replicação, interpolação bilinear e interpolação bicúbica.

A regressão linear é um método para a busca de um relacionamento linear entre variáveis aleatórias. O objetivo é estimar valores desconhecidos a partir de valores previamente observados. Essa relação é expressa por uma equação linear que é utilizada para estimar os valores desconhecidos e representa uma aproximação com um determinado erro (CHATTERJEE; HADI, 2006).

Nem todos os conjuntos de dados apresentam um relacionamento linear, portanto, tais

conjuntos produzem grandes erros nas estimativas quando aproximadas por um modelo linear. Por este motivo, transformações são comumente aplicadas ao conjunto de dados antes de realizar a regressão (MONTGOMERY; PECK; VINING, 2012). Estas transformações tem o objetivo de linearizar os dados minimizando o erro da estimativa. Uma transformação bastante utilizada para este propósito é a transformada de Box-Cox, que gera uma família de funções determinada por um parâmetro (SAKIA, 1992).

A área de processamento de imagens é uma área de pesquisa bastante ativa e muitos trabalhos já foram publicados, dentre eles diversos algoritmos para tratar do problema do *zoom*. Além dos já mencionados, existem métodos que utilizam outras técnicas e abordagens que incluem algoritmos baseados em equações diferencias parciais (GAO; SONG; TAI, 2009), que utiliza equações diferencias não lineares de alta ordem para estimar os novos pixels e busca preservar descontinuidades na imagem. Almira e Romero (2011) propuseram a utilização de teoremas de amostragens para a criação de três novos algoritmos de *zoom*. Sabrin e Ali (2014) desenvolveram a replicação inteligente, que decompõe a imagem em uma série de imagens binárias, onde são interpoladas e agregadas afim de obter a imagem ampliada. Já Battiato, Gallo e Stanco (2002) apresentam um algoritmo adaptativo que realiza uma interpolação com pesos controlados pelo gradiente da imagem.

Este trabalho apresenta um novo algoritmo de interpolação de imagens que será utilizado para propor um novo método de *zoom* que utiliza regressão linear para estimar as cores na imagem ampliada. A transformada de Box-Cox é aplicada antes da regressão afim de minimizar o erro, haja vista não haver garantia que os pixels que compõe o conjunto de dados utilizado na regressão apresentem um comportamento linear.

1.1 Problema

Uma imagem digital é uma matriz contendo valores de intensidade de cor, onde cada elemento dessa matriz é denominado pixel (GONZALEZ; WOODS, 2006). A ampliação de uma imagem consiste em aumentar o tamanho dessa matriz, introduzindo novos pixels. Dessa forma, é necessária a utilização de algum método para estimar o valor dos novos pixels criados.

Os métodos existentes na literatura em geral tendem a não preservar bordas e de adicionar bastante distorção a imagem em níveis mais altos de zoom. O algoritmo proposto tem como objetivo a preservação de bordas e minimização da distorção das imagens ampliadas em níveis mais altos de zoom.

1.2 Objetivos

Este trabalho apresenta o desenvolvimento de um novo algoritmo para ampliação de imagens. O método proposto faz uso da regressão linear para a estimação dos valores nos pixels

desconhecidos. Para aprimorar o tratamento de regiões em que a intensidade de cor não varia linearmente, é utilizada a transformada de Box-Cox para a estimação não-linear.

Os principais objetivos deste trabalho são:

- Descrever um algoritmo de *zoom* por regressão linear e Box-Cox
- Obtenção de ampliações melhores, na média, que o método de interpolação bilinear, um dos principais algoritmos de ampliação de imagens.
- Preservação das bordas na imagem ampliada, bem como redução do efeito de borramento.
- Apresentar um comparativo com os principais algoritmos de ampliação de imagens utilizando a métrica PSNR, que é comumente utilizada para medir o nível de distorção em uma imagem.

1.3 Justificativa

Devido a grande quantidade de áreas que utilizam a ampliação de imagens e os efeitos indesejados de serrilhamento e borramento causados por muitos dos algoritmos mais conhecidos, a pesquisa por novos e melhores métodos de ampliação de imagens é constante. Como já mencionado, um bom algoritmo de *zoom* deve manter, na medida do possível, as características originais da imagem. Para tanto, o que se procura é reduzir o efeito de borramento e a preservação das bordas da imagem, sendo isso um dos objetivos deste trabalho.

1.4 Organização

Além deste capítulo de introdução, o texto deste trabalho é dividido em mais 4 capítulos:

Capítulo 2 - Referencial Teórico
Este capítulo apresenta os aspectos mais relevantes da teoria utilizada no desenvolvimento do algoritmo proposto. São discutidos como uma imagem digital é representada e como algoritmos de ampliação funcionam. Os algoritmos de ampliação por replicação, interpolação linear e interpolação bicúbica são apresentados juntamente com uma breve análise dos resultados obtidos com a sua utilização. Os conceitos básicos da regressão linear e como transformações são utilizadas para melhorar as estimativas obtidas com a regressão também são apresentados.

Capítulo 3 - Zoom com regressão linear
Este capítulo descreve todo o funcionamento do método proposto e apresenta um pseudocódigo da implementação desenvolvida.

Capítulo 4 - Resultados

Este capítulo apresenta os resultados obtidos com a aplicação do método em diversas imagens digitais. Os resultados são comparados empiricamente, através da análise visual das imagens resultantes e numericamente através do cálculo do PSNR. O cálculo do PSNR é feito sobre as imagens obtidas com cada um dos algoritmos testados. O método proposto será comparado com três algoritmos presentes na literatura: ampliação por replicação, ampliação por interpolação linear e ampliação por interpolação bicúbica.

Capítulo 5 - Conclusão e trabalhos futuros

Este capítulo apresenta as conclusões para o algoritmo proposto e apresenta direcionamentos para trabalhos futuros.

2 Referencial Teórico

Este capítulo apresenta o referencial teórico necessário para o entendimento do método proposto. São apresentados os conceitos básicos que permeiam o processamento digital de imagens e o funcionamento dos algoritmos de ampliação mais utilizados atualmente. Também são apresentados os conhecimentos básicos acerca da regressão linear simples e múltipla e seu uso com a transformação de Box-Cox com o propósito de linearizar os pontos.

2.1 Processamento Digital de Imagens

O PDI é composto de vários processos que envolvem a aquisição, a representação e a manipulação de imagens digitais. O processo de aquisição de imagens pode ser realizado por dispositivos construídos para esse fim, como câmeras digitais e *Scanners*, por exemplo. A representação de uma imagem consiste em obter uma representação digital da informação visual. A representação deve ser simples e eficiente afim de permitir o processamento por meio de um computador (FLORIN; ARSINTE; MASTORAKIS, 2011).

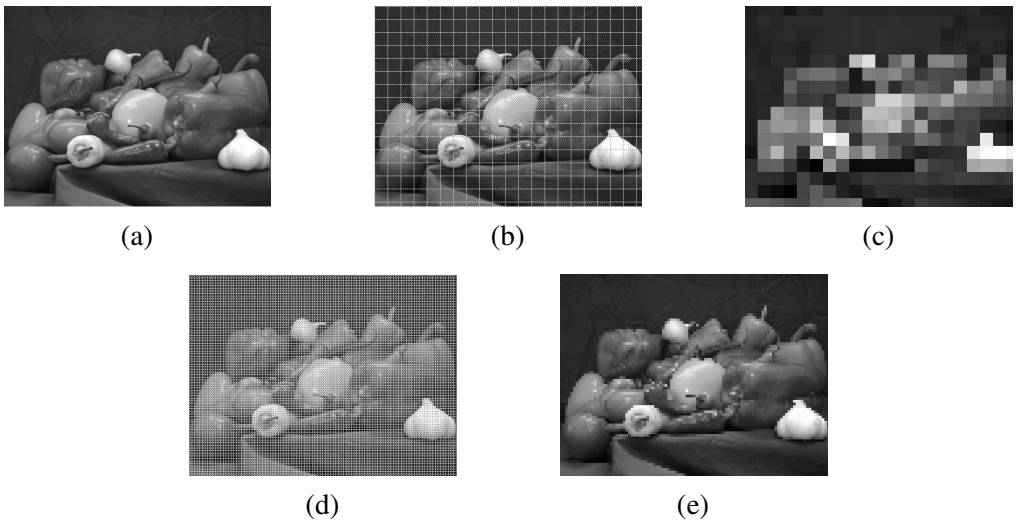
O mundo real apresenta a informação visual de forma contínua e portanto precisa ser discretizada para ser representada em um meio digital. O processo de discretização se refere a conversão do espaço contínuo em um espaço discreto e é divido em duas etapas: amostragem e quantização (THYAGARAJAN, 2011). A amostragem subdivide o espaço ocupado pela imagem em pequenas regiões, em que cada região é ocupada por uma única cor. Por outro lado, a quantização discretiza as intensidades das cores que aparecem na imagem.

No contexto de imagens digitais, a amostragem pode ser vista como o número de pontos (pixels) que serão utilizados para representar a imagem, sendo denominado resolução espacial da imagem. Já a quantização se refere ao número de valores possíveis que cada pixel na imagem pode assumir (LI; DREW; LIU, 2014). Um exemplo de amostragem pode ser visto na Figura 1 e um exemplo de quantização pode ser visto na Figura 2.

2.1.1 Representação de imagens digitais

Uma imagem pode ser interpretada como uma função bidimensional da intensidade da luz $f(x, y)$ em que x e y são as coordenadas de um pixel na imagem. O valor f é proporcional à cor da imagem naquele pixel (LI; DREW; LIU, 2014). Essa representação digital requer a definição de escalas tanto para as coordenadas x e y, oriundas da amostragem, como para os valores de intensidade de cada pixel, oriundas da quantização. A escala das coordenadas é conhecido como resolução da imagem, portanto, uma resolução de 512x256 indica que x varia

Figura 1 – Supondo uma imagem contínua (a) e um grid 25x25 em (b) a imagem amostrada utilizando este grid é vista em (c), em (d) um grid 5x5 e em (e) a imagem amostrada utilizando este grid.



(a) (b) (c)

(d) (e)

Fonte: Autoria Própria.

de 0 a 511 e que y varia de 0 a 255 e para qualquer x e y dentro desse intervalo, $f(x, y)$ dará o valor da intensidade do pixel naquela coordenada.

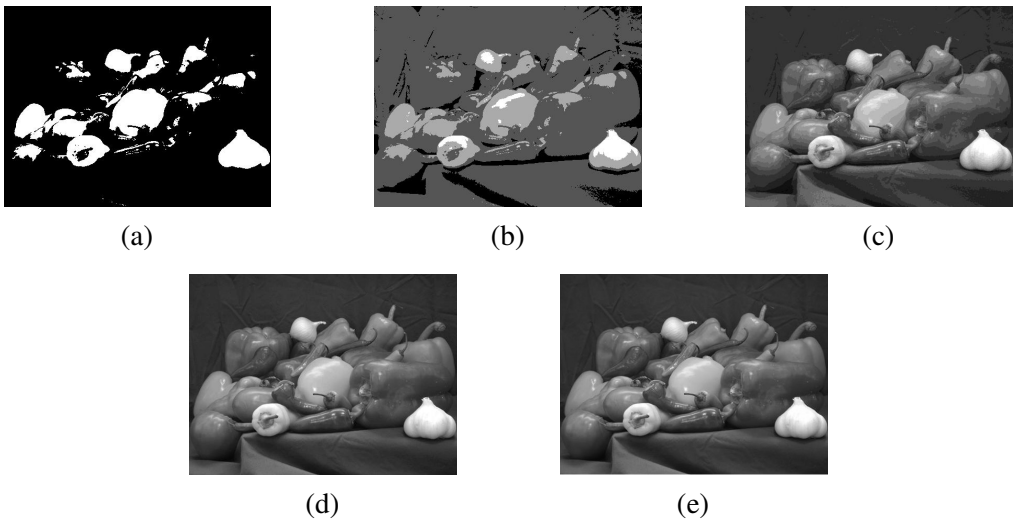
A intensidade de cada pixel está diretamente relacionada a quantização utilizada no processo de aquisição da imagem, uma imagem em que cada pixel assume 2 valores (preto ou branco) possui uma quantização (ou profundidade) de 1 bit (Figura 2(a)), ou seja, só será possível representar as intensidades com 0 ou 1, introduzindo uma grande perda de informação visual.

Aumentando a profundidade de cor é possível obter resultados muito melhores, como é possível visualizar claramente na Figura 2(e). Usualmente é utilizada uma profundidade de 8 bits (256 intensidades diferentes) para cada pixel (no caso de imagens monocromáticas). As imagens digitais são armazenadas em matrizes de dimensões iguais a de sua respectiva resolução, dessa forma a Figura 2(e), cuja resolução é igual a 512x384 pode ser representada matricialmente da forma que se segue

$$\text{f(x,y)} = \underbrace{\left.\begin{bmatrix} 43 & 44 & 46 & ... & 42 & 42 \\ 44 & 44 & 44 & ... & 39 & 39 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 89 & 91 & 91 & ... & 30 & 33 \end{bmatrix}\right\}_{h=384}}_{w=512}$$

em que w e h são a largura e altura da imagem, respectivamente e cada elemento da matriz representa a intensidade de luz naquele ponto.

Figura 2 – Em (a) imagem quantizada com 1 bit (2 níveis distintos), em (b) com 2 bits (4 níveis distintos), em (c) com 4 bits (16 níveis distintos), em (d) com 6 bits (64 níveis distintos) e em (e) com 8 bits (256 níveis distintos).



(a) (b) (c)

(d) (e)

Fonte: Autoria Própria.

2.1.2 Janelamento de imagens

Dois conceitos importantes são a vizinhança de um pixel e o de janelamento de imagens. Um pixel p_1 é dito vizinho de um outro pixel p_2 se estes dois pixels são adjacentes na horizontal, vertical ou diagonal (JAIN, 1989). Portanto, um dado pixel p(x,y) tem um total de 8 vizinhos caso ele esteja fora das extremidades da imagem, como é demonstrado na Figura 3.

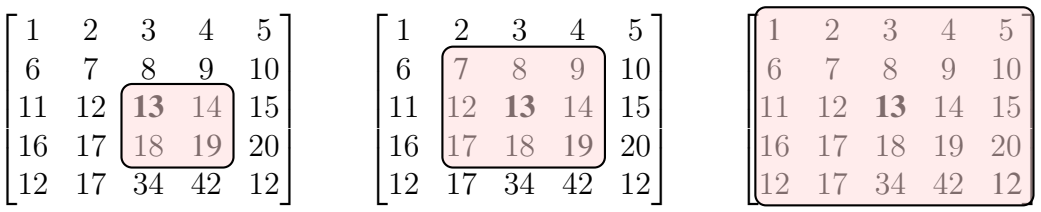
Figura 3 – A vizinhança de um pixel p(x,y).

$p(x-1, y-1)$	$p(x-1, y)$	$p(x-1, y+1)$
$p(x, y-1)$	**p(x,y)**	$p(x, y+1)$
$p(x+1, y-1)$	$p(x+1, y)$	$p(x+1, y+1)$

Fonte: Autoria Própria.

O conceito de janela se refere a uma submatriz composta pelos N pixels mais próximos de um dado pixel (x,y) (GONZALEZ; WOODS, 2006). Usualmente as janelas são matrizes quadradas de ordem ímpar e o pixel em questão está no centro dessa matriz. As janelas de ordem par não são muito utilizadas, com exceção da janela 2x2, cujo pixel (x,y) se localiza na coordenada (0,0) da janela. A Figura 4 apresenta alguns exemplos de janelas para uma imagem hipotética 5x5.

Figura 4 – Exemplo de janelas para o pixel (3,3). Em (a) Janela 2x2, em (b) Janela 3x3 e em (c) Janela 5x5.



(a) Janela 2x2 (b) Janela 3x3 (c) Janela 5x5

Fonte: Autoria Própria.

2.1.3 Imagens Coloridas

Imagens coloridas são uma extensão das imagens monocromáticas e possui múltiplos canais, onde através da combinação desses canais dá-se a percepção da imagem em cores. Existem diversos sistemas de cores, dentre eles destacam-se o RGB, HSL e CMYK (HUNT, 2005).

O objetivo de um modelo de cor é facilitar a especificação das cores em uma forma padronizada (WYSZECKI; STILES, 2000). O modelo RGB é um dos mais utilizados no ramo de processamento digitais de imagens e representa cada pixel por meio da combinação de suas componentes espectrais vermelho (R), verde (G) e azul (B). Neste modelo, cada cor varia de 0 a 255 (8 bits), sendo possível portanto a representação de mais de 16 milhões de cores (LI; DREW; LIU, 2014).

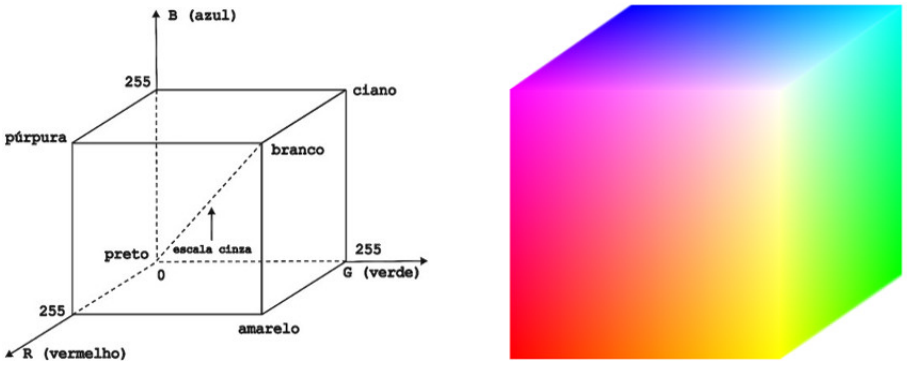
Na Figura 5(a) tem-se a representação do cubo formado pelo sistema de cores RGB, onde as cores primárias estão nos eixos, o preto está na origem (0,0,0), o branco está no vértice mais afastado da origem e as três cores secundárias (ciano, amarelo e magenta) estão nos outros três vértices. A Figura 5(b) mostra as combinações de cores possíveis.

Uma imagem colorida que utiliza o sistema de cor RGB é formada, portanto, pela combinação de três componentes de cor, que são representadas em matrizes independentes, também chamadas de canais (*channels*) de cor. A Figura 6, mostra cada canal de cor de uma imagem RGB, a combinação desses 3 canais forma a imagem colorida.

2.1.4 Zoom

A operação de Zoom consiste em aumentar a resolução de uma imagem (*Zoom In*) ou diminuir a resolução da mesma (*Zoom Out*). Ela também pode ser vista como um caso particular da operação de redimensionamento, onde a imagem é redimensionada pelo mesmo fator tanto em x quanto y não causando distorção na imagem uma vez que a razão entre largura e a altura da

Figura 5 – (a) Cubo formado pelas componentes RGB (b) Cubo de cores do RGB.



(a) (b)

Fonte: Gonzalez e Woods (2006).

Figura 6 – Imagem decomposta nos 3 canais de cores RGB.

(a) Canal R (b) Canal G (c) Canal B

Fonte: Gonzalez e Woods (2006).

imagem (*aspect ratio*) é mantida (GONZALEZ; WOODS, 2006).

A operação de redução é bem mais simples quanto comparada a operação de ampliação, e envolve basicamente o descarte de pixels e a reconstrução da mesma imagem em uma escala menor. Como o objetivo deste trabalho é a operação de *Zoom In*, algoritmos e técnicas de *Zoom Out* não serão aprofundados.

Na operação de *Zoom In* o tamanho da imagem aumenta, pois pixels são introduzidos na imagem e necessitam ser preenchidos afim de reconstruir a imagem na nova resolução. O problema está em justamente calcular os valores de intensidade para esses novos pixels de tal forma que a imagem reconstruída não perca suas características originais e que certos detalhes possam ser melhor observados e estudados na imagem. A Figura 7 apresenta um exemplo de uma imagem ampliada por um fator 2 com os pixels incluídos em preto, por serem inicializados com zero.

Na literatura, são propostas várias técnicas para o preenchimento dos pixels incluídos. A grande maioria desses algoritmos envolve o uso de alguma função de interpolação a fim de

Figura 7 – Imagem aumentada por um fator 2 sem reconstrução.

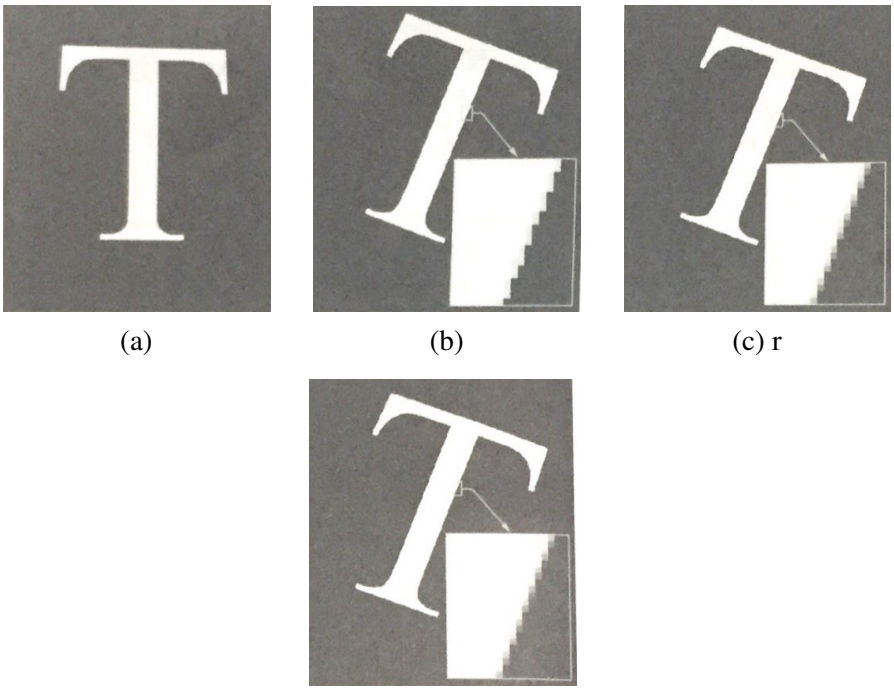
Fonte: Autoria Própria.

calcular os valores dos novos pixels. Estes algoritmos podem ser divididos em 3 categorias básicas: linear, não linear ou uma combinação dos anteriores (CHADDA; KAUR; THAKUR, 2012).

Há ainda uma outra divisão dos algoritmos de interpolação: adaptativos e não adaptativos (GONZALEZ; WOODS, 2006). Os algoritmos adaptativos podem tratar cada pixel de forma diferente, podendo detectar bordas e outras propriedades da imagem e alterar o método de interpolação. Já os não adaptativos tratam todos os pixels igualmente, independente da localização e das características dos seus pixels vizinhos. Exemplos de algoritmos não adaptativos incluem replicação, bilinear e bicúbica. Nas próximas seções estes três algoritmos serão explorados em detalhes.

A interpolação de imagens também é utilizada em outros tipos de operações. A rotação é um exemplo de operação que pode fazer uso da interpolação de imagens afim de reconstruir a imagem rotacionada (GONZALEZ; WOODS, 2006). Uma imagem rotacionada em 45º, por exemplo, precisa ser interpolada para que os pixels rotacionados sejam reconstruídos. Nem sempre os pixels rotacionados cairão em posições inteiras, sendo necessário a aplicação da interpolação por fatores não inteiros. A Figura 8 apresenta um exemplo de rotação em 21º utilizando os métodos de interpolação que serão discutidos na próxima seção. Também é possível aplicar *zoom* por fatores não inteiros, e os algoritmos apresentados nas próximas seções funcionam também para esses casos.

Figura 8 – Em (a) imagem original, em (b), (c) e (d) imagem rotacionada utilizando replicação, interpolação bilinear e interpolação bicúbica respectivamente.



(a) (b) (c) r

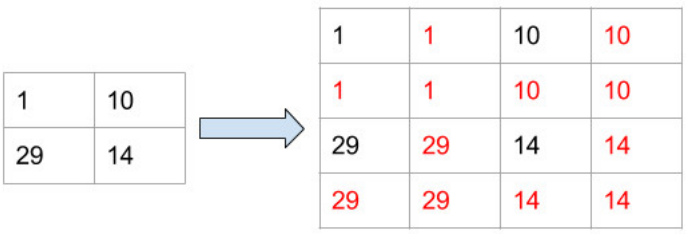
(d)

Fonte: Gonzalez e Woods (2006).

2.1.5 Zoom por Replicação

A replicação de pixels, ou *nearest replication* (RUSS, 2011) é, dentre os algoritmos conhecidos, o mais simples de todos e seu funcionamento se baseia na replicação dos pixels em sua vizinhança. O *zoom* por replicação simplesmente aumenta a área de um pixel e replica a cor nos pixel vizinhos criados. A Figura 9 exemplifica o funcionamento da ampliação por replicação, em que cada pixel na imagem original foi replicado para preencher os pixels introduzidos na ampliação.

Figura 9 – Uma imagem hipotética 2x2 ampliada por um fator 2 através de replicação.



Fonte: Autoria própria.

Observa-se que o método da replicação deixa a imagem ampliada com blocos perceptíveis de cor, por isso, quando o *zoom* por replicação é aplicado em imagens sob um fator maior que 2, ele tende a causar grandes distorções, principalmente nas bordas.[1] (CHADDA; KAUR; THAKUR, 2012). A Figura 10 mostra o resultado da aplicação do zoom por replicação, onde é possível observar a introdução de blocos na imagem.

Figura 10 – Ampliando porção de uma imagem por um fator 4 utilizando replicação.

(a) Imagem Original (b) Imagem ampliada

Fonte: Autoria própria.

A grande vantagem do *zoom* por replicação é sua baixa complexidade e fácil implementação. Entretanto, não é a melhor escolha para a maioria das aplicações. Um algoritmo para o *zoom* por replicação é apresentado no Algoritmo 1. Ele introduz um outro conceito importante e bastante utilizado nos algoritmos de zoom: o conceito de subpixel.

Um subpixel é um pixel imaginário situado entre dois pixels reais, o subpixel (0.5,0.5), por exemplo, está situado entre os pixels (0,0) e (1,1). Os subpixels são utilizados como uma forma de mapear um pixel da imagem aumentada na imagem original, facilitando o cálculo da intensidade do novo pixel. Em uma imagem aumentada por um fator 2, o pixel (1,1) representa agora um pixel vazio que precisa ser preenchido. Dessa forma ele é mapeado para o pixel (1/fator,1/fator), ou seja, (0.5,0.5). No caso do zoom por replicação a estratégia do algoritmo é replicar o pixel cujas coordenadas sejam iguais à parte inteira das coordenadas do subpixel.

2.1.6 Interpolação bilinear

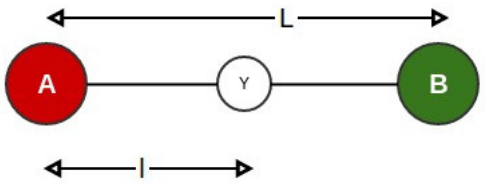
A ampliação por interpolação bilinear é um dos mais utilizados atualmente devido a sua rapidez e resultados satisfatórios. Na interpolação linear (MEIJERING, 2002), os pontos conhecidos são utilizados para estimar linearmente qualquer ponto arbitrário em uma reta imaginária formada pelos dois pontos. Considerando dois pontos A e B, a interpolação linear estima qual seria o valor de um ponto qualquer Y dentro dessa reta imaginária (Figura 11)

[1] Por bordas entende-se áreas na imagem onde há uma abrupta variação de intensidade de cor.

Algoritmo 1 Algoritmo de zoom por replicação.

```
1: fator ← N
2: nova_altura ← ROUND(img.altura * fator)
3: nova_largura ← ROUND(img.largura * fator)
4: para y = 0; i < nova_altura; i++ faça
5:     para x = 0; j < nova_largura; j++ faça
6:         sx ← x/fator
7:         sy ← y/fator
8:         nova_imagem(x, y) ← img(FLOOR(sx), FLOOR(sy))
9:     fim para
10: fim para
```

Figura 11 – Uma reta imaginária entre dois pontos A e B, onde se busca encontrar o valor de Y.



Fonte: Autoria Própria.

Por uma simples regra de três é possível estabelecer a relação base como mostrado na Equação (2.1) que deriva a equação da interpolação linear (2.2).

$$\frac{Y - A}{l} = \frac{B - A}{L}, \tag{2.1}$$

$$Y = A + l\frac{(B - A)}{L}. \tag{2.2}$$

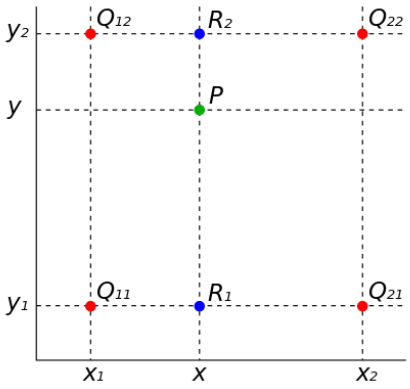
A aplicação do conceito de interpolação linear ao redimensionamento de imagens digitais deve ser feito em duas direções (MEIJERING, 2002), tanto em x quanto em y, sendo assim chamada de interpolação bilinear. Considerando quatro pontos Q_{11}, Q_{12}, Q_{21} e Q_{22} e um dado ponto P dentro da área formada por esses quatro pontos (Figura 12), é possível estimar o valor de P ao interpolar linearmente em x e depois em y. A interpolação na direção x resulta nas Equações (2.3) e (2.4), que calculam repectivamente os pontos R_1 e R_2 do gráfico da Figura 12,

$$f(x, y_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}), \tag{2.3}$$

$$f(x, y_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}), \tag{2.4}$$

em que $f(Q_{11}), f(Q_{12}), f(Q_{21})$ e $f(Q_{22})$ representam as cores dos 4 pixels vizinhos.

Figura 12 – Um ponto P a ser estimado com base nos quatro pontos vizinhos.



Fonte: Bocsika (2009).

A interpolação na direção y é realizada a partir da interpolação na direção x e resulta na obtenção do ponto P, de acordo com a Equação (2.5).

$$f(x,y) = \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2), \tag{2.5}$$

Nota-se que o zoom por interpolação bilinear utiliza os 4 pixels mais próximos do subpixel (janela 2x2) como base para o cálculo. Dessa forma é possível calcular o valor de qualquer pixel vazio na imagem aumentada utilizando as Equações (2.3), (2.4) e (2.5).

O *zoom* por interpolação bilinear apresenta bons resultados e é um algoritmo bastante rápido, uma de suas características porém é o borramento (Figura 13) das imagens aumentadas tendo como consequência a tendência de não preservar as bordas da mesma (CHADDA; KAUR; THAKUR, 2012).

O Algoritmo 2 apresenta o *zoom* por interpolação bilinear. O algoritmo é semelhante ao algoritmo da replicação, o que muda é a forma que a intensidade dos novos pixels são calculados. Os pixels da janela 2x2 utilizada pelo algoritmo estão contidos nas variáveis A,B,C e D, que são então utilizadas para o cálculo da interpolação. Este algoritmo faz uso de uma função chamada *lerp* que calcula a interpolação linear em uma direção.

2.1.7 Interpolação Bicúbica

A interpolação bicúbica, ou *bicubic interpolation* (KEYS, 1981), é semelhante a interpolação bilinear. Sua grande diferença é considerar uma janela 4x4, contendo, portanto, os 16 pixels mais próximos. Este algoritmo também atribui pesos ao pixels envolvidos, dando maior peso aos pixels mais próximos em detrimento dos pixels mais distantes. Nesse algoritmo, o valor

Algoritmo 2 Algoritmo de zoom por interpolação bilinear.

função LERP(c1,c2,v1,v2,x)
    se $v1 == v2$ **então**
        retorne c1
    senão
        retorne c1 + ((c2 - c1) / (v2 - v1)) * (x - v1)
    fim se
fim função
$fator \leftarrow N$
$nova\_altura \leftarrow \text{ROUND}(img.altura * fator)$
$nova\_largura \leftarrow \text{ROUND}(img.largura * fator)$
para $y = 0; i < nova\_altura; i++$ **faça**
    para $x = 0; j < nova\_largura; j++$ **faça**
        $sx \leftarrow x/fator$
        $sy \leftarrow y/fator$
        $dx \leftarrow \text{FLOOR}(sx)$
        $dy \leftarrow \text{FLOOR}(sy)$
        $A \leftarrow img(dx, dy)$
        $B \leftarrow img(dx, dy + 1)$
        $C \leftarrow img(dx + 1, dy)$
        $D \leftarrow img(dx + 1, dy + 1)$
        $fxy_1 \leftarrow \text{LERP}(A, B, dx, dx + 1, sx)$
        $fxy_2 \leftarrow \text{LERP}(C, D, dx, dx + 1, sx)$
        $fxy \leftarrow \text{LERP}(fxy_1, fxy_2, dy, dy + 1, sy)$
        $nova\_imagem(x, y) \leftarrow \text{ROUND}(fxy)$
    fim para
fim para

de intensidade de um dado pixel (x,y) é obtido através da Equação (2.6),

$$f(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j, \tag{2.6}$$

em que os 16 coeficientes a_{ij} são obtidos a partir das 16 equações das 16 incógnitas que podem ser escritas utilizados os 16 pixels mais próximos. Por ser um algoritmo extenso, sua implementação será omitida, mas pode ser encontrada em Demofox (2015).

Este algoritmo tende a apresentar resultados melhores que o *zoom* por interpolação bilinear, porém apresenta uma complexidade computacional um pouco maior. A Figura 14 mostra o resultado da aplicação deste algoritmo.

2.2 Métricas de avaliação

Comparar resultados de reconstrução de imagens requer uma medida de qualidade. Duas medidas bastante utilizadas são o MSE (*Mean Squared Error*) e PSNR (*Peak signal-to-*

Figura 13 – Ampliando uma porção de uma imagem por um fator 4 utilizando interpolação bilinear.

(a) Imagem Original (b) Imagem ampliada

Fonte: Autoria própria.

Figura 14 – Ampliando uma porção de uma imagem por um fator 4 utilizando interpolação bicúbica.

(a) Imagem Original (b) Imagem ampliada

Fonte: Autoria Própria.

noise ratio) (WANG et al., 2004). O PSNR (HORE; ZIOU, 2010) pode ser visto como um aprimoramento do MSE, uma vez que este é bastante sensível a profundidade de pixels da imagem e pode dificultar a interpretação de seus resultados e consequentemente comparações entre versões da mesma imagem com diferentes profundidades de pixels ou canais.

O PSNR resolve esse problema ao normalizar o MSE de acordo com a profundidade de pixels da imagem, gerando resultados mais fáceis de serem comparados e analisados. O MSE

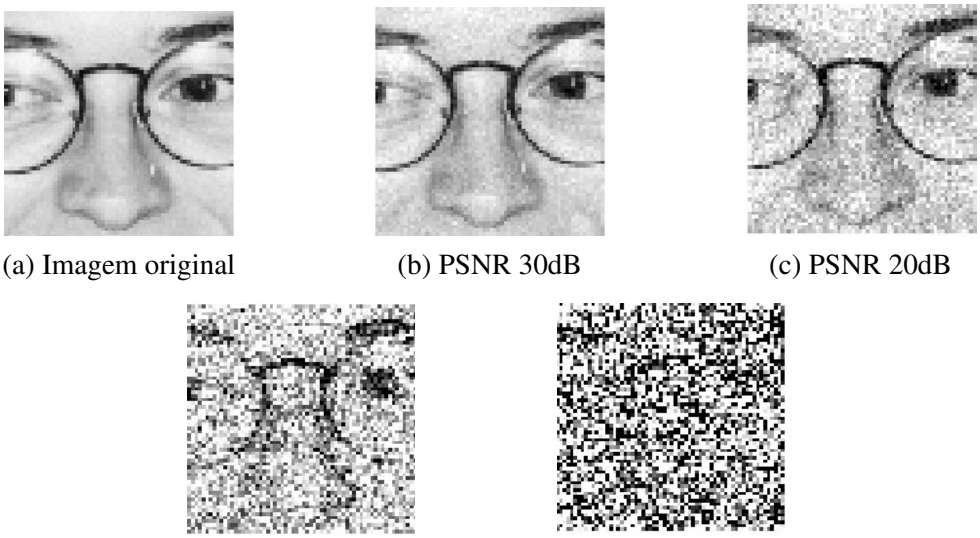
entre duas imagens $f(x, y)$ e $g(x, y)$ é dado pela Equação (2.7) e o PSNR pela Equação (2.8).

$$MSE = \frac{1}{MN} \sum_{n=1}^{M} \sum_{m=1}^{N} [f(x, y) - g(x, y)]^2, \tag{2.7}$$

$$PSNR = -10log_{10} \frac{MSE}{S^2}. \tag{2.8}$$

O PSNR é medido em decibéis (dB) e S é o pixel de maior intensidade na imagem. Quanto maior o valor do PSNR, maior é a qualidade da reconstrução. É interessante salientar, porém, que o PSNR só funciona para comparações de reconstrução de uma mesma imagem, isso significa que uma imagem X reconstruída com PSNR igual 20 dB não necessariamente é pior que outra imagem Y reconstruída com PSNR 30 dB (HUYNH-THU; GHANBARI, 2008). A Figura 15 compara diversos valores de PSNR para uma mesma imagem onde foi adicionado diversos níveis de ruído.

Figura 15 – Em (a) tem-se a imagem original. Em (b), (c), (d) e (e) versões da mesma imagem com diferentes níveis de ruído com PSNR iguais 30dB, 20dB, 10dB e 0dB respectivamente.



(a) Imagem original (b) PSNR 30dB (c) PSNR 20dB

(d) PSNR 10dB (e) PSNR 0dB

Fonte: Todd Veldhuizen (1998).

O PSNR é calculado tendo como base uma imagem "ideal", no sentido de não possuir ruído ou distorções. Uma metodologia para o uso do PSNR na mensuração da qualidade dos algoritmos de *zoom* será explicado mais adiante neste trabalho.

2.3 Regressão

Regressão é um método simples para a busca de relacionamento entre variáveis aleatórias (MONTGOMERY; PECK; VINING, 2012; CHATTERJEE; HADI, 2006). A relação é expressa na forma de uma equação ou modelo conectando as variáveis dependentes (ou de resposta) com as variáveis de previsão. A variável de resposta é denotada por Y e o conjunto de variáveis de previsão são denotadas por $X_1, X_2, ..., X_p$, onde p representa o número de variáveis de previsão. Na regressão linear, a relação entre as variáveis é determinada pela Equação (2.9),

$$\hat{y_i} = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_p x_{pi} + \epsilon, \tag{2.9}$$

em que ϵ é considerado um erro aleatório e representa a discrepância da aproximação e $\beta_0, \beta_1, ..., \beta_p$ são os chamados parâmetros de regressão ou coeficientes. Quando se tem somente uma variável de previsão, a regressão é denominada simples e quando se tem duas ou mais variáveis de previsão a regressão é denominada regressão múltipla (MONTGOMERY; PECK; VINING, 2012).

2.3.1 Conceitos Básicos

É comum organizar os dados de um problema de regressão em uma tabela. Considerando n observações de um determinado problema consistindo de uma variável de resposta Y e uma variável de previsão X esses dados seriam dispostos em uma tabela de acordo com a Tabela 1.

Tabela 1 – Notação utilizada para representar os dados usados em uma regressão simples

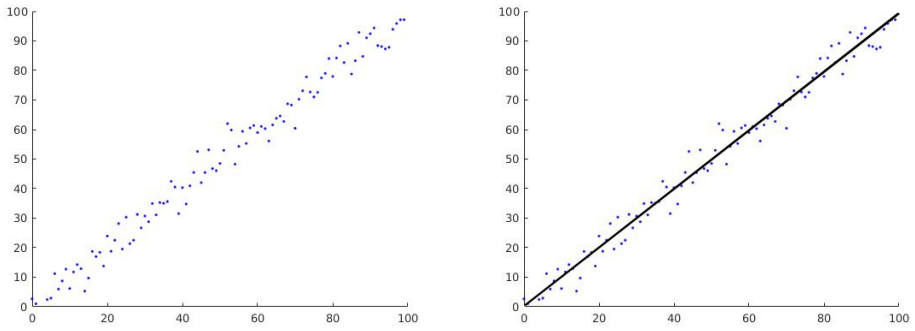
Observação	X	Y
1	x_1	y_1
2	x_2	y_2
⋮	⋮	⋮
n	x_n	y_n

Fonte: Autoria Própria.

O objetivo da regressão linear é encontrar uma reta que minimiza as distâncias entre os pontos. Supondo um conjunto hipotético de dados e seu gráfico de dispersão na Figura 16(a), a reta que mais se aproxima de todos os pontos e portanto minimiza as distâncias é a reta apresentada na Figura 16(b).

A reta é encontrada ao se calcular os coeficientes β, no caso dos dados hipotéticos da Figura 16, os coeficientes da equação $\hat{y_i} = \beta_0 + \beta_1 x_i$ são 2.0188 e 3.522 respectivamente. Isso significa que é possível estimar o valor de Y de um dado X não presente nos dados observados simplesmente substituindo x_i na equação.

Figura 16 – Em (a) tem-se o gráfico de dispersão e (b) a sua respectiva reta de regressão.



(a) (b)

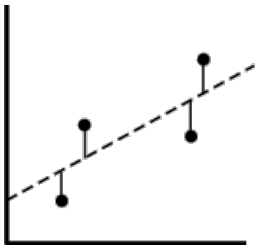
Fonte: Autoria Própria.

2.3.2 Estimando os parâmetros

O cálculo dos coeficientes β é uma das etapas mais importantes da regressão. Existem diversos métodos para se obter tais coeficientes, sendo que um dos mais utilizados é o LSE (*Least Squares Estimation*), também conhecido como método dos mínimos quadrados (LUENBERGER, 1997), que calcula a reta que minimiza a soma dos quadrados das distâncias de cada ponto à reta.

A distância de cada ponto à reta, representa o erro caso o ponto seja estimado utilizando a equação dessa reta (Figura 17). Esses erros podem ser obtidos através da Equação (2.10) e a soma dessas distâncias é dada pela Equação (2.11).

Figura 17 – Distâncias entre pontos reais e a reta de regressão.



Fonte: Weisstein, Eric W (2016).

$$\epsilon_i = y_i - \beta_0 - \beta_1 x_i, i = 1, 2..., n, \tag{2.10}$$

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} (\epsilon_i)^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2. \tag{2.11}$$

O método LSE consiste em minimizar $S(\beta_0, \beta_1)$. Os valores de β_0 e β_1 que minimizam $S(\beta_0, \beta_1)$ são dados pelas Equações (2.12) e (2.13),

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(y_i - \overline{y})(x_i - \overline{x})}{\sum_{i=1}^{n}(x_i - \overline{x})^2}, \tag{2.12}$$

$$\hat{\beta}_0 = y - \beta_1 \overline{x}, \tag{2.13}$$

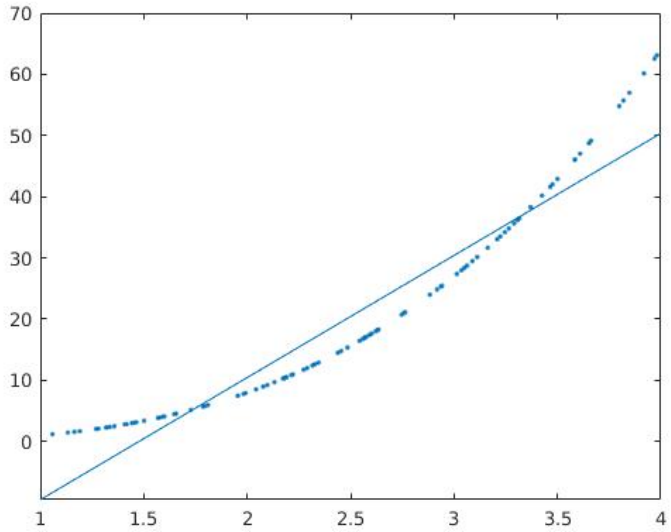
em que $\overline{x}$ e $\overline{y}$ são, respetivamente, as médias das variáveis X e Y.

2.3.3 Transformações

Os dados observados nem sempre estão em um formato apropriado para a regressão. No caso particular da regressão linear, o conjunto de dados original pode se aproximar mais de uma função quadrática do que de uma função linear e a utilização da regressão linear apresentaria um erro muito grande, visto que grande parte dos pontos estariam distantes da reta.

Na Figura 18 tem-se o gráfico da função $y = x^3$ e a sua respectiva reta de regressão. É possível notar claramente que a reta de regressão possui uma distância muito grande para a grande maioria dos pontos no gráfico de dispersão e consequentemente introduzindo um erro considerável nas estimativas de qualquer y_i.

Figura 18 – A reta de regressão para a função $y = x^3$.



Fonte: Autoria Própria.

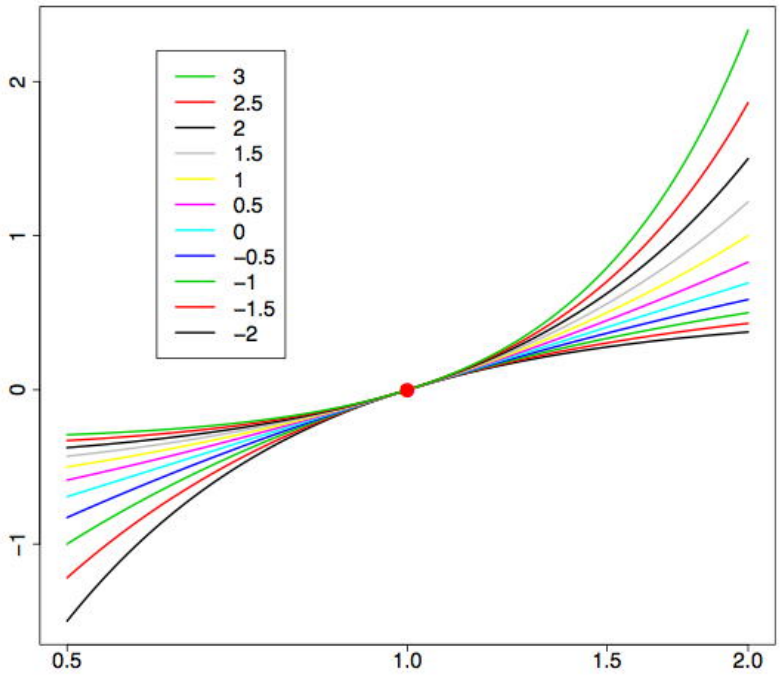
As transformações são aplicadas, justamente para atingir certos objetivos no conjunto de dados, tais como a obtenção de linearidade (LALONDE, 2005). É comumente necessário

aplicar alguma transformada antes de realizar a regressão, afim de obter melhores resultados ao minimizar o erro.

Existem diversas transformadas, uma delas é a Box-Cox (BOX; COX, 1981) que consiste na representação de uma família de funções, onde cada função é determinada pela utilização de um parâmetro. Esta transformação é mostrada na Equação (2.14). Na Figura 19 são apresentadas as funções geradas pela transformada Box-Cox com diferentes valores de λ para a função $f(x) = log(x)$.

$$T(y_i) = \begin{cases} \lambda \neq 0, & \frac{y_i^{\lambda}-1}{\lambda} \\ \lambda = 0, & ln(y_i). \end{cases} \tag{2.14}$$

Figura 19 – Funções geradas para vários valores de λ para a função $f(x) = log(x)$.



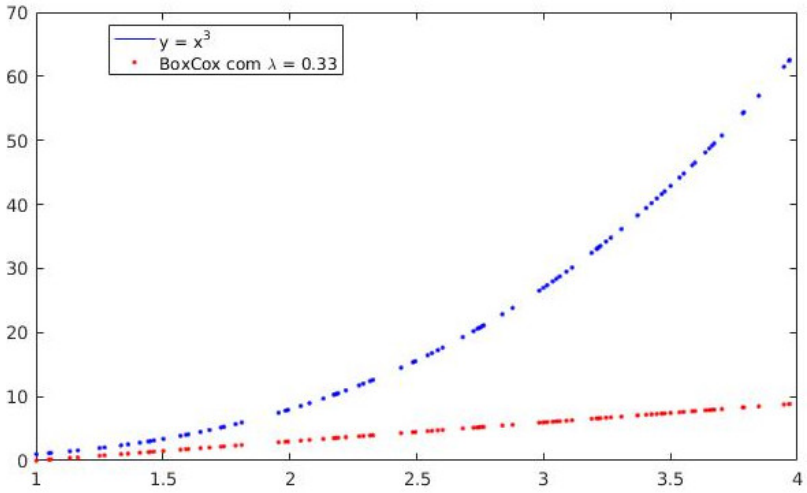
Fonte: David M. Lane1 (2007).

Um exemplo da aplicação da Box-Cox é mostrada na Figura 20, onde tem-se o gráfico da função x^3 no intervalo [0,4] e a mesma função transformada utilizando Box-Cox com $\lambda = 0.33$. Visualmente é fácil perceber que a transformada linearizou os pontos e estes agora podem ser estimados por uma reta.

A transformada de Box-Cox admite inversa, desta forma, é possível obter os dados originais a partir de dados transformados. A inversa pode ser obtida com a Equação (2.15),

$$y_i = \begin{cases} \lambda \neq 0, & e^{\frac{log[(\lambda * T(y_i))+1]}{\lambda}} \\ \lambda = 0, & e^x. \end{cases} \tag{2.15}$$

Figura 20 – Exemplo da Transformada de Box-Cox.



Fonte: Autoria Própria.

2.3.4 Regressão Múltipla

A regressão múltipla é uma extensão da regressão simples. A regressão é dita múltipla quando a variável de resposta Y depende de p variáveis de previsão $X_1, X_2, ...X_p$. A relação entre Y e $X_1, X_2, ...X_p$, no modelo linear, é formulada pela Equação (2.16) (MONTGOMERY; PECK; VINING, 2012),

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_p x_{pi} + \epsilon_i, \tag{2.16}$$

em que $\beta_0, \beta_1, \beta_2, ..., \beta_p$ são as constantes parciais de coeficientes parciais da regressão.

A regressão múltipla também pode ser representada em notação de matrizes na forma que se segue (CHATTERJEE; HADI, 2006)

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & ... & x_{1p} \\ 1 & x_{21} & ... & x_{2p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & ... & x_{np} \end{bmatrix}, \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix} \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

O modelo linear pode então, ser representado de acordo com a Equação (2.17),

$$Y = X\beta + \epsilon, \tag{2.17}$$

em que coeficientes β são então obtidos minimizando $S(\beta)$. Em notação matricial, $S(\beta)$ é obtido de acordo com a Equação (2.18),

$$S(\beta) = (Y - X\beta)^T (Y - X\beta), \tag{2.18}$$

em que a minimização de $S(\beta)$ leva ao sistema de equações (2.19),

$$(X^T X)\beta = X^T Y, \tag{2.19}$$

considerando que $(X^T X)$ admite inversa, $\hat{\beta}$ pode ser obtido de acordo com a Equação (2.20),

$$\hat{\beta} = (X^T X)^{-1} X^T Y, \tag{2.20}$$

e a estimativa de um ponto y_i é obtida através da Equação 2.21.

$$\hat{Y} = X\hat{\beta}. \tag{2.21}$$

2.4 Ferramentas de Desenvolvimento

O desenvolvimento de *software* tem-se tornado cada dia mais complexo, por este motivo, ferramentas de apoio ao programador surgiram com o objetivo de auxiliar e facilitar a tarefa de desenvolvimento ou prototipação. Uma ferramenta de desenvolvimento bastante comum é a IDE (*Integrated Development Enviroment*) e consiste basicamente de um software integrado a alguma linguagem de programação que provê diversos recursos que agilizam o processo de desenvolvimento de *software* (REHMAN; PAUL, 2002). A seguir, serão apresentados as principais ferramentas utilizadas para o desenvolvimento deste trabalho.

O *MatLab* (MATLAB, 2016), é uma IDE que provê uma linguagem de script simples e voltada a prototipação que permite programadores, engenheiros, estudantes, pesquisadores ou qualquer usuário com conhecimentos mínimo de programação, prototipar, testar e validar, por exemplo, problemas matemáticos, computacionais e problemas voltados a engenharia. Na Figura 21(a) é possível visualizar uma tela típica do programa. Neste trabalho o *MatLab* foi utilizado como ferramenta de prototipação e validação.

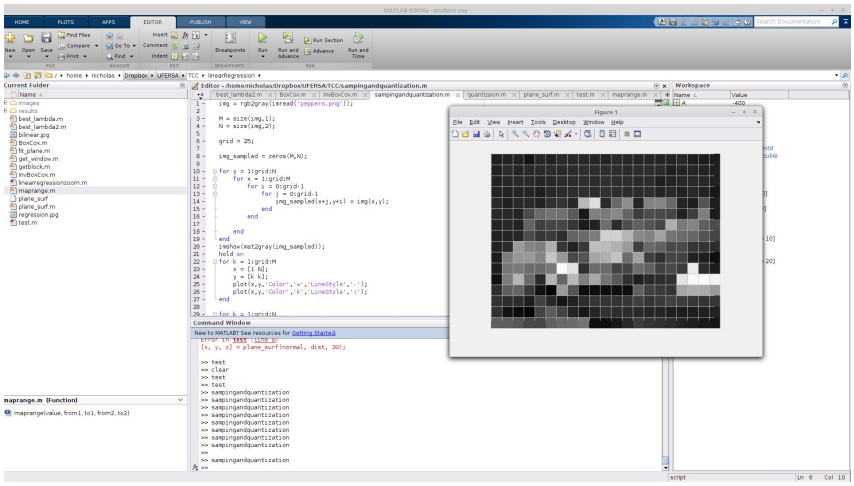
Uma vez que uma ideia tenha sido validada no *MatLab*, o programador pode decidir implementá-la em alguma linguagem de programação de uso geral, tal como C++. C++ é uma linguagem de programação de alto nível (STROUSTRUP, 2000; PRATA, 2001) bastante popular e foi a linguagem escolhida para a implementação final do algoritmo. Seus principais usos estão relacionados a programas que demandam alto poder de processamento, tais como aplicações de processamento de imagens, editores gráficos e jogos de computador.

É possível escrever programas em C++ sem o uso de uma IDE, desde que o usuário faça o uso do compilador. Um compilador é o software responsável por compilar o programa e convertê-lo em uma linguagem que os computadores entendam (AHO; SETHI; ULLMAN, 1986). Apesar disso, o uso de uma IDE voltada para a linguagem C++ facilita o ambiente de desenvolvimento ao automatizar tarefas repetitivas e auxiliar o programador. Um exemplo de IDE para C++ é o *QT Creator* (QTCREATOR, 2016) apresentado na Figura 21(b).

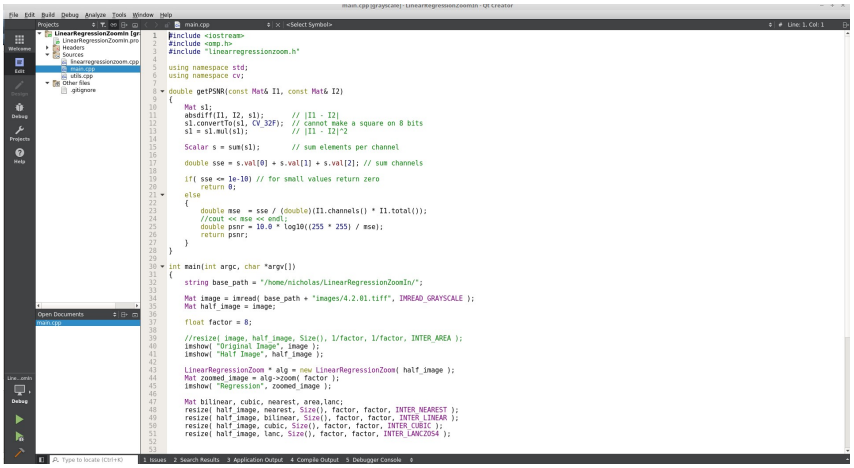
Algoritmos de processamento de imagens precisam ser rápidos e eficientes, por isso, uma das técnicas utilizadas nas implementações destes algoritmos é a técnica de paralelização, que consiste na execução de uma mesma tarefa de forma simultânea (TANENBAUM, 2007). Atualmente a grande maioria dos computadores disponíveis no mercado possuem múltiplos núcleos de processamento, permitindo que programas obtenham ganho de performance ao executar tarefas custosas de forma paralela.

A paralelização pode ser obtida diretamente, por meio de funções de baixo nível expostas pelo sistema operacional ou por bibliotecas que facilitam esse trabalho. O *OpenMP* é um exemplo de biblioteca que permite o desenvolvimento de aplicações de forma paralela utilizando C++ (DAGUM; MENON, 1998). O *OpenMP* possui uma sintaxe que permite ao programador especificar trechos de código que serão executados em paralelo sem a necessidade de programar utilizando a interface de baixo nível do sistema operacional.

Figura 21 – Em (a) tela típica do MatLab e em (b) interface do Qt Creator.



(a) Interface do MatLab



(b) IDE Qt Creator

Fonte: Autoria Própria.

3 Zoom com Regressão Linear

Este capítulo apresenta e descreve o algoritmo de ampliação de imagens desenvolvido. São utilizadas a regressão linear e a transformação de Box-Cox para estimar as cores criadas no processo de ampliação. Em uma das etapas do algoritmo busca-se a função que apresenta o menor erro de interpolação, dadas as cores em uma vizinhança. O algoritmo proposto está dentro da categoria dos algoritmos não-adaptativos, pois tratam todos os pixels da imagem da mesma forma.

A diferença básica entre os algoritmos de *zoom* que utilizam métodos de interpolação é a função de interpolação. No *zoom* por replicação, por exemplo, a função de interpolação é a replicação dos pixels enquanto na interpolação bilinear a função é a interpolação linear aplicada em dois eixos. No algoritmo desenvolvido, a função de interpolação é uma função linear que define a reta que mais se aproxima dos pixels vizinhos de um dado pixel cuja cor é desconhecida.

O algoritmo proposto apresenta algumas semelhanças com os algoritmos de ampliação já apresentados. A diferença como já mencionada, é a função de interpolação. Dessa forma o algoritmo apresentado no Algoritmo 3 serve como base para as discussões subsequentes. Este algoritmo percorre toda a imagem ampliada, afim de preencher seus espaços vazios. A intensidade dos pixels desconhecidos é calculado pela função *RegressionInterpolation*, que é discutida em detalhes nas próximas seções.

Algoritmo 3 Algoritmo base para o *zoom* por regressão linear.

```
1: fator ← N
2: nova_altura ← ROUND(img.altura * fator)
3: nova_largura ← ROUND(img.largura * fator)
4: para y = 0; i < nova_altura; i++ faça
5:     para x = 0; j < nova_largura; j++ faça
6:         sx ← x/fator
7:         sy ← y/fator
8:         nova_imagem(x, y) ← REGRESSIONINTERPOLATION(sx, sy, img)
9:     fim para
10: fim para
```

Assim como acontece na interpolação bilinear, cada pixel vazio da nova imagem é mapeado para um subpixel da imagem original e os pixels que compõem a sua janela 2x2 são utilizados no cálculo da estimativa. O motivo da escolha dessa janela é que com uma janela maior há uma tendência de a imagem resultante ficar com uma aparência de "bloco", pois pixels mais distantes estariam envolvidos na estimativa do novo pixel, com um mesmo peso dos pixels mais próximos.

Devido ao fato de que a intensidade de cada pixel depende de duas variáveis, x e y, que

compõem a coordenada daquele ponto, é necessário o uso da regressão múltipla. Por este motivo, a cor de cada pixel é estimada com a Equação (3.1),

$$\hat{y_i} = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \hat{\beta}_2 x_{2i}, \tag{3.1}$$

em que x_{1i} é i-ésima coordenada na horizontal, x_{2j} é a j-ésima coordenada na vertical e y_i é i-ésimo pixel a ser estimado.

3.1 Estimando os novos pixels

Os coeficientes β são calculados utilizando a notação matricial apresentada na seção 2.3.4 e a Equação (2.20). O primeiro passo é a construção das matrizes X e Y. A matriz X é a matriz que contém as variáveis independentes, em que toda a primeira coluna é igual a 1, a segunda coluna é igual as coordenadas x e a terceira coluna é igual as coordenadas y dos pixels envolvidos na regressão, a saber, os 4 pixels mais próximos do subpixel a ser calculado.

Na Figura 22, é exemplificado como seriam as matrizes X e Y para o subpixel $(0.5, 0.5)$ de uma dada imagem 4x4. Considerando que está sendo aplicado um *zoom* de fator 2, o subpixel $(0.5, 0.5)$ é mapeado no novo pixel $(1, 1)$ na imagem aumentada.

Figura 22 – Em (a) tem-se uma imagem monocromática 4x4, em (b) a respectiva matriz X para o subpixel $(0.5, 0.5)$ e em (c) tem-se a matriz Y contendo os valores de intensidade da vizinhança do subpixel $(0.5, 0.5)$.

$$\begin{bmatrix} 124 & 126 & 127 & 130 \\ 120 & 112 & 125 & 127 \\ 115 & 140 & 120 & 123 \\ 121 & 123 & 129 & 126 \end{bmatrix}$$

(a) Imagem Monocromática 4x4

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

(b) Matriz X

$$\begin{bmatrix} 124 \\ 126 \\ 120 \\ 112 \end{bmatrix}$$

(c) Matriz Y

Fonte: Autoria Própria.

Após o cálculo da Equação (2.20) utilizando as matrizes da Figura 22, são obtidos os valores de 126.5, -9 e -3 respectivamente para $\beta_0, \beta_1 e \beta_2$. O cálculo da estimativa do subpixel

$(0.5, 0.5)$ é então realizado substituindo os valores na Equação (3.2),

$$\begin{aligned} \hat{y} &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \\ &= 126.5 - 9x_1 - 3x_2 \\ &= 126.5 - 9 * 0.5 - 3 * 0.5 = 120.5 \end{aligned} \tag{3.2}$$

o valor 121 (120.5 arredondado para o inteiro mais próximo) representa então a estimativa do subpixel $(0.5, 0.5)$, este por sua vez é mapeado para o pixel real $(1, 1)$ na imagem aumentada. Repetindo este processo para os demais pixels, obtêm-se a reconstrução completa da imagem. O Algoritmo 4 sintetiza todo esse procedimento, porém ainda não incorpora a transformada de Box-Cox.

Algoritmo 4 Algoritmo da função RegressionInterpolation sem Box-Cox.

função REGRESSIONINTERPOLATION(sx,sy,img)
  $dx \leftarrow$ FLOOR(sx) $dy \leftarrow$ FLOOR(sy)
  $A \leftarrow img[dx, dy]$
  $B \leftarrow img[dx, dy + 1]$
  $C \leftarrow img[dx + 1, dy]$
  $D \leftarrow img[dx + 1, dy + 1]$
  $\mathrm{X} \leftarrow \begin{bmatrix} 1 & dx & dy \\ 1 & dx & dy + 1 \\ 1 & dx + 1 & dy \\ 1 & dx + 1 & dy + 1 \end{bmatrix}$
  $\mathrm{Y} \leftarrow \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}$
  $\beta \leftarrow (X^T X)^{-1} X^T Y$
  retorne ROUND$(\beta[0] + sx * \beta[1] + sy * \beta[2])$
fim função

A Figura 23 mostra um trecho de uma imagem ampliada por um fator 2 com alguns pixels não preenchidos. De acordo com o algoritmo, os pixels com posições (0,0), (0,1), (1,0) e (1,1) da imagem ampliada irão utilizar os pixels A, B,C e D da imagem original no cálculo da regressão. Nota-se que a regressão é a mesma para todos esses três pixels, o que muda são os valores de x_{1i} e x_{1j}, representados por sx e sy respectivamente, utilizados na Equação (3.1).

3.2 Transformada Box-Cox

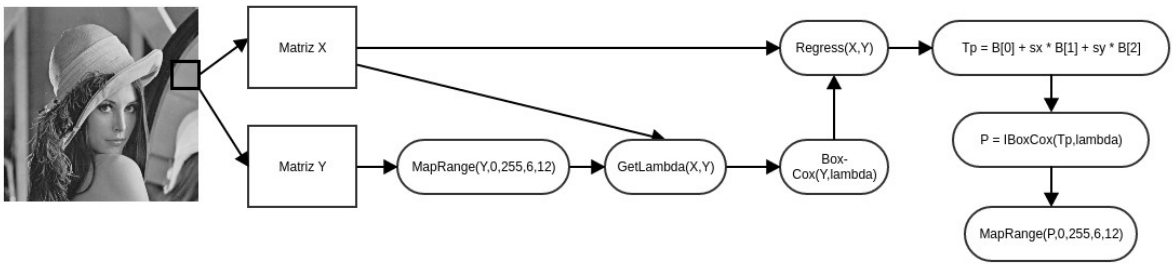
Com o objetivo de otimizar os resultados e diminuir o erro da estimativa, a transformada Box-Cox é utilizada para aproximar relação entre as coordenadas e intensidades dos pixels à uma função linear. É importante salientar que ao transformar a intensidade dos pixels utilizando Box-Cox (Equação (2.14)), se faz necessário após o cálculo da regressão e da estimativa do pixel, agora transformado, converter este valor de volta para o domínio original. Para tanto basta tirar a inversa da transformada de Box-Cox

Figura 23 – Trecho de uma imagem ampliada, contendo pixels não preenchidos.

Fonte: Autoria Própria.

de acordo com a Equação (2.15). A Figura 24 sintetiza todos os passos necessários para a aplicação da transformada.

Figura 24 – Fluxograma da aplicação da transformada.



Fonte: Autoria Própria.

Para minimizar problemas de computação numérica, as intensidades das cores são convertidas para o intervalo $[6, 12]$. A função utilizada para realizar essa conversão é apresentada no Algoritmo 5. Essa função converte um determinado valor no intervalo [x,y] para o intervalo [z,k].

Algoritmo 5 Função para mapear um valor pertence a um determinado intervalo a um outro intervalo.

```
1: função MAPRANGE(value,x,y,z,k)
2:     retorne (value − x)/(y − x) ∗ (k − z) + z
3: fim função
```

Como discutido anteriormente, a transformada de Box-Cox depende de um parâmetro denominado lambda (λ). Encontrar o melhor λ é uma tarefa primordial para obter bons resultados com a transformada Box-Cox. Para minimizar problemas de computação numérica, foi escolhido o intervalo [-5,5] para encontrar o melhor lambda. O método para a busca consiste em buscar neste intervalo o lambda gerador

de uma função, que quando calculada a inversa, apresente o menor erro em relação a função original. O procedimento para o cálculo do erro é descrito na Equação (3.3),

$$\begin{aligned} H &= X * (X^T X)^{-1} X^T \\ T_y &= BoxCox(Y, \lambda) \\ HT_y &= H * T_y \\ H_y &= IBoxCox(HT_y, \lambda) \\ err &= (H_y - Y)^T * (H_y - Y) \end{aligned} \tag{3.3}$$

em que $BoxCox$ e $IBoxCox$ são funções que calculam a transformada de Box-Cox e sua inversa respectivamente.

A implementação do algoritmo *GetLambda* utilizado para encontrar o melhor lambda é descrito no Algoritmo 6. Vale destacar que X e Y são as matrizes utilizadas na regressão e não as coordenadas da imagem, já que estas são referenciadas por x e y. O Algoritmo 7 apresenta a função *RegressionInterpolation* modificada para utilizar Box-Cox antes do cálculo da regressão.

Algoritmo 6 Função GetLambda.

```
1:  função GETLAMBDA(X,Y)
2:      min_err ← 1^10
3:      H ← X * (X^T X)^-1 X^T
4:      para k = −5; k <= 5; k = k + 0.01 faça
5:          λ ← k
6:          T_y ← BOXCOX(Y, λ)
7:          HT_y ← H * T_y
8:          H_y ← IBOXCOX(HT_y, λ)
9:          err ← (H_y − Y)^T * (H_y − Y)
10:         se min_err > err então
11:             min_err ← err
12:             best_lambda ← λ
13:         fim se
14:     fim para
15:     retorne best_lambda
16: fim função
```

3.3 Otimizações

Como já mencionado anteriormente e exemplificado através da Figura 23, o cálculo dos coeficientes de regressão sempre será o mesmo para os subpixels dentro da mesma janela. Por este motivo, é possível otimizar o algoritmo e armazenar os coeficientes já calculados para cada janela, afim de reutilizá-los e evitar repetição de cálculos. O Algoritmo 8 apresenta o algoritmo modificado com esta otimização.

Algoritmo 7 Função RegressionInterpolation com BoxCox.

função REGRESSIONINTERPOLATION(sx,sy,img)
    $dx \leftarrow$ FLOOR(sx) $dy \leftarrow$ FLOOR(sy)
    $A \leftarrow img[dx, dy]$
    $B \leftarrow img[dx, dy + 1]$
    $C \leftarrow img[dx + 1, dy]$
    $D \leftarrow img[dx + 1, dy + 1]$
    $\mathrm{X} \leftarrow \begin{bmatrix} 1 & dx & dy \\ 1 & dx & dy+1 \\ 1 & dx+1 & dy \\ 1 & dx+1 & dy+1 \end{bmatrix}$
    $\mathrm{Y} \leftarrow \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}$
    $Y \leftarrow$ MAPRANGE$(Y, 0, 255, 6, 12)$
    $\lambda \leftarrow$ GETLAMBDA(X, Y)
    $T_y \leftarrow$ BOXCOX(Y, λ)
    $\beta \leftarrow (X^T X)^{-1} X^T T_y$
    $T_p \leftarrow \beta[0] + sx * \beta[1] + sy * \beta[2]$
    $p \leftarrow$ IBOXCOX(T_p, λ)
    $p \leftarrow$ MAPRANGE$(p, 6, 12, 0, 255)$
    retorne ROUND(p)
fim função

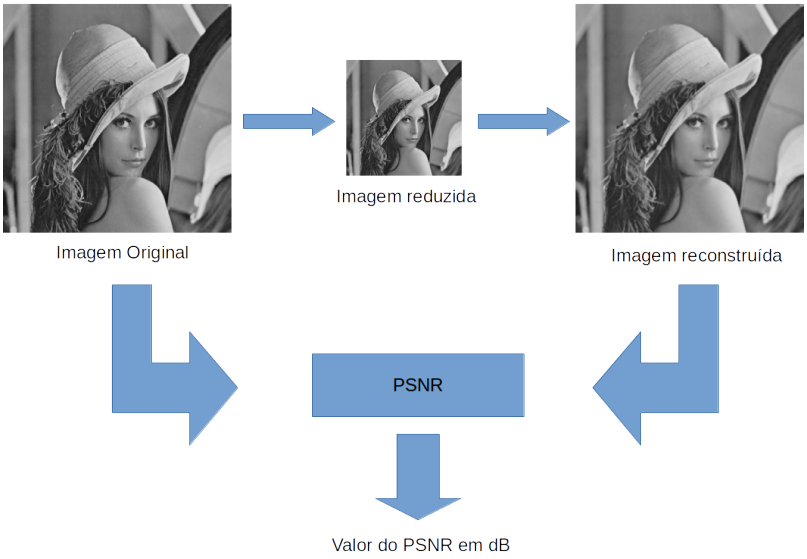
Algoritmo 8 Versão otimizada do Algoritmo 7.

função REGRESSIONINTERPOLATION(sx,sy,img)
  $dx \leftarrow \text{FLOOR}(sx)$
  $dy \leftarrow \text{FLOOR}(sy)$
  se $c\beta[dx][dy]$ **então**
    $\beta \leftarrow c\beta[dx][dy]$
    $\lambda \leftarrow lambdas[dx][dy]$
    $T_p \leftarrow \beta[0] + sx * \beta[1] + sy * \beta[2]$
    $p \leftarrow \text{IBOXCOX}(T_p, \lambda)$
    $p \leftarrow \text{MAPRANGE}(p, 6, 12, 0, 255)$
  senão
    $A \leftarrow img[dx, dy]$
    $B \leftarrow img[dx, dy + 1]$
    $C \leftarrow img[dx + 1, dy]$
    $D \leftarrow img[dx + 1, dy + 1]$
    $\mathrm{X} \leftarrow \begin{bmatrix} 1 & dx & dy \\ 1 & dx & dy+1 \\ 1 & dx+1 & dy \\ 1 & dx+1 & dy+1 \end{bmatrix}$
    $\mathrm{Y} \leftarrow \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix}$
    $Y \leftarrow \text{MAPRANGE}(Y, 0, 255, 6, 12)$
    $\lambda \leftarrow \text{GETLAMBDA}(X, Y)$
    $T_y \leftarrow \text{BOXCOX}(Y, \lambda)$
    $\beta \leftarrow (X^T X)^{-1} X^T T_y$
    $T_p \leftarrow \beta[0] + sx * \beta[1] + sy * \beta[2]$
    $p \leftarrow \text{IBOXCOX}(T_p, \lambda)$
    $p \leftarrow \text{MAPRANGE}(p, 6, 12, 0, 255)$
    $c\beta[dx][dy] \leftarrow \beta$
    $lambdas[dx][dy] \leftarrow \lambda$
  fim se
  retorne ROUND(p)
fim função

4 Resultados

Neste capítulo são apresentados os resultados obtidos com o algoritmo descrito no capítulo anterior. Quatro imagens com características distintas foram escolhidas para os testes. A metodologia dos testes consiste em realizar uma operação de redução (*Zoom Out*) e em seguida realizar a operação de ampliação com cada um dos algoritmos com o objetivo de retornar a imagem reduzida a sua dimensão original. Essa metodologia é sintetizada na Figura 25.

Figura 25 – Metodologia dos testes.



Fonte: Autoria Própria;

Através dessa metodologia é possível comparar a imagem original com as imagens reconstruídas e calcular o seu respectivo PSNR, uma vez que elas possuirão a mesma resolução. Essa metodologia é amplamente utilizada na comparação e na avaliação de algoritmos de ampliação (GONZALEZ; WOODS, 2006; GAO; SONG; TAI, 2009; ALMIRA; ROMERO, 2011; SABRIN; ALI, 2014). Este procedimento será realizado com os algoritmos de replicação, interpolação bilinear, interpolação bicúbica e o algoritmo proposto. Afim de obter uma análise acerca do ganho com a utilização da transformada Box-Cox, também serão apresentados os resultados da aplicação do algoritmo proposto com e sem a transformada. É importante destacar que quanto maior o valor do PSNR melhor é a reconstrução da imagem e consequentemente menor é a distorção.

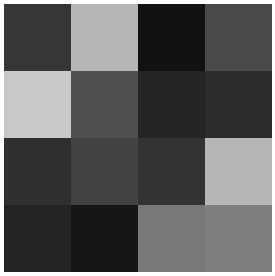
As quatro imagens utilizadas nos testes são apresentadas na Figura 26. Todas as imagens possuem a dimensão de 512x512 e serão aplicados os seguintes fatores de zoom: 2,4,8 e 16. Nas próximas seções serão apresentados os resultados para cada uma das imagens e discutidos os resultados obtidos.

Figura 26 – Conjunto de imagens utilizadas para os testes

(a) Imagem 1 - Lena

(b) Imagem 2 - Peppers

(c) Imagem 3 - Rose



(d) Imagem 4 - Squares

Fonte: ImageProcessingPlace (2006).

4.1 Imagem 1 - Lena

A imagem da lena é uma imagem considerada padrão e bastante utilizada no PDI, ela é atrativa pelo fato de possuir uma boa mistura de detalhes, regiões planas, sombreamento e texturas que se mostram bastante interessante para o teste de diversos algoritmos de processamento de imagens, inclusive para algoritmos de ampliação. A Figura 27 apresenta um comparativo gráfico dos resultados obtidos, enquanto que a Tabela 2 elenca os valores de PSNR obtidos com cada algoritmo e para cada fator de *zoom* para a primeira imagem testada.

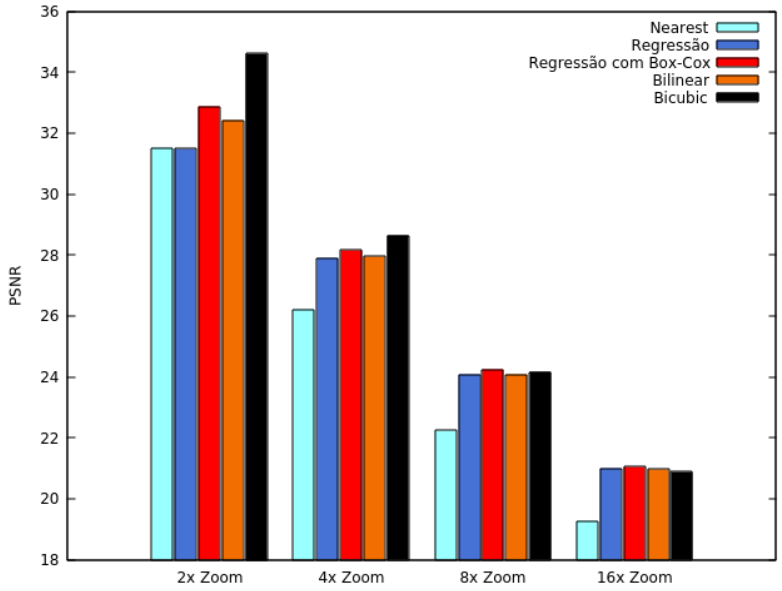
Tabela 2 – Valores do PSNR obtidos na reconstrução da imagem 1 com os algoritmos testados.

Fator	Replicação	Regressão	Regressão com Box-Cox	Bilinear	Bicúbica
2x	31.5086 dB	31.4943 dB	32.8691 dB	32.4271 dB	34.6219 dB
4x	26.2055 dB	27.9036 dB	28.1627 dB	27.9588 dB	28.6356 dB
8x	22.2397 dB	24.0807 dB	24.243 dB	24.0835 dB	24.1435 dB
16x	19.264 dB	20.997 dB	21.0558 dB	20.9735 dB	20.8902 dB

Fonte: Autoria Própria

Nesta imagem, o algoritmo proposto apresentou resultados melhores que a interpolação bilinear em todos os fatores testados. É possível notar também que houve uma notável diferença entre o algoritmo proposto sem Box-Cox e a versão com Box-Cox para os fatores de *zoom* 2 e 4. A medida que o fator aumenta, entretanto, a tendência é de equiparação entre os algoritmos testados, com exceção do algoritmo de replicação, que para fatores de *zoom* maiores que 2 apresenta considerável nível de distorção nas imagens. Para os fatores de *zoom* 8 e 16, o algoritmo proposto apresentou níveis de PSNR um pouco

Figura 27 – Resultados obtidos com a primeira imagem testada.



Fonte: Autoria Própria.

melhores que a bicúbica. Na Figura 28 é possível ver as imagens reconstruídas com fator 4 para cada algoritmo testado.

4.2 Imagem 2 - Peppers

A segunda imagem testada apresenta como característica principal a presença de diversas bordas na imagem, ou seja, áreas que apresentam uma abrupta variação na intensidade de cor de pixels próximos. Essa imagem também é uma imagem comumente encontrada na literatura e bastante utilizada como imagem de teste. Os resultados para esta imagem são apresentados na Figura 29 e na Tabela 3.

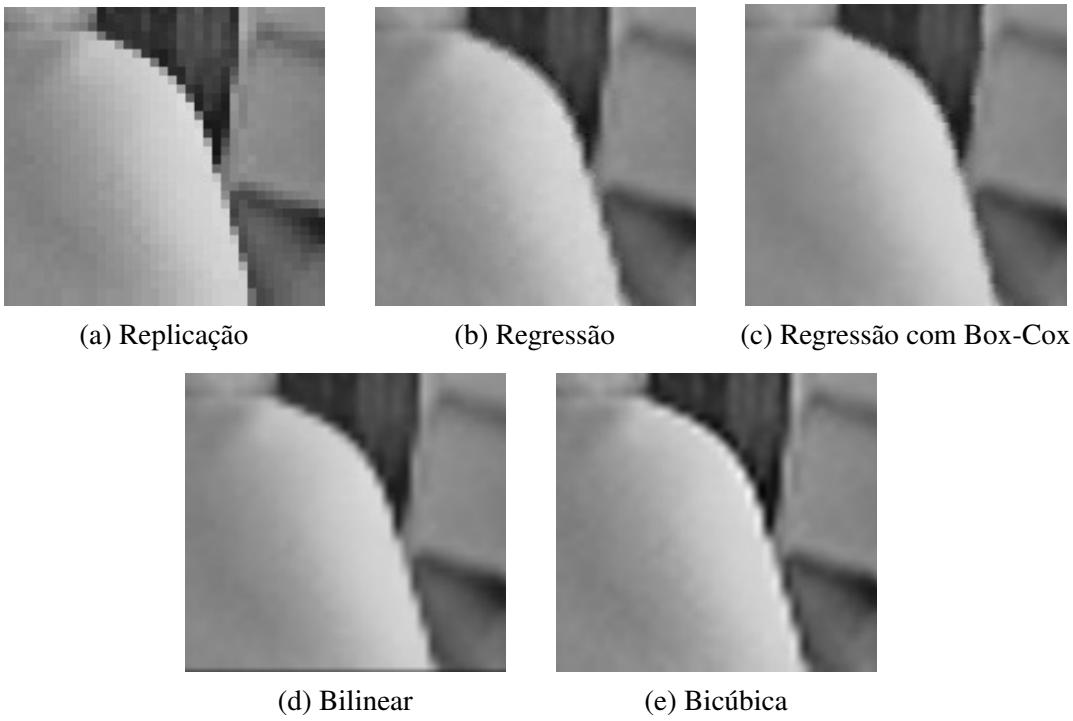
Tabela 3 – Valores do PSNR obtidos na reconstrução da imagem 2 com os algoritmos testados.

Fator	Replicação	Regressão	Regressão com Box-Cox	Bilinear	Bicúbica
2x	29.7885 dB	30.1467 dB	31.0636 dB	30.6941 dB	31.9527 dB
4x	25.2087 dB	26.804 dB	27.0772 dB	26.7146 dB	27.2483 dB
8x	21.4475 dB	23.5529 dB	23.7569 dB	23.5266 dB	23.7695 dB
16x	17.939 dB	20.1081 dB	20.1828 dB	20.1237 dB	20.1 dB

Fonte: Autoria Própria.

O algoritmo proposto mais uma vez apresentou resultados melhores que a interpolação bilinear para todos os fatores de ampliação testados. O algoritmo proposto também apresentou níveis de distorção próximos a da interpolação bicúbica, chegando a ultrapassá-la no fator de *zoom* 16. Os resultados também mostram que, novamente, houve uma notável diferença entre a versão do algoritmo sem Box-Cox e com Box-Cox. Na Figura 30 é possível ver as imagens reconstruídas com fator 4 para cada algoritmo testado.

Figura 28 – Primeira imagem testada, reconstruída com fator 4 com cada algoritmo de Zoom.



(a) Replicação (b) Regressão (c) Regressão com Box-Cox

(d) Bilinear (e) Bicúbica

Fonte: Autoria Própria.

4.3 Imagem 3 - Rose

A terceira imagem testada também é bastante utilizada como imagem de teste e apresenta um pano de fundo preto e homogêneo com uma rosa no centro. Há uma abrupta variação nas bordas da rosa e ela é bem definida, apresentando vários contornos e sombreamentos. Apesar dessa abrupta variação nas bordas, todo o restante da imagem apresenta pouca variação de cor. A Figura 31 apresenta um comparativo gráfico dos resultados. Enquanto que a Tabela 4 elenca os valores de PSNR obtidos com cada algoritmo para cada fator de zoom.

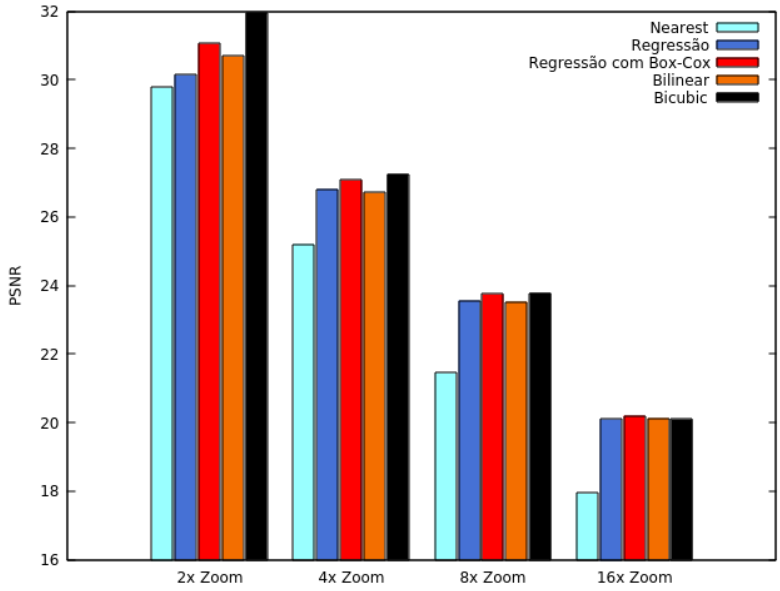
Tabela 4 – Valores do PSNR obtidos na reconstrução da imagem 3 com os algoritmos testados.

Fator	Replicação	Regressão	Regressão com Box-Cox	Bilinear	Bicúbica
2x	34.1105 dB	34.3083 dB	36.0802 dB	35.5583 dB	37.3503 dB
4x	28.8497 dB	30.852 dB	31.3548 dB	31.1714 dB	31.8016 dB
8x	24.6126 dB	26.6858 dB	26.9422 dB	26.8444 dB	27.0218 dB
16x	19.264 dB	22.9684 dB	23.2244 dB	23.0537 dB	21.0171 dB

Fonte: Autoria Própria.

Novamente o algoritmo proposto apresentou resultados melhores que o *zoom* por interpolação bilinear e se aproximou bastante da interpolação bicúbica, ultrapassando-a para o fator de zoom 16. Houve um distanciamento um pouco maior entre a versão sem Box-Cox e a versão com Box-Cox do algoritmo proposto principalmente no fator de zoom 2. Tal diferença se torna clara ao comparar a imagem

Figura 29 – Resultados obtidos com a segunda imagem testada.



Fonte: Autoria Própria.

reconstruída pela versão do algoritmo sem Box-Cox com a versão do algoritmo com Box-Cox. Há uma notória introdução de ruído na imagem, como é possível observar na Figura 32. Na Figura 33 é possível ver as imagens reconstruídas com fator 4 para cada algoritmo testado.

4.4 Imagem 4 - Squares

A última imagem testada é uma imagem artificial criada especificamente para os testes realizados neste trabalho. Essa imagem é compostas por vários quadrados homogêneos de tonalidades diferentes, semelhante a um tabuleiro de xadrez. Os resultados obtidos para essa imagem estão na Figura 34 e na Tabela 5.

Tabela 5 – Valores do PSNR obtidos na reconstrução da imagem 4 com os algoritmos testados.

Fator	Replicação	Regressão	Regressão com Box-Cox	Bilinear	Bicúbica
2x	inf	36.3632 dB	39.9639 dB	38.8251 dB	39.9593 dB
4x	inf	35.8685 dB	35.964 dB	34.9042 dB	35.8752 dB
8x	inf	32.7509 dB	32.7708 dB	31.6123 dB	32.6276 dB
16x	inf	29.6441 dB	29.7793 dB	28.5587 dB	29.5866 dB

Fonte: Autoria Própria.

A primeira observação é que o melhor algoritmo para esta imagem foi o algoritmo de replicação, que não adicionou nenhuma distorção a imagem e por isso obteve uma reconstrução perfeita da imagem (PSNR tende ao infinito). Isto se deve ao fato a imagem ser quadrada (número de linhas é igual ao número de colunas) composta por diversos quadrados homogêneos e pelo fato dos fatores de zoom aplicados

Figura 30 – Segunda imagem testada, reconstruída com fator 4 com cada algoritmo de Zoom.

(a) Replicação

(b) Regressão

(c) Regressão com Box-Cox

(d) Bilinear

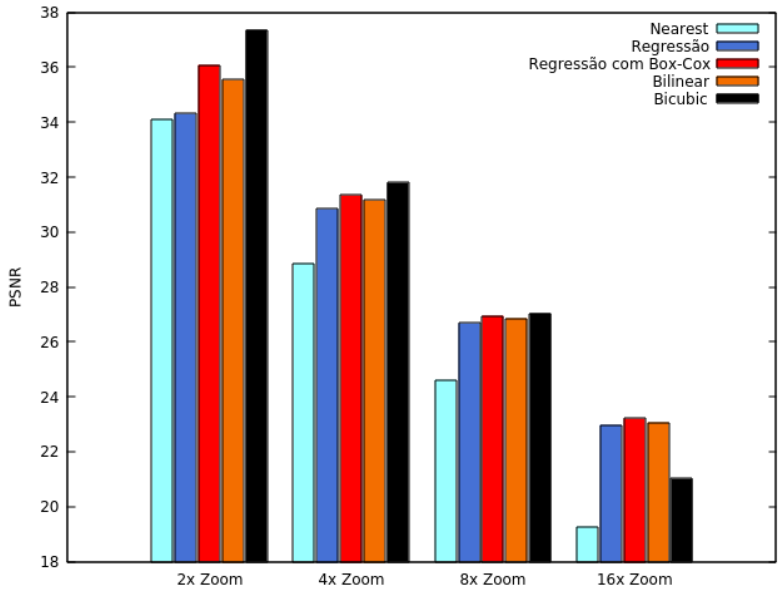
(e) Bicúbica

Fonte: Autoria Própria.

serem todos múltiplos de 2. Como já explicado, o algoritmo de replicação aumenta a área de um pixel, esta ação nesta imagem em particular, não causa nenhum efeito serrilhado ou de borramento. Por este motivo, o algoritmo de replicação não foi adicionado ao comparativo da Figura 34.

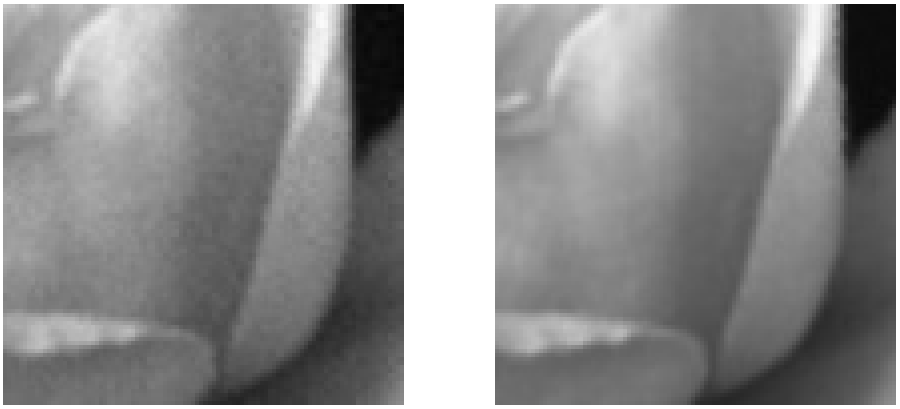
Por também apresentar áreas com pouca ou nenhuma variação de intensidade de cor, a imagem reconstruída com a versão do algoritmo sem Box-Cox adicionou bastante ruído na imagem resultante, como é possível observar claramente na Figura 35. Já na Figura 36 é possível ver as imagens reconstruídas com fator 4 para cada algoritmo testado.

Figura 31 – Resultados obtidos com a terceira imagem testada.



Fonte: Autoria Própria.

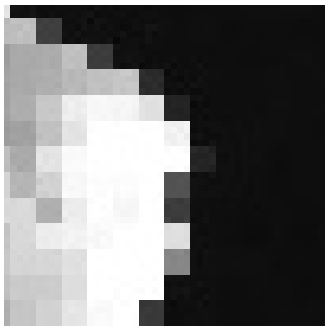
Figura 32 – Em (a) trecho da imagem reconstruída sem Box-Cox, em (b) trecho da imagem reconstruída com Box-Cox.



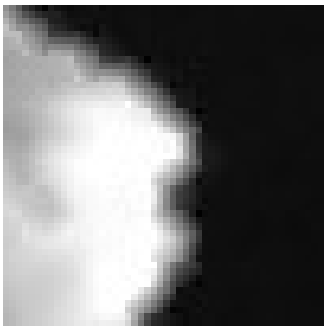
(a) Sem Box-Cox (b) Com Box-Cox

Fonte: Autoria Própria.

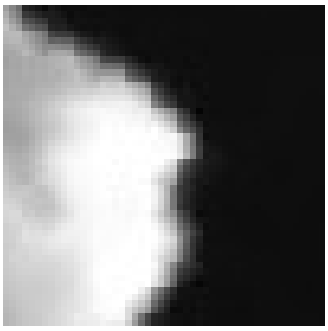
Figura 33 – Terceira imagem testada, reconstruída com fator 4 com cada algoritmo de Zoom.



(a) Replicação



(b) Regressão



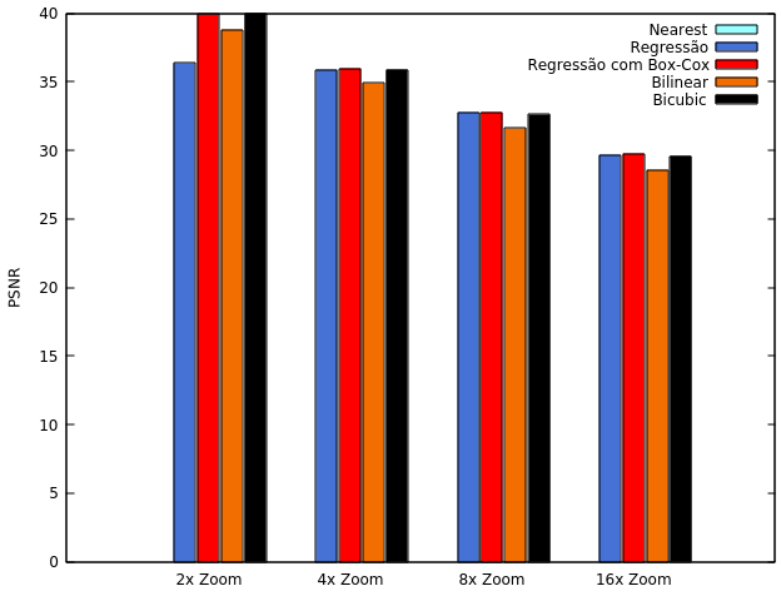
(c) Regressão com Box-Cox

(d) Bilinear

(e) Bicúbica

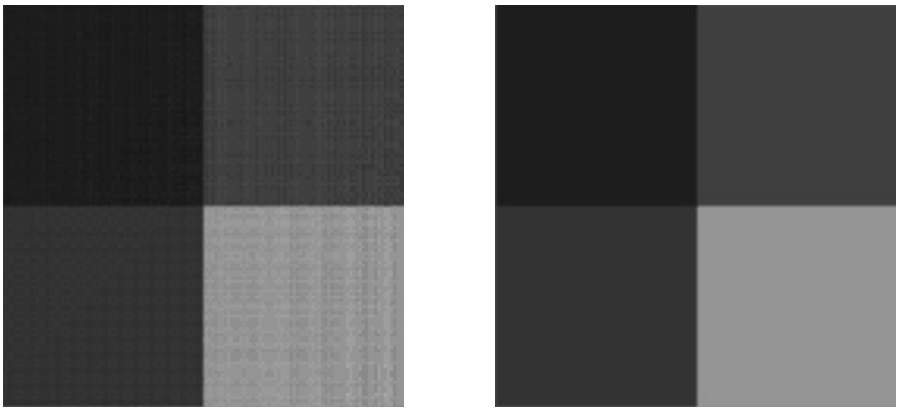
Fonte: Autoria Própria.

Figura 34 – Resultados obtidos com a quarta imagem testada.



Fonte: Autoria Própria

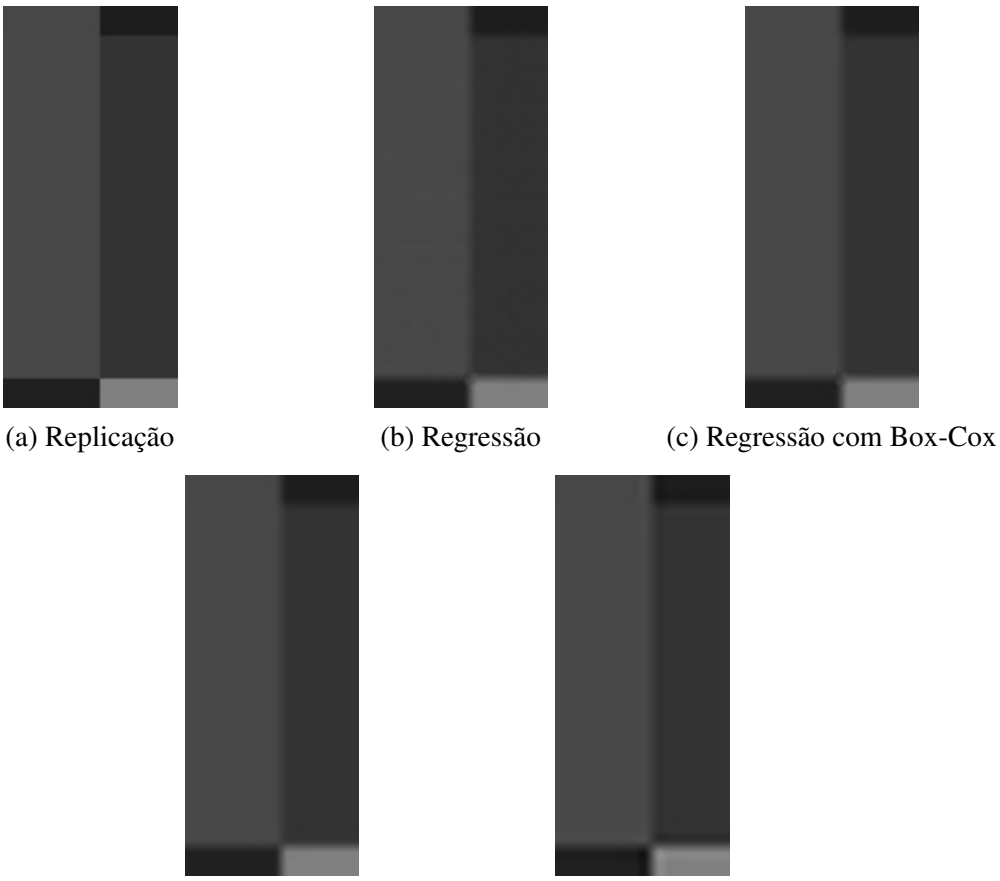
Figura 35 – Em (a) trecho da imagem reconstruída sem Box-Cox, em (b) trecho da imagem reconstruída com Box-Cox.



(a) Sem Box-Cox (b) Com Box-Cox

Fonte: Autoria Própria.

Figura 36 – Quarta imagem testada, reconstruída com fator 4 com cada algoritmo de Zoom.



(a) Replicação (b) Regressão (c) Regressão com Box-Cox

(d) Bilinear (e) Bicúbica

Fonte: Autoria Própria

5 Conclusão e Trabalhos Futuros

Este trabalho propôs um novo método de interpolação de imagens que utiliza regressão linear e Box-Cox para estimar as cores dos pixels no processo de reconstrução da imagem. O algoritmo de interpolação desenvolvido foi utilizado para propor um novo método de ampliação de imagens. Apesar disso, o método de interpolação proposto pode ser utilizado em qualquer outra operação que necessite reconstruir imagens através de interpolação.

Na regressão os quatro pixels mais próximo do pixel a ser estimado são utilizados como variáveis dependentes. Já as coordenadas x e y desses pixels são utilizadas como variáveis de previsão. Como cada variável de resposta depende de duas variáveis de previsão, a regressão aplicada é a regressão múltipla. A transformada de Box-Cox foi utilizada para linearizar os pontos e minimizar o erro.

A utilização de uma janela composta por mais que quatro pixels se mostrou insatisfatória, pois a imagem resultante ficou com uma aparência de "blocos". Isto se deve ao fato de que pixels mais distantes são utilizados no cálculo da regressão com o mesmo peso dos pixels mais próximos. A literatura mostra que para janelas maiores que 2x2, se faz necessário a aplicação de pesos a cada um dos pixels utilizados na interpolação afim de priorizar os pixels mais próximos (GONZALEZ; WOODS, 2006).

Os resultados obtidos com a utilização do algoritmo de interpolação proposto na ampliação de imagens se mostraram satisfatórios. O algoritmo proposto apresentou melhores resultados quando comparado a interpolação bilinear em todas as imagens testadas. Para alguns fatores de zoom, o algoritmo proposto chegou a inclusive ultrapassar a interpolação bicúbica, que utiliza uma janela maior, composta pelos 16 pixels mais próximos.

Também foi apresentado um comparativo entre o algoritmo proposto com e sem Box-Cox. Foi possível observar numericamente e empiricamente que a transformada de Box-Cox se mostrou uma peça fundamental para a estimativa dos novos pixels. Em todos os testes, houve uma diferença considerável nos níveis de PSNR entre ambas as versões.

Em duas das imagens testadas foi possível observar de forma clara, que a versão do algoritmo sem Box-Cox adicionou bastante ruído nas porções da imagem onde há pouca ou nenhuma variação de cor. Essas áreas apresentam janelas contendo intensidade de cores não lineares e consequentemente a estimativa da reta para essas áreas apresentou um grande erro. O uso da transformada de Box-Cox permitiu a utilização de um único e simples modelo de regressão ao buscar a linearização dos pontos. Apesar de não ser possível garantir a linearidade, a utilização dessa transformada se mostrou bastante satisfatória.

Um outro resultado importante é que o algoritmo proposto tende a apresentar melhores resultados em altos fatores de ampliação. Nos resultados obtidos, o algoritmo proposto apresentou melhores níveis de PSNR para todas as imagens testadas para o fator de *zoom* 16 e em três das imagens testadas para o fator de *zoom* 8.

Como trabalhos futuros, propõe-se a utilização do método *Weighted Least Squares* (WLS)

(MONTGOMERY; PECK; VINING, 2012) para o cálculo da regressão e a utilização de outras transformadas afim de linearizar os pontos. O método WLS permite atribuir peso a cada uma das variáveis de resposta ao realizar o cálculo da regressão. A utilização desse método pode abrir portas para o aumento da janela utilizada pelo algoritmo e assim apresentar melhores resultados para fatores de zoom intermediários. O WLS introduz uma matriz W, em que os pesos atribuídos a cada variável de resposta se encontram na diagonal dessa matriz. A estimativa dos coeficientes β podem então ser obtidos através da Equação (5.1).

$$\hat{\beta} = (X^T W X)^{-1} X^T W Y \tag{5.1}$$

Além da transformada de Box-Cox existem outras transformadas utilizadas para tornar a relação aproximadamente linear, como por exemplo, as transformações logarítmicas e de potência (CHATTERJEE; HADI, 2006; MONTGOMERY; PECK; VINING, 2012). Propõe-se um estudo mais aprofundado sobre essas transformações afim de verificar se elas são apropriadas para o uso com o algoritmo desenvolvido.

Referências

AHO, A. V.; SETHI, R.; ULLMAN, J. D. *Compilers: Principles, Techniques, and Tools*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1986. ISBN 0-201-10088-6.

ALMIRA, J.; ROMERO, A. Image zooming based on sampling theorems. *Materials matemàtics*, p. 0001–35, 2011.

BATTIATO, S.; GALLO, G.; STANCO, F. A locally adaptive zooming algorithm for digital images. *Image and vision computing*, Elsevier, v. 20, n. 11, p. 805–812, 2002.

Bocsika. *Bilinear interpolation*. 2009. [Online; acessado em 05 de Outubro de 2016]. Disponível em: <https://commons.wikimedia.org/wiki/File:BilinearInterpolation.svg>.

BOX, G.; COX, D. *An Analysis of Transformations Revisited, Rebutted.* [S.l.], 1981.

CHADDA, S.; KAUR, N.; THAKUR, R. Zooming techniques for digital images: A survey 1. Citeseer, 2012.

CHATTERJEE, S.; HADI, A. S. *Regression Analysis by Example*. [S.l.]: John Wiley, Sons, Inc., 2006. ISBN 9780470055465.

DAGUM, L.; MENON, R. Openmp: An industry-standard api for shared-memory programming. *IEEE Comput. Sci. Eng.*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 5, n. 1, p. 46–55, jan. 1998. ISSN 1070-9924. Disponível em: <http://dx.doi.org/10.1109/99.660313>.

David M. Lane1. *Introduction to Statistics*. 2007. [Online; acessado em 10 de Outubro de 2016]. Disponível em: <http://onlinestatbook.com/2/index.html>.

Demofox. *Resizing Images With Bicubic Interpolation*. 2015. [Online; acessado em 27 de Outubro de 2016]. Disponível em: <http://blog.demofox.org/2015/08/15/resizing-images-with-bicubic-interpolation/>.

FLORIN, T.; ARSINTE, R.; MASTORAKIS, N. E. Resolution analysis of an image acquisition system. In: *Proceedings of the 5th European Conference on European Computing Conference*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2011. (ECC'11), p. 382–391. ISBN 978-960-474-297-4. Disponível em: <http://dl.acm.org/citation.cfm?id=1991016.1991085>.

GAO, R.; SONG, J.-P.; TAI, X.-C. Image zooming algorithm based on partial differential equations technique. *International Journal of Numerical Analysis and Modeling*, v. 6, n. 2, p. 284–292, 2009.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 013168728X.

HORE, A.; ZIOU, D. Image quality metrics: Psnr vs. ssim. In: IEEE. *Pattern recognition (icpr), 2010 20th international conference on*. [S.l.], 2010. p. 2366–2369.

HUNT, R. *The Reproduction of Colour (6rd Edition)*. [S.l.]: John Wiley, Sons, Ltd, 2005. ISBN 9780470024270.

HUYNH-THU, Q.; GHANBARI, M. Scope of validity of psnr in image/video quality assessment. *Electronics letters*, IET, v. 44, n. 13, p. 800–801, 2008.

ImageProcessingPlace. *Images Processing Database*. 2006. [Online; acessado em 01 de Outubro de 2016]. Disponível em: <http://www.imageprocessingplace.com/root_files_V3/image_databases.htm>.

JAIN, A. K. *Fundamentals of Digital Image Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1989. ISBN 0-13-336165-9.

KEYS, R. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 29, n. 6, p. 1153–1160, Dec 1981. ISSN 0096-3518.

LALONDE, S. M. Transforming variables for normality and linearity—when, how, why and why not's. In: *SAS Conference Proceedings NESUG*. [S.l.: s.n.], 2005. p. 11–14.

LI, Z.-N.; DREW, M. S.; LIU, J. *Fundamentals of Multimedia*. 2nd. ed. [S.l.]: Springer Publishing Company, Incorporated, 2014. ISBN 3319052896, 9783319052892.

LUENBERGER, D. G. *Optimization by Vector Space Methods*. 1st. ed. New York, NY, USA: John Wiley & Sons, Inc., 1997. ISBN 047118117X.

MATLAB. *version 9.0.0 (R2016a)*. Natick, Massachusetts: The MathWorks Inc., 2016.

MEIJERING, E. A chronology of interpolation: From ancient astronomy to modern signal and image processing. *Proceedings of the IEEE*, v. 90, n. 3, p. 319–342, March 2002.

MONTGOMERY, D.; PECK, E.; VINING, G. *Introduction to Linear Regression Analysis*. Wiley, 2012. (Wiley Series in Probability and Statistics). ISBN 9780470542811. Disponível em: <https://books.google.com.br/books?id=0yR4KUL4VDkC>.

PITAS, I. *Digital image processing algorithms and applications*. [S.l.]: John Wiley & Sons, 2000.

PRATA, S. *C++ Primer Plus (Fourth Edition)*. 4th. ed. Indianapolis, IN, USA: Sams, 2001. ISBN 0672322234.

QTCREATOR. *version 4.0.3*. Bertel Jungin aukio D3A,Finland: The Qt Company., 2016.

REHMAN; PAUL, C. *The Linux Development Platform: Configuring, Usin and Mainting a Complete Programming Environment*. [S.l.]: Prentice Hall Professional Technical Reference, 2002. ISBN 0130091154.

RUSS, J. C. *The Image Processing Handbook, Sixth Edition*. 6th. ed. Boca Raton, FL, USA: CRC Press, Inc., 2011. ISBN 1439840458, 9781439840450.

SABRIN, K. M.; ALI, M. H. An intelligent pixel replication technique by binary decomposition for digital image zooming. *arXiv preprint arXiv:1405.3195*, 2014.

SAKIA, R. The box-cox transformation technique: a review. *The statistician*, JSTOR, p. 169–178, 1992.

STROUSTRUP, B. *The C++ Programming Language*. 3rd. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000. ISBN 0201700735.

TANENBAUM, A. S. *Modern Operating Systems*. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2007. ISBN 9780136006633.

THYAGARAJAN, K. *Still Image and Video Compression with MATLAB*. Wiley, 2011. ISBN 9781118097762. Disponível em: <https://books.google.com.br/books?id=adv71MkRIWYC>.

Todd Veldhuizen. *Measures of image quality*. 1998. [Online; acessado em 02 de Outubro de 2016]. Disponível em: <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VELDHUIZEN/node18.html>.

WANG, Z. et al. Image quality assessment: From error visibility to structural similarity. *Trans. Img. Proc.*, IEEE Press, Piscataway, NJ, USA, v. 13, n. 4, p. 600–612, abr. 2004. ISSN 1057-7149. Disponível em: <http://dx.doi.org/10.1109/TIP.2003.819861>.

Weisstein, Eric W. *Least Squares Fitting*. 2016. [Online; acessado em 10 de Outubro de 2016]. Disponível em: <http://mathworld.wolfram.com/LeastSquaresFitting.html>.

WYSZECKI, G.; STILES, W. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley, 2000. (Wiley Series in Pure and Applied Optics). ISBN 9780471399186. Disponível em: <https://books.google.com/books?id=_51HDcjWZPwC>.