Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Engenharia Eletrônica e Computação

INTERPOLAÇÃO DE IMAGENS USANDO REDES NEURAIS ARTIFICIAIS

Autor:

Fabiano Malhard Araújo de Barros

Orientador:

Luiz Pereira Caloba

Examinador:

Eduardo Antonio Barros da Silva

Examinador:

Gelson Vieira Mendonça

DEL

RIO DE JANEIRO

Março/2009

Dedicatória:

Dedico este trabalho àqueles que contribuíram de forma direta ou indireta para a conclusão de mais esta etapa.

Aos amigos Hudson, Ary, Fernando, René, Alex e a todos os outros, deixo aqui meu abraço.

Aos irmãos Monique e Fellipe, espero que sirva de motivação.

À minha mãe, que sempre lutou para que um dia isso fosse possível, meus maiores sentimentos de gratidão que nunca serão suficientes para expressar todo meu amor.

Agradecimentos:

Agradeço especialmente ao amigo Fernando, que me ajudou em diversos momentos da graduação, oferecendo material, ajuda e apoio em trabalhos conjuntos, e o mais importante o incentivo pra sempre seguir em frente.

Resumo:

Muitas vezes as imagens disponíveis possuem tamanhos reduzidos, ou mesmo as imagens que possuem tamanho significativo contém pequenas regiões, ricas em informações, difíceis de serem trabalhadas.

Nos dois casos ocorrem os seguintes problemas: são imagens difíceis de serem visualizadas, pois têm algumas características específicas, como por exemplo, objetos, bordas, arestas, cantos, que devem ser identificadas e extraídas.

Por isso, a solução direta para esse problema é a ampliação da imagem usando algum método de interpolação.

Durante o projeto tentamos desenvolver um modelo em Redes Neurais que tivesse desempenho ao menos similar aos já tradicionais métodos de interpolação utilizados comercialmente nos softwares de processamento de imagens.

Mostraremos alguns métodos desenvolvidos, detalharemos todo o código para que eles possam ser reproduzidos e apresentaremos os resultados.

Estes estarão descritos matematicamente através de comparação de Erro Médio Quadrático e Relação Sinal Ruído, mas também apresentaremos todas as imagens utilizadas e geradas pelos diversos métodos, para que seja possível uma avaliação visual dos resultados de cada abordagem.

Palavras-chave:

Interpolação, Redes Neurais, MatLab.

Sumário:

Lista de Figuras:

Lista de Tabelas:

Capítulo 1

Introdução

Tema

O trabalho engloba as áreas de Processamento de Imagens e Redes Neurais. Desenvolvemos vários métodos para realizar a interpolação de imagens mas apresentaremos somente os 3 que tiveram rendimentos semelhantes aos métodos tradicionais de interpolação e, em um dos casos, melhor.

Delimitação

O projeto visa atender os requisitos acadêmicos do Departamento de Engenharia Eletrônica e de Computação da Universidade Federal do Rio de Janeiro na disciplina Projeto Final.

O mesmo ainda atende a profissionais de diversas áreas que lidem com imagens, desde que respeitada a limitação de ser rodado sobre o MatLab, ou feita a transposição para uma outra linguagem.

Justificativa

A ampliação de imagens, também conhecida como reamostragem, é uma operação espacial que, a partir de uma imagem em escala pequena, produz uma outra em escala maior, ou seja, a área de cobertura de cada pixel na cena é aumentada de uma imagem para a outra, geralmente envolvendo

processos de interpolação.

Essa técnica é extremamente usada em tarefas de análise de imagens e visualizações científicas, entretenimento, scanners, câmeras digitais, entre outros, porque nem sempre as imagens originais possuem uma dimensão adequada ao grau de detalhamento que é buscado.

A ampliação está relacionada diretamente à criação de uma nova imagem que contém mais pixels do que a imagem original. Por exemplo, ampliar uma imagem utilizando um fator dois implica em dizer que a imagem conterá quatro vezes mais pixels, o dobro de linhas e colunas, para que a mesma área na cena possa ser coberta. Para produzir esses elementos adicionais é necessário estimá-los a partir dos dados da imagem original utilizando-se algum método de interpolação.

As técnicas convencionais de interpolação encontram dificuldades em, ao mesmo tempo, preservar detalhes e suavizar a imagem. Essas dificuldades podem gerar efeitos visuais não desejados na figura ampliada como, por exemplo, a imagem pode parecer borrada, assim como o serrilhamento das bordas, granulação ou até aparecimento de artefatos, como pontos e quadrados.

Objetivo

Inspirado na combinação desses dois problemas, a preservação de bordas sem perder a suavidade da imagem, esse trabalho propôs a criação de um algoritmo baseado em redes neurais artificiais, RNA (que por serem aproximadoras universais se tornam boas candidatas a interpoladoras), capaz de dobrar a resolução de imagens preservando suas

características, gerando assim uma imagem mais nítida e suave.

As imagens resultantes dos métodos desenvolvidos serão comparadas com as originadas pelos métodos tradicionais, onde avaliaremos o Erro Médio Quadrático e uma aproximação da Relação Sinal/Ruído entre a imagem ampliada e a original.

Metodologia

Após a finalização dos estudos em Redes Neurais e Processamento de Imagens, não encontramos trabalhos que reunissem essas duas áreas para que pudéssemos melhorá-los.

Devido a inexistência de bibliografia específica, nos aprofundamos nas documentações do próprio MatLab (Image Processing e Neural Network Toolbox).

A parir daí verificamos como trabalhar as imagens por blocos e tracei alguns relacionamentos entre a imagem original e o resultado da ampliação.

Foram testados diversos relacionamentos gráficos entre as imagens e os de melhor resultado foram:

- 1 pixel para 4 pixels (quadrado)
- Cruz de cinco pixels (pixel central pivô) para 4 pixels (quadrado)
- Quadrado com nove pixels (pixel central pivô) para 4 pixels (quadrado)

Experimentamos diversas arquiteturas de Redes Neurais, bem como Funções de Treinamento e Transferência, e finalmente optei por utilizar somente uma camada intermediária com 3 neurônios e as funções "*tansig*" e

"*purelim*" e a otimização de Levenberg-Marquardt.

Utilizamos imagens capturadas por microscópios, satélites e câmeras digitais, para validar os métodos e tornar sua aplicabilidade a mais ampla possível.

As imagens variam entre escalas de cinza e RGB, com distintos formatos de entrada e níveis de zoom, onde cada uma foi escolhida devido a uma particularidade, seja para avaliar fumaça, fog de uma cachoeira, água, pixels vizinhos desconexos, etc.

Descrição

Para descrever todo o processo de desenvolvimento e mostrar também os resultados, dividimos a estrutura do documento em diversos capítulos e apêndices.

Iniciamos introduzindo o conceito de Redes Neurais e como utiliza-las no MatLab no capítulo 2.

No capítulo 3 apresentamos os tipos de imagem para o MatLab e como o mesmo as processa.

Os métodos tradicionais de interpolação de imagens são brevemente introduzidos no capítulo 4.

No capítulo 5 descrevo em detalhes como funcionam os métodos desenvolvidos durante o projeto, para no capítulo 6 apresentar matematicamente os resultados comparativos.

Em seguida, concluo o trabalho e apresento a bibliografia utilizada, incluindo nos apêndices o código fonte e todas as imagens (originais e resultantes de cada método).

Capítulo 2

Redes Neurais:

2.1 Introdução

"Uma rede neural é um processado paralelo e maciçamente distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro em dois aspectos:

O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem.

Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido."[1]

As redes neurais por possuírem a capacidade de generalização se tornam candidatas à resolução de problemas complexos, porém, a solução em geral, vem quando integradas em uma abordagem consistente de engenharia de sistemas. Nestes casos é comum sua decomposição em subconjuntos de tarefas relativamente simples que coincidam com suas capacidades inerentes: não-linearidade, mapeamento entrada-saída, adaptabilidade, resposta a evidências, informação contextual, tolerância à falhas, sem contar a analogia neurobiológica.

[1] Haykin, Simon, Redes Neurais: princípios e práticas. 2.ed. - Porto Alegre, 2001.

Um Breve Histórico de Redes Neurais

É comum associar o surgimento da era moderna de redes neurais ao artigo de 1943 de McCulloch e Pitts, no qual descrevem um cálculo lógico de redes neurais unificando os estudos de neurofisiologia e lógica matemática, criando um modelo simples e formal de um neurônio que seguia uma lei "tudo ou nada".

O próximo passo importante foi dado em 1949 com a publicação do livro de Hebb, no qual pela primeira vez foi apresentada uma regra explícita de aprendizagem fisiológica para as modificações sinápticas.

A partir do livro de Hebb e durante toda a década de 50 e 60 o assunto foi expandido pra diversas outras áreas do conhecimento desde os primeiros modelos computacionais para testar a teoria neural, passando por estudos sobre: eficiência sináptica e a relação estatística entre os estados em ambos os lados da mesma, o que aproximou o campo das idéias por trás da teoria da informação de Shanon; teses de comportamento adaptativo; inteligência artificial e filtros adaptativos não lineares.

Em 1958, quinze anos após a publicação do clássico de McCulloch e Pitts, uma nova abordagem para o reconhecimento de padrões foi introduzida por Rosenblatt em seu trabalho sobre o perceptron, que atingiu seu ápice na demonstração do teorema da convergência do perceptron.

Até 1969 parecia que os perceptrons poderiam realizar qualquer coisa, quando surgiu o trabalho de Minsky e Papert, que demonstrou que existiam limites para o que os perceptrons de camada única podiam resolver. Os dois foram

ainda mais além e afirmaram que não havia razão pra supor que esta limitação seria superada numa versão de camadas múltiplas.

Somente nos anos 80 voltamos a encontrar pesquisas relevantes e amplamente divulgadas na área, inclusive solucionando estes problemas básicos. Umas das razões para esse intervalo de tempo seria a dificuldade de se implementar um ambiente de teste que implicava na construção dos sistemas com dispositivos analógicos e a grande incerteza instaurada por Minsky e Papert que deixou de encorajar pesquisadores e agências financiadoras.

A partir de 1982 voltamos a notar um crescimento no campo com o trabalho de Hopfield, sucedido por diversos pesquisadores, sem deixar de mencionar Ackley, Hinton e Sejnowski, que em 1985, desenvolveram a máquina de Boltzman, que foi a primeira realização bem-sucedida de uma rede neural de múltiplas camadas.

2.2 Modelo Neural

O neurônio é a unidade básica de processamento de informação de uma rede neural. A figura 1 nos ajuda a entendê-lo melhor:

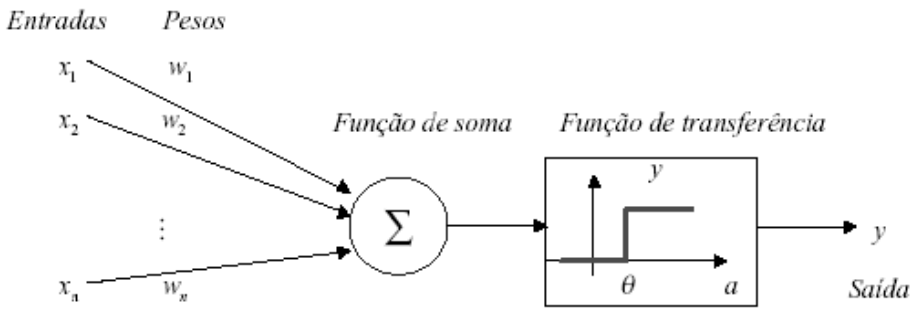

Figura 1 – Modelo Neural

Os 3 elementos básicos são:

- O conjunto de sinapses ou elos de conexão, formados pelos diversos sinais de entrada e os pesos sinápticos que efetuarão a multiplicação.
- Um somador para os sinais de entrada (e possivelmente uma polarização ou "bias") que gera o campo local induzido, ou em algumas representações mais simples, a saída do combinador linear.

$$\mathbf{V = \sum XiWi + b}$$

- Uma função de ativação que é responsável por introduzir uma não linearidade e geralmente restringir a saída de um neurônio.

Além deles podemos ter ainda um "bias" aplicado no somador, que tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação, ou melhor, modificar a relação entre o campo local induzido ou potencial de ativação e a saída do combinador linear.

Podemos substituir o valor do "bias" por mais um elo de conexão com uma entrada fixa em 1 e peso sináptico de igual valor.

As funções de ativação ou transferências mais utilizadas são de três tipos: função de limiar, linear ou sigmóide.

Um neurônio passa adiante um estímulo conforme a força dos estímulos recebidos provenientes dos neurônios que com ele, estão conectados.

No mundo artificial, o mesmo efeito é simulado; assim

a soma dos impulsos recebidos e com quais neurônios um neurônio estabelece ligações são os fatores que determinam a propagação de um estimulo, como mostra a figura 2.

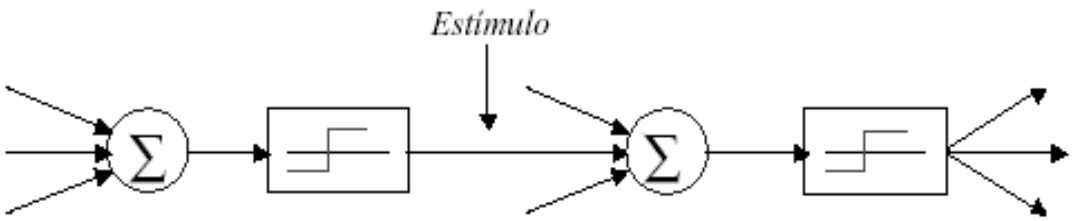

Figura 2 – Funcionamento de uma Rede

O tipo de conexão, número de camadas de neurônios e o tipo de treinamento, são os aspectos que diferem os tipos de redes neurais existentes, como descrito na figura 3.

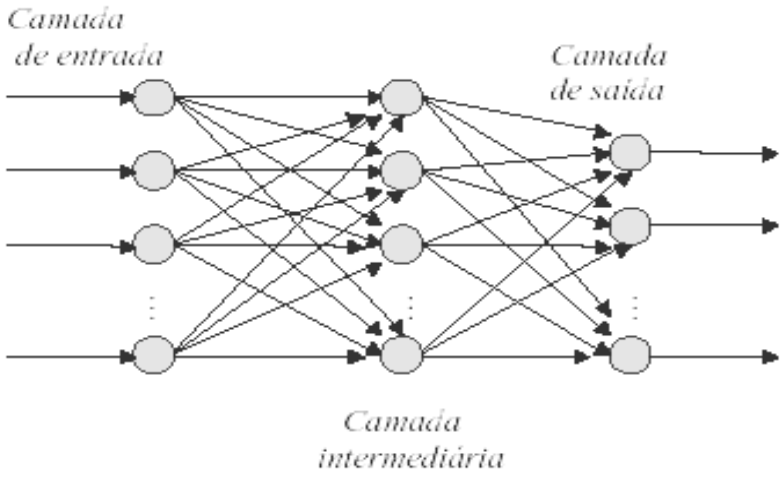

Figura 3 – Múltiplas Camadas – Versão Simples

2.3 Redes Neurais no MATLAB

As redes neurais são tratadas pelo MatLab como um objeto específico, possuindo vários parâmetros que variam de acordo com o tipo de rede utilizada assim como a sua arquitetura, porém alguns parâmetros são comuns para todos os tipos de rede: matriz de entrada, matriz de saída, matriz de pesos, matriz de indução "bias", função de

transferência ou ativação, número de camadas, número de épocas e o erro mínimo.

Podemos observar na figura 4 uma representação completa de uma arquitetura para facilitar seu entendimento:

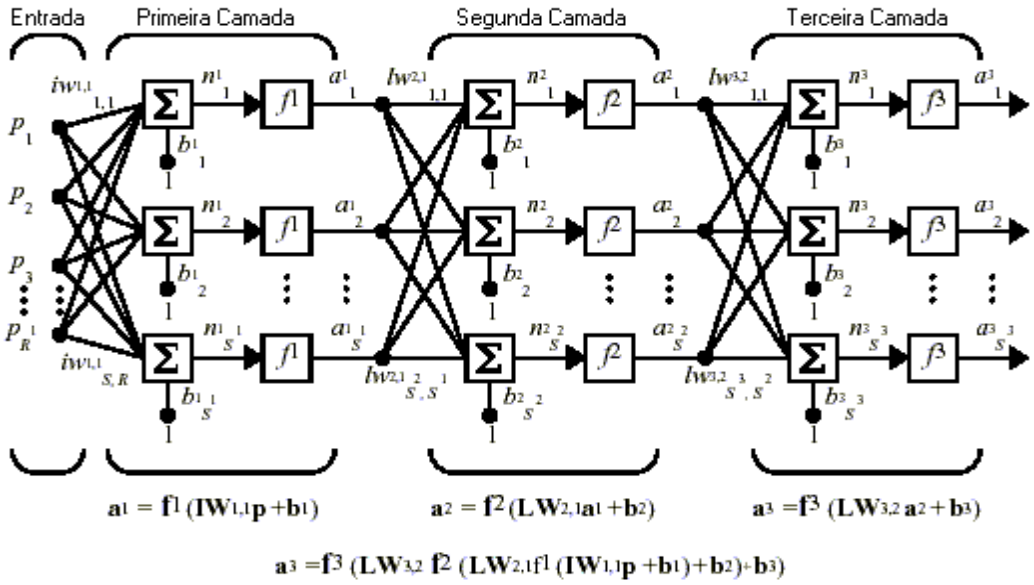

Figura 4 – Múltiplas Camadas – Versão Completa

Na figura 5 vemos sua representação resumida, onde temos os diversos sinais representados por seus vetores e matrizes de entrada e saída:

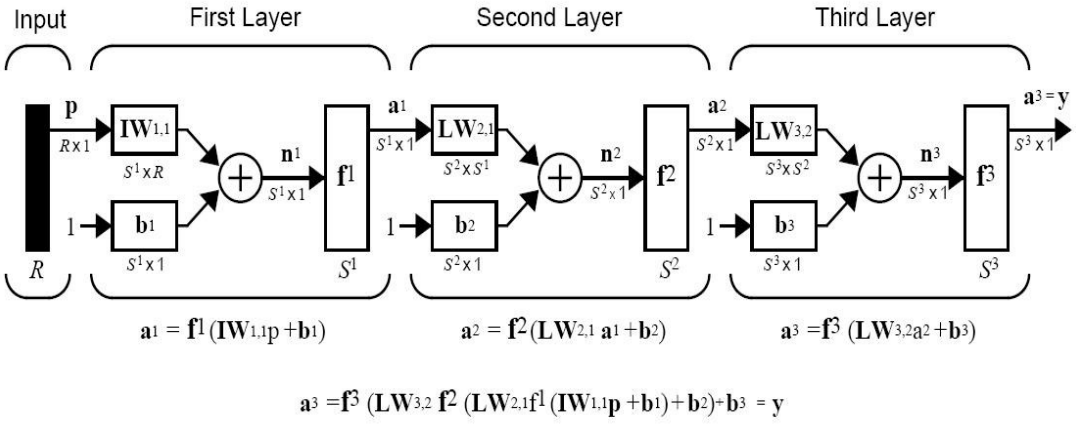

Figura 5 – Múltiplas Camadas – Versão Resumida

A matriz de entrada determina quantos neurônios a

camada de entrada terá. Nessa matriz cada linha representa um valor de entrada na rede, e cada coluna um conjunto de entradas. Por exemplo, se for definido uma matriz de entrada 3 x 6, então essa rede terá 3 neurônios na camada de entrada e 6 conjuntos de entradas.

A matriz de saída é definida pelo número de neurônios na camada de saída de uma rede, ou seja, se uma rede tiver um neurônio na camada de saída, essa matriz terá apenas uma linha. Assim como a matriz de entrada, na matriz de saída cada linha representa um valor de saída da rede e cada coluna um conjunto de saídas. Por exemplo, se uma rede gerar uma matriz de saída 3 x 6 é porque a rede possui 3 neurônios na última camada e 6 conjuntos de valores de saída.

A rede neural precisa corresponder os valores de entrada e de saída para conseguir achar a relação correta entre esse valores, e o MatLab faz isso pela coluna da matriz de entrada e de saída.

As funções de ativação determinam qual tipo função será usada para disparar os neurônios de uma camada para outra, uma rede pode ter várias funções de transferências tudo depende do número de camadas que uma rede possui. O MATLAB possui uma grande variedade de funções de transferências, mas também é possível determinar uma função de transferência própria.

A figura 6 nos mostra essas funções de ativação.

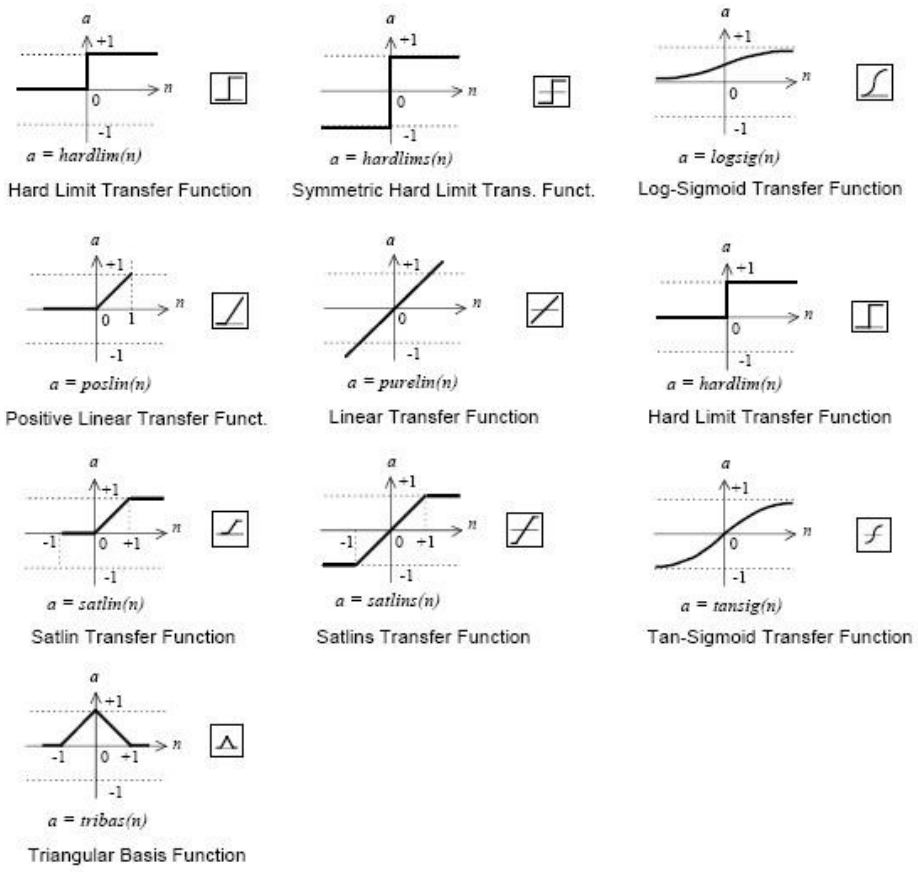

Figura 6 – Funções de Ativação

As funções mais utilizadas já foram descritas anteriormente, mas voltamos a elas aqui para melhor detalhar o seu uso específico. Em geral, a função hardlim (Hard Limit) ou de limiar é muito utilizada em sistemas de classificação, a função purelim (Linear) ou linear por partes é mais usada para filtros adaptativos, enquanto a função logsig (Log-Sigmoid) ou sigmóide é mais comumente usada em redes de retro propagação por sua propriedade de ser diferenciável.

Uma rede pode ter várias matrizes de pesos e de indução (bias) tudo depende do número de camadas que a rede possui. Essas matrizes possuem os pesos das conexões entres os neurônios de duas camadas adjacentes, e o tamanho dessas matrizes varia com o número de neurônios de cada camada.

O MATLAB atribui valores quaisquer para essas matrizes quando a rede é criada, e durante o treinamento essas matrizes são ajustadas visando a melhor aproximação entre os conjuntos de entrada e saída, ou seja, o ajuste que possui o menor erro.

O número de épocas e o erro mínimo são parâmetros utilizados somente durante o treinamento da rede e determinam o número de vezes que o conjunto de treinamento deve passar pela rede e o erro mínimo desejado entre a entrada e saída, respectivamente.

Se o número de épocas for ultrapassado ou o erro mínimo for superado, o treinamento é finalizado.

Nesse trabalho utilizamos redes do tipo "back propagation", que consiste em redes de múltiplas camadas com um algoritmo de treinamento baseado na retro propagação. O MATLAB possui várias variações de algoritmos de "back propagation", sendo que nesse trabalho foram usadas somente duas delas: "Resilent Backpropagation", "Levenberg-Marquardt".

O algoritmo "Levenberg-Marquardt" tem a vantagem de ter um menor tempo de treinamento, porém necessita de muita memória, tornando inviável o treinamento de um conjunto muito grande.

O algoritmo "Resilent Backpropagation" possui um treinamento bem mais lento, porém utiliza bem menos memória, sendo ideal para conjuntos de treinamentos muito grandes.

Como exemplo, a tabela 1 faz uma comparação entre esse dois algoritmos para um mesmo caso, usando como base o tempo e o número médio de épocas para a conversão ao erro mínimo da rede neural.

Tabela 1 – Comparação entre os algoritmos de treinamento

Algoritmo	**Treinamento (min.)**	**Quantidade de épocas**
Resilent	12,95	185
Levenberg-Marquardt	1,87	6

Para se criar uma rede neural no MATLAB é necessário informar alguns parâmetros: número de camadas escondidas, numero de neurônios na camada escondida e de saída, numero de entradas, funções de transferência e o algoritmo de treinamento a ser usado.

Os demais valores, como matriz de pesos e influencia, números de épocas são definidos automaticamente com valores padrões, podendo ser alterado se assim for necessário.

O comando abaixo mostra como criar uma rede com 3 entradas, uma saída, uma camada escondida e usar o algoritmo resilent backpropagation para treinar a rede.

```
net = newff([0 1;0 1;0 1],[3,1],
{'tansig','purelin'},'trainrp');
```

A figura 7 mostra o que cada parâmetro significa para a construção de uma rede neural artificial no MATLAB.

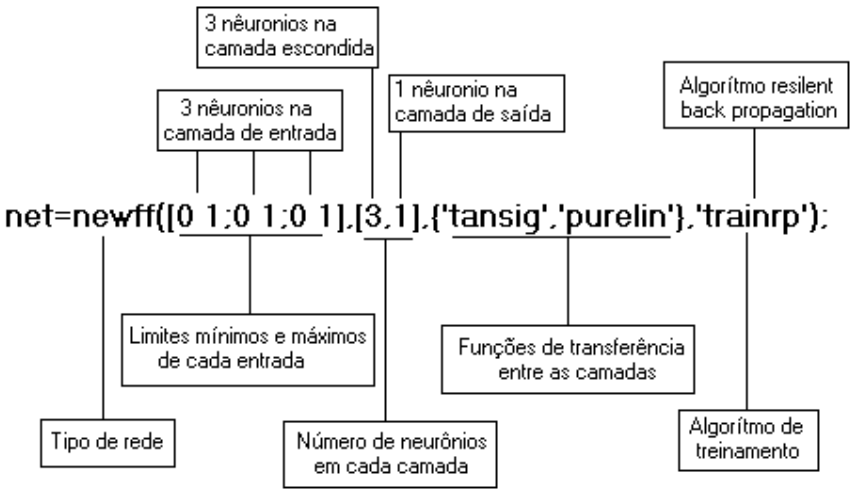

Figura 7 – Criando uma Rede Neural no MatLab

Para treinar uma rede é necessário possuirmos um conjunto de treinamento, que no MATLAB quer dizer uma matriz de entrada e uma matriz de saída com um número suficiente de valores para treinar apropriadamente a rede.

A quantidade de colunas entre a matriz de entrada e de saída tem que ser iguais, caso contrário o MATLAB não irá treinar a rede.

O treinamento da rede anterior é feito pelo seguinte comando:

```
net = train(net,Minput,Moutput);
```

A figura 8 descreve cada parâmetro para o treinamento de uma rede neural no MATLAB.

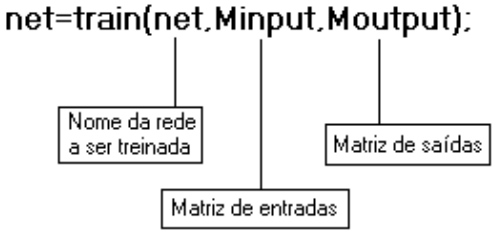

Figura 8 – Treinando uma Rede Neural no MatLab

Após a execução desse comando a rede estará treinada e pronta para ser utilizada.

2.4 Comandos básicos do Neural Network Toolbox

net = **newff**(PR,[S1 S2...SNl],{TF1 TF2...TFNl},BTF,BLF,PF)
→ Cria a nova rede neural feed-forward backpropagation, seguindo os parâmetros acima.

No trabalho usamos quase sempre a seguinte sintaxe:

net=newff(minmax(Me),[3,4],
{'tansig','purelin'},'trainbr');

Onde:

PR = minmax(Me) → Mínimos e máximos extraídos diretamente da Matriz de Entrada.
Si = [3,4] → Redes com 3 camadas (incluindo-se a de entrada), onde a camada intermediária possui 3 neurônios e a de saída 4.
TFi = {'tansig','purelin'} → Funções de transferência distintas por camada.
BTF = trainbr → Função de treinamento baseada na otimização de Levenberg-Marquardt.

[net,tr,Y,E,Pf,Af] = **train**(net,P,T,Pi,Ai,VV,TV)
→ Treina a rede neural criada previamente de acordo com os parâmetros acima.

No trabalho usamos quase sempre a seguinte sintaxe:

net=train(net,Me,Ms);

Onde:

net = net → Rede Neural
P = Me → Matriz de Entrada
T = Ms → Matriz de Saída, que representa o alvo da rede.

Ainda definimos os seguintes parâmetros de treinamento:

netB.trainParam.epochs=12;
netB.trainParam.show=3;
netB.trainParam.goal=1e-4;

[Y,Pf,Af,E,perf] = **sim**(net,P,Pi,Ai,T)
→ Simula a rede neural já treinada e gera a saída desejada, de acordo com os parâmetros acima.

No trabalho usamos quase sempre a seguinte sintaxe:

Mout=sim(net,Me);

Onde:

Y = Mout → A Matriz de Saída produzida pela rede, que no nosso caso ainda sofrerá algumas manipulações algébricas para a geração do "Array" que representa a imagem ampliada.
net = net → A rede propriamente dita, já treinada.
P = Me → Matriz de Entrada da Rede.

Capítulo 3

Imagens:

3.1 Processamento de Imagens no MatLab

Quando estivermos trabalhando com imagens no MatLab devemos lembrar de vários aspectos, tais como: utilizar o formato correto para abertura, gravar a imagem em diferentes tipos, como visualizar a imagem e suas propriedades, como converter entre os tipos suportados; além é claro dos comandos mais avançados definidos pelo "toolbox" que nos possibilitam trabalhar com a imagem.

O MATLAB trata imagens como uma matriz que varia de acordo com o formato da imagem, mas em geral cada elemento da matriz representa o valor de um pixel. O "software" aceita 4 tipos de formatos de imagens, cada um com suas características.

3.2 Tipos de Imagem

3.2.1 Imagens Binárias: (Binary)

Cada pixel é representado de forma binária (0 off - preto e 1 on - branco), formando uma matriz de zeros e uns, como mostra a figura 9.

Figura 9 – Imagem Binária

3.2.2 Imagens em Tons de Cinza: (Intensity ou Gray scale)

Uma imagem em tons de cinza é representada por uma matriz onde cada elemento pode assumir um valor dentro de um intervalo, definido a partir da classe da mesma: "double", "unit8" ou "unit16".

Cada elemento na matriz representa um valor de intensidade ou tom de cinza, onde 0 em geral representa o preto e 1, 255 ou 65535 de acordo com as classes definidas anteriormente. Isso pode ser percebido na figura 10.

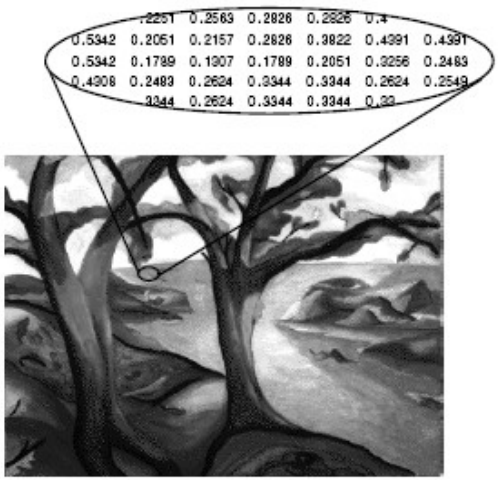

Figura 10 – Imagem em Tons de Cinza

3.2.3 Imagens RGB

As imagens RGB, também denominadas de "truecolor", pela capacidade de reprodução de milhões de cores, são armazenadas no MatLab por três matrizes de mesma dimensão, cada uma representando uma das cores que compõem cada pixel (R – "red", vermelho; G – "green", verde; B – "blue", azul).

Assim como a imagens em tons de cinza, as RGB podem ser da classe "double", unit8 ou unit16; Sendo assim, um

pixel definido pelo conjunto (0,0,0) representa preto enquanto outro (1,1,1) é mostrado como branco, assim como na figura 11.

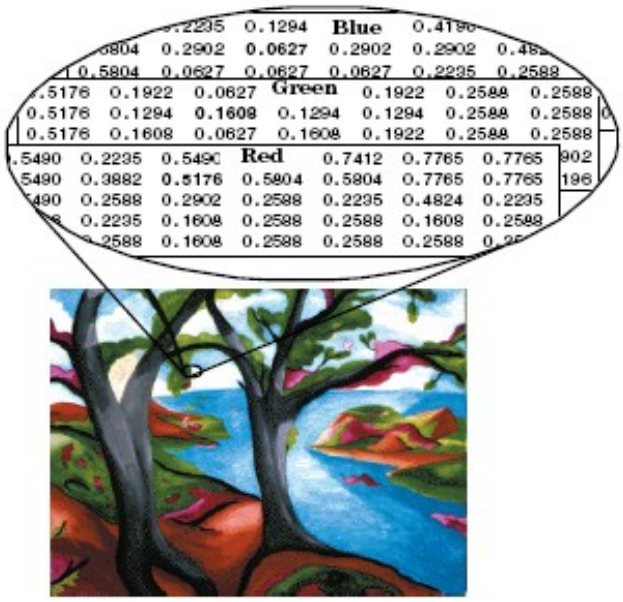

Figura 11 – Imagem RGB

3.2.4 Imagens Indexadas

Nesse formato, as imagens são representadas por duas matrizes, uma de tamanho igual a imagem n x n e outra p x 3.

Esta segunda matriz é chamada mapa de cor ou colormap, onde cada linha representa uma intensidade de cor que está sendo usada na imagem, com cada uma das três colunas representando um dos componentes RGB.

Na primeira matriz, cada elemento é um índice da linha a ser utilizada na matriz de cores, conforme mostrado na figura 12:

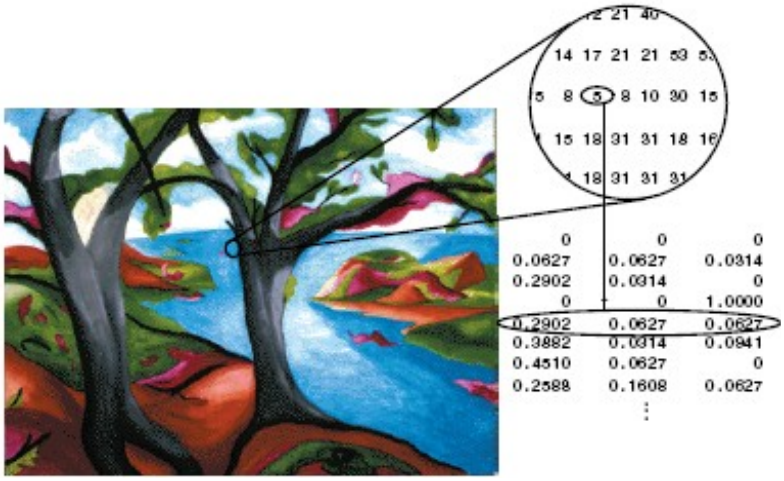

Figura 12 – Imagem Indexada

3.3 Comandos básicos do Image Processing Toolbox

X=**imread**('caminho\arquivo','formato');

→ Coloca no Array X a imagem presente no arquivo descrito.

imshow(X)

→ Exibe em uma nova janela a imagem previamente armazenada no Array X

imwrite(X,'caminho\arquivo','formato');

→ Cria o arquivo de imagem descrito com o conteúdo do Array X.

Y=**imresize**(X, scale); ou

Y=**imresize**(X, [mrows ncols]);

→ Cria um novo Array Y com o conteúdo de X com o devido redimensionamento.

Y=**blkproc**(X, [m n], função(f)); ou

Y=**blkproc**(X, [m n], [mborder nborder], função(f));

→ Processa o "Array" X em blocos m,n completando com 0's quando necessário, aplicando a função(f). mborder e nborder determinam quanto haverá de sobreposição entre os distintos blocos.

Z=**imadd**(X,Y)

→ Soma duas ou mais "Arrays" de imagem. A soma algébrica de X + Y pode não ser possível devido a quantidade de bits utilizada para representar o mapa de cores das mesmas.

whos

→ Lista as variáveis da área de trabalho; para o nosso caso, detalha o número de linhas e colunas do "Array", o total de bytes e a profundidade (quantidade de bits).

[Y,map]=**gray2ind**(X,n);

→ Converte uma imagem em escala de cinza para indexada.

Y=**ind2gray**(X,map);

→ Converte uma imagem indexada em escala de cinza.

[Y,map]=**rgb2ind**(X,n);

→ Converte uma imagem em RGB para indexada.

Y=**ind2rgb**(X,map);

→ Converte uma imagem indexada em RGB.

Y=**rgb2gray**(X);

→ Converte uma imagem RGB em escala de cinza.

Y=**gray2rgb**(X);

→ Converte uma imagem em escala de cinza para RGB.

[x y z]=**size**(X);

→ x e y receberam as dimensões da imagem, podendo ser 512 e 512; enquanto z receberá o tipo da mesma indicando uma imagem binária ou em escala de cinza quando for 1 e RGB quando for 3, por exemplo.

isind(X)

→ Geralmente usado em testes de validade (if), já que seu código de retorno é 1 se a imagem for indexada e 0 caso contrário.

Y=**im2double**(X)

→ Pega a imagem X de entrada e retorna uma imagem de classe double, fazendo as devidas correções se necessário.

B=**uint8**(A)

→ Converte o vetor A de qualquer classe em inteiro sem sinal; e no caso do uint8, com intervalo de 0 a 255

```
>> A=[356.000 -3.123 4.125];
>> B=uint8(A)
B =
   255     0     4
```

D=**cat**(dim,A,B,C);

→ Concatena A, B e C de acordo com a variável dim, como abaixo:

```
>> A=[1 2 3; 1 2 3];
>> B=[3 2 1; 3 2 1];
>> C=[4 5 6; 4 5 6];
>> D=cat(1,A,B,C)
D =
     1     2     3
     1     2     3
     3     2     1
     3     2     1
     4     5     6
     4     5     6

>> E=cat(2,A,B,C)
E =
     1     2     3     3     2     1     4     5     6
     1     2     3     3     2     1     4     5     6

>> F=cat(3,A,B,C)

F(:,:,1) =          F(:,:,2) =          F(:,:,3) =
     1     2     3
     1     2     3        3     2     1        4     5     6
                            3     2     1        4     5     6
```

Capítulo 4

Métodos Tradicionais de Interpolação:

Abordaremos a seguir alguns métodos já conhecidos de interpolação de imagens, e por esse motivo serão chamados de tradicionais.

4.1 Bilinear

É um método de interpolação que usa os 4 pixels vizinhos para obter o valor do novo pixel, estes pixels são os de cima, de baixo, da esquerda e o da direita. Esse valor é calculado usando uma média ponderada baseada na distância dos pixels.

Essa técnica tende a gerar uma imagem borrada, isso devido à redução do contraste causado pelo cálculo da média ponderada dos pixels vizinhos, porém o serrilhado das bordas é reduzido e a imagem parece mais uniforme.

4.2 Bicubic

A interpolação "Bicubic", assim como o método anterior, determina o valor dos novos pixels pelo cálculo da média ponderada baseado na distância dos pixels mais próximos, só que com uma matriz 4x4 de pixels vizinhos.

Assim como o método "Bilinear", esse método gera uma imagem muito mais uniforme que o método "Nearest neighbor", e também borra a imagem gerada. Para corrigir esse efeito é comum aplicar alguma técnica que corrija o contraste entre os pixels, mais conhecida como "sharpening". Em alguns algoritmos desse método é incluído um parâmetro extra que é

usado para fazer essa regulagem.

Esse método é considerado o método padrão para ampliação de imagens.

4.3 B-spline

Essa técnica, assim como a Bicubic, também utiliza um conjunto de 16-pixels, uma matriz 4x4 de pixels vizinhos, mas essa técnica utiliza aproximações polinomiais para determinar o valor do novo pixel. Cada ponto da imagem é colocado em uma curva "spline" onde os coeficientes dessa curva são determinados de uma forma a garantir uma melhor suavização.

Esse método tende a preservar mais os detalhes da imagem, além de controlar o "borramento", principalmente por não aplicar uma média entre os pixels vizinhos.

4.4 Lanczos

Essa é uma técnica um pouco mais sofisticada que as demais, pois para obtenção do valor do novo pixel ela utiliza uma variação do método de interpolação criado por Cornelius Lanczos (1893-1974).

Essa técnica utiliza matrizes de 4x4, 6x6 ou 8x8 de pixels vizinhos.

Capítulo 5

Método Aplicado:

5.1 Introdução

Nesse trabalho foram testadas várias formas de construir entradas e saídas, bem como arquiteturas de redes, todas visando melhorar a qualidade da imagem final obtida por cada método.

Alguns desses métodos não obtiveram resultados satisfatórios e não foram incluídos no grupo de comparação.

Os três métodos que serão discutidos nesse trabalho possuem a mesma sistemática para formar o conjunto de treinamento das redes, e que consiste em tendo uma imagem Io qualquer, também chamada de imagem original, utiliza-se uma função em MatLab (média dos 4 pixels vizinhos) para diminuir a sua resolução pela metade gerando uma nova imagem Ir. Essa nova imagem é chamada de imagem de baixa resolução, como mostra a figura 13:

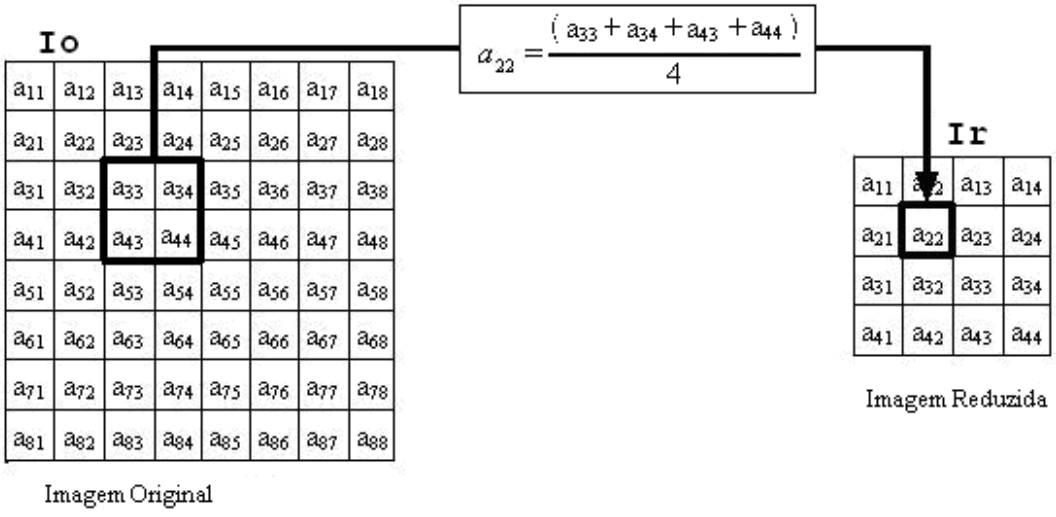

Figura 13 - Método de Redução – Fator 2

Após a obtenção da imagem Ir, é necessário convertê-las (Ir e Io) para matrizes de entrada e saída de rede, respectivamente, isso devido ao MATLAB processar uma matriz de entrada coluna por coluna, e uma matriz de saída linha por linha.

Nesse processo de conversão não há perda de pixels, e cada um dos métodos realizados nesse trabalho possui uma maneira diferente para converter essas imagens em matrizes de entrada.

As matrizes de saída, como será visto posteriormente, possuem a mesma forma independentemente do método, e todas foram geradas por uma única função.

A figura 14 mostra a conversão das imagens em matrizes de entrada e saída, e depois essas matrizes são associadas a uma rede neural e só posteriormente essa rede é treinada.

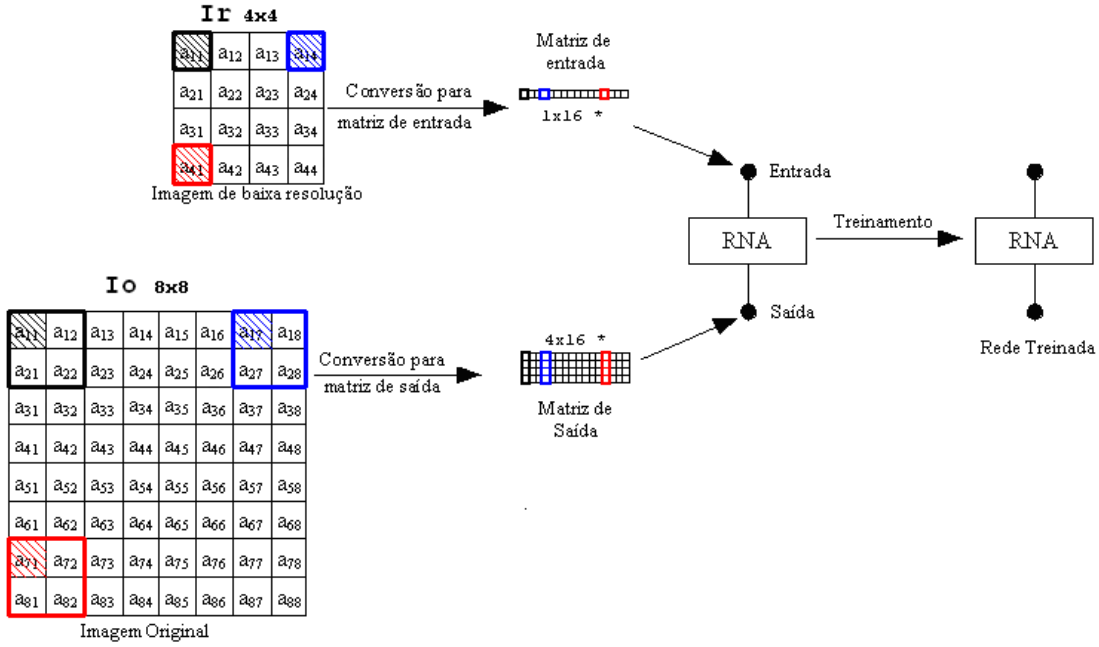

Figura 14 – Conversão de imagens em matrizes de entrada e saída

Quando o processo de treinamento chegar ao fim, a rede

neural será capaz de duplicar a resolução da imagem em questão, ou seja, converte-se a original Io para uma imagem ampliada Ia, com o dobro da resolução de Io, e esta é chamada de imagem de alta resolução.

O processo de ampliação da imagem também envolve conversão entre matrizes. Primeiramente esse processo irá converter a imagem original em uma matriz de entrada, após isso essa matriz será colocada nas entradas da rede já treinada. Essa rede irá gerar uma matriz de saída e só então que essa matriz será convertida em uma nova imagem, que terá o dobro da resolução da imagem original. A figura 15 mostra de forma clara esse processo.

É desejável que imagem de alta resolução Ia, seja nítida e de ótima qualidade. A figura 15 ilustra o processo.

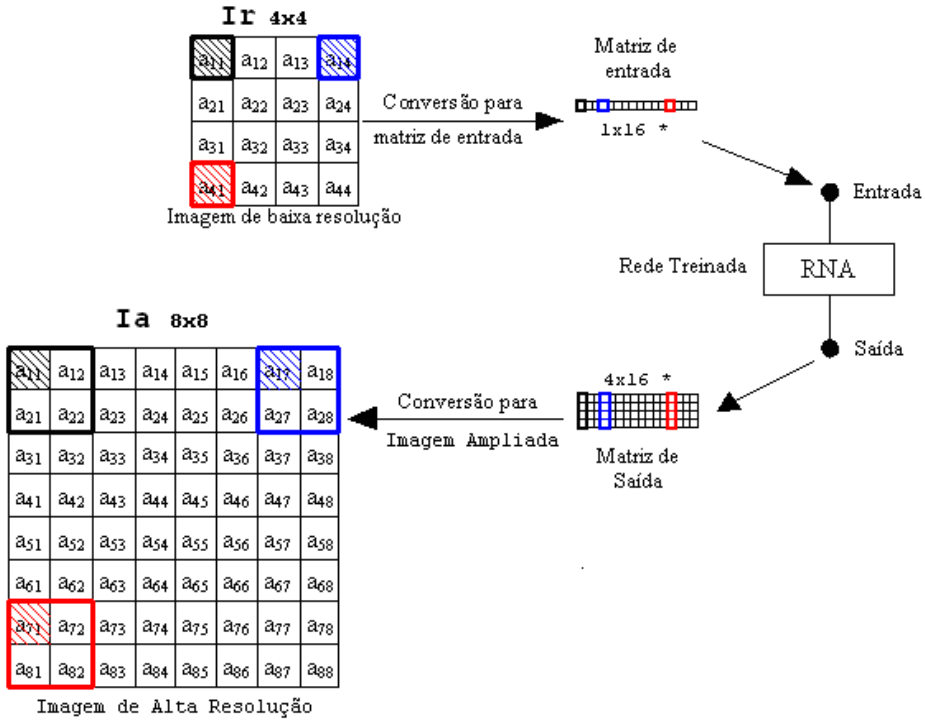

Figura 15 – Obtenção de Imagem de Alta Resolução

A diferença entre cada uns dos métodos está na forma em que os pixels da imagem de baixa resolução Ir, são relacionados com os pixels da imagem original Io.

No decorrer do desenvolvimento desse projeto observou-se que as imagens coloridas, RGB, apresentavam melhor desempenho quando se usavam 3 redes neurais, uma pra cada cor, ou seja, uma rede para os tons de vermelho, outra para os tons de verde e mais uma para os tons de azul, ao invés de usar uma única rede neural para todas as cores.

Esse comportamento também é encontrado quando tentamos usar a função imresize, que realiza a interpolação de cada plano individualmente para as imagens RGB.

Também observamos um melhor desempenho das redes quando se utilizava 3 neurônios na camada escondida delas, como mostra a figura 16.

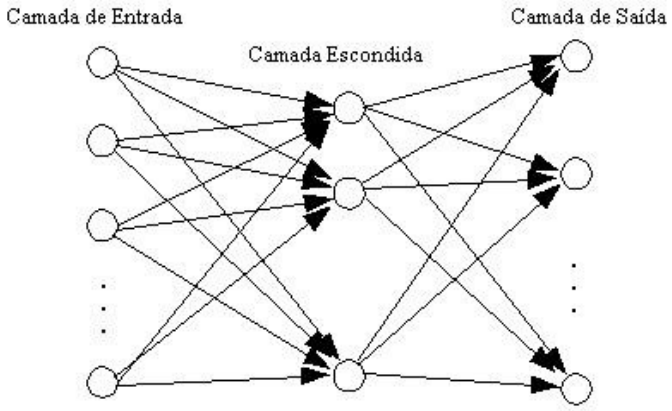

Figura 16 – Arquitetura das Redes Neurais utilizadas

Imagens em tons de cinza apresentaram ótimos resultados usando apenas uma única rede neural e com 3 neurônios na camada escondida.

5.2 Explicação

Como o desejo do projeto é encontrar um método capaz de obter resultados melhores ou ao menos semelhantes aos métodos tradicionais de interpolação, não faremos como na figura 15.

Ao invés de utilizarmos a imagem original Io para entrada da rede, utilizaremos a imagem de baixa resolução Ir, que será duplicada pelo método, gerando a imagem ampliada Ia, que nesse caso terá as mesmas dimensões da imagem original.

Fizemos isso para termos de maneira simples e matemática, uma base de comparação; neste caso a própria imagem original.

A imagem original foi reduzida no MatLab, na razão 2 para 1, para gerar a imagem de baixa resolução.

Para os métodos tradicionais de interpolação, utilizamos o "software" PhotoBrush, onde a imagem reduzida foi novamente ampliada.

A saída, a imagem ampliada oriunda dos métodos tradicionais (PhotoBrush) ou das Redes Neurais (MatLab), foi então comparada, matematicamente, com a imagem original segundo o Erro Médio Quadrático no MatLab.

Incluímos ainda a Relação Sinal Ruído, que possui relação estreita com o EMQ, somente para ilustrar.

Com base nesses valores para diversas fotografias utilizadas pudemos avaliar cada um dos métodos

desenvolvidos.

Os resultados estão descritos nas tabelas de cada fotografia do capítulo 6 Estudo de Casos.

Os métodos desenvolvidos para o projeto serão explicados a seguir.

Em todos os 3 métodos apresentados, após as conversões necessárias, treinamos a rede com uma imagem, até atingirmos um número determinado de épocas ou um nível de erro aceitável.

Após o treinamento, essa rede é utilizada para todas as outras imagens apresentadas aqui no trabalho; ficando ainda disponível para qualquer outra imagem.

A imagem utilizada para o treinamento e as utilizadas para as simulações, bem como as resultantes de cada método encontram-se no apêndice.

5.3 Descrição dos Métodos Implementados

5.3.1 Método V1

Esse método se baseia na idéia de um relacionamento direto entre os pixels das imagens Io e Ir, ou seja, não utiliza um grupo de pixels vizinhos da imagem de baixa resolução Ir para fazer a relação com a imagem original Io, diferentemente dos demais métodos realizados nesse trabalho.

Ou seja, usa um único pixel da imagem Ir e o relaciona com um grupo de 4 pixels da imagem original Io, uma matriz 2 x 2, como descrito na figura 17.

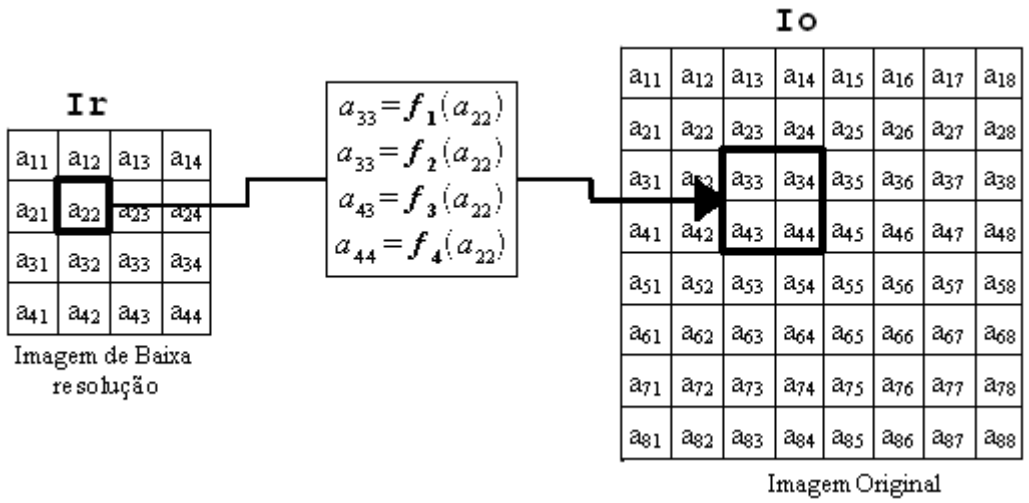

Figura 17 – Método V1

Como nesse método 1 pixel de Ir equivale a 4 pixels de Io, então a arquitetura da rede neural desse método será de um neurônio na camada de entrada, 3 na camada escondida e 4 na de saída, como observado na figura 18.

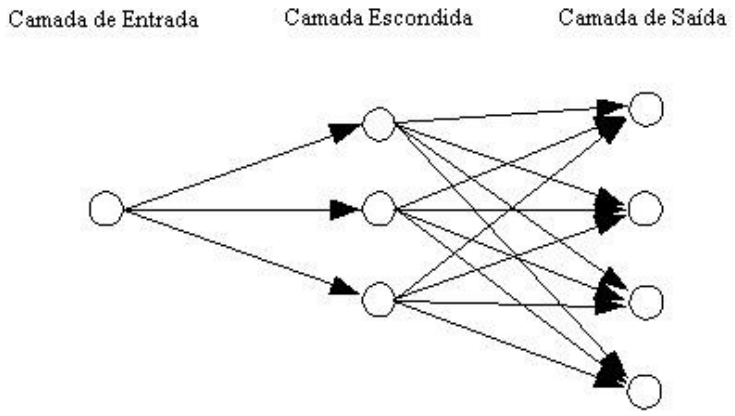

Figura 18 – Arquitetura da RNA para o Método V1

A conversão das imagens em matrizes de entrada e saída assim como o treinamento da rede neural desse método é a mais rápida entre os demais, visto que esse possui menos pontos nas matrizes de entrada.

A regra de conversão das imagens é bem simples, como na equação abaixo:

$$\mathbf{Io}\begin{vmatrix} a_{i,j} & a_{i,j+1} \\ a_{i+1,j} & a_{i+1,j+1} \end{vmatrix} = f\left(\mathbf{Ir}\left[a_{\frac{(i-1)}{2}+1,\frac{(j-1)}{2}+1}\right]\right)$$

Onde Io[($a_{i,j}$),($a_{i,j+1}$),...] é o grupo de 4 pixels vizinhos de Io e Ir($a_{i,j}$) é o pixel correspondente, sendo que i e j são sempre ímpares, variando de 1 a n-1, onde n é o número de colunas (ou linhas, por lidarmos sempre com matrizes quadradas).

Esta função, assim como para os métodos seguintes, é uma função não linear aprendida e implementada pela Rede Neural.

A conversão é feita linha por linha da imagem.

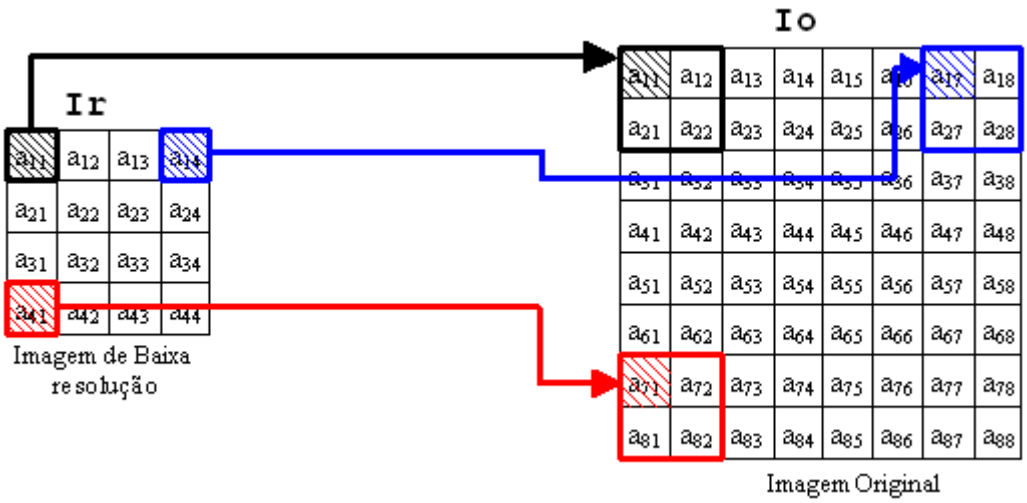

Figura 19 – Relação entre os pixels de Ir e Io – V1

Os pixels pivôs estão marcados na figura 19 e observa-se que somente esses são usados para fazer a relação com os pixels de Io.

Por exemplo, o pixel pivô marcado de preto na imagem de baixa resolução Ir, $[a_{11}]$ está relacionado com o grupo de 4 pixels que estão envolvidos de preto na imagem original Io, que são $[a_{11}, a_{12}, a_{21}, a_{22}]$, esse processamento continua linha por linha até o último pixel da imagem de baixa resolução Ir.

Na tabela 2 temos a forma da matriz de entrada após a conversão da imagem de baixa resolução.

Essa será uma matriz 1 x m, onde m é o numero total de pixels da imagem de baixa resolução.

Tabela 2 – Matriz de Entrada – V1

a_{11}	a_{12}	a_{13}	a_{14}	a_{21}	a_{22}	a_{23}	a_{24}	a_{31}	a_{32}	a_{33}	a_{34}	a_{41}	a_{42}	a_{43}	a_{44}

A tabela 3 mostra a matriz de saída após a conversão da imagem original, essa será uma matriz 4 x n, onde n é um quarto do número total de pixels da figura original, ou

seja, o mesmo número de pixels da imagem reduzida, n=m.

Com isso as duas matrizes terão o mesmo número de colunas, satisfazendo a condição de treinamento de redes do MATLAB.

Tabela 3 – Matriz de Saída – V1

a_{11}	a_{13}	a_{15}	a_{17}	a_{31}	a_{33}	a_{35}	a_{37}	a_{51}	a_{53}	a_{55}	a_{57}	a_{71}	a_{73}	a_{75}	a_{77}
a_{12}	a_{14}	a_{16}	a_{18}	a_{32}	a_{34}	a_{36}	a_{38}	a_{52}	a_{54}	a_{56}	a_{58}	a_{72}	a_{74}	a_{76}	a_{78}
a_{21}	a_{23}	a_{25}	a_{27}	a_{41}	a_{43}	a_{45}	a_{47}	a_{61}	a_{63}	a_{65}	a_{67}	a_{81}	a_{83}	a_{85}	a_{87}
a_{22}	a_{24}	a_{26}	a_{28}	a_{42}	a_{44}	a_{46}	a_{48}	a_{62}	a_{64}	a_{66}	a_{68}	a_{82}	a_{84}	a_{86}	a_{88}

A imagem de alta resolução Ia gerada por essa rede possui uma boa qualidade, porém a imagem tende ficar um pouco serrilhada nas bordas.

5.3.2 Método V2

Esse método pode-se dizer que é uma extensão do método anterior que usa um relacionamento direto, porém esse tem como principal característica usar um pixel central, o pixel que era usado no relacionamento direto, e os pixels vizinhos que o circundam, fazendo, com isso, uma analogia aos métodos de interpolação mais tradicionais.

Esse método utiliza um grupo de 9 pixels vizinhos, uma matriz 3 x 3, da imagem de baixa resolução Ir, para fazer a correspondência com um grupo de 4 pixels, uma matriz 2 x 2, da imagem original Io, então cada conjunto de 4 pixels da imagem Io é relacionado com um conjunto de 9 pixels da imagem Ir. O método é exemplificado na figura 20.

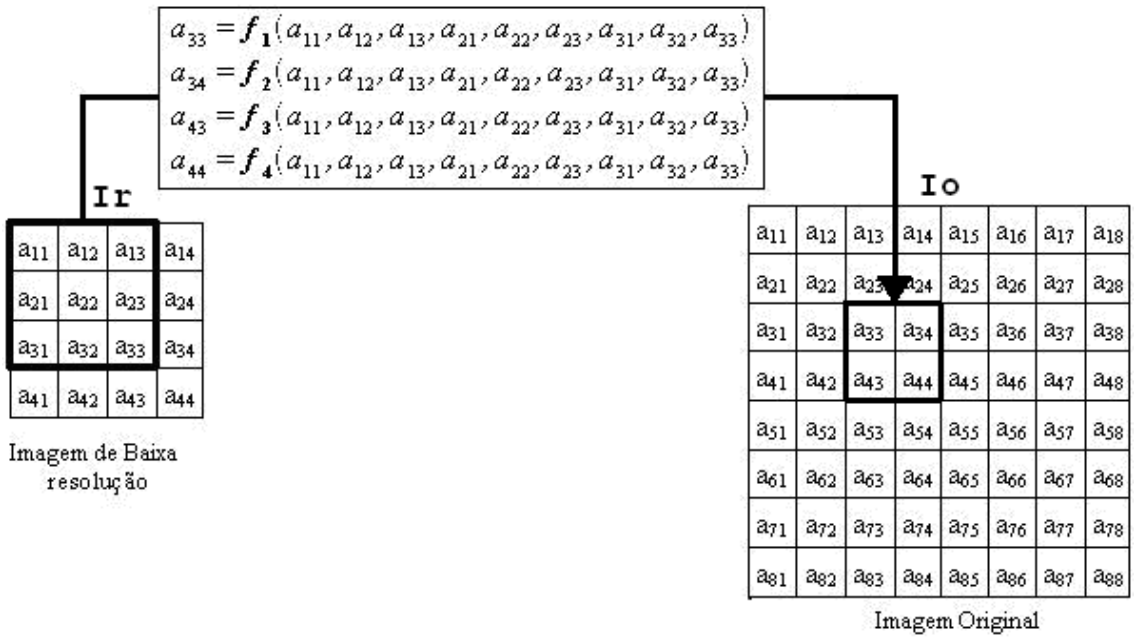

Figura 20 – Método V2

Visto que esse método utiliza-se de um conjunto de 9 pixels de Ir e os relacionam com 4 pixels de Io, a arquitetura da rede neural desse método terá 9 neurônios na camada de entrada, 3 na camada escondida e 4 na de saída,

como mostrado na figura 21:

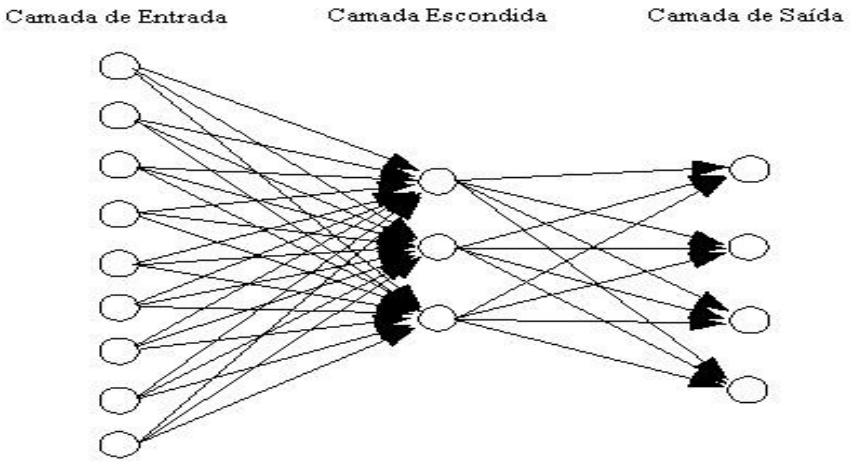

Figura 21 - Arquitetura da RNA para o Método V2

Os pontos que passam os limites da imagem I2 também são substituídos por zero, a velocidade de conversão e treinamento desse método é lenta, visto que esse apresenta um maior número de pontos na matriz de entrada. Esse método segue a regra mostrada abaixo para a conversão das imagens em matrizes.

$$\mathbf{Io}\begin{vmatrix} a_{i,j} & a_{i,j+1} \\ a_{i+1,j} & a_{i+1,j+1} \end{vmatrix} = f(\mathbf{Ir}\begin{bmatrix} a_{\frac{(i-1)}{2},\frac{(j-1)}{2}}, a_{\frac{(i-1)}{2},\frac{(j-1)}{2}+1}, a_{\frac{(i-1)}{2},\frac{(j-1)}{2}+2}, a_{\frac{(i-1)}{2}+1,\frac{(j-1)}{2}}, a_{\frac{(i-1)}{2}+1,\frac{(j-1)}{2}+1}, a_{\frac{(i-1)}{2}+1,\frac{(j-1)}{2}+2}, \\ a_{\frac{(i-1)}{2}+2,\frac{(j-1)}{2}}, a_{\frac{(i-1)}{2}+2,\frac{(j-1)}{2}+1}, a_{\frac{(i-1)}{2}+2,\frac{(j-1)}{2}+2} \end{bmatrix})$$

A figura 22 mostra que os pixels pivôs da imagem Ir são os pixels centrais da matriz 3 x 3 de pixels vizinhos, onde o pivô grifado de preto, o pixel [a_{11}], determina o grupo de pixel a ser usado na relação, que são [0,0,0,0,a_{11},a_{12},0,a_{21},a_{22}] em Ir, e [a_{11},a_{12},a_{21},a_{22}] na imagem original Io.

Esse processamento é feito em todos os pixels, linha por linha, da imagem de baixa resolução Ir até seu último

pixel.

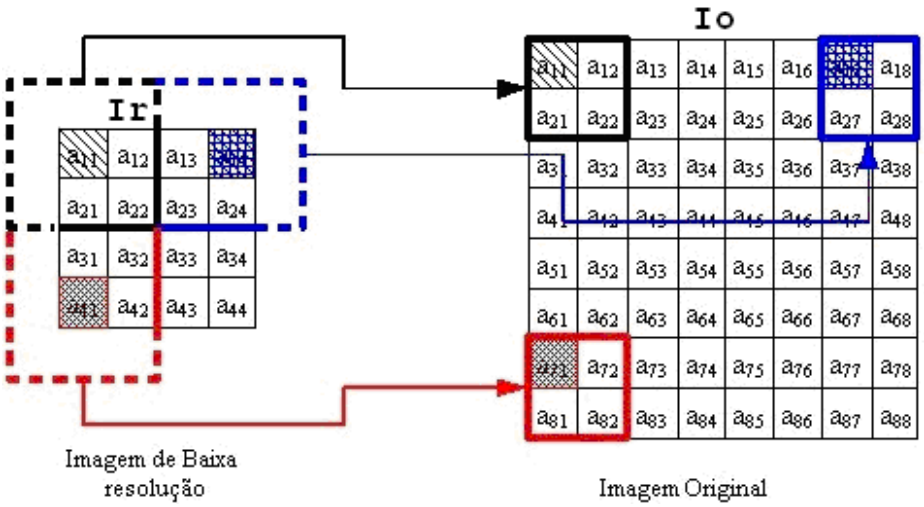

Figura 22 – Relação entre os pixels de Ir e Io – V2

Entendida a relação, mostramos a seguir como será a forma da matriz de entrada após a conversão da imagem de baixa resolução Ir.

Essa será uma matriz 9 x m, onde m é o número total de pixels da imagem de baixa resolução Ir, detalhada na tabela 4.

Tabela 4 – Matriz de Entrada – V2

0	0	0	0	0	a_{11}	a_{12}	a_{13}	0	a_{21}	a_{22}	a_{23}	0	a_{31}	a_{32}	a_{33}
0	0	0	0	a_{11}	a_{12}	a_{13}	a_{14}	a_{21}	a_{22}	a_{23}	a_{24}	a_{31}	a_{32}	a_{33}	a_{34}
0	0	0	0	a_{12}	a_{13}	a_{14}	0	a_{22}	a_{23}	a_{24}	0	a_{32}	a_{33}	a_{34}	0
0	a_{11}	a_{12}	a_{13}	0	a_{21}	a_{22}	a_{23}	0	a_{31}	a_{32}	a_{33}	0	a_{41}	a_{42}	a_{43}
a_{11}	***a_{12}***	***a_{13}***	***a_{14}***	***a_{21}***	***a_{22}***	***a_{23}***	***a_{24}***	***a_{31}***	***a_{32}***	***a_{33}***	***a_{34}***	***a_{41}***	***a_{42}***	***a_{43}***	***a_{44}***
a_{12}	a_{13}	a_{14}	0	a_{22}	a_{23}	a_{24}	0	a_{32}	a_{33}	a_{34}	0	a_{42}	a_{43}	a_{44}	0
0	a_{21}	a_{22}	a_{23}	0	a_{31}	a_{32}	a_{33}	0	a_{41}	a_{42}	a_{43}	0	0	0	0
a_{21}	a_{22}	a_{23}	a_{24}	a_{31}	a_{32}	a_{33}	a_{34}	a_{41}	a_{42}	a_{43}	a_{44}	0	0	0	0
a_{22}	a_{23}	a_{24}	0	a_{32}	a_{33}	a_{34}	0	a_{42}	a_{43}	a_{44}	0	0	0	0	0

A matriz de saída, descrita na tabela 5, possui a mesma forma do método anterior, visto que o alvo na imagem Io são os mesmos do método anterior para cada pixel pivô de Ir.

Tabela 5 – Matriz de Saída – V2

a_{11}	a_{13}	a_{15}	a_{17}	a_{31}	a_{33}	a_{35}	a_{37}	a_{51}	a_{53}	a_{55}	a_{57}	a_{71}	a_{73}	a_{75}	a_{77}

Esse método foi capaz de gerar imagens de alta resolução de excelente qualidade, muito suaves e nítidas.

5.3.3 Método V3

Esse método é uma simplificação do método anterior, já que esse utiliza um grupo menor de pixels da imagem Ir para estabelecer a relação entre as duas imagens.

Esse método só usa um grupo de 5 pixels, o mesmo pixel central usado no método anterior, e os pixels de cima, de baixo, à direita e à esquerda, ou seja, um grupo em forma de cruz.

Esse grupo de pixels vizinhos é o mesmo que é usado no algoritmo de interpolação bilinear, e assim como no método anterior, esse grupo de pixels é relacionado com um grupo de 4 pixels da imagem Io, uma matriz 2x2, como mostra a figura 23.

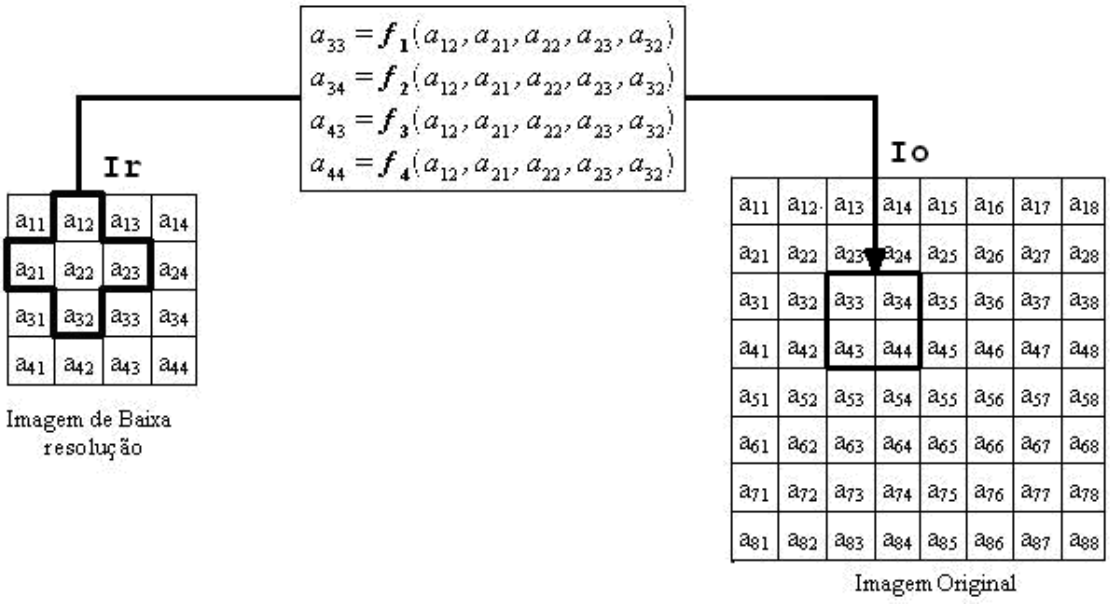

Figura 23 – Método V3

Como esse método usa 5 pixels da imagem Ir e 4 pixels de Io, a rede neural desse método terá 5 neurônios na camada de entrada, 3 na escondida e 4 na de saída, assim como na figura 24.

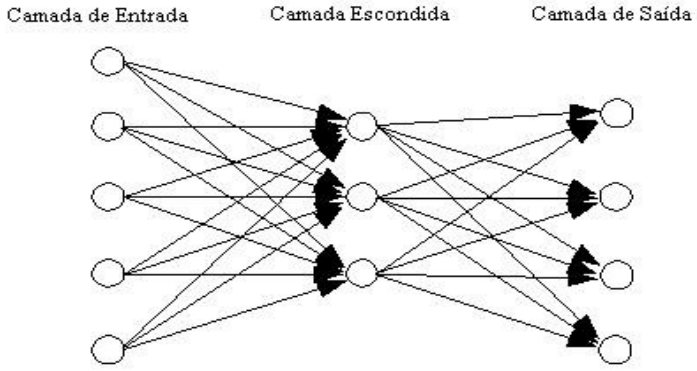

Figura 24 - Arquitetura da RNA do método V3

A conversão das imagens em matrizes ocorre da mesma forma que o método anterior, e os pontos que passam os limites da imagem Ir também são preenchidos com zeros. A velocidade de conversão e de treinamento desse método é ligeiramente mais rápida que o anterior, visto que esse utiliza menos pontos na matriz de entrada. Logo a seguir temos a relação que é seguida para a conversão das imagens em matrizes.

$$\mathbf{Io}\begin{vmatrix} a_{i,j} & a_{i,j+1} \\ a_{i+1,j} & a_{i+1,j+1} \end{vmatrix} = f(\mathbf{Ir}\left[a_{\frac{(j-1)}{2},\frac{(j-1)}{2}+1}, a_{\frac{(i-1)}{2}+1,\frac{(j-1)}{2}}, a_{\frac{(i-1)}{2}+1,\frac{(j-1)}{2}+1}, a_{\frac{(i-1)}{2}+1,\frac{(j-1)}{2}+2}, a_{\frac{(i-1)}{2}+2,\frac{(j-1)}{2}+1} \right])$$

Como foi dito anteriormente, os pixels pivôs desse método são os pixels centrais do grupo de 5 pixels na imagem Ir, por exemplo, o pivô marcado com preto, $[a_{11}]$, engloba o grupo $[0,0,a_{11},a_{12},a_{21}]$ e o grupo $[a_{11},a_{12},a_{21},a_{22}]$ na imagem original Io.

Assim como os demais métodos esse processamento irá passar por todos os pixels de Ir, linha por linha e os grupos de 5 pixels estão descritos na imagem 25.

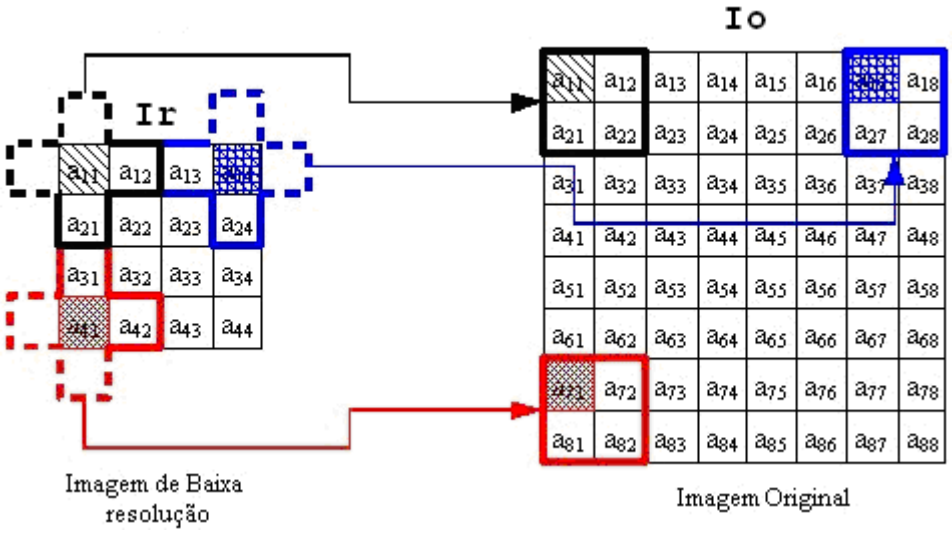

Figura 25 – Relação entre os pixels de Ir e Io – V3

A matriz de entrada desse método, tabela 6, após a conversão da imagem de baixa resolução Ir é uma matriz de 5 x m, onde m é o número total de pixels de Ir, semelhante ao método anterior, apenas com menos pixels.

Tabela 6 – Matriz de Entrada – V3

0	0	0	0	a_{11}	a_{12}	a_{13}	a_{14}	a_{21}	a_{22}	a_{23}	a_{24}	a_{31}	a_{32}	a_{33}	a_{34}
0	a_{11}	a_{12}	a_{13}	0	a_{21}	a_{22}	a_{23}	0	a_{31}	a_{32}	a_{33}	0	a_{41}	a_{42}	a_{43}
a_{11}	a_{12}	a_{13}	a_{14}	a_{21}	a_{22}	a_{23}	a_{24}	a_{31}	a_{32}	a_{33}	a_{34}	a_{41}	a_{42}	a_{43}	a_{44}
a_{12}	a_{13}	a_{14}	0	a_{22}	a_{23}	a_{24}	0	a_{32}	a_{33}	a_{34}	0	a_{42}	a_{43}	a_{44}	0
a_{21}	a_{22}	a_{23}	a_{24}	a_{31}	a_{32}	a_{33}	a_{34}	a_{41}	a_{42}	a_{43}	a_{44}	0	0	0	0

A matriz de saída, mostrada na tabela 7, é igual ao dos métodos anteriores, ou seja, uma matriz 4 x n onde n é um quarto do total de pixels da imagem original Io.

Tabela 7 – Matriz de Saída – V3

a_{11}	a_{13}	a_{15}	a_{17}	a_{31}	a_{33}	a_{35}	a_{37}	a_{51}	a_{53}	a_{55}	a_{57}	a_{71}	a_{73}	a_{75}	a_{77}
a_{12}	a_{14}	a_{16}	a_{18}	a_{32}	a_{34}	a_{36}	a_{38}	a_{52}	a_{54}	a_{56}	a_{58}	a_{72}	a_{74}	a_{76}	a_{78}
a_{21}	a_{23}	a_{25}	a_{27}	a_{41}	a_{43}	a_{45}	a_{47}	a_{61}	a_{63}	a_{65}	a_{67}	a_{81}	a_{83}	a_{85}	a_{87}
a_{22}	a_{24}	a_{26}	a_{28}	a_{42}	a_{44}	a_{46}	a_{48}	a_{62}	a_{64}	a_{66}	a_{68}	a_{82}	a_{84}	a_{86}	a_{88}

O desempenho desse método foi ligeiramente pior que o anterior, devido ao menor número de pontos para relacionar nas duas imagens.

5.4 Observação

Poderíamos concluir erradamente que para obtermos resultados melhores bastaria aumentar ainda mais o número de pixels circundantes ao pivô, porém isso não se comprovou.

Ao se aumentar cada vez mais a cruz, passamos a obter uma aproximação de um círculo ao redor do pivô, o que intuitivamente parece o ideal. Porém, mesmo estes pixels distantes do pivô, terem adquirido peso sináptico muito baixo, o tempo necessário para as conversões, treinamento e simulação da rede tornaram-se proibitivos quando comparados aos ganhos de Erro Médio Quadrático.

Como já mencionado anteriormente, outros métodos foram implementados durante a elaboração do projeto, porém estes apresentaram resultados insatisfatórios.

Um destes utilizou 25 pixels (5x5 incluindo o pivô), outro 21 pixels (uma cruz ainda maior ao redor do pivô).

Outra abordagem totalmente distinta, oposta ao método V1, onde relacionamos 4 pixels de Ir a 1 pixel de Io também não se mostrou eficaz.

Dessa forma, tratando a imagem através de representações gráficas dos pixels, as melhores representações encontradas foram as descritas pelos métodos aqui presentes: 1 pixel, 1 cruz com 5 pixels e 1 quadrado com 9 pixels (o de melhor resultado).

Capítulo 6

Estudos de casos:

Em todos os próximos itens estaremos mostrando a imagem original e logo em seguida uma tabela com o erro médio quadrático entre a imagem original e a ampliada; e a relação sinal ruído, para termos uma avaliação matemática, afastando-nos da subjetividade da avaliação visual.

6.1 Célula

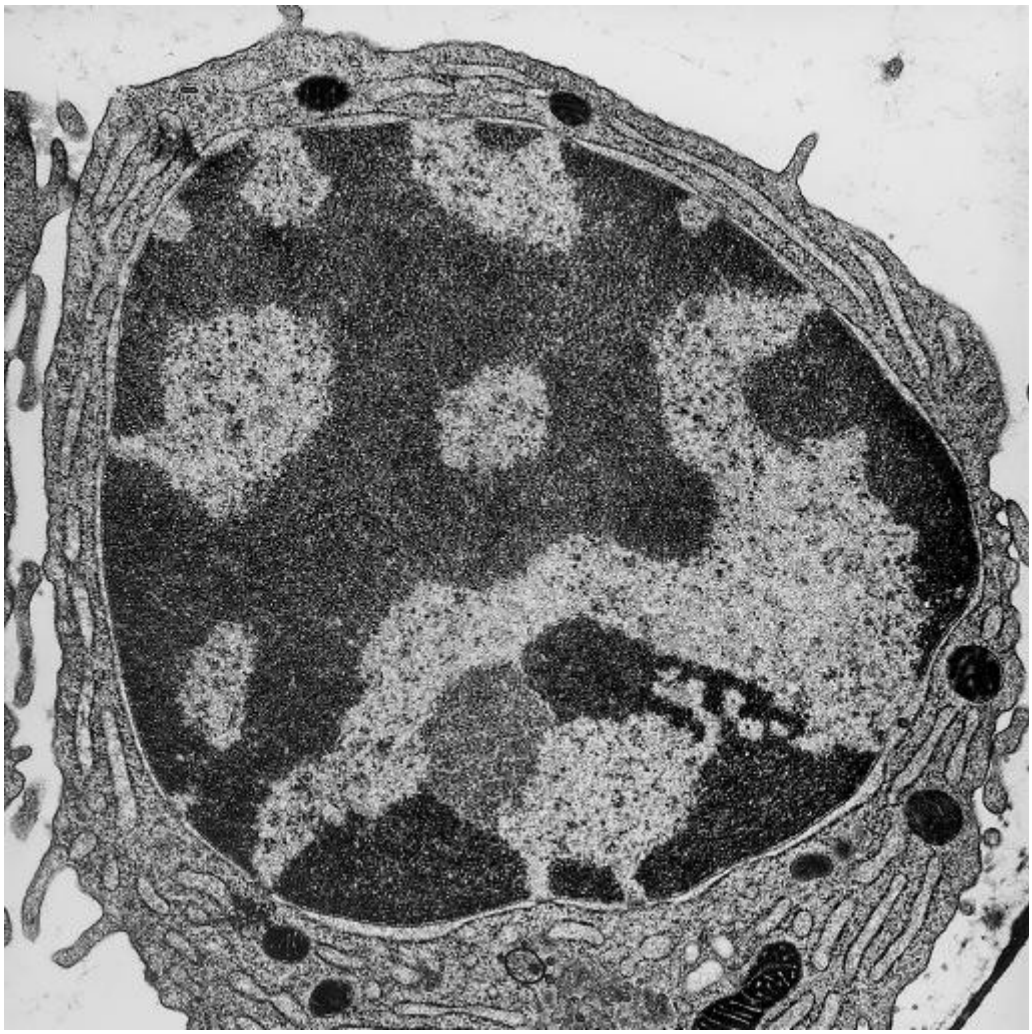

Figura 26 – Célula

Tabela 8 – Célula

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.1342	16.0206
Bicubic	0.1395	15.1683
B-Spline	0.1416	14.3268
Lanczos	0.1394	15.1588
Método V1	**0.1261**	**18.1716**
Método V2	**0.1256**	**18.4373**
Método V3	0.1273	17.6177

Obs.: Simulada somente para fins de validação, já que foi utilizada durante o treinamento da rede.

6.2 Fagocitose

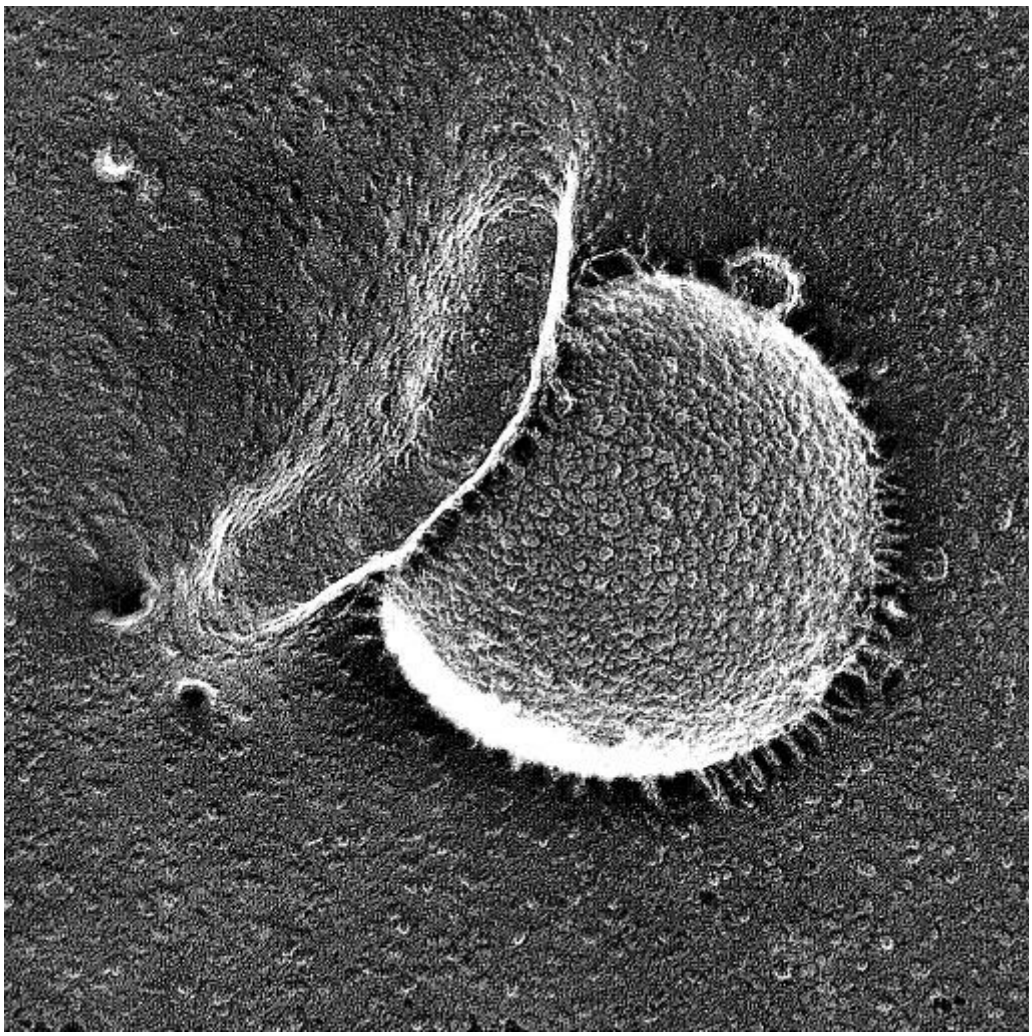

Figura 27 – Fagocitose

Tabela 9 – Fagocitose

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.1836	3.9924
Bicubic	0.1901	3.8907
B-Spline	0.1920	3.5585
Lanczos	0.1900	3.8758
Método V1	**0.1742**	**4.6046**
Método V2	0.1760	**4.5725**
Método V3	**0.1755**	4.5407

6.3 Alvéolos

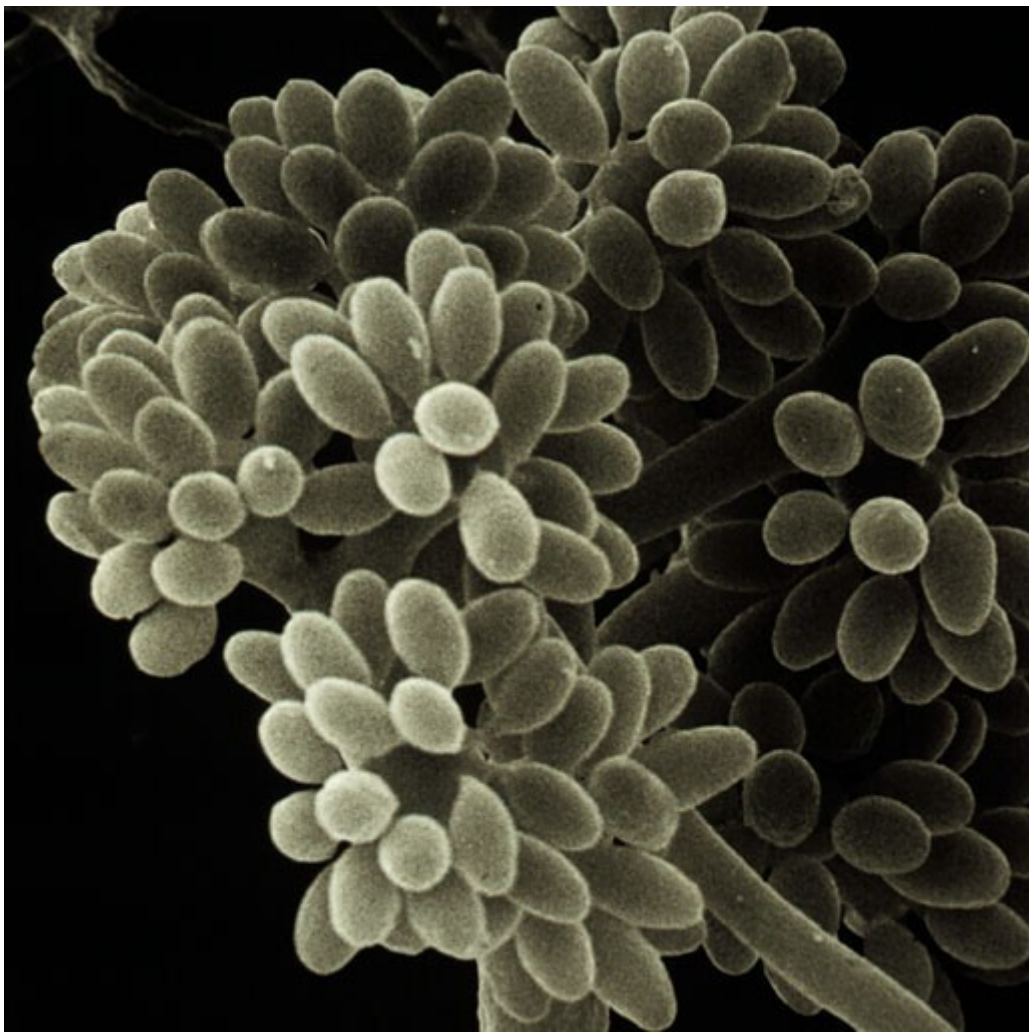

Figura 28 – Alvéolos

Tabela 10 – Alvéolos

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.0320	79.8838
Bicubic	0.0374	59.7965
B-Spline	0.0431	43.3446
Lanczos	0.0371	60.3175
Método V1	0.0341	71.8897
Método V2	**0.0227**	**162.8390**
Método V3	**0.0282**	**105.5262**

6.4 Músculo

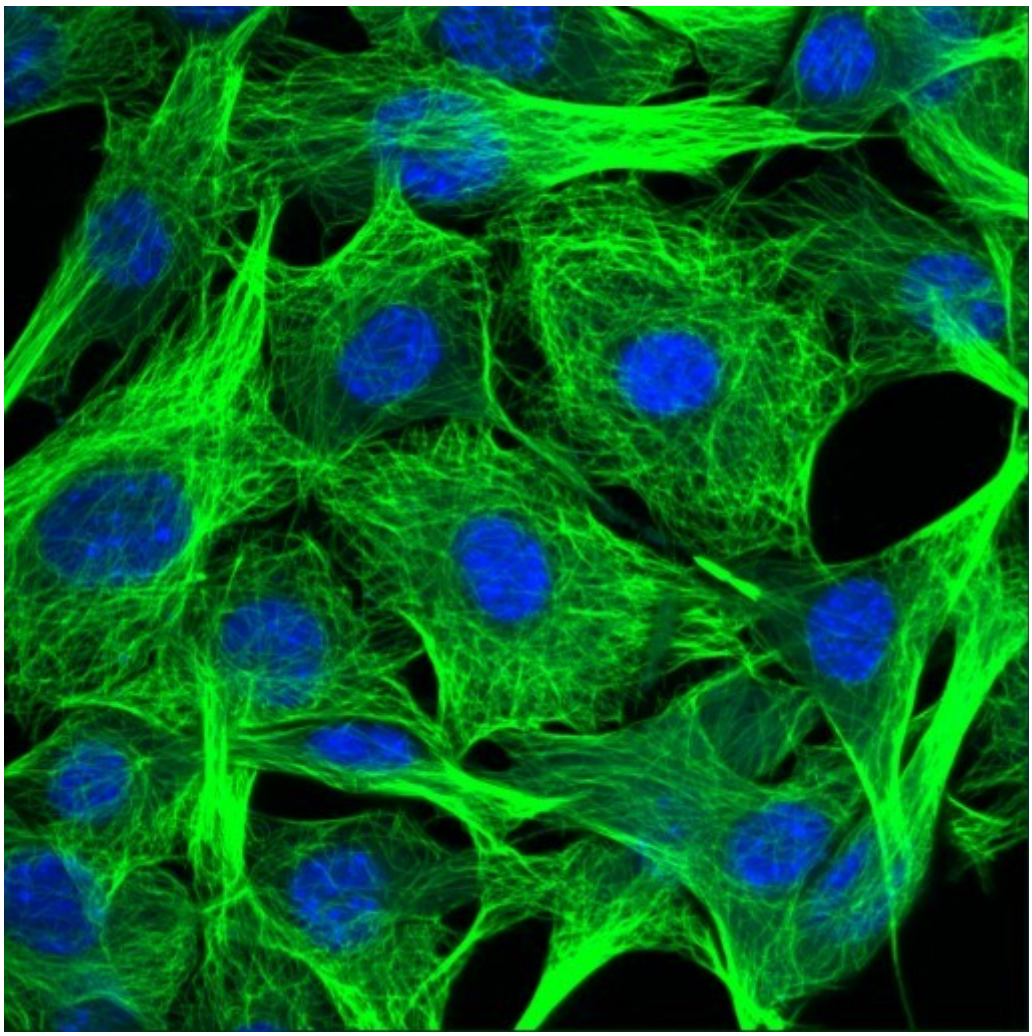

Figura 29 – Músculo

Tabela 11 – Músculo

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.0513	30.2258
Bicubic	0.0539	27.8474
B-Spline	0.0541	27.0026
Lanczos	0.0537	27.9162
Método V1	0.0318	54.1118
Método V2	**0.0235**	**70.0987**
Método V3	**0.0285**	**59.1619**

6.5 Barco

Figura 30 – Barco

Tabela 12 – Barco

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.0591	83.2057
Bicubic	0.0626	74.8727
B-Spline	0.0647	69.5290
Lanczos	0.0625	75.1179
Método V1	**0.0544**	**97.5097**
Método V2	**0.0537**	**100.9239**
Método V3	0.0612	79.2554

6.6 Peixes

Figura 31 – Peixes

Tabela 13 – Peixes

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.0341	213.9538
Bicubic	0.0367	185.5711
B-Spline	0.0373	179.6393
Lanczos	0.0366	187.0903
Método V1	**0.0274**	341.0377
Método V2	**0.0259**	**376.6082**
Método V3	0.0291	**344.9680**

6.7 Maracanã

Figura 32 – Maracanã

Tabela 14 – Maracanã

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.0356	170.0370
Bicubic	0.0388	144.2460
B-Spline	0.0430	116.2348
Lanczos	0.0385	145.7793
Método V1	0.0309	236.7787
Método V2	**0.0224**	**483.9403**
Método V3	**0.0301**	**262.8501**

6.8 Cachoeira

Figura 33 – Cachoeira

Tabela 15 – Cachoeira

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.0650	66.0011
Bicubic	0.0677	61.2206
B-Spline	0.0715	54.4508
Lanczos	0.0677	61.3070
Método V1	**0.0630**	**71.1324**
Método V2	**0.0620**	**72.0938**
Método V3	0.0673	59.4634

6.9 *Baía de Guanabara*

Figura 34 – Baía de Guanabara

Tabela 16 – Baía de Guanabara

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	**0.0193**	**658.4185**
Bicubic	0.0207	572.9947
B-Spline	0.0243	417.8918
Lanczos	0.0206	578.5715
Método V1	0.0199	625.4489
Método V2	**0.0178**	**788.3681**
Método V3	0.0205	584.4509

6.10 Rio de Janeiro

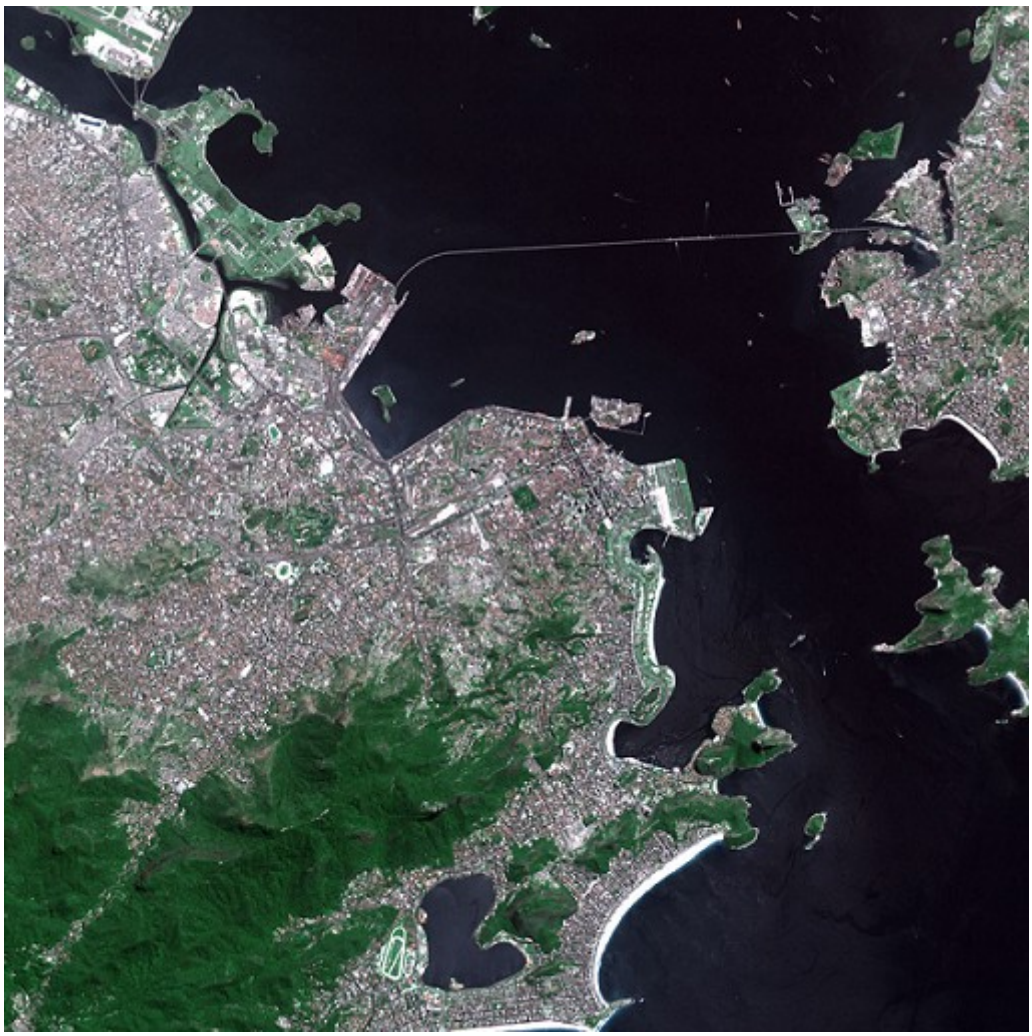

Figura 35 – Rio de Janeiro

Tabela 17 – Rio de Janeiro

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.0782	19.9513
Bicubic	0.0819	18.6026
B-Spline	0.0872	15.8595
Lanczos	0.0818	18.5931
Método V1	0.0746	22.6506
Método V2	**0.0712**	**24.9414**
Método V3	**0.0731**	**23.4067**

6.11 Lagoa Rodrigo de Freitas

Figura 36 – Lagoa Rodrigo de Freitas

Tabela 18 – Lagoa Rodrigo de Freitas

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.0585	24.6065
Bicubic	0.0612	23.4246
B-Spline	0.0704	16.5755
Lanczos	0.0609	23.4613
Método V1	0.0561	27.9551
Método V2	**0.0452**	**43.6664**
Método V3	**0.0502**	**35.1544**

6.12 Aeroporto Santos Dummont

Figura 37 – Aeroporto Santos Dummont

Tabela 19 – Aeroporto Santos Dummont

Método	Erro médio quadrático	Relação sinal-ruído
Bilinear	0.0522	41.2633
Bicubic	0.0568	35.8613
B-Spline	0.0659	25.4306
Lanczos	0.0565	36.0915
Método V1	0.0524	42.3814
Método V2	**0.0404**	**72.0174**
Método V3	**0.0463**	**54.0138**

Capítulo 7:

Conclusão:

O objetivo inicial do trabalho foi atingido.

Conseguimos desenvolver métodos que se utilizaram das vantagens inerentes das Redes Neurais como candidatas a interpoladoras de imagem, e atingiram resultados semelhantes aos dos métodos clássicos de interpolação utilizados amplamente pelo mercado.

Melhor que isso, conseguimos até mesmo encontrar um método onde os resultados obtidos, através não somente da avaliação visual (subjetiva), mas também de comparações matemáticas (Erro Médio Quadrático e Relação Sinal Ruído), foram melhores que os dos modelos clássicos.

Acredito que o trabalho possa e deva ser continuado, e creio que as melhorias iniciais seriam:

- Tratar não somente imagens quadradas.
- Procurar novas arquiteturas de Redes, um pouco mais complexas.
- Evitar o uso da aproximação abs na Matriz de Saída, utilizando preferencialmente uma saturação quando os valores ultrapassarem os limiares.
- Evitar o preenchimento com zeros, e utilizar a replicação das bordas.
- Utilizar a relação PSNR.
- Utilizar incremental ou batch training.
- Associar filtros ao final do processamento a fim de aumentar ainda mais a Relação Sinal Ruído.

Bibliografia:

[1]ZINNEMAN, T., "Interpolation by T. Zinneman", site http://www.nut-n-but.net/CCPCUG/TechBrief Articles/TB0204.html.

[2]Manual "Imaging Processing Toolbox for use in MATLAB, version 2", Mathworks Inc, 1999.

[3]Manual "Neural Network Toolbox for use in MATLAB, version 3.0", Mathworks Inc, 1999.

[4]JÄHNE, P., "Digital Image Processing", Ed. Springer-Verlag, New York, 1997.

[5]SOILLE, P., "Morphological Image Analysis. Principles and Applications", Ed. Springer-Verlag, New York, 1999.

[6]ATKINS, C. Brain, "Classification-Based Methods in optimal image interpolation", Ed. Purdue University, West Lafayette, Indiana, 1998.

[7]HAYKIN, Simon, "Redes Neurais - Princípios e Práticas", Ed. Bookman, Rio de janeiro, RJ, 2000.

[8]WOODS, Richard, "Digital Image Processing", Ed. Prentice Hall, Rio de Janeiro, RJ, 2001.

[9]STEIN, Michael L., "Interpolation of Spatial Data: Some Mathematics for Kriging", Springer-Verlag, New York, 1999.

Apêndice A - Código Fonte (MatLab):

a) Redução da Imagem Original

```
% Reduzir Imagem por 2
% Fabiano Malhard Araujo de Barros

function Ir=reduzir_imagem_por_2(Io)

[x y z]=size(Io);

if (isind(Io)==1)
% Funcao isind retorna o valor 1 se a imagem for indexada e
0 se nao
  load clown
  if z==1
% Funcao size retorna 1 se a imagem for em escala de cinza
e 3 se RGB
    RBG=ind2gray(Io,map);
  else
    RGB=ind2rgb(Io,map);
  end;
end;

percent=waitbar(0,'Reduzindo Imagem');

if z==3
% Se a imagem for RGB z=3, tratamos uma cor por vez
  red=blkproc(Io(:,:,1),[2 2],'mean2(x)');
  waitbar(1/3,percent);
  green=blkproc(Io(:,:,2),[2 2],'mean2(x)');
  waitbar(2/3,percent);
  blue=blkproc(Io(:,:,3),[2 2],'mean2(x)');
  close(percent);
  Ir=uint8(cat(3,red,green,blue));
else
    Ir=blkproc(im2double(Io),[2 2],'mean2(x)');
end;
```

b) Relação Sinal Ruído

```
% Calcular a Razao Sinal Ruido
% Fabiano Malhard Araujo de Barros

function x=SNR(Io,Ia)
% Onde Ia - imagem ampliada, Io - imagem original

s=sum(sum(sum((im2double(Ia).^2))));
r=sum(sum(sum( ((im2double(Ia)-im2double(Io)).^2) )));
x=s/r;
```

c) Erro Médio Quadrático

```
% Calcular o Erro Medio Quadratico
% Fabiano Malhard Araujo de Barros

function x=EMQ(Io,Ia)
% Onde Ia - imagem ampliada, Io - imagem original

x=sqrt(mean2((im2double(Io)-im2double(Ia)).^2));
```

d) Gerar Matriz de Alvo para o treinamento da Rede

```
% Converter a Imagem Original para a Matriz de Alvos da
Rede para o processo de treinamento
% Fabiano Malhard Araujo de Barros

function Mt=gerar_matriz_de_alvo(Io)
% Onde Io - imagem original, Mt - matriz de alvo para o
treinamento
[X Y Z]=size(Io);

k=0;
percent=waitbar(0,'Gerando Matriz de Saida da Rede');

% Cada pixel da imagem reduzida se relaciona com 4 pixels
da imagem original
for i=1:2:X
  waitbar(i/(X),percent);
  for j=1:2:Y
    k=k+1;
    M(1,k,1)=Io(i,j,1);
    M(2,k,1)=Io(i,j+1,1);
    M(3,k,1)=Io(i+1,j,1);
    M(4,k,1)=Io(i+1,j+1,1);

    if Z==3
      M(1,k,2)=Io(i,j,2);
      M(2,k,2)=Io(i,j+1,2);
      M(3,k,2)=Io(i+1,j,2);
      M(4,k,2)=Io(i+1,j+1,2);

      M(1,k,3)=Io(i,j,3);
      M(2,k,3)=Io(i,j+1,3);
      M(3,k,3)=Io(i+1,j,3);
      M(4,k,3)=Io(i+1,j+1,3);
    end;
   end;
end;

close(percent);
Mt=M;
```

e) Converter Matriz de Saída

```
% Coverter Matriz de Saida para a Imagem Ampliada
% Fabiano Malhard Araujo de Barros

function C=converter_matriz_de_saida(Mout,col)
% Onde Ia - imagem ampliada, Mout - matriz de saida da rede

[X Y Z]=size(Mout);

percent=waitbar(0,'Gerando Imagem Ampliada');
i=1;
k=1;

for j=1:Y
   waitbar(j/Y,percent);
   R(i,2*k-1,1)=Mout(1,j,1);
   R(i,2*k,1)=Mout(2,j,1);
   R(i+1,2*k-1,1)=Mout(3,j,1);
   R(i+1,2*k,1)=Mout(4,j,1);

   if Z==3
      R(i,2*k-1,2)=Mout(1,j,2);
      R(i,2*k,2)=Mout(2,j,2);
      R(i+1,2*k-1,2)=Mout(3,j,2);
      R(i+1,2*k,2)=Mout(4,j,2);

      R(i,2*k-1,3)=Mout(1,j,3);
      R(i,2*k,3)=Mout(2,j,3);
      R(i+1,2*k-1,3)=Mout(3,j,3);
      R(i+1,2*k,3)=Mout(4,j,3);
   end;

   k=k+1;
   if k-1>=(col/2)
      i=i+2;
      k=1;
   end;
end;

close(percent);
Ia=R;
```

f) Método V1

i) Gerar Matriz de Entrada V1

```
% Converter a Imagem Reduzida para a Matriz de Entrada da
Rede
% Fabiano Malhard Araujo de Barros

function Me=gerar_matriz_de_entrada_V1(Ir);
% Onde Me - matriz de entrada, Ir - imagem reduzida

[X Y Z]=size(Ir);

col=1;

percent=waitbar(0,'Gerando Matriz de Entrada da Rede');

% Cada pixel da imagem reduzida e pivo, e de grupo unico
for i=1:X
   waitbar(i/(X),percent);
   Me(1,col:col+(Y-1),1)=Ir(i,:,1);

   if Z==3
      Me(1,col:col+(Y-1),2)=Ir(i,:,2);
      Me(1,col:col+(Y-1),3)=Ir(i,:,3);
   end;
   col=col+Y;
end;

close(percent);
```

ii) Treinar Rede V1

```
% Treinar a Rede Neural V1
% Fabiano Malhard Araujo de Barros

function [netR,netG,netB]=treinar_rede_V1(Me,Mt)
% Onde netX - sao as redes RGB individualizadas, Me -
matriz de entrada, Mt - matriz de alvos

[x y z]=size(Me);

InNet=im2double(Me);
OutNet=im2double(Mt);

nNet=3;

netR=newff(minmax(InNet(:,:,1)),[nNet,4],
{'tansig','purelin'},'trainbr');
% Criacao da rede com 3 neuronios na camada intermediaria e
4 na ultima camada; utilizando as funcoes de transferencia
```

```
tansig e purelin; e a funcao de treinamento trainbr que
representa a otimizacao de Levenberg-Marquardt.
% Nos casos de imagens RGB, esta rede tratara os tons de
vermelho
netR.trainParam.epochs=21;
netR.trainParam.show=3;
netR.trainParam.goal=1e-4;
netR=train(netR,InNet(:,:,1),OutNet(:,:,1));

if z == 3
                    netG=newff(minmax(InNet(:,:,2)),[nNet,4],
{'tansig','purelin'},'trainbr');
%Criacao de uma nova rede para os tons de verde no caso de
imagens RGB
   netG.trainParam.epochs=21;
   netG.trainParam.show=3;
   netG.trainParam.goal=1e-4;
   netG=train(netG,InNet(:,:,2),OutNet(:,:,2));

                    netB=newff(minmax(InNet(:,:,3)),[nNet,4],
{'tansig','purelin'},'trainbr');
%Criacao de uma nova rede para os tons de azul no caso de
imagens RGB
   netB.trainParam.epochs=21;
   netB.trainParam.show=3;
   netB.trainParam.goal=1e-4;
   netB=train(netB,InNet(:,:,3),OutNet(:,:,3));
end;
```

iii) Simular Rede V1

```
% Simular a Rede Neural V1 para a Imagem Original
% Fabiano Malhard Araujo de Barros

function Ia=simular_rede_V1(Io,netR,netG,netB)
% Onde Ia - imagem ampliada, Io - imagem original, netX -
redes RGB individuais

Ir=reduzir_imagem_por_2(Io);

[X Y Z]=size(Ir);

InRede=im2double(gerar_matriz_de_entrada_V1(Ir));

MR=sim(netR,InRede(:,:,1));

if Z==3
   MG=sim(netG,InRede(:,:,2));
   MB=sim(netB,InRede(:,:,3));
   M=cat(3,MR,MG,MB);
else
```

```
    M=MR;
end;

Ia=converter_matriz_de_saida_V1(M,2*Y);
```

g) Método V2

i) Gerar Matriz de Entrada V2

```
% Converter a Imagem Original para a Matriz de Alvos da
Rede para o processo de treinamento
% Fabiano Malhard Araujo de Barros

function Me=gerar_matriz_de_entrada_V2(Ir);
% Onde Me - matriz de entrada, Ir - imagem reduzida

[X Y Z]=size(Ir);

% Copiando a linha n para a posicao n+1 para nao perder
nenhum dado
% Logo comecamos da 256 e a copiamos para a linha 257
for i=X:-1:1
   Ir(i+1,:,:)=Ir(i,:,:);
end;
% Substituicao da linha 1, que foi copiada para a linha 2,
por zeros
Ir(1,:,:)=zeros(1,Y,Z);

% Copiando a coluna n para a posicao n+1 para nao perder
nenhum dado
% Logo comecamos da 256 e a copiamos para a coluna 257
for j=Y:-1:1
   Ir(:,j+1,:)=Ir(:,j,:);
end;
% Substituicao da coluna 1, que foi copiada para a coluna
2, por zeros
Ir(:,1,:)=zeros(X+1,1,Z);

% Adicionando uma linha de zeros na ultima linha
Ir(X+2,1,:)=0;

% Adicionando uma coluna de zeros na ultima coluna
Ir(1,Y+2,:)=0;

col=1;
percent=waitbar(0,'Gerando Matriz de Entrada da Rede');

% Cada pixel da imagem reduzida e pivo, e central de um
grupo de 9 pixels, em quadrado
for i=2:X+1
  for j=2:Y+1
    Me(1:3,col,1)=Ir(i-1:i+1,j-1,1);
    Me(4:6,col,1)=Ir(i-1:i+1,j,1);
    Me(7:9,col,1)=Ir(i-1:i+1,j+1,1);

    if Z==3
```

```
      Me(1:3,col,2)=Ir(i-1:i+1,j-1,2);
      Me(4:6,col,2)=Ir(i-1:i+1,j,2);
      Me(7:9,col,2)=Ir(i-1:i+1,j+1,2);

      Me(1:3,col,3)=Ir(i-1:i+1,j-1,3);
      Me(4:6,col,3)=Ir(i-1:i+1,j,3);
      Me(7:9,col,3)=Ir(i-1:i+1,j+1,3);
    end;
    col=col+1;
  end;
  waitbar((i/X),percent);
end;

close(percent);
```

ii) Treinar Rede V2

```
% Treinar a Rede Neural V2
% Fabiano Malhard Araujo de Barros

function [netR,netG,netB]=treinar_rede_V2(Me,Mt)
% Onde netX - sao as redes RGB individualizadas, Me -
matriz de entrada, Mt - matriz de alvos

[x y z]=size(Me);

InNet=im2double(Me);
OutNet=im2double(Mt);

nNet=3;

netR=newff(minmax(InNet(:,:,1)),[nNet,4],
{'tansig','purelin'},'trainbr');
% Criacao da rede com 3 neuronios na camada intermediaria e
4 na ultima camada; utilizando as funcoes de transferencia
tansig e purelin; e a funcao de treinamento trainbr que
representa a otimizacao de Levenberg-Marquardt.
% Nos casos de imagens RGB, esta rede tratara os tons de
vermelho
netR.trainParam.epochs=21;
netR.trainParam.show=3;
netR.trainParam.goal=1e-4;
netR=train(netR,InNet(:,:,1),OutNet(:,:,1));

if z == 3
                  netG=newff(minmax(InNet(:,:,2)),[nNet,4],
{'tansig','purelin'},'trainbr');
%Criacao de uma nova rede para os tons de verde no caso de
imagens RGB
  netG.trainParam.epochs=21;
```

```
  netG.trainParam.show=3;
  netG.trainParam.goal=1e-4;
  netG=train(netG,InNet(:,:,2),OutNet(:,:,2));

                  netB=newff(minmax(InNet(:,:,3)),[nNet,4],
{'tansig','purelin'},'trainbr');
%Criacao de uma nova rede para os tons de azul no caso de
imagens RGB
  netB.trainParam.epochs=21;
  netB.trainParam.show=3;
  netB.trainParam.goal=1e-4;
  netB=train(netB,InNet(:,:,3),OutNet(:,:,3));
end;
```

iii) Simular Rede V2

```
% Simular a Rede Neural V2 para a Imagem Original
% Fabiano Malhard Araujo de Barros

function Ia=simular_rede_V2(Io,netR,netG,netB)
% Onde Ia - imagem ampliada, Io - imagem original, netX -
redes  RGB individuais

Ir=reduzir_imagem_por_2(Io);

[X Y Z]=size(Ir);

InRede=im2double(gerar_matriz_de_entrada_V2(Ir));

MR=sim(netR,InRede(:,:,1));

if Z==3
   MG=sim(netG,InRede(:,:,2));
   MB=sim(netB,InRede(:,:,3));
   M=cat(3,MR,MG,MB);
else
   M=MR;
end;

Ia=abs(converter_matriz_de_saida_V2(M,2*Y));
```

h) Método V3

i) Gerar Matriz de Entrada V3

```
% Converter a Imagem Reduzida para a Matriz de Entrada da
Rede
% Fabiano Malhard Araujo de Barros

function Me=gerar_matriz_de_entrada_V3(Ir);
% Onde Me - matriz de entrada, Ir - imagem reduzida

[X Y Z]=size(Ir);

% Copiando a linha n para a posicao n+1 para nao perder
nenhum dado
% Logo comecamos da 256 e a copiamos para a linha 257
for i=X:-1:1
   Ir(i+1,:,:)=Ir(i,:,:);
end;
% Substituicao da linha 1, que foi copiada para a linha 2,
por zeros
Ir(1,:,:)=zeros(1,Y,Z);

% Copiando a coluna n para a posicao n+1 para nao perder
nenhum dado
% Logo comecamos da 256 e a copiamos para a coluna 257
for j=Y:-1:1
   Ir(:,j+1,:)=Ir(:,j,:);
end;
% Substituicao da coluna 1, que foi copiada para a coluna
2, por zeros
Ir(:,1,:)=zeros(X+1,1,Z);

% Adicionando uma linha de zeros na ultima linha
Ir(X+2,1,:)=0;

% Adicionando uma coluna de zeros na ultima coluna
Ir(1,Y+2,:)=0;

col=1;
percent=waitbar(0,'Gerando Matriz de Entrada da Rede');

% Cada pixel da imagem reduzida e pivo, e central de um
grupo de 5 pixels, em forma de cruz
for i=2:X+1
   for j=2:Y+1
      Me(1,col,1)=Ir(i-1,j,1);
      Me(2,col,1)=Ir(i-1,j-1,1);
      Me(3,col,1)=Ir(i,j,1);
      Me(4,col,1)=Ir(i,j+1,1);
      Me(5,col,1)=Ir(i+1,j,1);
```

```
      if Z==3
         Me(1,col,2)=Ir(i-1,j,2);
         Me(2,col,2)=Ir(i-1,j-1,2);
         Me(3,col,2)=Ir(i,j,2);
         Me(4,col,2)=Ir(i,j+1,2);
         Me(5,col,2)=Ir(i+1,j,2);

         Me(1,col,3)=Ir(i-1,j,3);
         Me(2,col,3)=Ir(i-1,j-1,3);
         Me(3,col,3)=Ir(i,j,3);
         Me(4,col,3)=Ir(i,j+1,3);
         Me(5,col,3)=Ir(i+1,j,3);
      end;
      col=col+1;
   end;
   waitbar((i/X),percent);
end;

close(percent);
```

ii) Treinar Rede V3

```
% Treinar a Rede Neural V3
% Fabiano Malhard Araujo de Barros

function [netR,netG,netB]=treinar_rede_v3(Me,Mt)
% Onde netX - sao as redes RGB individualizadas, Me -
matriz de entrada, Mt - matriz de alvos

[x y z]=size(Me);

InNet=im2double(Me);
OutNet=im2double(Mt);

nNet=3;

netR=newff(minmax(InNet(:,:,1)),[nNet,4],
{'tansig','purelin'},'trainbr');
% Criacao da rede com 3 neuronios na camada intermediaria e
4 na ultima camada; utilizando as funcoes de transferencia
tansig e purelin; e a funcao de treinamento trainbr que
representa a otimizacao de Levenberg-Marquardt.
% Nos casos de imagens RGB, esta rede tratara os tons de
vermelho
netR.trainParam.epochs=12;
netR.trainParam.show=3;
netR.trainParam.goal=1e-4;
netR=train(netR,InNet(:,:,1),OutNet(:,:,1));

if z == 3
```

```
                    netG=newff(minmax(InNet(:,:,2)),[nNet,4],
{'tansig','purelin'},'trainbr');
%Criacao de uma nova rede para os tons de verde no caso de
imagens RGB
   netG.trainParam.epochs=12;
   netG.trainParam.show=3;
   netG.trainParam.goal=1e-4;
   netG=train(netG,InNet(:,:,2),OutNet(:,:,2));

                    netB=newff(minmax(InNet(:,:,3)),[nNet,4],
{'tansig','purelin'},'trainbr');
%Criacao de uma nova rede para os tons de azul no caso de
imagens RGB
   netB.trainParam.epochs=12;
   netB.trainParam.show=3;
   netB.trainParam.goal=1e-4;
   netB=train(netB,InNet(:,:,3),OutNet(:,:,3));
end;
```

iii) Simular Rede V3

```
% Simular a Rede Neural V3 para a Imagem Original
% Fabiano Malhard Araujo de Barros

function Ia=simNetV4(Io,netR,netG,netB)
% Onde Ia - imagem ampliada, Io - imagem original, netX -
redes RGB individuais

Ir=reduzir_imagem_por_2(Io);

[X Y Z]=size(Ir);

InRede=im2double(gerar_matriz_de_entrada_V3(Ir));

MR=sim(netR,InRede(:,:,1));

if Z==3
   MG=sim(netG,InRede(:,:,2));
   MB=sim(netB,InRede(:,:,3));
   M=cat(3,MR,MG,MB);
else
   M=MR;
end;

Ia=abs(converter_matriz_de_saida_V3(M,2*Y));
```

Apêndice B - Fotografias:

a) Célula

i) Original

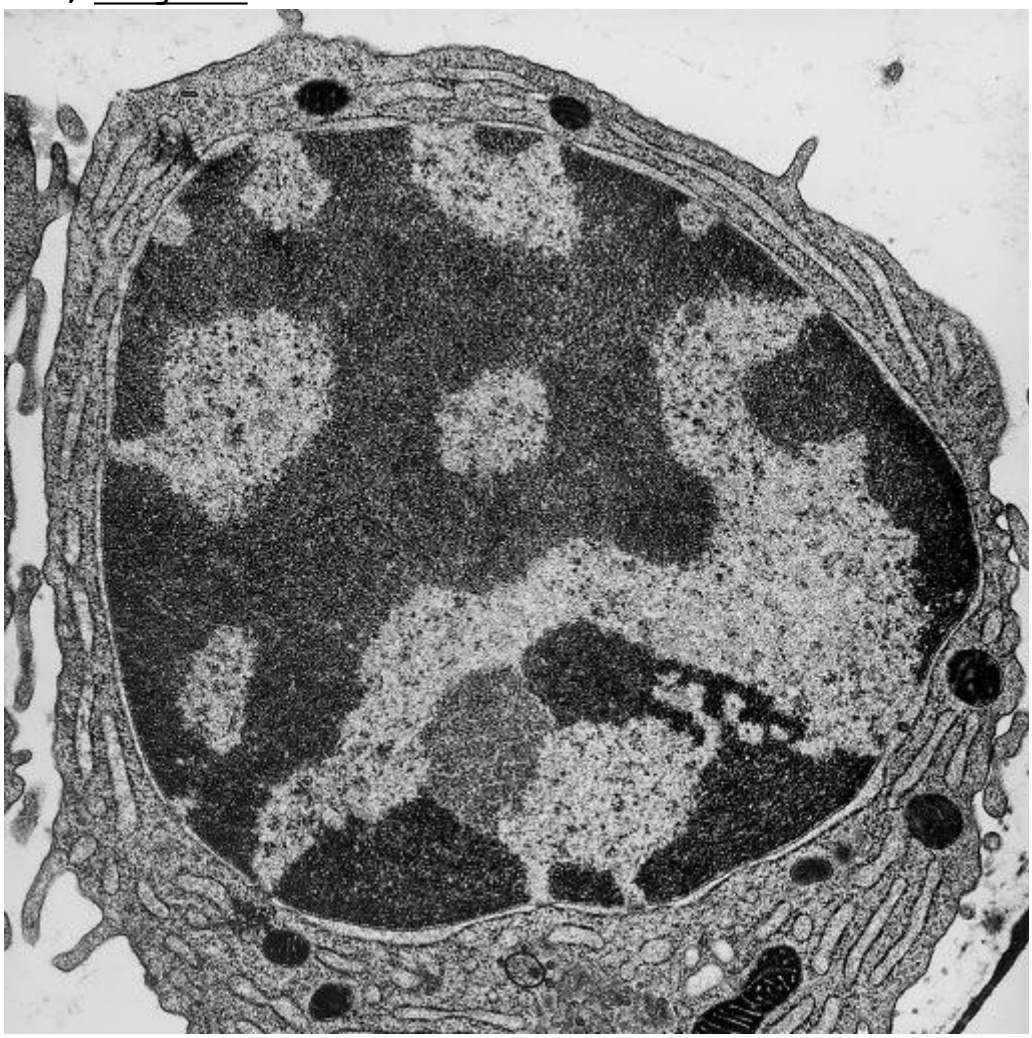

Figura 38 – Célula Original

ii) Reduzida

Figura 39 – Célula Reduzida

Obs.: Simulada somente para fins de validação, já que foi utilizada durante o treinamento da rede.

iii) Bilinear

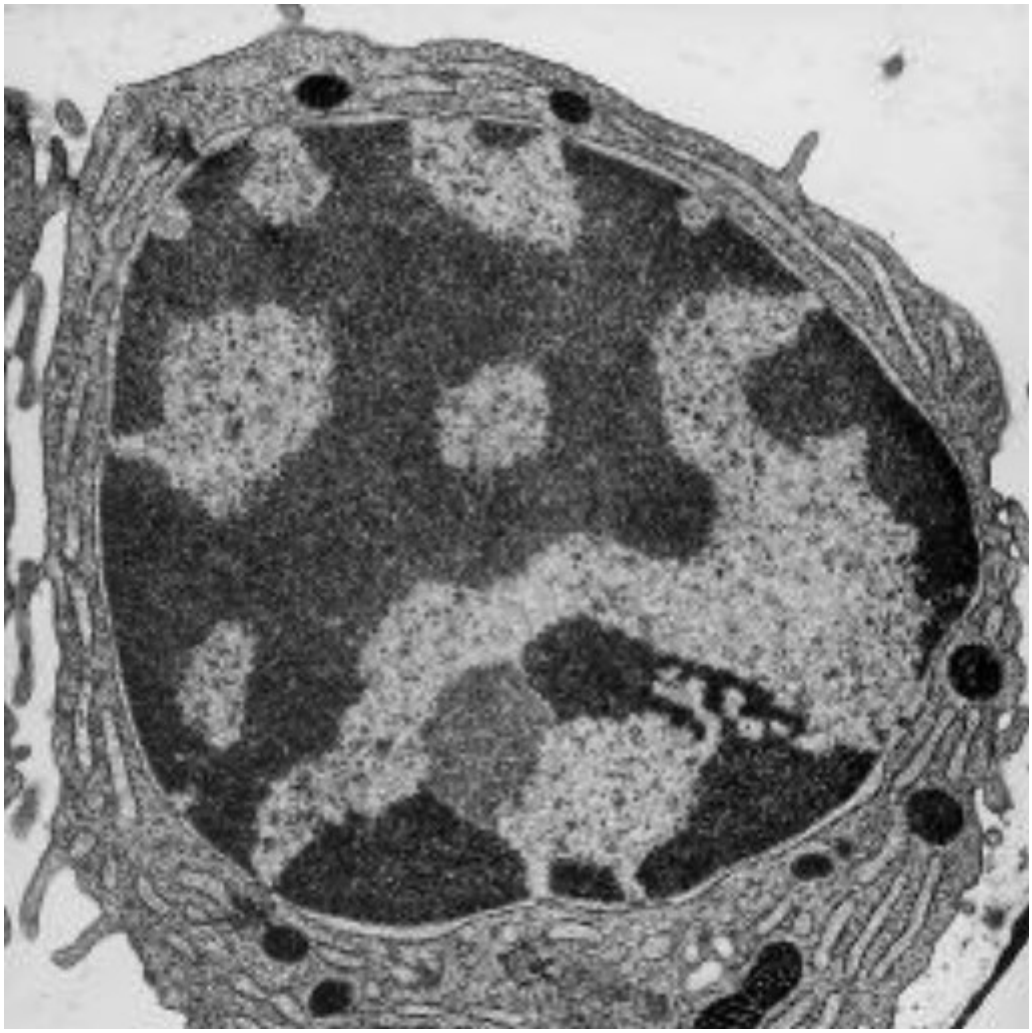

Figura 40 – Célula Bilinear

iv) Bicubic

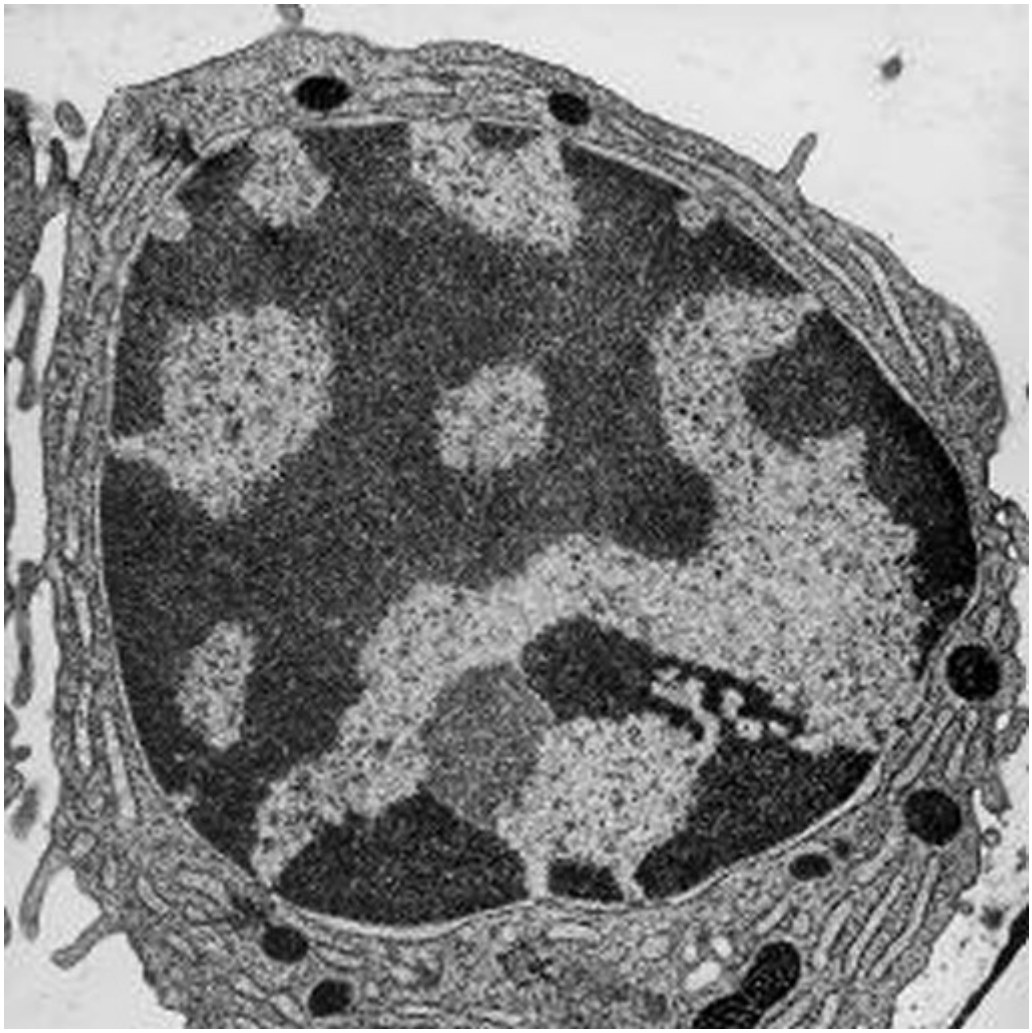

Figura 41 – Célula Bicubic

v) B-Spline

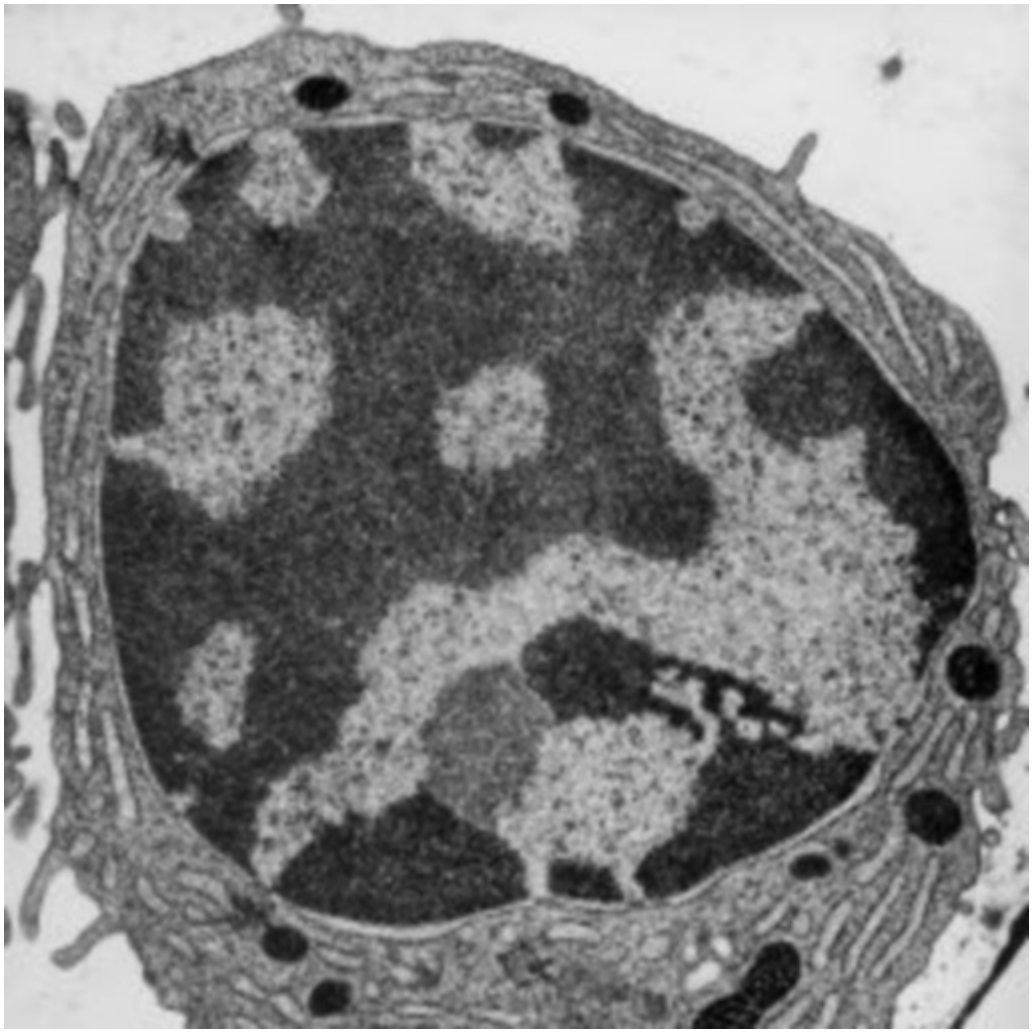

Figura 42 – Célula B-Spline

vi) Lanczos

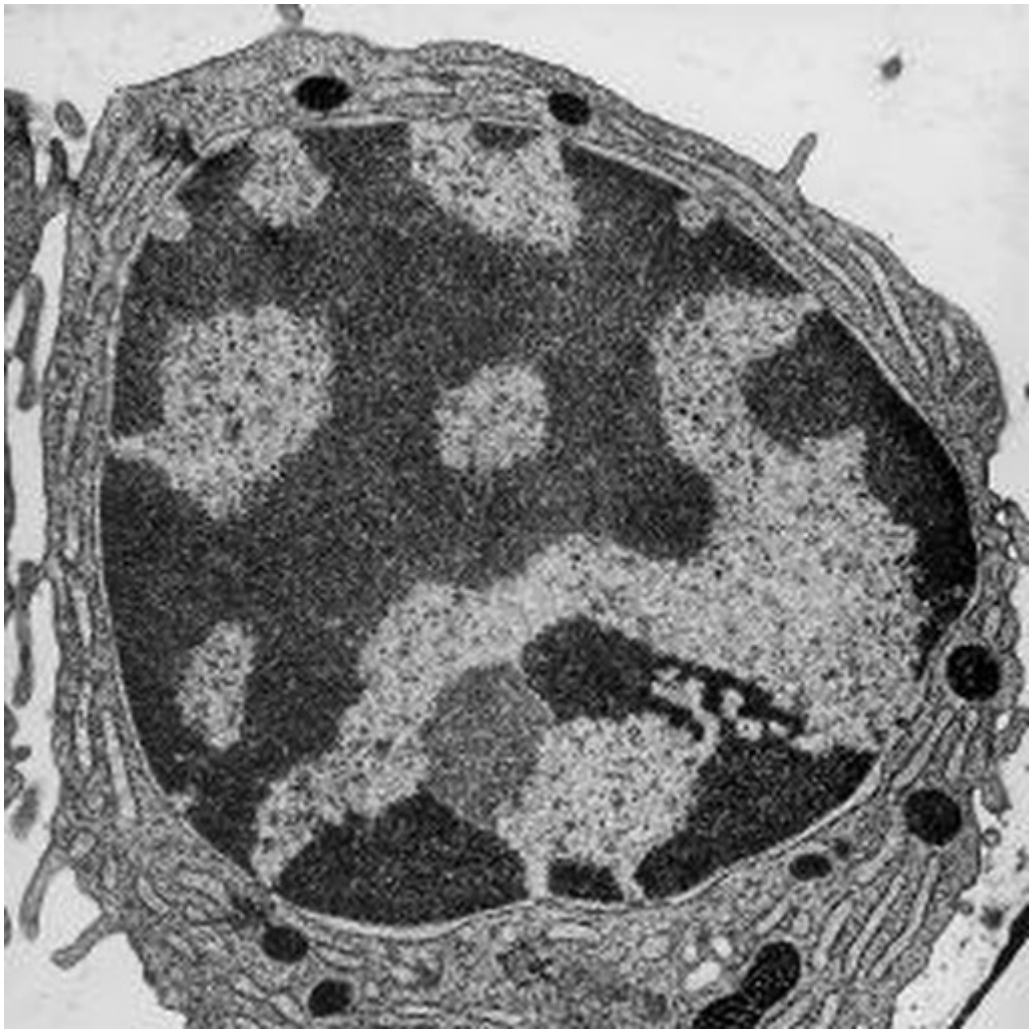

Figura 43 – Célula Lanczos

vii) Método V1

Figura 44 – Célula V1

viii) **Método V2**

Figura 45 – Célula V2

ix) Método V3

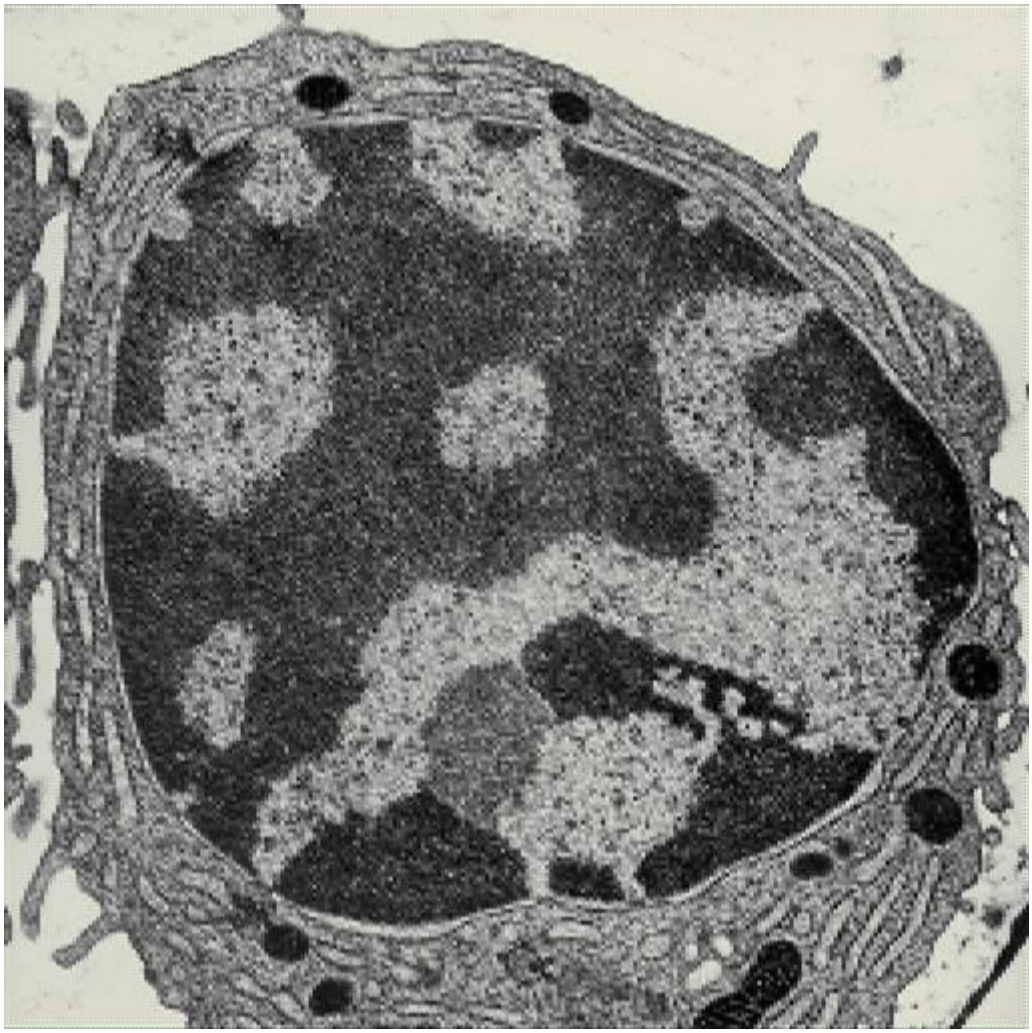

Figura 46 – Célula V3

b) Fagocitose

i) <u>Original</u>

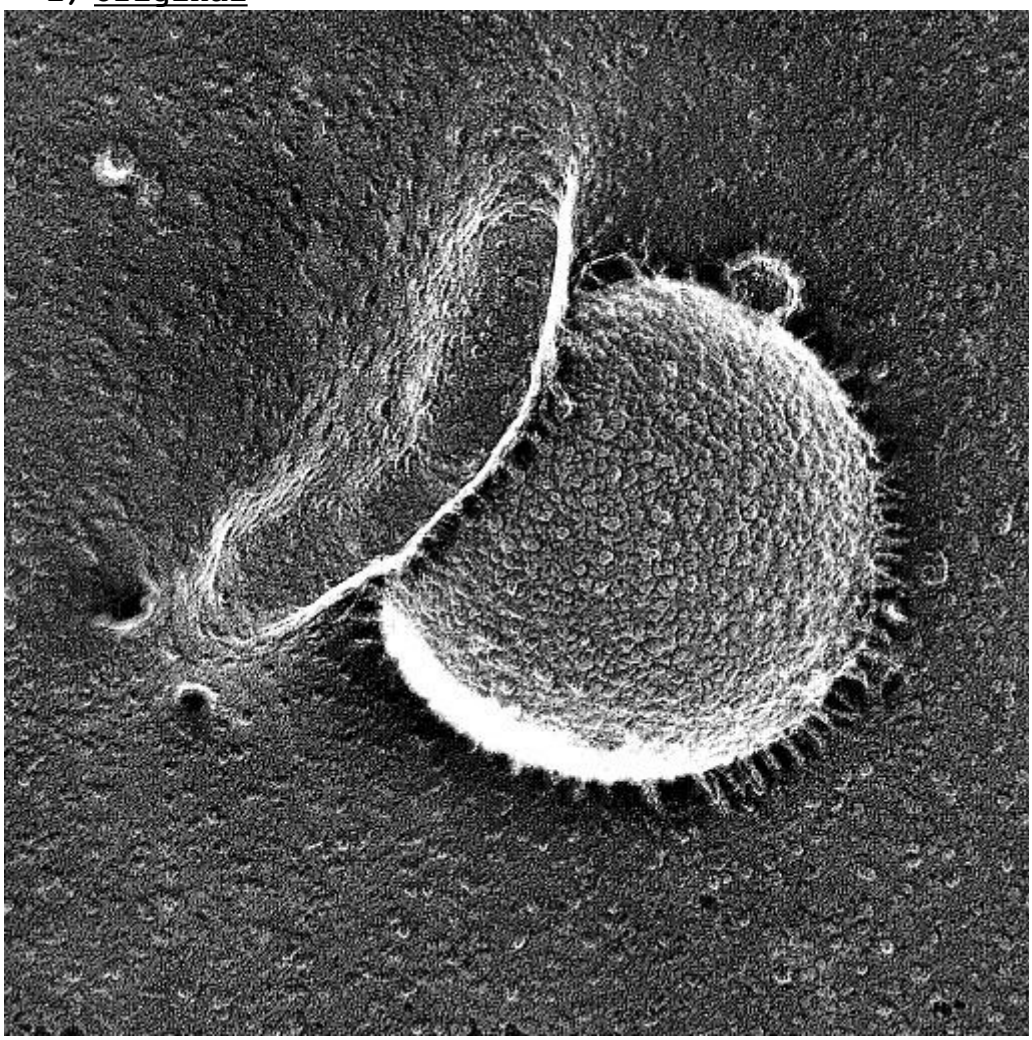

Figura 47 – Fagocitose Original

ii) <u>Reduzida</u>

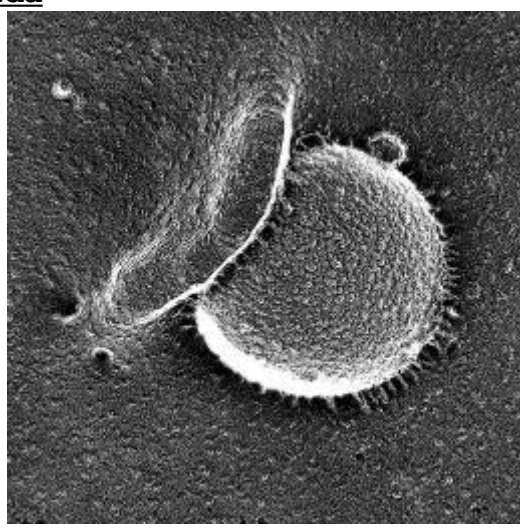

Figura 48 – Fagocitose Reduzida

iii) Bilinear

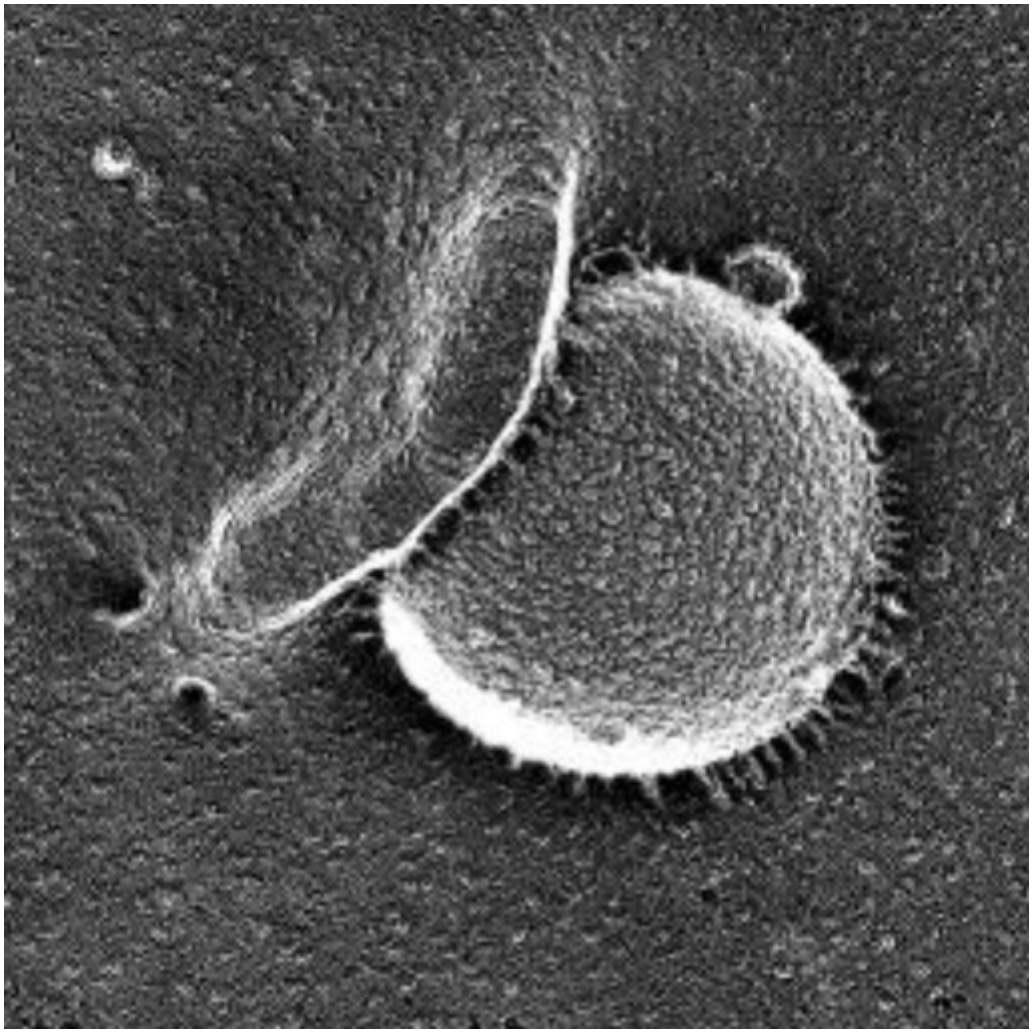

Figura 49 – Fagocitose Bilinear

iv) Bicubic

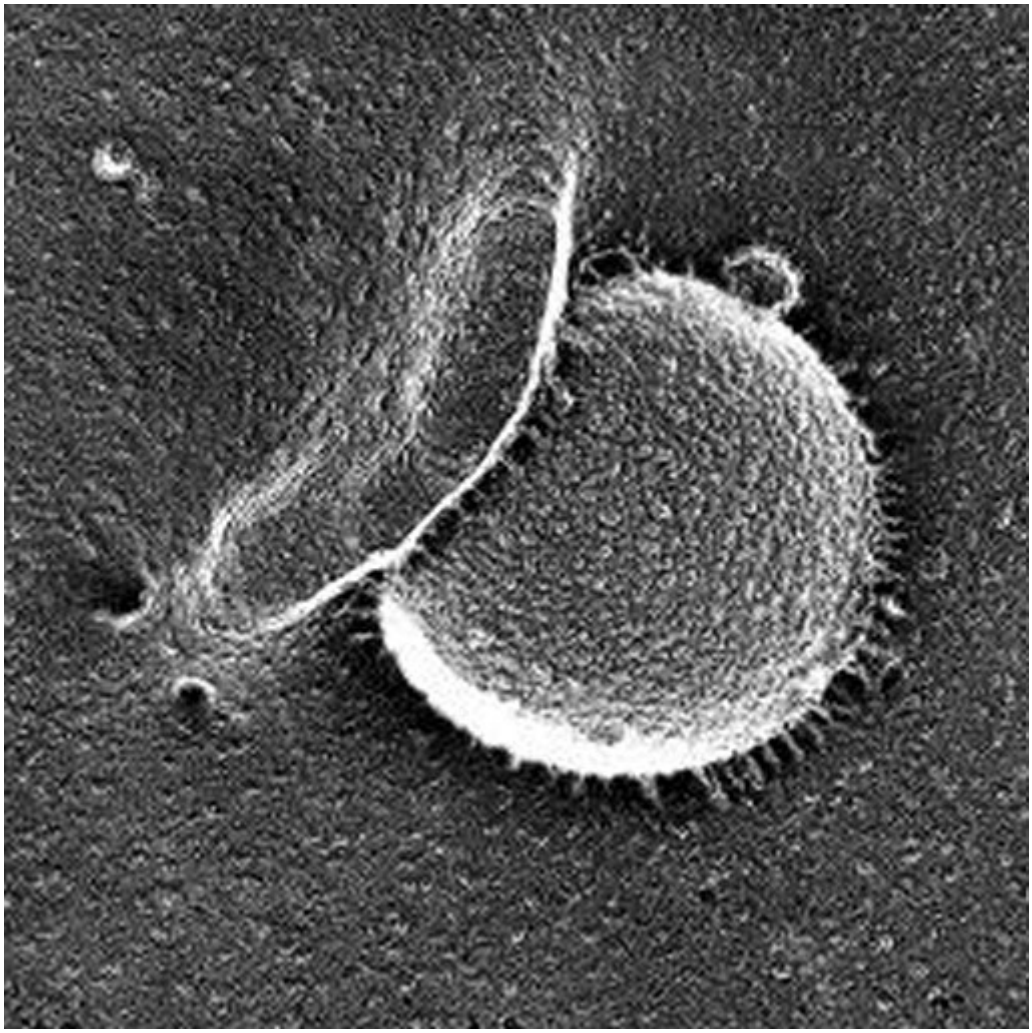

Figura 50 – Fagocitose Bicubic

v) B-Spline

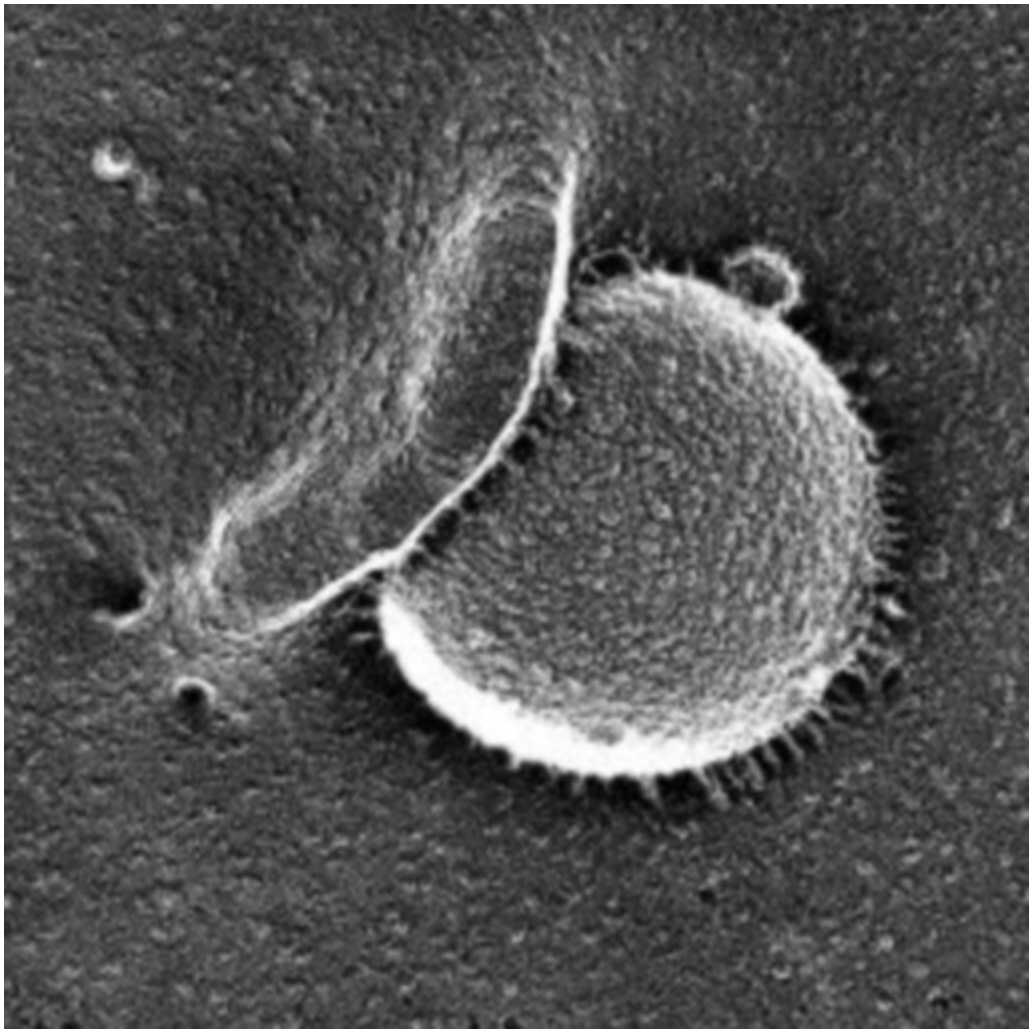

Figura 51 – Fagocitose B-Spline

vi) Lanczos

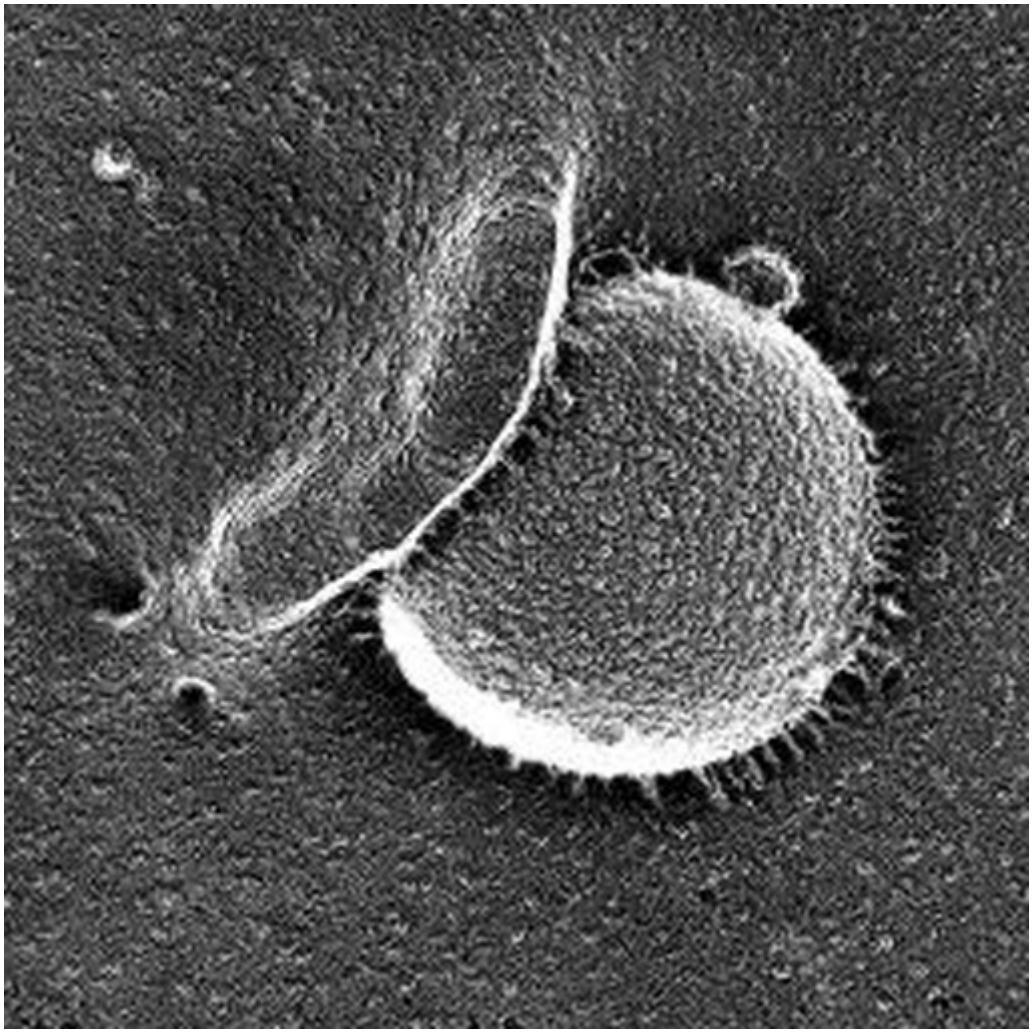

Figura 52 – Fagocitose Lanczos

vii) Método V1

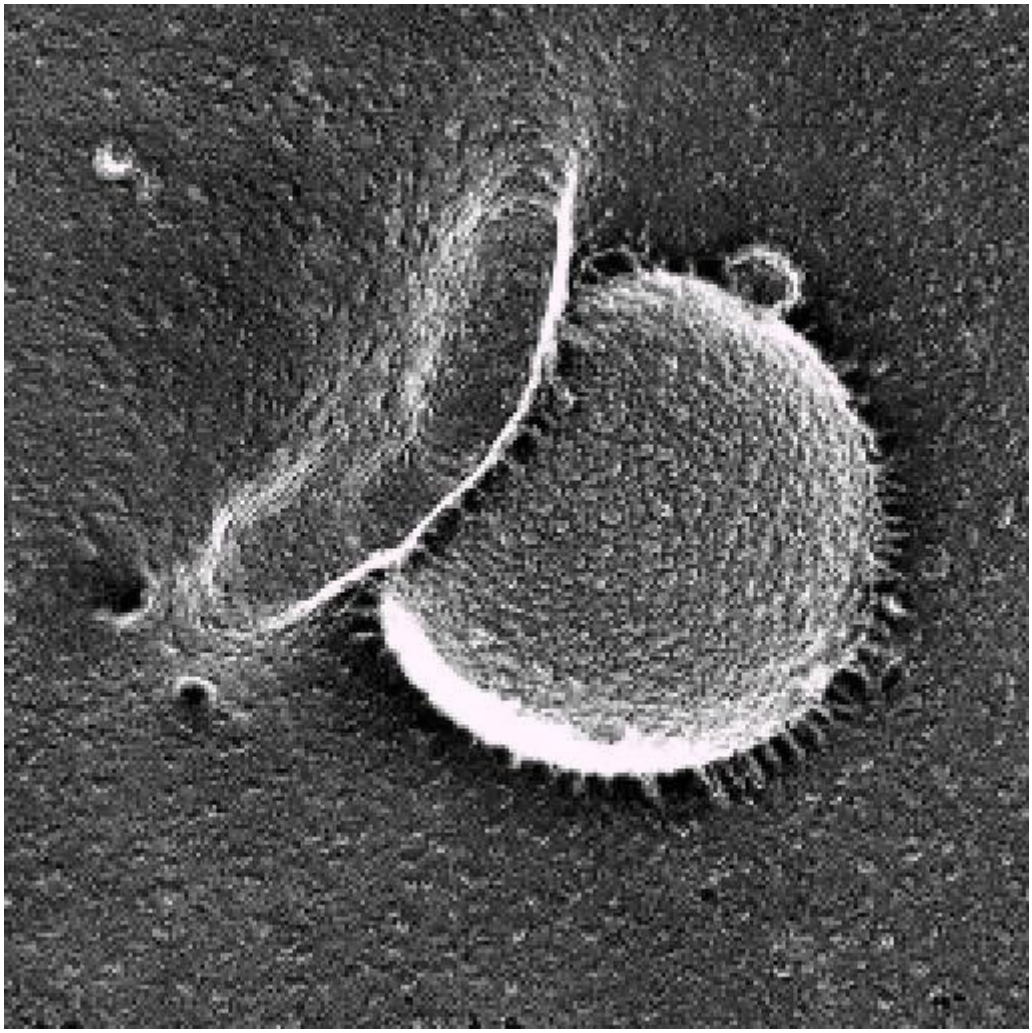

Figura 53 – Fagocitose V1

viii) **Método V2**

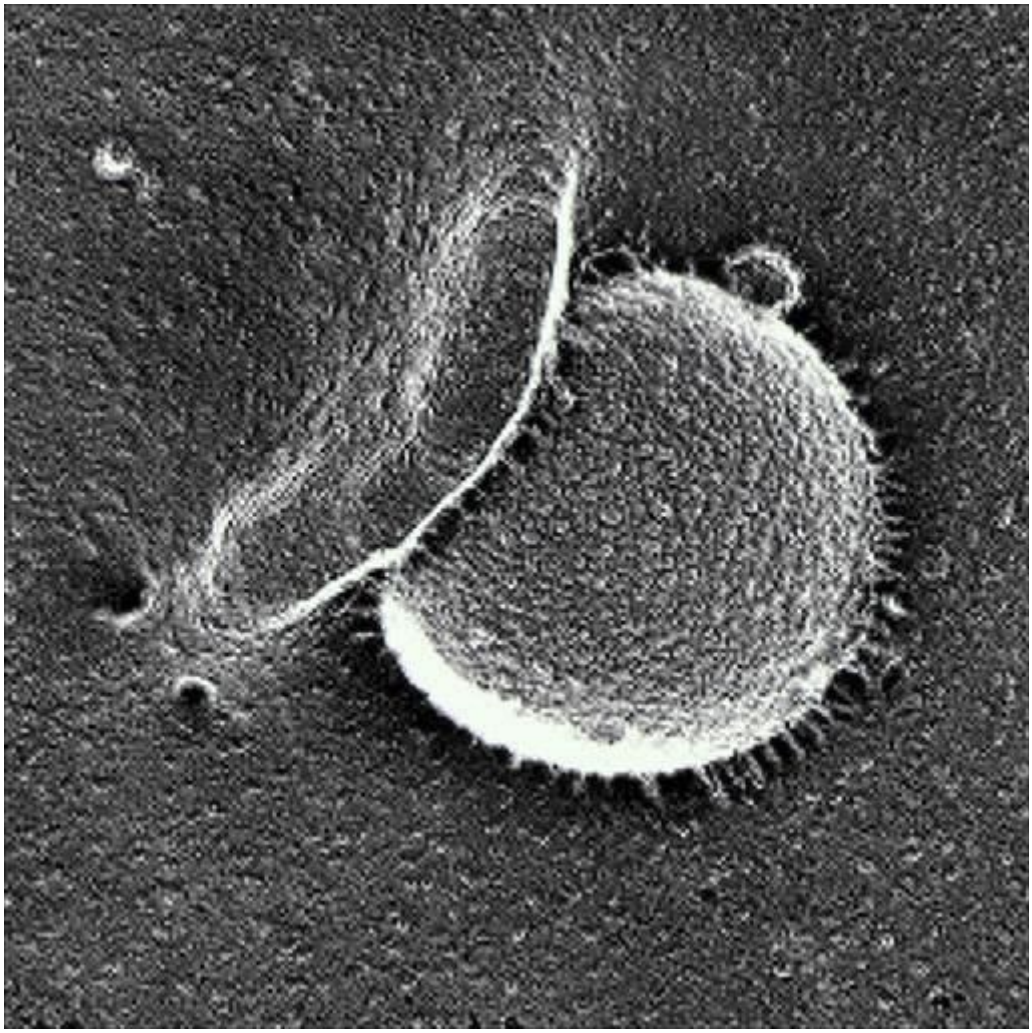

Figura 54 – Fagocitose V2

ix) Método V3

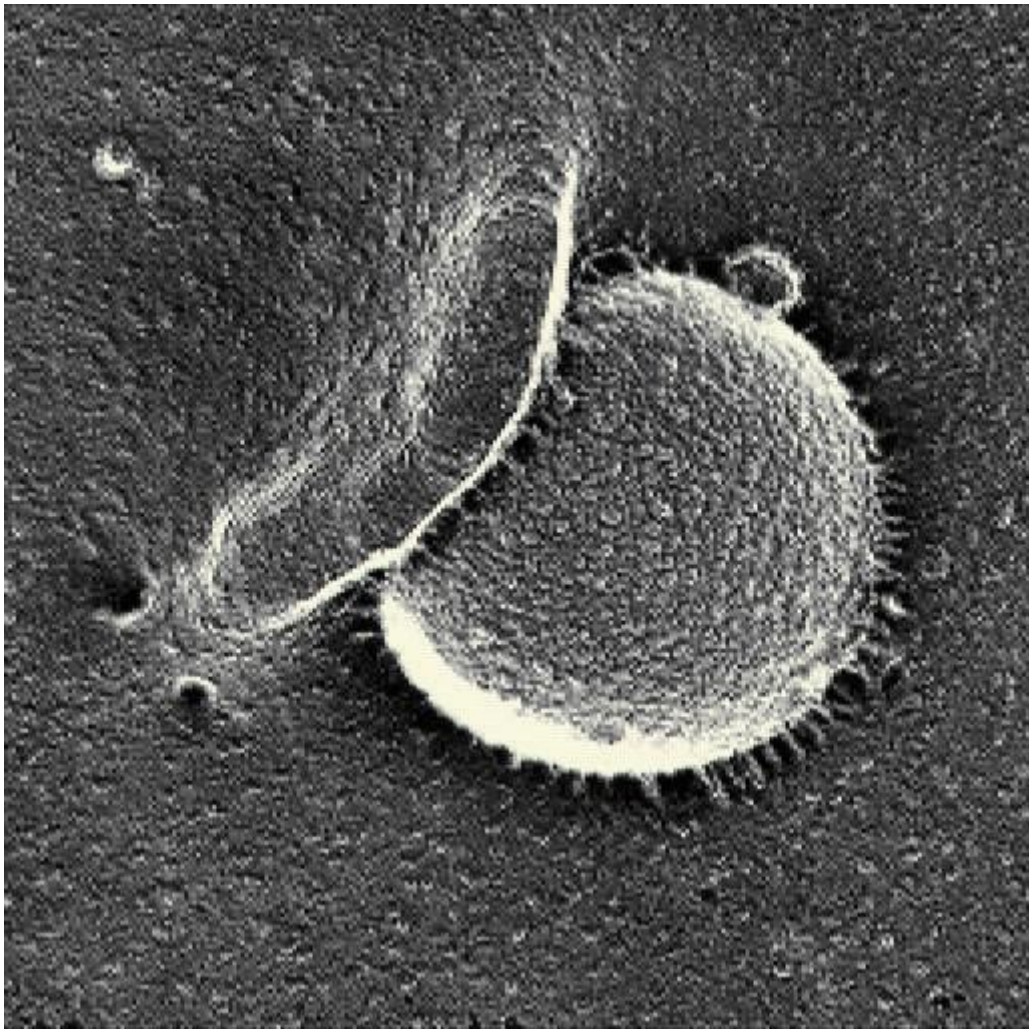

Figura 55 – Fagocitose V3

c) Alvéolos

i) Original

Figura 56 – Alvéolos Original

ii) Reduzida

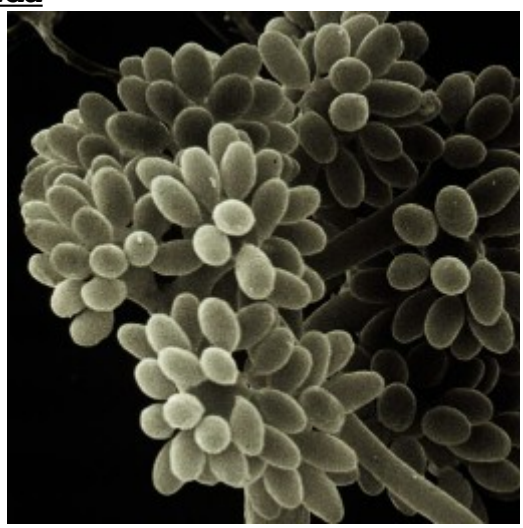

Figura 57 – Alvéolos Reduzida

iii) Bilinear

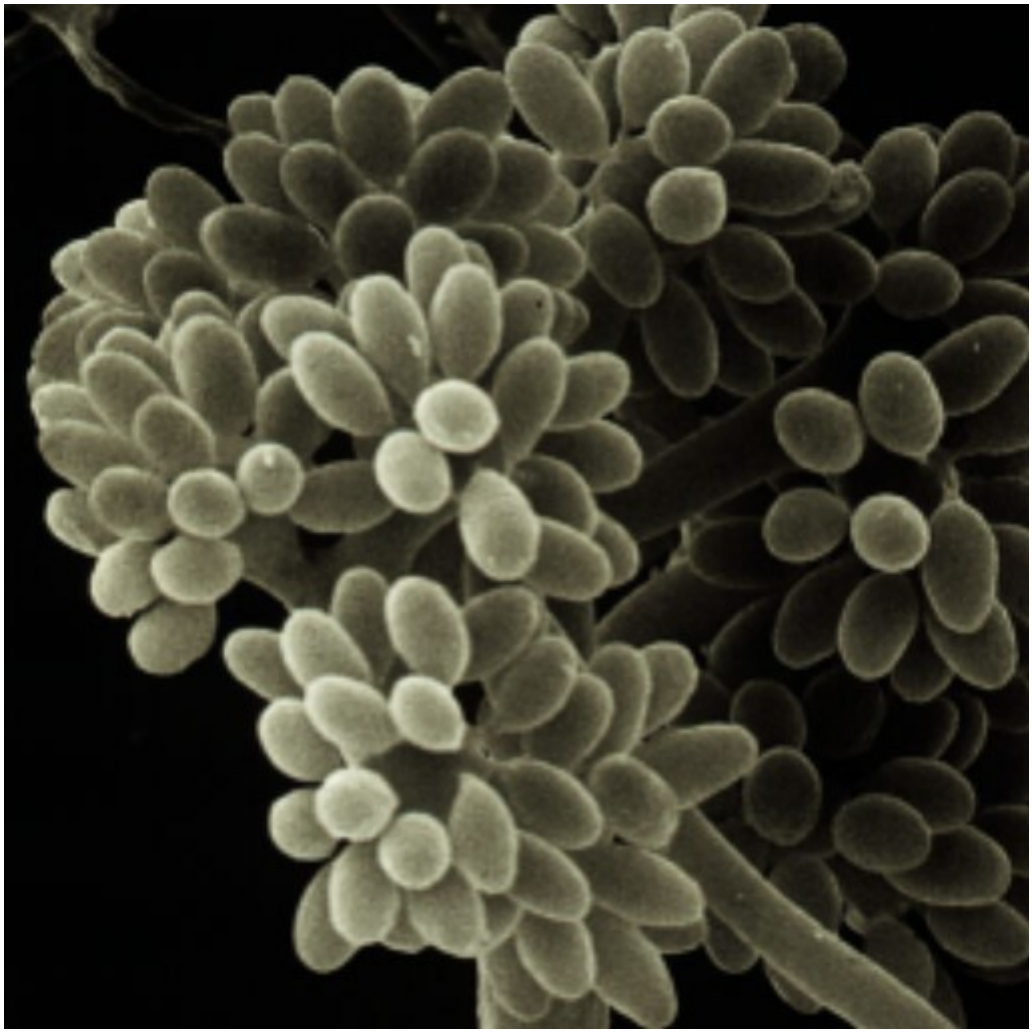

Figura 58 – Alvéolos Bilinear

iv) Bicubic

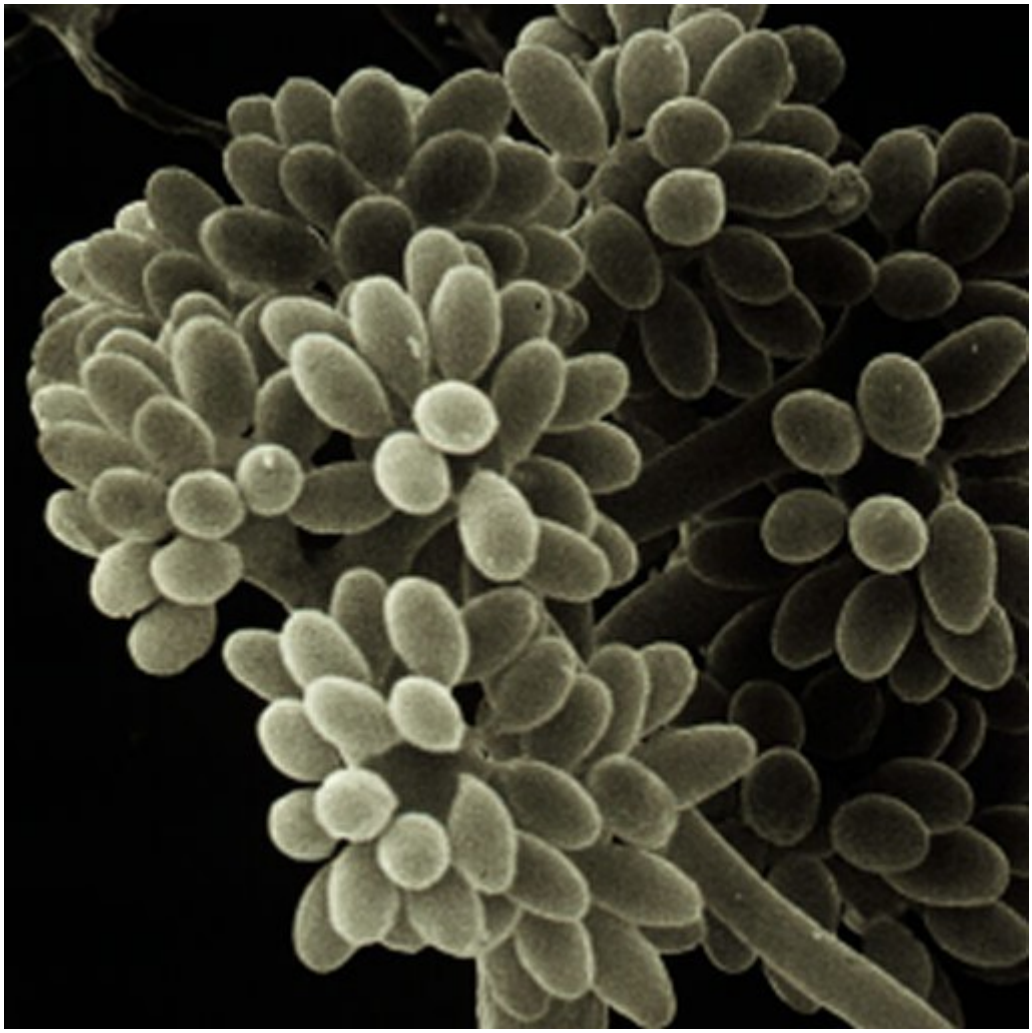

Figura 59 – Alvéolos Bicubic

v) B-Spline

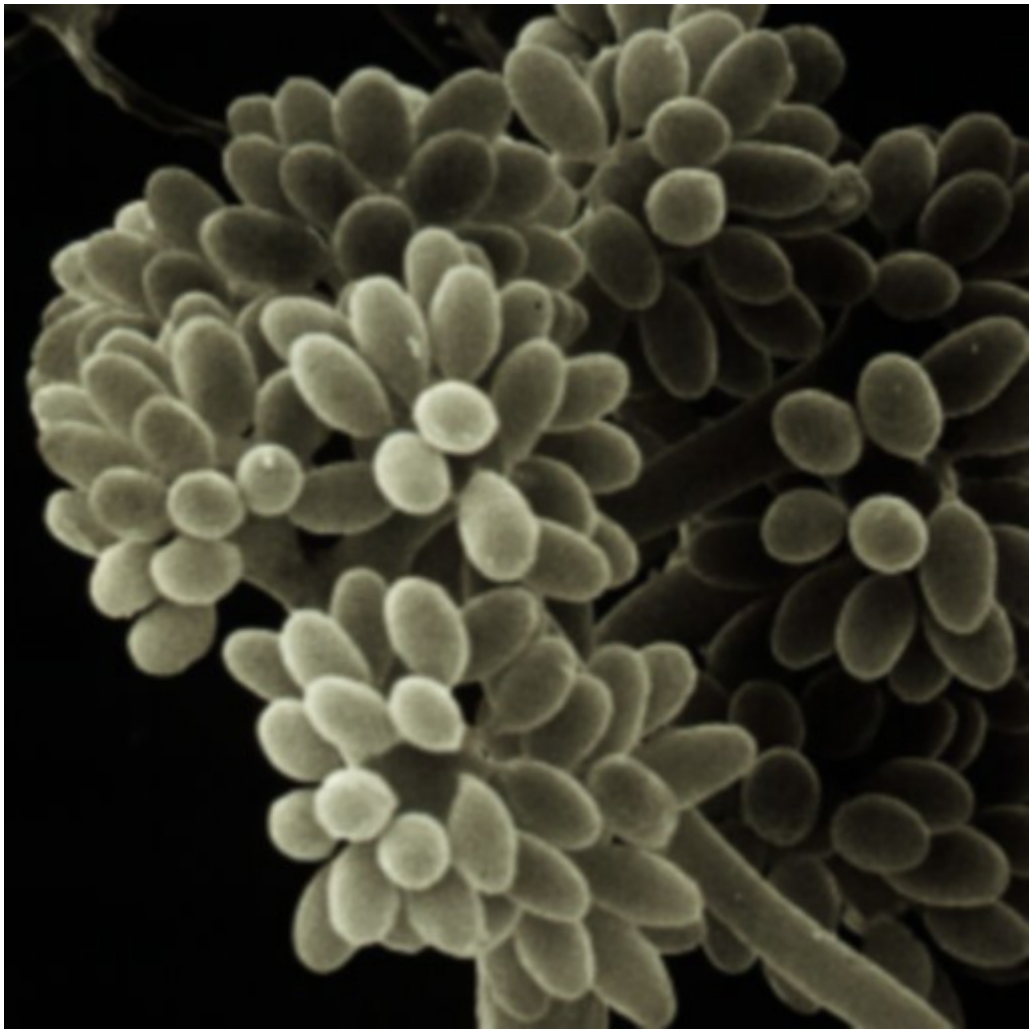

Figura 60 – Alvéolos B-Spline

vi) Lanczos

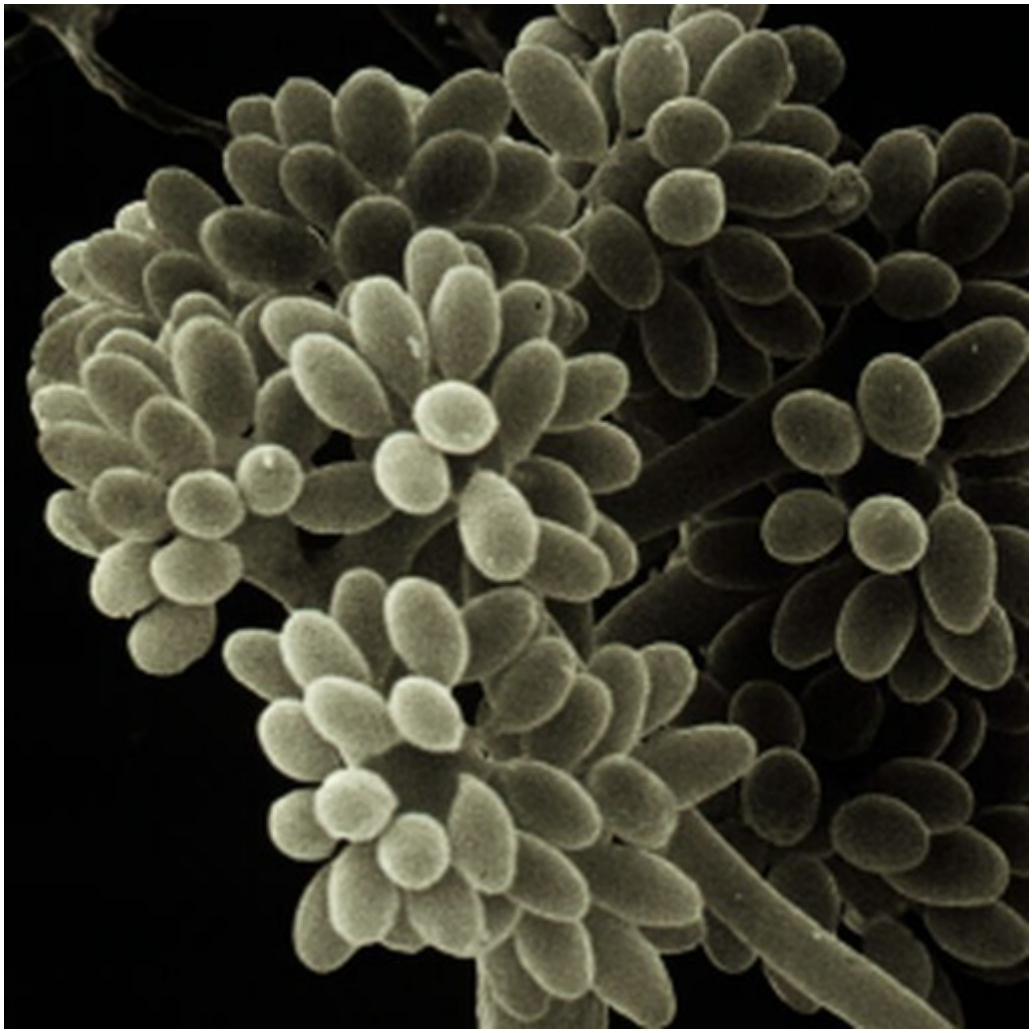

Figura 61 – Alvéolos Lanczos

vii) Método V1

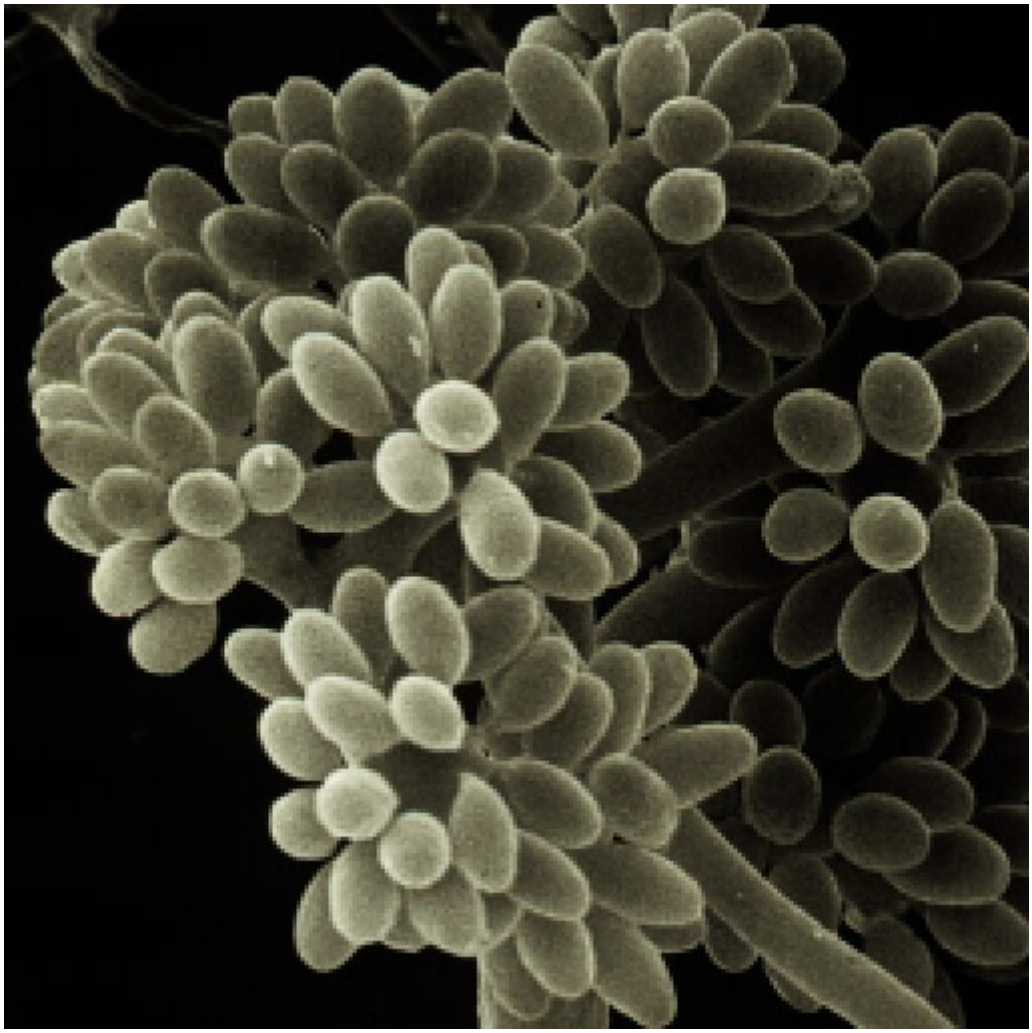

Figura 62 – Alvéolos V1

viii) **Método V2**

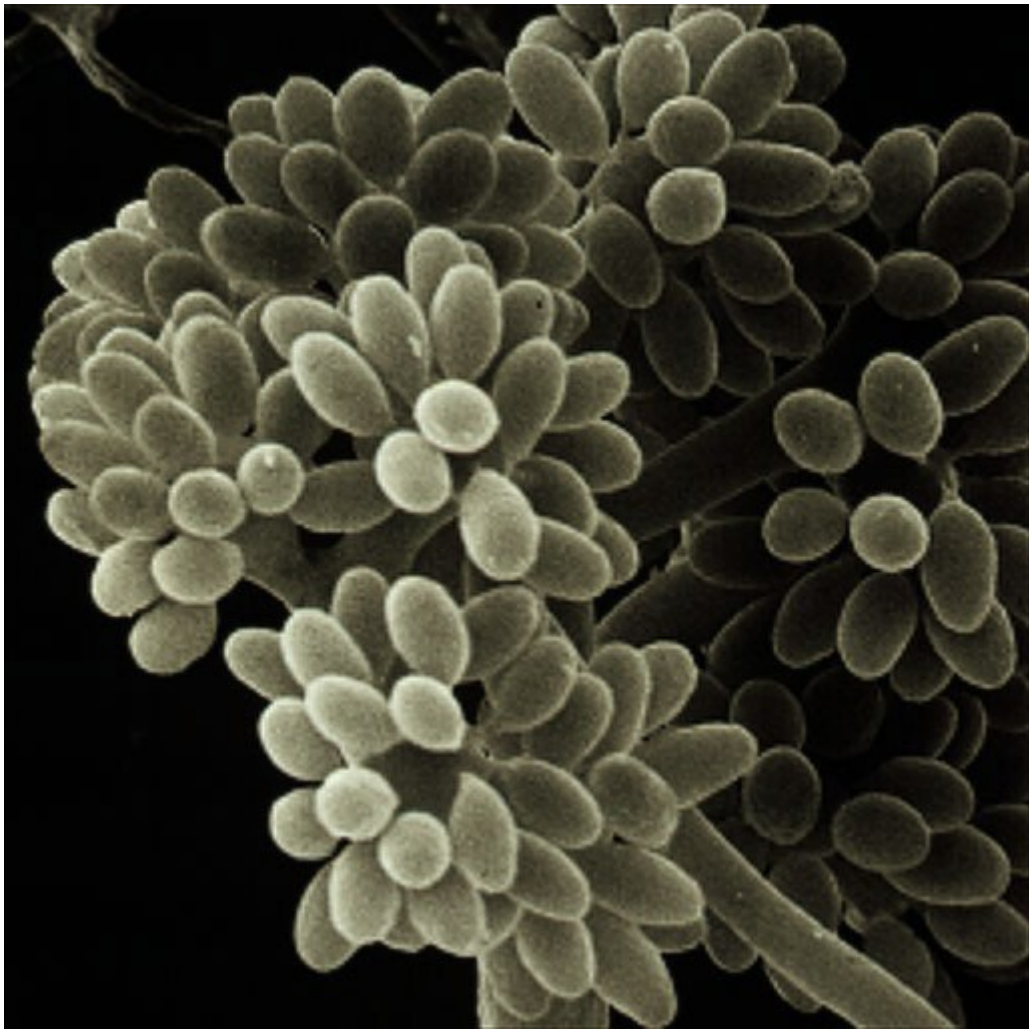

Figura 63 – Alvéolos V2

ix) Método V3

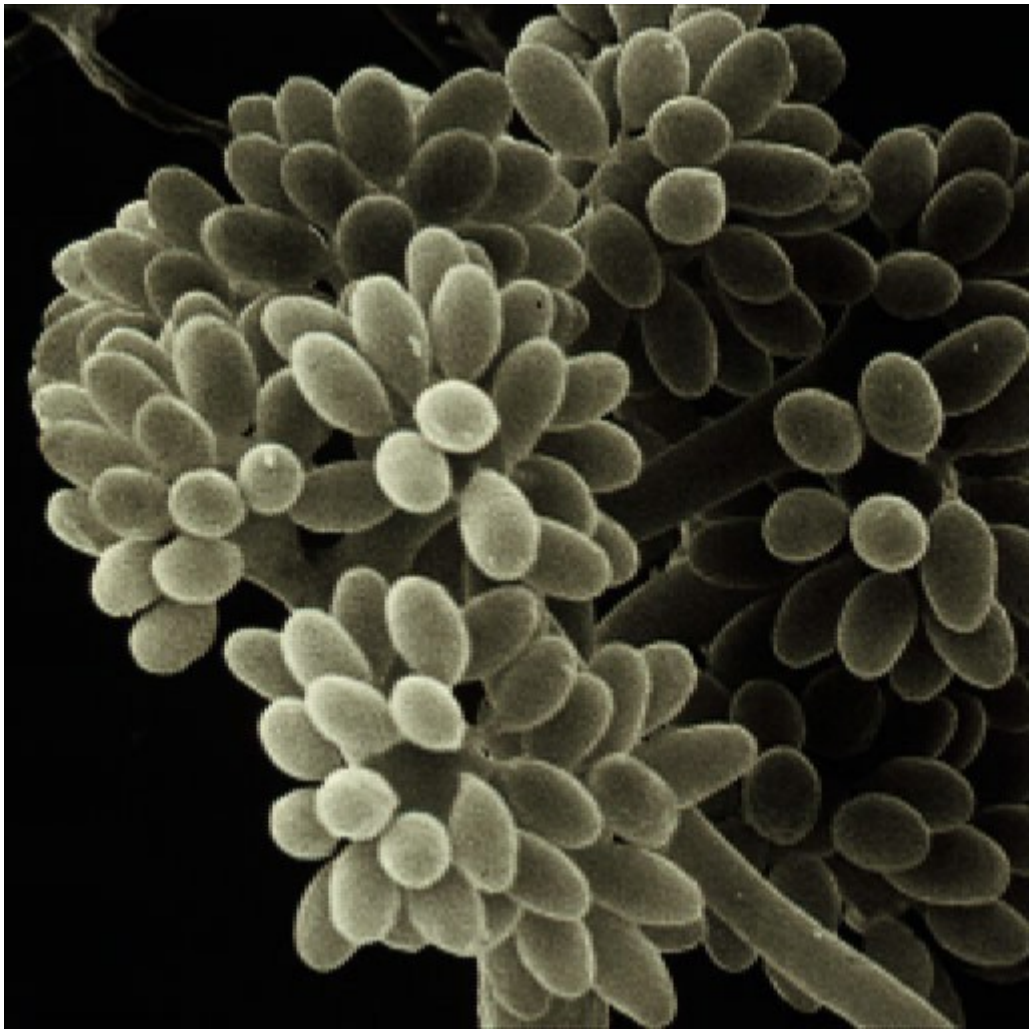

Figura 64 – Alvéolos V3

d) Músculo

i) Original

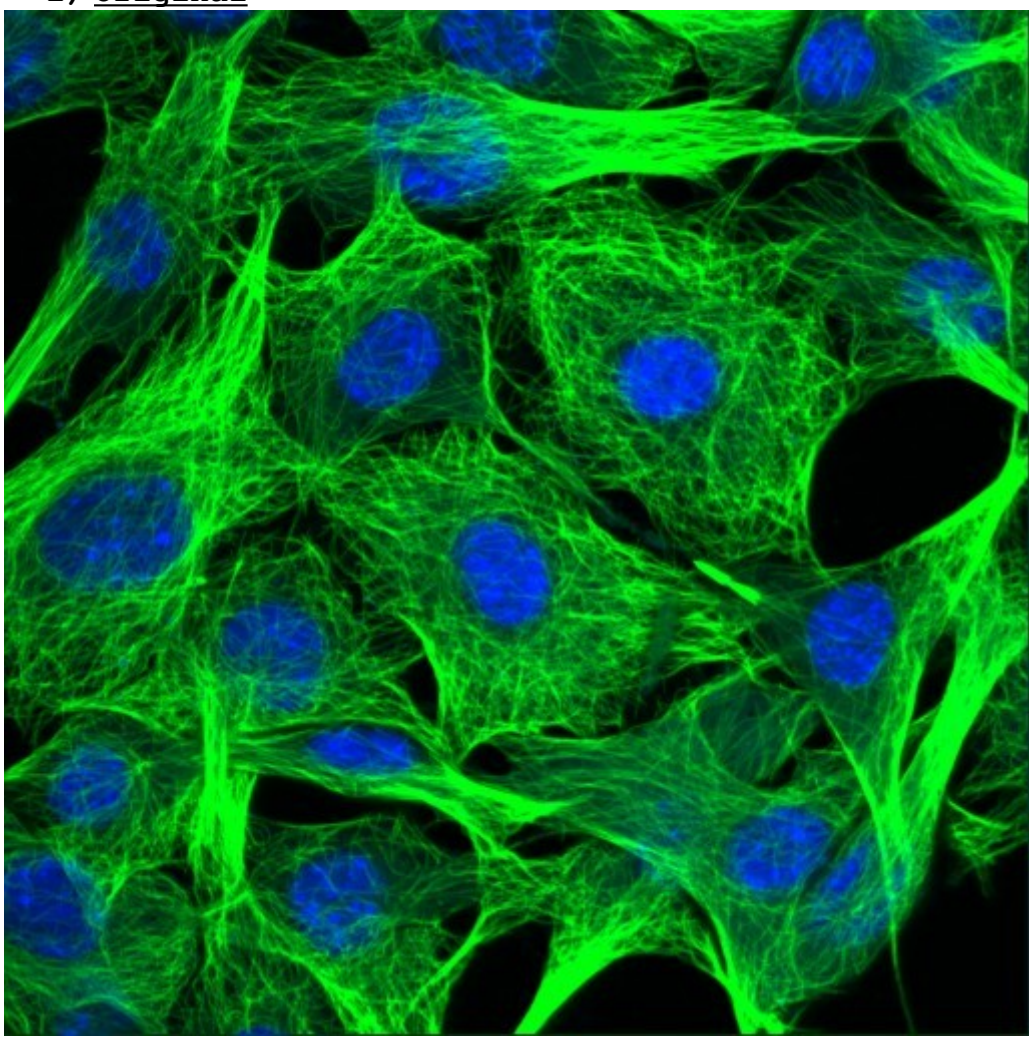

Figura 65 – Músculos Original

ii) Reduzida

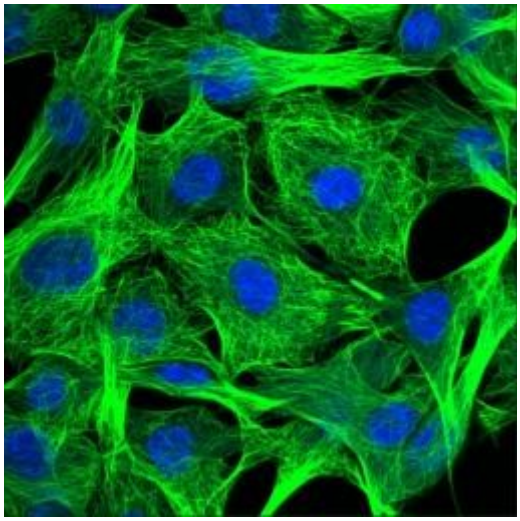

Figura 66 – Músculos Reduzida

iii) Bilinear

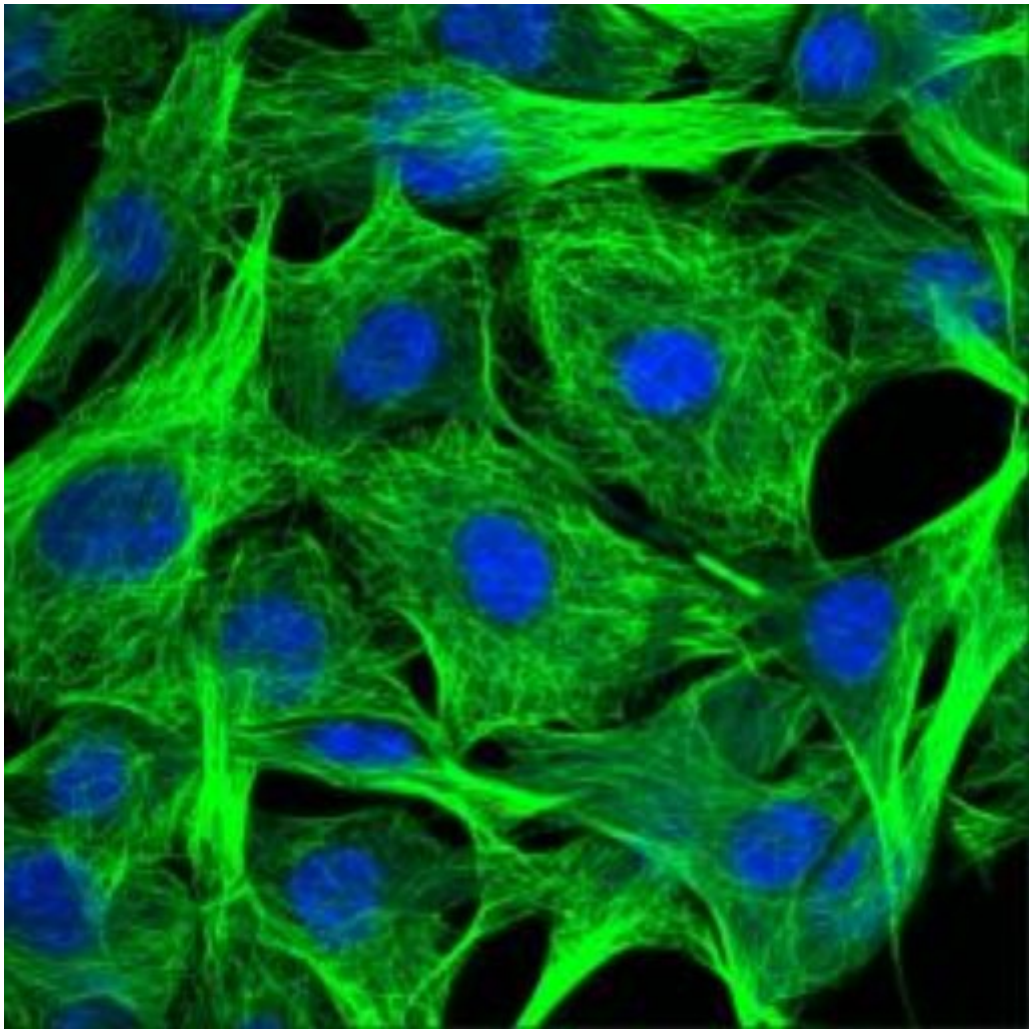

Figura 67 – Músculos Bilinear

iv) Bicubic

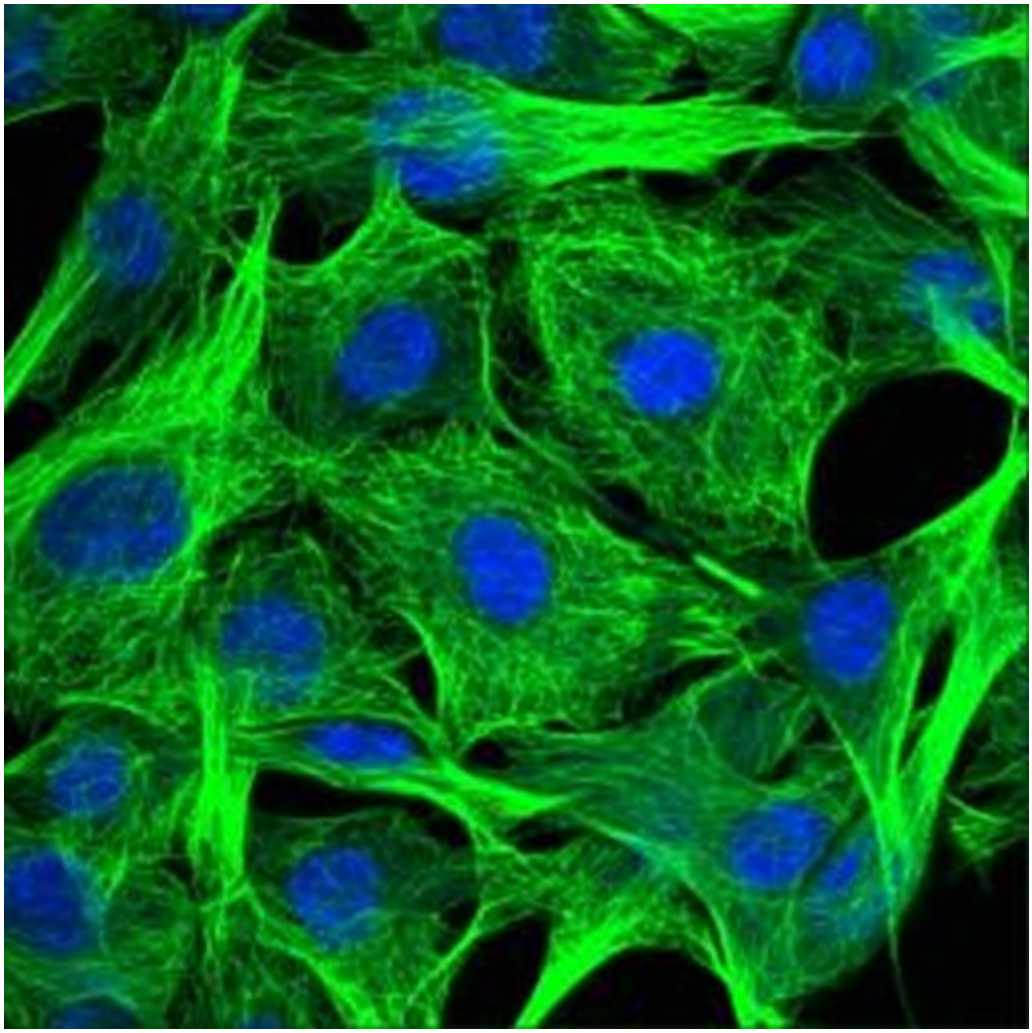

Figura 68 – Músculos Bicubic

v) B-Spline

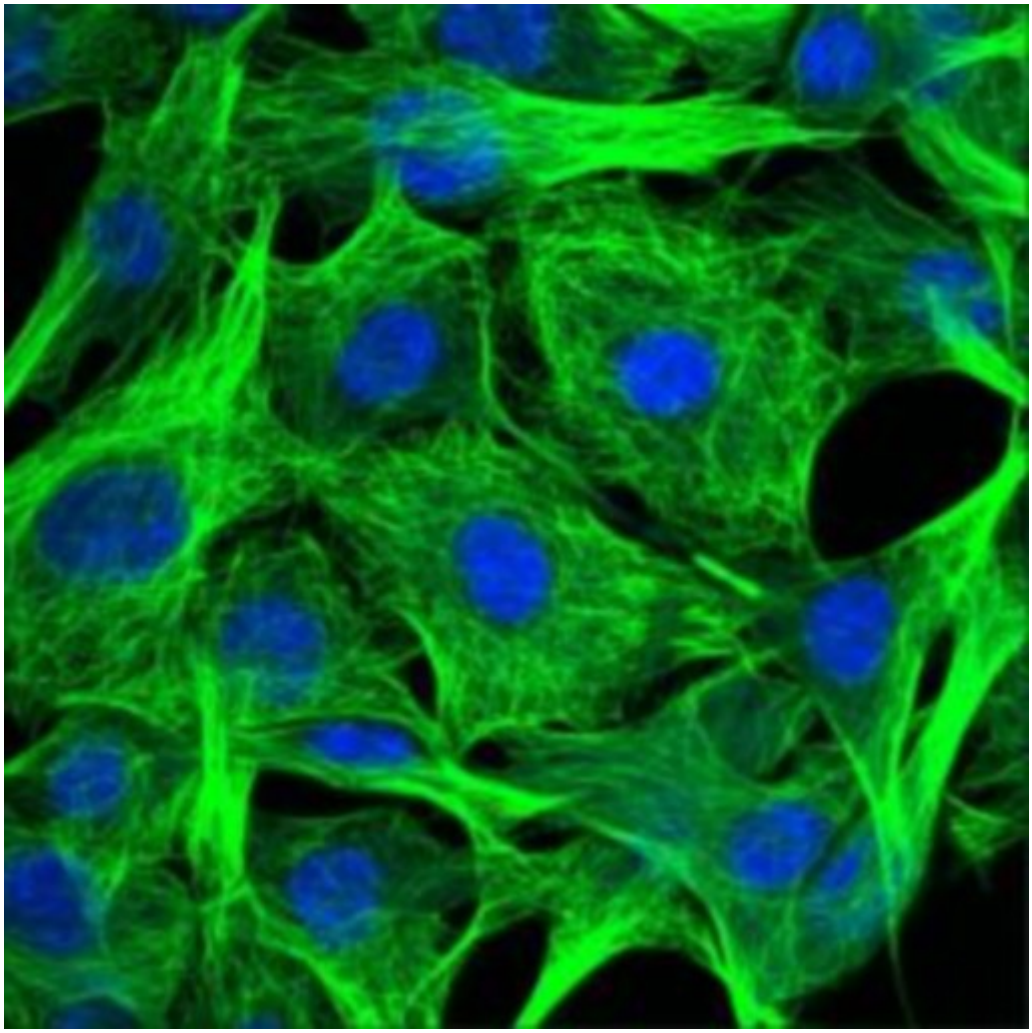

Figura 69 – Músculos B-Spline

vi) Lanczos

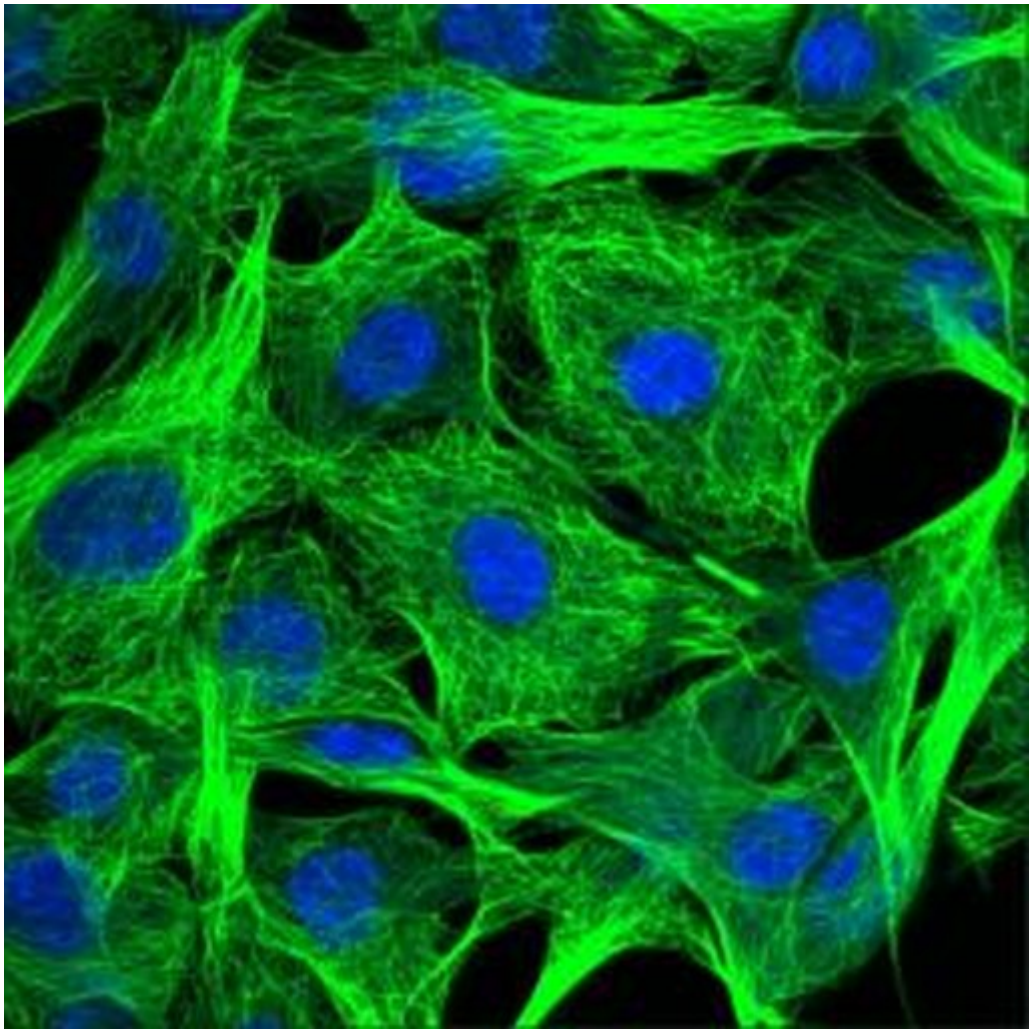

Figura 70 – Músculos Lanczos

vii) Método V1

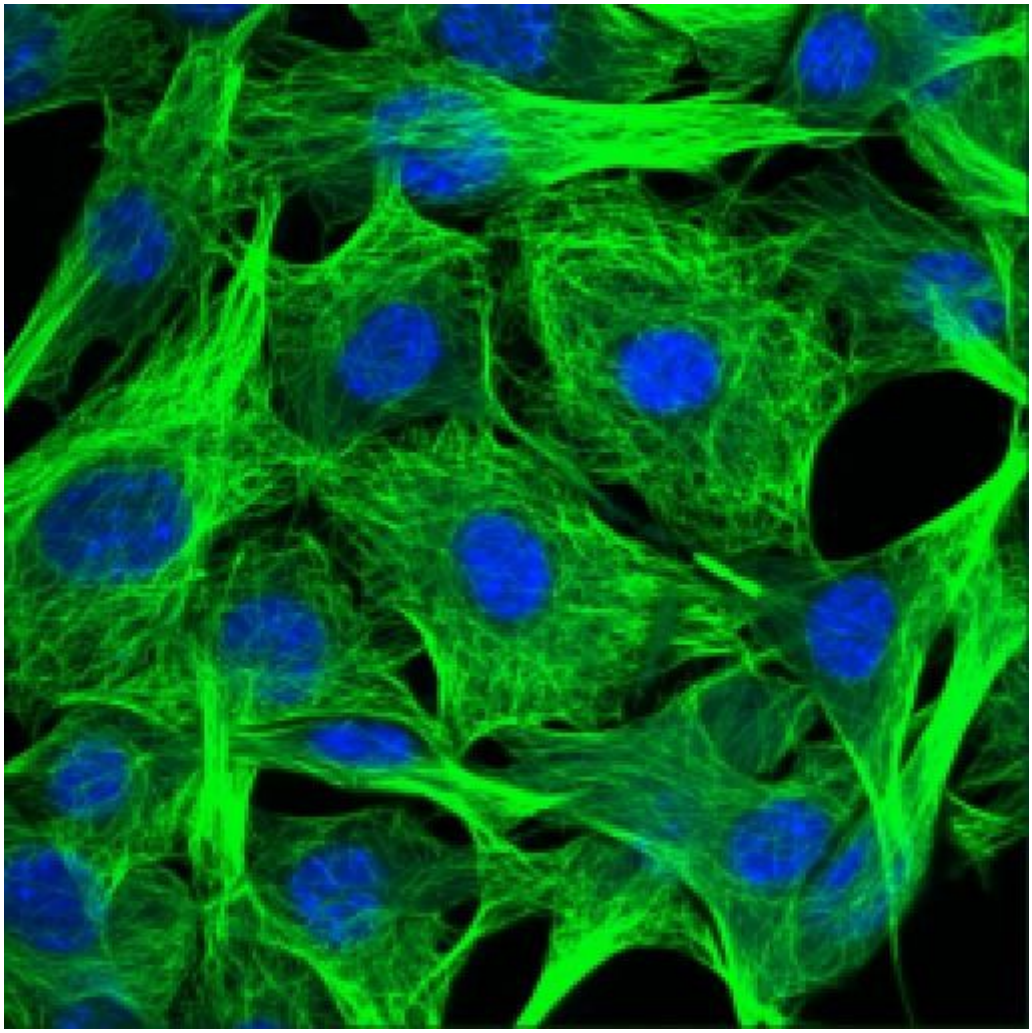

Figura 71 – Músculos V1

viii) **Método V2**

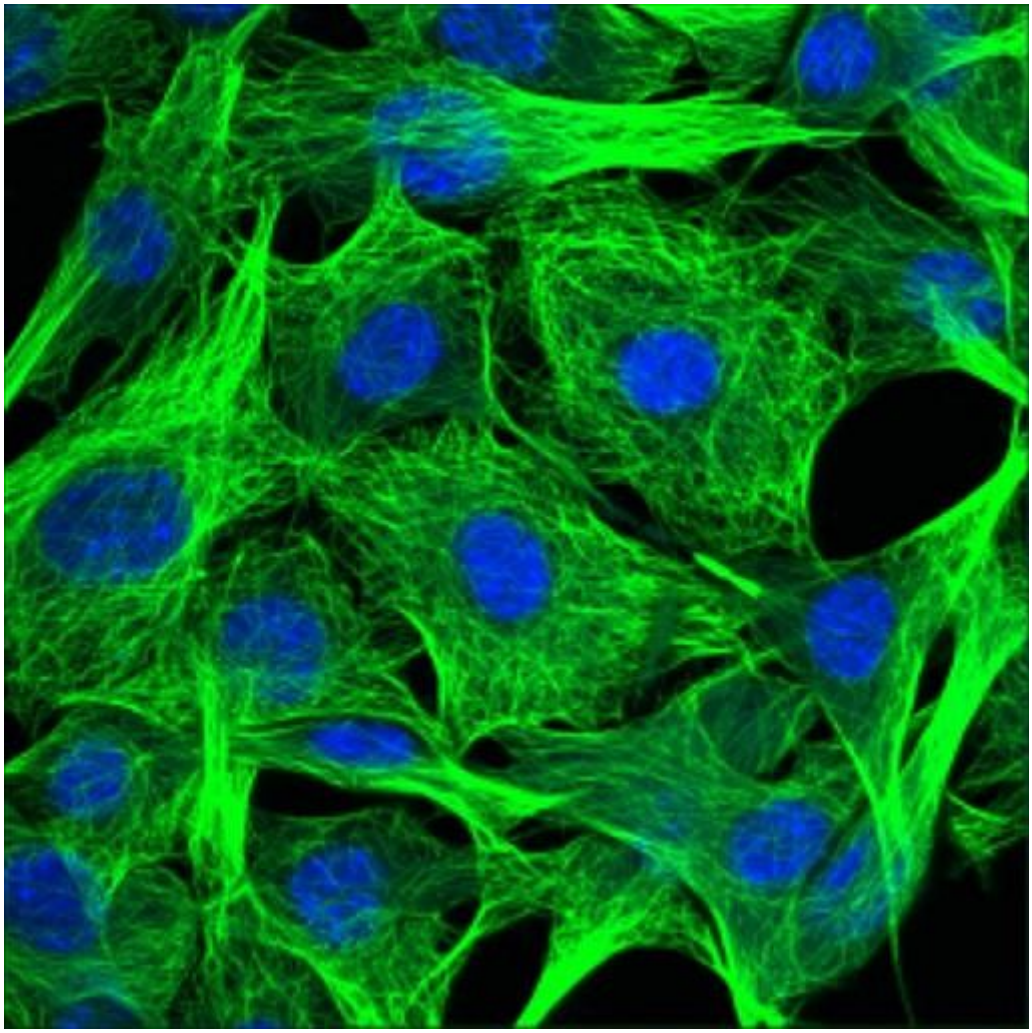

Figura 72 – Músculos V2

ix) Método V3

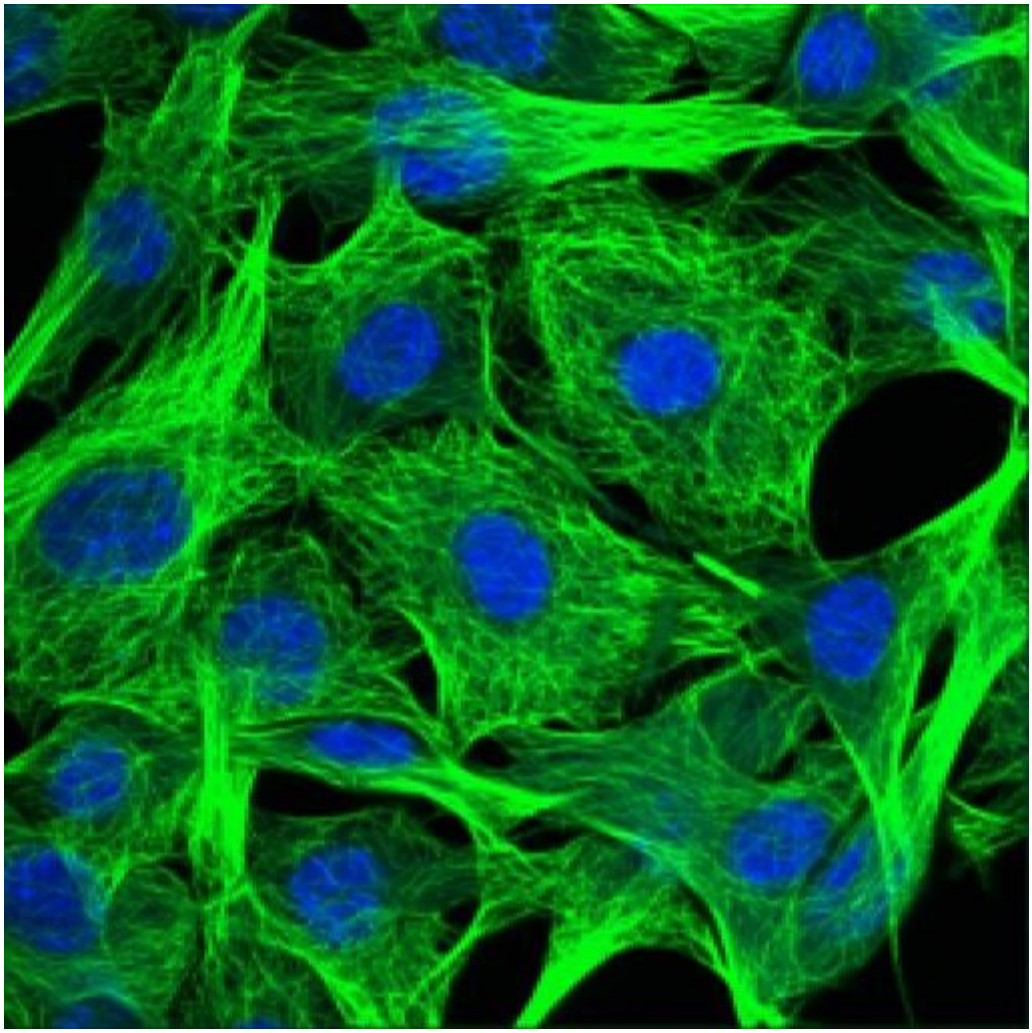

Figura 73 – Músculos V3

e) Barco

i) <u>Original</u>

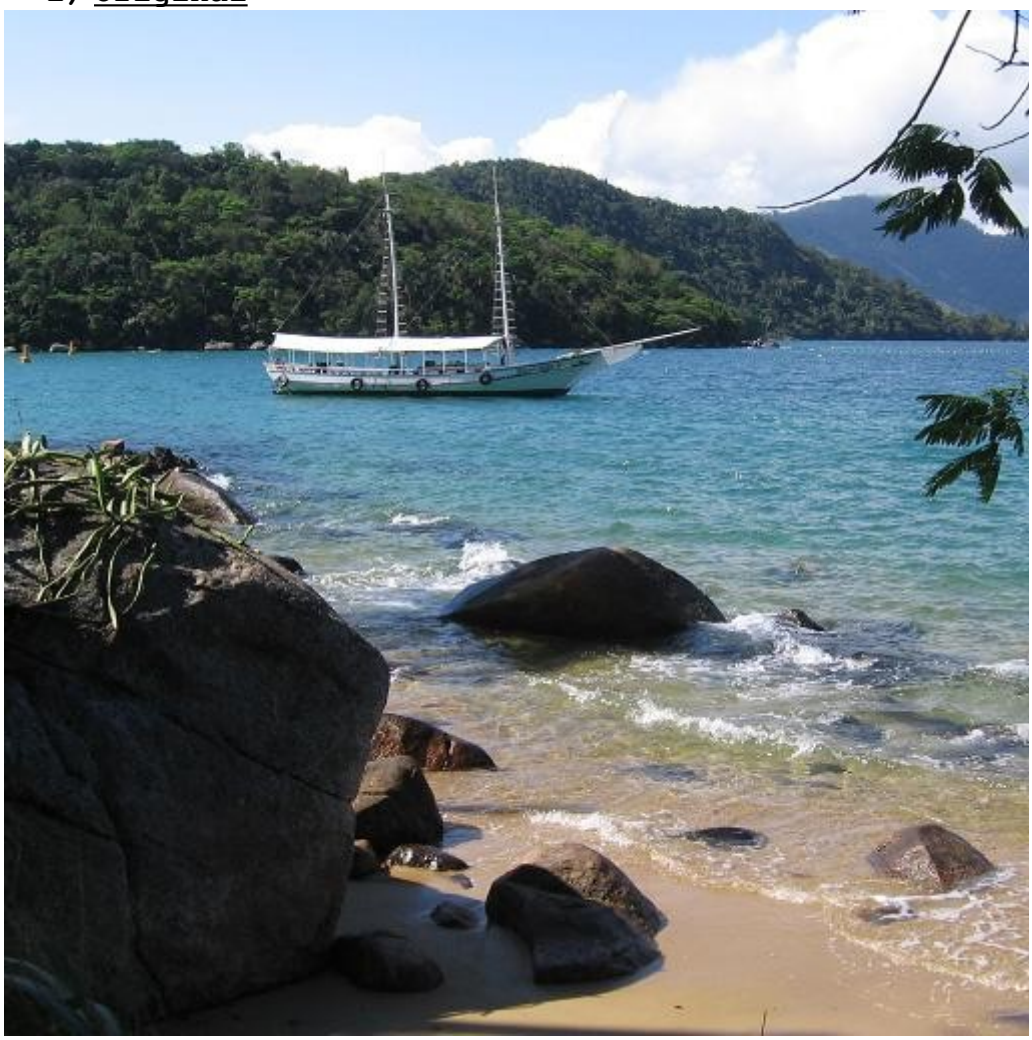

Figura 74 – Barco Original

ii) <u>Reduzida</u>

Figura 75 – Barco Reduzida

iii) Bilinear

Figura 76 – Barco Bilinear

iv) Bicubic

Figura 77 – Barco Bicubic

v) B-Spline

Figura 78 – Barco B-Spline

vi) Lanczos

Figura 79 – Barco Lanczos

vii) Método V1

Figura 80 – Barco V1

viii) **Método V2**

Figura 81 – Barco V2

ix) Método V3

Figura 82 – Barco V3

f) Peixes

i) Original

Figura 83 – Peixes Original

ii) Reduzida

Figura 84 – Peixes Reduzida

iii) Bilinear

Figura 85 – Peixes Bilinear

iv) Bicubic

Figura 86 – Peixes Bicubic

v) B-Spline

Figura 87 – Peixes B-Spline

vi) Lanczos

Figura 88 – Peixes Lanczos

vii) Método V1

Figura 89 – Peixes V1

viii) **Método V2**

Figura 90 – Peixes V2

ix) Método V3

Figura 91 – Peixes V3

g) Maracanã

i) <u>Original</u>

Figura 92 – Maracanã Original

ii) <u>Reduzida</u>

Figura 93 – Maracanã Reduzida

iii) Bilinear

Figura 94 – Maracanã Bilinear

iv) Bicubic

Figura 95 – Maracanã Bicubic

v) B-Spline

Figura 96 – Maracanã B-Spline

vi) Lanczos

Figura 97 – Maracanã Lanczos

vii) Método V1

Figura 98 – Maracanã V1

viii) **Método V2**

Figura 99 – Maracanã V2

ix) Método V3

Figura 100 – Maracanã V3

h) Cachoeira

i) <u>Original</u>

Figura 101 – Cachoeira Original

ii) <u>Reduzida</u>

Figura 102 – Cachoeira Reduzida

iii) Bilinear

Figura 103 – Cachoeira Bilinear

iv) Bicubic

Figura 104 – Cachoeira Bicubic

v) B-Spline

Figura 105 – Cachoeira B-Spline

vi) Lanczos

Figura 106 – Cachoeira Lanczos

vii) Método V1

Figura 107 – Cachoeira V1

viii) Método V2

Figura 108 – Cachoeira V2

ix) Método V3

Figura 109 – Cachoeira V3

i) Baía de Guanabara

i) <u>Original</u>

Figura 110 – Guanabara Original

ii) <u>Reduzida</u>

Figura 111 – Guanabara Reduzida

iii) Bilinear

Figura 112 – Guanabara Bilinear

iv) Bicubic

Figura 113 – Guanabara Bicubic

v) B-Spline

Figura 114 – Guanabara B-Spline

vi) Lanczos

Figura 115 – Guanabara Lanczos

vii) Método V1

Figura 116 – Guanabara V1

viii) Método V2

Figura 117 – Guanabara V2

ix) Método V3

Figura 118 – Guanabara V3

j) Rio de Janeiro

i) Original

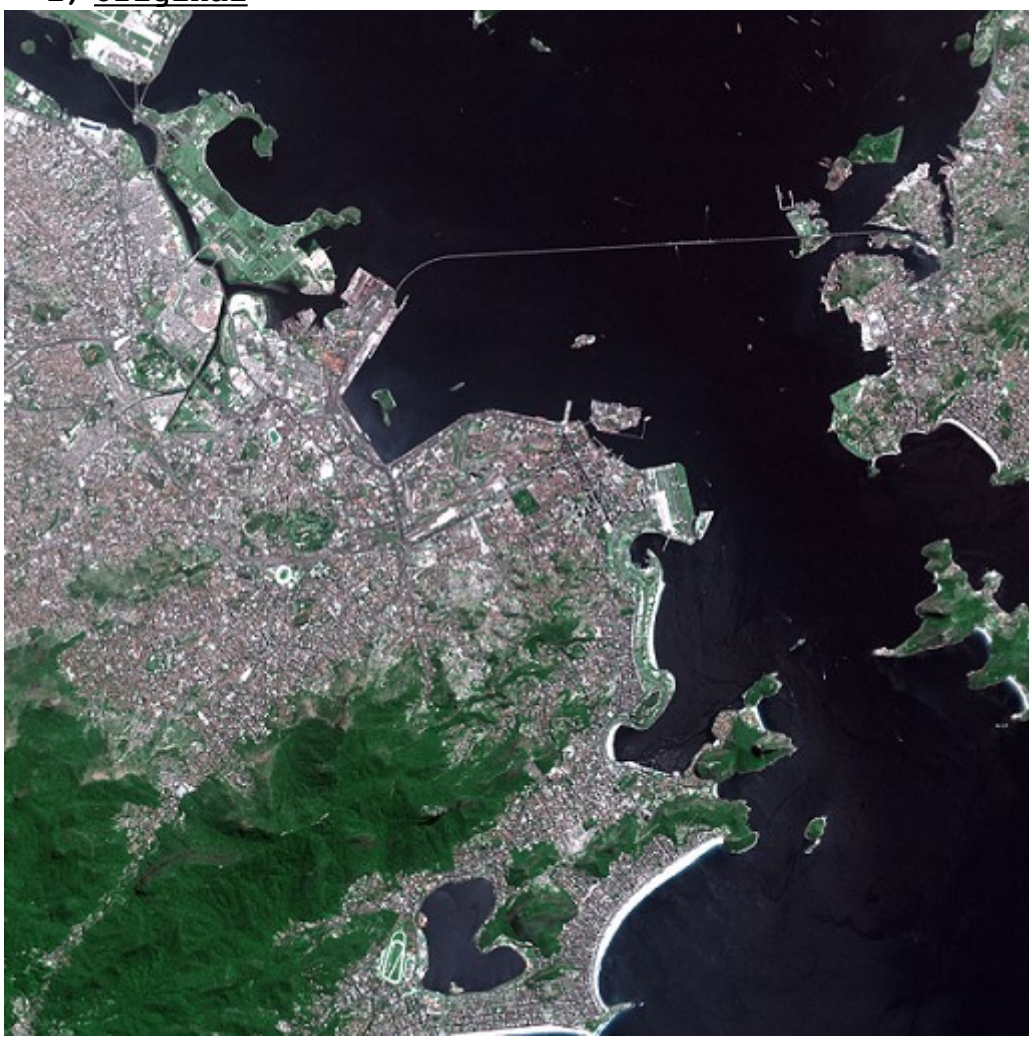

Figura 119 – Rio de Janeiro Original

ii) Reduzida

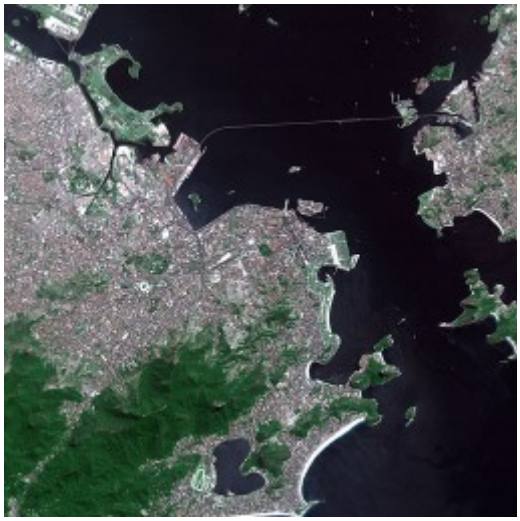

Figura 120 – Rio de Janeiro Reduzida

iii) Bilinear

Figura 121 – Rio de Janeiro Bilinear

iv) Bicubic

Figura 122 – Rio de Janeiro Bicubic

v) B-Spline

Figura 123 – Rio de Janeiro B-Spline

vi) Lanczos

Figura 124 – Rio de Janeiro Lanczos

vii) Método V1

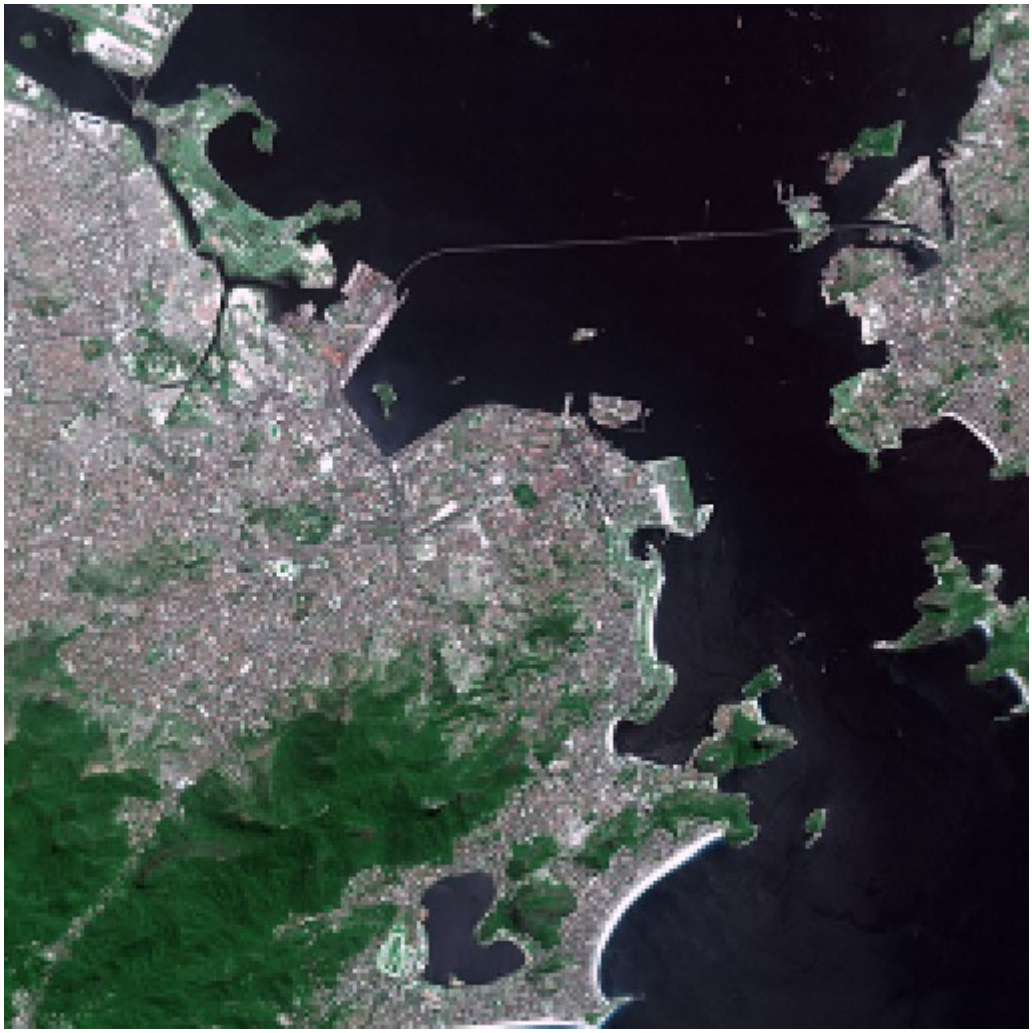

Figura 125 – Rio de Janeiro V1

viii) **Método V2**

Figura 126 – Rio de Janeiro V2

ix) Método V3

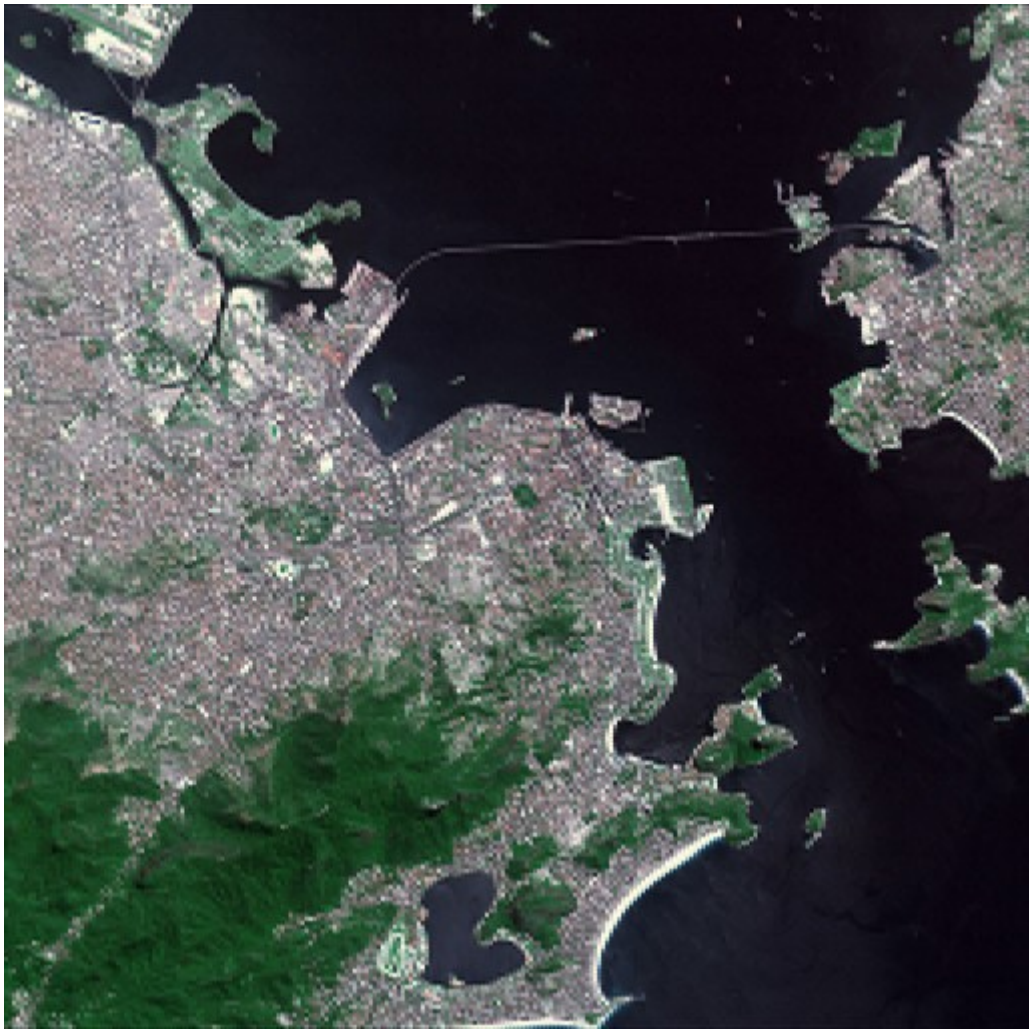

Figura 127 – Rio de Janeiro V3

k) Lagoa Rodrigo de Freitas

i) <u>Original</u>

Figura 128 – Rodrigo de Freitas Original

ii) <u>Reduzida</u>

Figura 129 – Rodrigo de Freitas Reduzida

iii) Bilinear

Figura 130 – Rodrigo de Freitas Bilinear

iv) Bicubic

Figura 131 – Rodrigo de Freitas Bicubic

v) B-Spline

Figura 132 – Rodrigo de Freitas B-Spline

vi) Lanczos

Figura 133 – Rodrigo de Freitas Lanczos

vii) <u>Método V1</u>

Figura 134 – Rodrigo de Freitas V1

viii) **Método V2**

Figura 135 – Rodrigo de Freitas V2

ix) Método V3

Figura 136 – Rodrigo de Freitas V3

1) Aeroporto Santos Dummont

i) Original

Figura 137 – Santos Dummont Original

ii) Reduzida

Figura 138 – Santos Dummont Reduzida

iii) Bilinear

Figura 139 – Santos Dummont Bilinear

iv) Bicubic

Figura 140 – Santos Dummont Bicubic

v) B-Spline

Figura 141 – Santos Dummont B-Spline

vi) Lanczos

Figura 142 – Santos Dummont Lanczos

vii) Método V1

Figura 143 – Santos Dummont V1

viii) Método V2

Figura 144 – Santos Dummont V2

ix) Método V3

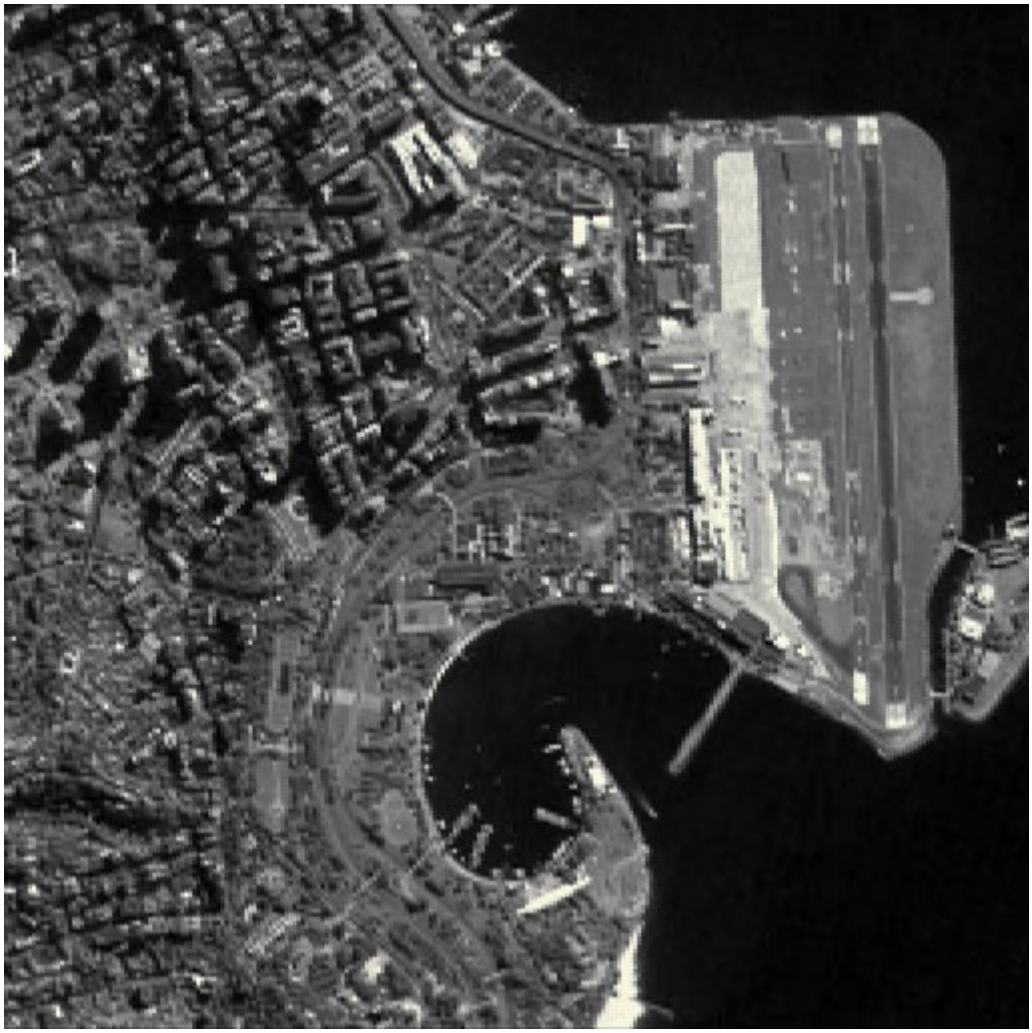

Figura 145 – Santos Dummont V3