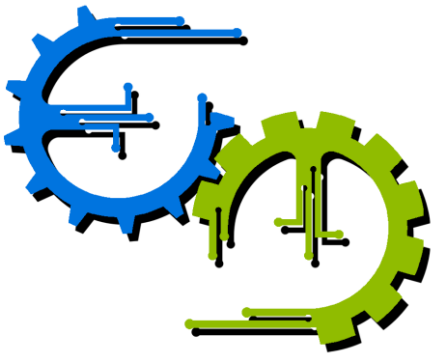
TRABALHO DE GRADUAÇÃO

RECONHECIMENTO DE FALHAS EM ESPAÇADORES DE LINHAS DE TRANSMISSÃO UTILIZANDO RECONSTRUÇÃO 3D E REDES NEURAIS ARTIFICIAIS

Hugo Caetano da Silva Júnior

Brasília, Julho de 2011.



ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

RECONHECIMENTO DE FALHAS EM ESPAÇADORES DE LINHAS DE TRANSMISSÃO UTILIZANDO RECONSTRUÇÃO 3D E REDES NEURAIS ARTIFICIAIS

Hugo Caetano da Silva Júnior

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Controle e Automação.

Banca examinadora

Prof. Adolfo Bauchspiess, UnB/ENE (Orientador) _____

Prof. Antônio Padilha Lanari Bó, UnB/ENE _____

Prof. Renato Alves Borges, UnB/ENE _____

Brasília, Julho de 2011

FICHA CATALOGRÁFICA

CAETANO DA SILVA JR., HUGO
Reconhecimento de falhas em espaçadores de linhas de transmissão utilizando reconstrução 3D e redes neurais artificiais [Distrito Federal] 2011.
xii, 80p., 210 x 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2011).
Trabalho de Graduação – Universidade de Brasília, Faculdade de Tecnologia.
1. Linhas de transmissão
2. Visão estéreo
3. Processamento de imagens
4. Redes neurais artificiais
I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

CAETANO S. J., H. (2011). Reconhecimento de falhas em espaçadores de linhas de transmissão utilizando reconstrução 3D e redes neurais artificiais, Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-n° 013/2011, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 80p.

CESSÃO DE DIREITOS

Hugo Caetano da Silva Júnior
SQS 406 Bloco R Asa Sul
CEP 70255-180 - Brasília - DF - Brasil

AGRADECIMENTOS

Agradeço à minha família, em especial às minhas tias,
por todo apoio que têm me dado.
Aos meus amigos, pelas noites que viramos juntos, seja
estudando ou nos divertindo.
Aos professores que compartilharam seus conhecimentos
e experiências.
Ao colega Elder Oroski, pela ajuda neste trabalho.

Hugo Caetano da Silva Júnior

RESUMO

A inspeção de linhas de transmissão de energia geralmente é feita por um helicóptero tripulado. Essa é uma forma perigosa, pois o veículo precisa manter uma distância pequena dos cabos de alta tensão. O objetivo deste trabalho é ajudar a tornar essa tarefa mais rápida e segura e menos dispendiosa. Os espaçadores são um dos elementos que necessitam de inspeção. Eles são dispositivos ligados aos cabos da linha de transmissão para evitar que os mesmos se toquem. Pretende-se verificar a integridade das garras desses espaçadores a partir de imagens do contorno da garra, adquiridas através de um sistema estéreo, em que são utilizadas duas câmeras para simular a visão humana e obter a noção de profundidade. A partir dos pontos no espaço, é feita uma reprojeção. Serão calculados os espectros da imagem reprojetada e uma rede neural artificial será treinada para verificar falhas. Com o procedimento proposto obteve-se uma taxa de acerto da classificação automatizada das garras da ordem de 83%. A robustez dos algoritmos ainda precisa ser melhorada, mas serve de ponto de partida para pesquisas relacionadas ao reconhecimento de falhas utilizando reconstrução 3D.

Palavras-chave: contorno, sistema estéreo, rede neural artificial, reconhecimento de falhas, reconstrução 3D.

ABSTRACT

The inspection of power transmission lines is usually done by manned helicopters. This task is very dangerous, because the vehicle needs to stay close to high voltage cables. This work aims to make this task faster, safer and less expensive. Transmission line spacers are some elements which require inspection. They are devices connected to the cables of transmission lines to prevent them from touching each other. The main goal of this work is verifying the integrity of the spacers' claws using images of the claw's profile, which are acquired through a stereo system, in which two cameras are used to simulate the human vision and achieve the depth perception. A reprojection is done from the three-dimensional points. The specter of the reprojected image will be calculated and verified by an artificial neural network. The proposed procedure provided a success rate of automated classification of the claws of nearly 83 %. The robustness of the algorithms needs to be improved, but serves as a starting point for research related to the recognition of failures using 3D reconstruction.

Keywords: profile, stereo system, artificial neural network, recognition of failures, 3D reconstruction.

SUMÁRIO

1 INTRODUÇÃO 1
1.1 Definição do problema 2

2 FUNDAMENTOS 5
2.1 Linhas de transmissão 5
2.2 Imagens digitais 6
2.2.1 Segmentação de imagens 7
2.2.2 Representação e código de cadeia 8
2.2.3 Séries de Fourier 9
2.3 Modelo de câmera 12
2.3.1 Calibração das câmeras 14
2.4 Sistema estéreo 19
2.4.1 Correspondência 20
2.4.2 Geometria epipolar 21
2.4.3 Retificação 23
2.4.4 Reconstrução 3D 24
2.5 Redes neurais artificiais 25

3 PROCEDIMENTOS 29
3.1 Aparato experimental 29
3.2 Calibração 31
3.3 Aquisição de imagens 32
3.4 Detecção de bordas 32
3.5 Visão estéreo 33
3.6 Reprojeção 34
3.7 Treinamento da rede neural 35

4 RESULTADOS 36
4.1 Calibração 36
4.2 Detecção de bordas e retificação 39
4.3 Correspondência 43
4.4 Reconstituição 3D e reprojeção 45
4.5 Rede Neural 48

5 CONCLUSÕES 50

REFERÊNCIAS BIBLIOGRÁFICAS 51

ANEXOS **53**

LISTA DE FIGURAS

1.1 Espaçador. 1
1.2 VANT onde as câmeras serão acopladas. 2
1.3 Garra presa corretamente. 3
1.4 Garra aberta. 3
1.5 Sequencia de passos realizados. 4

2.1 Linhas de transmissão (espaçadores em destaque) [Leal, 2010]. 5
2.2 (a) Uma imagem com restrições esparsas: rabiscos brancos indicam *foreground*, rabiscos pretos indicam background. (b) Segmentação ruim obtida por [Chuang *et al.*, 2001]. (c) Boa segmentação obtida por [Rother *et al.*, 2004] e [Li *et al.*, 2004]. (d) Máscaras obtidas automaticamente podem resultar em pequenas perdas. Uma obtenção acurada (e) é necessária para produzir (f) uma excelente segmentação [Levin *et al.*, 2006]., ... 8
2.3 Código de cadeia. 8
2.4 Contorno correspondente à sequencia 0005676644422123. 9
2.5 Projeção x da Figura 2.4 [de Oliveira & de Araújo Dias, 2007]. 10
2.6 Projeção y da Figura 2.4 [de Oliveira & de Araújo Dias, 2007]. 10
2.7 Aproximação do contorno da Figura 2.4 para n = 1, 2, 3, 4 [Giardina Dougherty, 1988]. 11
2.8 Modelo da câmara escura de orifício [Rodrigues & de Almeida Mencari, 1999]. 12
2.9 Centro óptico C, ponto principal c, eixo focal e, plano focal $\mathcal{F}$, plano de retina $\mathcal{R}$ e projeção m do ponto M. 13
2.10 O plano de retina está à frente do plano focal. 13
2.11 Sistemas de coordenadas. 14
2.12 Projeção do ponto M no plano de retina. 15
2.13 Projeção do raio de luz proveniente do ponto M no plano $y'z'$. 15
2.14 Sistemas de coordenadas uv e ij em relação à imagem. 16
2.15 Posição do sistema de coordenadas de câmera em relação ao sistema externo. . 18
2.16 Plano epipolar (E). 22
2.17 Linha epipolar (p). 23
2.18 Câmeras em posição canônica. 23
2.19 Esquema de imagens retificadas: As projeções m_L e m_R estão na mesma linha epipolar p (horizontal) 24
2.20 Modelo do neurônio de MacCulloch [net, n.d.]. 26
2.21 Modelo de neurônio [NNT, 2006]. 26
2.22 Exemplos de funções de ativação [NNT, 2006]. 27
2.23 Camadas de uma rede neural [net, n.d.]. 27

3.1 Amostra de cabo presente no LARA. 29
3.2 Câmera STH-MDCS3-VARX. 30
3.3 Interface do software SRI Small Vision System. 30
3.4 Imagens do tabuleiro usado na calibração. 31
3.5 Posição do sistema de coordenadas de mundo. 32
3.6 Exemplo de imagens adquiridas com as câmeras. 32
3.7 Máscaras. 33
3.8 Câmera virtual (mostrada em azul). 34

4.1 ROI (garra sem defeitos). 40
4.2 ROI (garra com defeito). 40
4.3 Máscaras (garra sem defeitos). 40
4.4 Máscaras (garra com defeito). 41
4.5 Bordas (garra sem defeitos). 41
4.6 Bordas (garra com defeito). 41
4.7 Bordas com imagem em tamanho original (garra sem defeitos). 42
4.8 Bordas com imagem em tamanho original (garra com defeito). 42
4.9 Imagens retificadas (garra sem defeitos). 42
4.10 Imagens retificadas (garra com defeito). 43
4.11 Detecção de bordas para uma imagem com angulação de 30°). 43
4.12 Plot 3D. 45
4.13 Reprojeção dos pontos. 46
4.14 Outra vista dos gráficos da Figura 4.13. 47
4.15 Reprojeção de pontos. 47
4.16 Cada linha mostra duas vistas da reconstituição dos pontos e a reprojeção dos pares de imagens 2, 4, 21 e 23, respectivamente. 49

LISTA DE TABELAS

2.1 Comprimento dos segmentos 9
2.2 Tabela verdade da função XNOR..... 21

4.1 Equivalência dos parâmetros intrínsecos no MATLAB. 36
4.2 Calibração individual da câmera esquerda. 36
4.3 Calibração individual da câmera direita..... 37
4.4 Calibração estéreo. 37
4.5 Calibração estéreo retificada. 39
4.6 Matriz de correspondência. 44
4.7 Resultados da rede neural..... 48

Símbolos Latinos

$f(x, y)$	Função de intensidade luminosa
$i(x, y)$	Função de iluminação do meio
$r(x, y)$	Função de reflectância do material
A_0	Coeficiente de Fourier DC da direção x
A_n	n-ésimo coeficiente de Fourier cosenoidal na direção x
B_n	n-ésimo coeficiente de Fourier senoidal na direção x
C_0	Coeficiente de Fourier DC da direção y
C_n	n-ésimo coeficiente de Fourier cosenoidal na direção y
D_n	n-ésimo coeficiente de Fourier senoidal na direção y
$\mathcal{F}$	Plano focal
$\mathcal{R}$	Plano de retina
C	Centro óptico
c	Ponto principal da imagem
f	Distância focal
e	Eixo focal
xyz	Sistema de coordenadas externo (coordenadas de mundo)
$x'y'z'$	Sistema de coordenadas de câmera
uv	Sistema de coordenadas da imagem (em milímetros)
ij	Sistema de coordenadas da imagem (em pixels)
du	Tamanho do pixel na direção u
dv	Tamanho do pixel na direção v
i	Linha do pixel na imagem
j	Coluna do pixel na imagem
i_0	Coordenada i do ponto principal
j_0	Coordenada j do ponto principal
U	Coordenada homogênea relacionada a j
V	Coordenada homogênea relacionada a i
S	Escala das coordenadas homogêneas

P	Matriz de projeção perspectiva (matriz de calibração)
K	Matriz dos parâmetros intrínsecos
L	Matriz dos parâmetros extrínsecos
T	Vetor de translação
R	Matriz de rotação
$\mathcal{E}$	Plano epipolar
p	Linha epipolar

Subscritos

$_L$	relativo à câmera ou à imagem da esquerda
$_R$	relativo à câmera ou à imagem da direita

Siglas

VANT	Veículo Aéreo Não Tripulado
GPS	*Global Positioning System*
LARA	Laboratório de Automação e Robótica
IEEE	*Institute of Electrical and Electronics Engineers*
GB	Gigabytes
GHz	Gigahertz
RAM	*Random Access Memory*
USB	*Universal Serial Bus*
FPS	Frames por segundo
ROI	*Region of interest*

1 INTRODUÇÃO

As linhas de transmissão são utilizadas para transmitir energia elétrica de uma fonte geradora para uma carga consumidora. No Brasil, as fontes geradoras geralmente são usinas hidrelétricas, termelétricas e eólicas. Devido a questões de segurança ou mesmo por fatores geográficos, elas costumam ficar distante da população. Por isso, as linhas de transmissão possuem uma grande extensão, e inspecioná-las é uma tarefa árdua.

A inspeção é necessária para garantir o constante fornecimento de energia elétrica. A interrupção deste serviço implica em pesadas multas para as empresas responsáveis. A inspeção geralmente é feita por um helicóptero tripulado. Essa é uma forma perigosa, pois o veículo precisa manter uma distância pequena dos cabos de alta tensão. A inspeção também pode ser terrestre, mas essa é uma forma muito limitada, pois o acesso muitas vezes é restrito, devido à localização geográfica. O objetivo deste trabalho é ajudar a tornar essa tarefa mais rápida e segura e menos dispendiosa.

Dentre os componentes da linha que necessitam de inspeção, estão os espaçadores (Figura 1.1). Estes são dispositivos ligados aos cabos da linha de transmissão para evitar que os mesmos se toquem em função de cargas de vento. A condução é feita por três cabos para que se aumente o valor da corrente elétrica, que é conduzida pela borda do cabo. Os cabos ligados pelos espaçadores possuem a mesma fase. Usar um único cabo para conduzir a mesma fase seria desperdício de material. O motivo de os cabos não poderem tocar-se é para evitar danos físicos.

Figura 1.1: Espaçador.

Pode-se pensar em vários métodos para a verificação destes componentes. Existe a verificação manual, atualmente realizada. Outros métodos que podem ser utilizados são um sistema mecânico de sensores colocadosem cada garra ou uma análise de uma única imagem. Neste trabalho, pretende-se verificar a integridade das garras desses espaçadores a

partir de imagens. As imagens serão adquiridas através de um sistema estéreo, em que são utilizadas duas câmeras para simular a visão humana e obter a noção de profundidade. Dessa forma, pode-se ter uma noção dos valores das dimensões da garra e, assim, padronizar o objeto de análise. Serão analisados os espectros de imagens obtidas através da reprojeção dos pontos tridimensionais e uma rede neural será treinada para verificar falhas.

O trabalho de [de Oliveira & de Araújo Dias, 2007] trata da inspeção das garras através da análise de uma única imagem. Logo, existe um banco de dados para imagens adquiridas numa posição padrão. Utilizando um sistema estéro, será possível a adquirir as imagens em diferentes posições e aplicar uma transformação nos pontos tridimensionais de forma que eles assumam a posição padrão.

As câmeras serão acopladas a um Veículo Aéreo Não Tripulado (VANT). Trata-se de um pequeno helicóptero (Figura 1.2) que será guiado via GPS seguindo as linhas de transmissão de modo a evitar colisões, com pontos de parada próximos aos espaçadores [Beckmann, 2008].

Figura 1.2: VANT onde as câmeras serão acopladas.

1.1 DEFINIÇÃO DO PROBLEMA

As falhas mais comuns que envolvem os espaçadores são em relação à sua integridade física e ao deslocamento do cabo para fora da garra de fixação. Esse trabalho será focado na garra de fixação. Verificar-se-á se a garra precisa de ajuste. Ela pode estar frouxa, completamente aberta ou quebrada (Figuras 1.3 e 1.4).

Figura 1.3: Garra presa corretamente.

Figura 1.4: Garra aberta.

Pretende-se, a partir de um sistema de visão estéreo, montar um modelo tridimensional do cabo com a garra. O processamento será feito utilizando o MATLAB, que possui as ferramentas necessárias para o tratamento de imagens.

Depois de adquiridas as imagens, é feita a detecção de bordas. A segmentação do espaçador é um problema muito complexo. Em vez disso, será feita a segmentação do cabo. Depois de detectadas as bordas do cabo, faz-se a retificação das imagens e a reconstrução 3D. Os pontos reconstruídos serão reprojetados a partir de uma câmera imaginária, cuja posição será de frente para a garra, de forma a conseguir a melhor imagem possível. Quando a borda for reprojetada na nova imagem, serão calculados seus coeficientes de Fourier.

Os coeficientes de algumas imagens pré-definidas servirão para o treinamento de uma rede neural. Depois disso, serão feitos testes para verificar a eficácia do método. A Figura 1.5 mostra sequencialmente todo o processo.

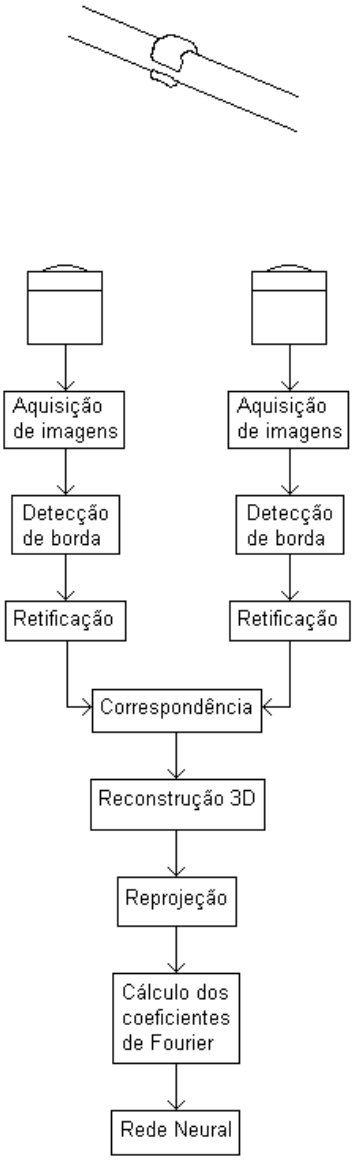


Figura 1.5: Sequencia de passos realizados.

2 FUNDAMENTOS

Neste capítulo serão apresentados os conceitos e a teoria necessários para o entendimento do presente projeto.

2.1 LINHAS DE TRANSMISSÃO

As linhas de transmissão são utilizadas para transmitir energia elétrica de uma fonte geradora para uma carga consumidora, geralmente através de corrente alternada (60 Hz) e alta tensão (até 750 kV). Pode-se observar uma linha de transmissão na Figura 2.1.

Os principais constituintes das linhas de transmissão são os condutores, o aterramento, as fundações, os isoladores, os para-raios, as esferas de sinalização, e as estruturas.

Os condutores são os cabos, geralmente de alumínio, por onde a corrente trafega. O aterramento é feito com cabos de cobre e serve para descarregar cargas acumuladas para a terra. As fundações servem de base para as estruturas. Os isoladores são geralmente de vidro, cerâmica ou polímeros e servem para fixar os condutores nas estruturas. Os para-raios evitam a formação de arco-elétricos causados por raios. As esferas de sinalização são geralmente laranja e feitas em resina polimérica reforçada com fibra de vidro e servem para alertar as aeronaves sobre a presença dos cabos. As estruturas dão suporte aos condutores e aos para-raios.

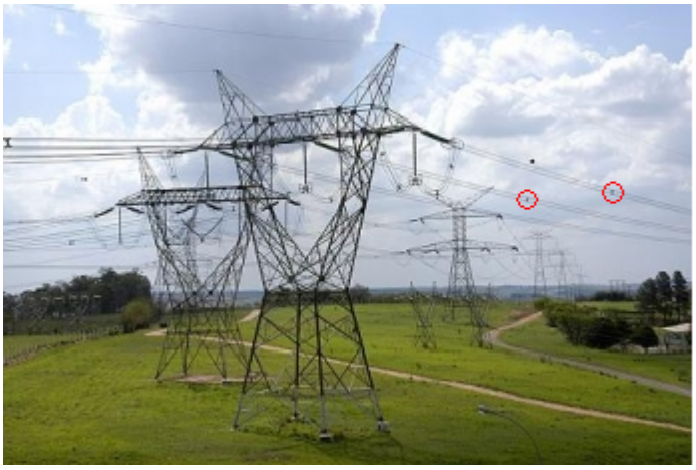


Figura 2.1: Linhas de transmissão (espaçadores em destaque) [Leal, 2010].

Caso haja algum problema com a linha de transmissão, o fornecimento de energia pode ser interrompido. Por isso, é muito importante a manutenção. No Brasil, as fontes geradoras geralmente são usinas hidrelétricas, termelétricas e eólicas. Devido a questões de segurança

ou mesmo por fatores geográficos, elas costumam ficar distante da população. Por isso, as linhas de transmissão possuem uma grande extensão, e inspecioná-las é uma tarefa árdua.

A manutenção traz muitos benefícios, independente de onde seja aplicada. Com uma manutenção adequada, aumenta-se a segurança e a qualidade, pois diminui-se as chances de possíveis erros. Além disso, os custos diminuem, pois os equipamentos funcionam de forma mais eficiente.

A manutenção deve ser realizada em três estágios [Oliveira, 2009]:

- Manutenção do terreno: para permitir o trânsito de veículos e pessoas e evitar que a vegetação cause algum dano à instalação.
- Manutenção da torre: para garantir a conservação da estrutura e evitar acidentes.
- Manutenção dos isoladores e cabos condutores: para garantir o funcionamento correto dos cabos condutores, bem como dos para-raios e isoladores. Aqui entram os espaçadores, que são o escopo deste trabalho.

A manutenção também pode ser corretiva, preventiva ou preditiva. A manutenção corretiva ocorre para reparar o dano causado a algum componente. Ela deve ser evitada, mas costuma ocorrer principalmente devido a fenômenos naturais, como as tempestades. A manutenção preventiva é a troca de componentes de tempos em tempos, mesmo que o fornecimento de energia ainda não esteja prejudicado. A manutenção preditiva consiste no ajuste de pequenas falhas visíveis (como aperto de parafusos e troca de isoladores) que podem gerar algum problema.

2.2 IMAGENS DIGITAIS

Uma imagem é uma função de intensidade luminosa bidimensional. Seja $f(x, y)$ uma função que denota uma imagem nas coordenadas espaciais x e y, temos que $0 < f(x, y) < \infty$, pois a luz é uma forma de energia, portanto finita e positiva. $f(x, y)$ pode ser interpretado como o produto de duas grandezas: iluminação e reflectância. A primeira depende do meio, da quantidade de luz emitida. A segunda é uma propriedade do material. Sendo $i(x, y)$ a iluminação no ponto (x, y) e $r(x, y)$ a reflectância no ponto (x, y), obtém-se a equação (2.1), onde $0 < i(x, y) < \infty$ e $0 < r(x, y) < 1$ [Gonzalez & Woods, 2000].

$$f(x, y) = i(x, y)r(x, y) \tag{2.1}$$

Para que uma imagem seja usada num processamento digital, ela precisa ser discretizada. Isso deve ocorrer tanto no espaço quanto na amplitude. Assim, uma imagem digital pode ser expressa por uma matriz, onde cada elemento representa um determinado pixel e seu valor

representa a intensidade luminosa. Numa imagem monocromática, (x, y) é a coordenada do pixel e $f(x, y)$ é um valor que representa o nível de cinza, tal que x e y são números inteiros cujo valor máximo depende da resolução da imagem. Para uma imagem colorida, são definidas três funções, $f_R(x, y)$, $f_G(x, y)$ e $f_B(x, y)$, representando os níveis vermelho, verde e azul, respectivamente. Estas três cores juntas, variando apenas a intensidade de cada uma, podem representar qualquer outra cor.

2.2.1 Segmentação de imagens

Segmentar uma imagem significa subdividi-la em suas partes ou objetos constituintes. A segmentação geralmente cessa quando os objetos de interesse já foram isolados. Os algoritmos de segmentação costumam ser baseados nas seguintes propriedades de níveis de cinza:

- Descontinuidade: busca-se segmentar baseado em mudanças bruscas nos níveis de cinza.
- Similaridade: a segmentação ocorre baseada em regiões com nível de cinza parecido.

Os tipos básicos de descontinuidade são: pontos, linhas e bordas. Para fazer o reconhecimento, toma-se como base um pixel e verifica-se o nível de cinza dos pixels adjacentes. A detecção de borda é a mais utilizada na prática, pois pontos e linhas finas isoladas não são frequentes na maioria das aplicações.

As principais abordagens dos algoritmos fundamentados em similaridades baseiam-se em limiarização, crescimento de regiões e divisão e fusão de regiões [Gonzalez Woods, 2000].

Para a extração de um objeto, muitas vezes é necessária a intervenção do usuário. É determinada uma região da imagem pertencente ao objeto (*foreground*) e uma região que não pertence ao objeto (*background*). Uma imagem α, com o objeto em destaque, é determinada a partir de medições de cores, comparando o restante da imagem com o *foreground* e o *background* (Figura 2.2).

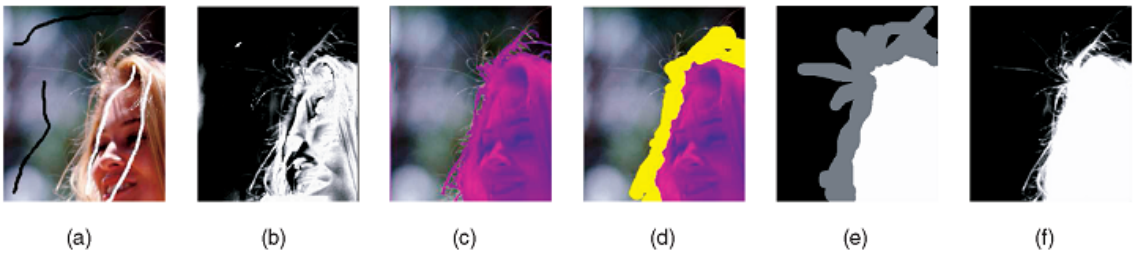


Figura 2.2: (a) Uma imagem com restrições esparsas: rabiscos brancos indicam *foreground*, rabiscos pretos indicam background. (b) Segmentação ruim obtida por [Chuang *et al.*, 2001]. (c) Boa segmentação obtida por [Rother *et al.*, 2004] e [Li *et al.*, 2004]. (d) Máscaras obtidas automaticamente podem resultar em pequenas perdas. Uma obtenção acurada (e) é necessária para produzir (f) uma excelente segmentação [Levin *et al.*, 2006].,

2.2.2 Representação e código de cadeia

Depois de detectada a borda, uma maneira de representá-la pode ser útil. Para isso serve o código de cadeia. A borda é representada por um conjunto de pequenos segmentos de determinado tamanho e direção. O segmento deve ser um dos mostrados na Figura 2.3.

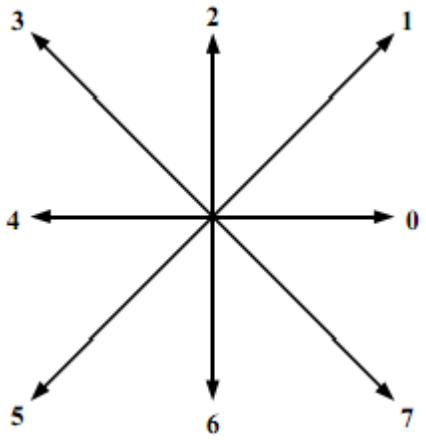


Figura 2.3: Código de cadeia.

Cada segmento possui tamanho diferente, conforme a Tabela 2.1, onde du é o comprimento do pixel na direção x e dv é o comprimento do pixel na direção y.

Tabela 2.1: Comprimento dos segmentos

Segmento	Comprimento
0	du
1	$\sqrt{du^2 + dv^2}$
2	dv
3	$\sqrt{du^2 + dv^2}$
4	du
5	$\sqrt{du^2 + dv^2}$
6	dv
7	$\sqrt{du^2 + dv^2}$

Por exemplo, a sequência 0005676644422123 está associada à Figura 2.4.

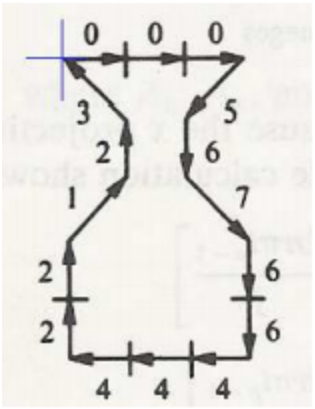


Figura 2.4: Contorno correspondente à sequencia 0005676644422123.

2.2.3 Séries de Fourier

Uma das formas de representar uma imagem é através da transformada de Fourier, amplamente utilizada em processamento de imagens. Para utilizá-la, é recomendável separar as coordenadas espacias x e y. Tomando como base a Figura 2.4, as projeções x e y são dadas pelas Figuras 2.5 e 2.6, respectivamente, onde a unidade é o comprimento do pixel ($du = dv = 1$) e tm é o comprimento do arco de acordo com a Tabela 2.1.

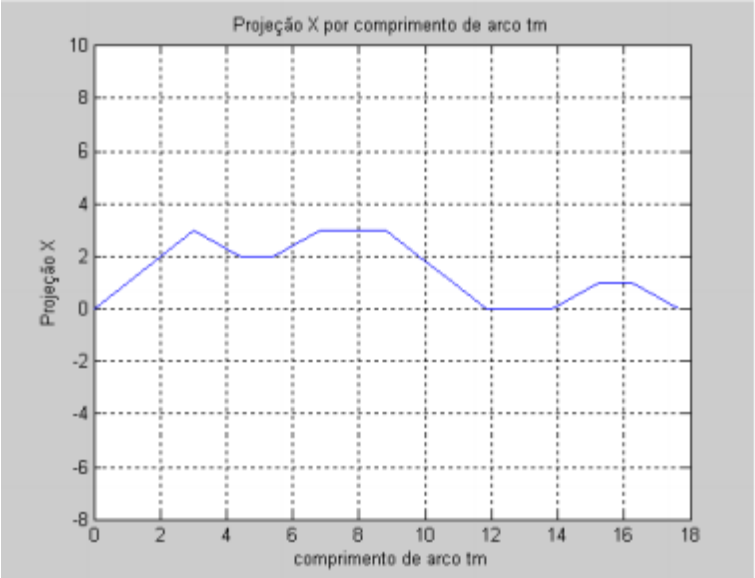


Figura 2.5: Projeção x da Figura 2.4 [de Oliveira & de Araújo Dias, 2007].

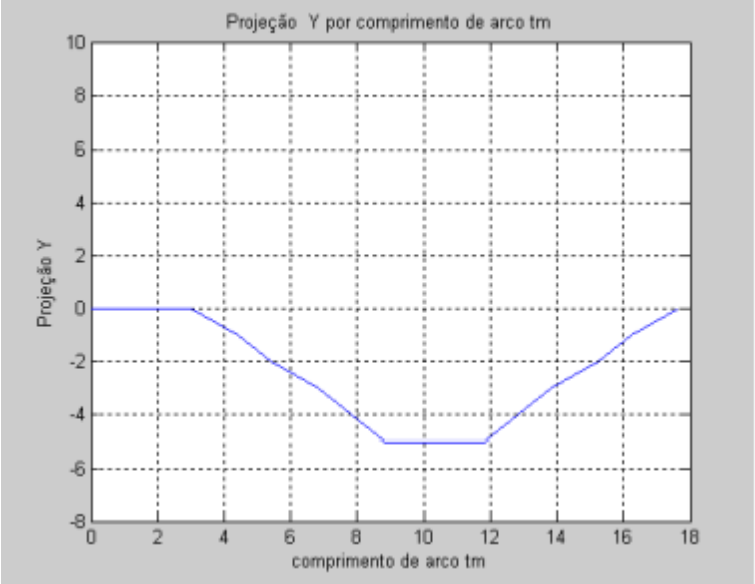


Figura 2.6: Projeção y da Figura 2.4 [de Oliveira & de Araújo Dias, 2007].

Os três primeiros símbolos da sequancia são 0. Cada 0 significa um avanço em x e nenhuma alteração em y. Depois um 5, que significa um recuo em x e um recuo em y. 6 significa um recuo em y e nenhuma alteração em x. Enfim, um avanço em x é dado por 0, 1 e 7; um avanço em y é dado por 1, 2 e 3; um recuo em x é dado por 3, 4 e 5; um recuo em y é dado por 5, 6 e 7.

Agora que estão sepados x e y, pode-se expandir cada uma das projeções em séries de Fourier, resultando nas equações (2.2) e (2.3).

$$x(t) = A_0 + \sum_{n=1}^{\infty} \left(A_n \cos\left(\frac{2\pi tn}{T}\right) + B_n \sin\left(\frac{2\pi tn}{T}\right) \right) \tag{2.2}$$

$$y(t) = C_0 + \sum_{n=1}^{\infty} \left(C_n \cos\left(\frac{2\pi tn}{T}\right) + D_n \sin\left(\frac{2\pi tn}{T}\right) \right) \tag{2.3}$$

Os coeficientes de Fourier podem ser determinados pelas equações (2.4), (2.5), (2.6), (2.7), (2.8) e (2.9).

$$A_0 = \frac{1}{T} \int_0^T x(t)dt \tag{2.4}$$

$$A_n = \frac{2}{T} \int_0^T x(t) \cos\left(\frac{2\pi tn}{T}\right) dt \tag{2.5}$$

$$B_n = \frac{2}{T} \int_0^T x(t) \sin\left(\frac{2\pi tn}{T}\right) dt \tag{2.6}$$

$$C_0 = \frac{1}{T} \int_0^T y(t)dt \tag{2.7}$$

$$C_n = \frac{2}{T} \int_0^T y(t) \cos\left(\frac{2\pi tn}{T}\right) dt \tag{2.8}$$

$$D_n = \frac{2}{T} \int_0^T y(t) \sin\left(\frac{2\pi tn}{T}\right) dt \tag{2.9}$$

Utilizando os coeficientes de Fourier, é possível reconstruir o contorno. Quanto mais alto o valor de n alcançar, mais próximo do esperado será o resultado [de Oliveira de Araújo Dias, 2007]. A Figura 2.7 mostra a aproximação para os quatro primeiros harmônicos do contorno mostrado na Figura 2.4.

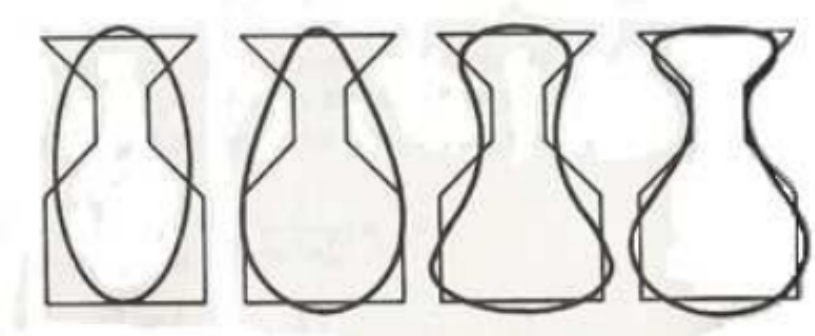


Figura 2.7: Aproximação do contorno da Figura 2.4 para n = 1, 2, 3, 4 [Giardina Dougherty, 1988].

2.3 MODELO DE CÂMERA

Uma câmera pode ser modelada a partir de um sistema bem simples. Esse modelo é a câmara escura de orifício. Consiste de uma caixa opaca com um pequeno furo, por onde a luz penetra. A Figura 2.8 ilustra este esquema. O ponto C é o orifício, cujas dimensões devem ser desprezadas neste modelo. AB é um objeto qualquer representado. Como C é um ponto, apenas um raio de luz proveniente de cada ponto do objeto penetra a câmera. Os raios de luz formam a imagem $A'B'$ no plano coincidente com o fundo da caixa.

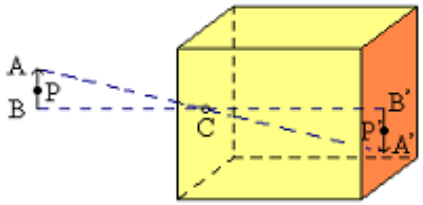


Figura 2.8: Modelo da câmara escura de orifício [Rodrigues & de Almeida Mencari, 1999].

Alguns elementos importantes a serem destacados são: o plano de retina, o centro óptico, o plano focal, o eixo óptico, a distância focal e o ponto principal.

- O plano de retina $\mathcal{R}$ é o plano onde as imagens são formadas.
- O centro óptico é o ponto C, por onde passam todos os raios de luz que formam a imagem.
- O plano focal $\mathcal{F}$ é o plano paralelo ao plano de retina e que contém C.
- O eixo óptico é a reta perpendicular ao plano de retina que passa pelo ponto C.
- A distância focal é a distância do plano de retina até o centro óptico.
- O ponto principal é a interseção do eixo óptico com o plano de retina.

A Figura 2.9 ilustra estes elementos.

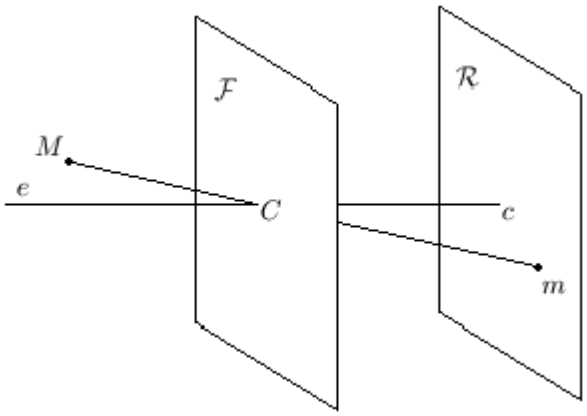


Figura 2.9: Centro óptico C, ponto principal c, eixo focal e, plano focal $\mathcal{F}$, plano de retina $\mathcal{R}$ e projeção m do ponto M.

Na Figura 2.9, é importante notar que a imagem aparece invertida em relação ao objeto. Para evitar isso e facilitar o raciocínio, pode-se transladar o plano de retina de duas vezes a distância focal em direção ao centro óptico. O resultado é o mostrado na Figura 2.10.

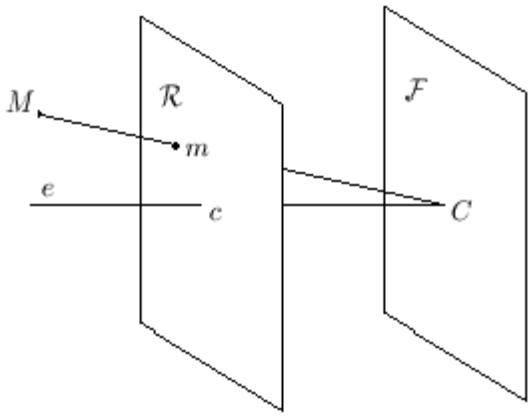


Figura 2.10: O plano de retina está à frente do plano focal.

Existem quatro sistemas de coordenadas a serem considerados. O primeiro é o sistema de coordenadas de mundo ou sistema de coordenadas externo xyz, cujas origem e orientação são arbitrárias. O segundo é o sistema de coordenadas de câmera $x'y'z'$. Neste sistema, a origem coincide com o centro óptico da câmera, a coordenada z' coincide com o eixo focal, e o eixo x' é paralelo ao eixo u da imagem. O terceiro é o sistema de coordenadas de imagem uv, bidimensional, dado em milímetros, cujo plano é coincidente com o plano focal e cuja origem é o ponto principal. O último é o sistema de coordenadas de imagem ij, bidimensional, dado em pixels, cujo plano é coincidente com o plano focal. A origem do sistema ij está no canto superior esquerdo da imagem. Os eixos i e j são paralelos aos eixos

v e u, respectivamente. Os eixos u e j possuem a mesma orientação. Os eixos v e i possuem orientação contrária. A Figura 2.11 mostra esses quatro sistemas de coordenadas.

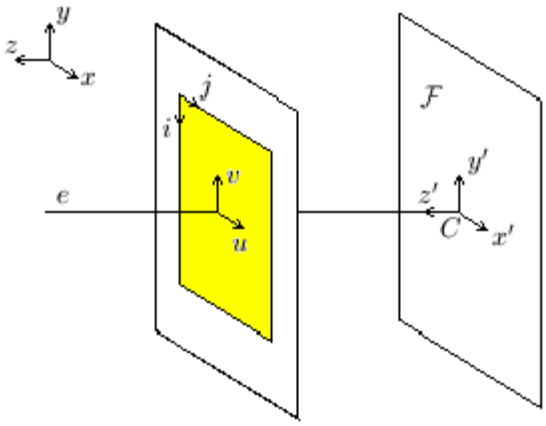


Figura 2.11: Sistemas de coordenadas.

2.3.1 Calibração das câmeras

A calibração consiste na determinação dos parâmetros intrínsecos e extrínsecos da câmera. Com estes dados, é possível montar a matriz de projeção perspectiva, também conhecida como matriz de calibração. Essa matriz determina a relação entre as coordenadas x, y, z de um ponto no espaço e a posição do pixel correspondente na imagem gerada. Sendo P a matriz de calibração, tem-se (2.10), onde U, V e S são coordenadas homogêneas, conforme (2.11) e (2.12), e i e j representam a posição do pixel. O pixel (i, j) é aquele localizado na linha i e na coluna j.

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = P \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.10}$$

$$i = \frac{V}{S} \tag{2.11}$$

$$j = \frac{U}{S} \tag{2.12}$$

Para achar a matriz de calibração, é preciso descobrir os parâmetros da câmera. Os parâmetros intrínsecos são a distância focal, o tamanho do pixel e as coordenadas do ponto principal. Os parâmetros extrínsecos são a matriz de rotação R e a vetor de translação T da câmera em relação ao sistema de coordenadas externo.

2.3.1.1 Parâmetros intrínsecos

Para encontrar as coordenadas do ponto na imagem, em milímetros, deve-se observar a Figura 2.12, onde M é um ponto no espaço e m é sua imagem. x', y' e z' são coordenadas de câmera, e u e v são coordenadas de imagem. Pode-se achar a coordenada v observando a projeção no plano $y'z'$ (Figura 2.13). Através de semelhança de triângulos encontra-se (2.13). Analogamente, encontra-se a coordenada u pela projeção no plano $x'z'$. Chega-se à equação (2.14).

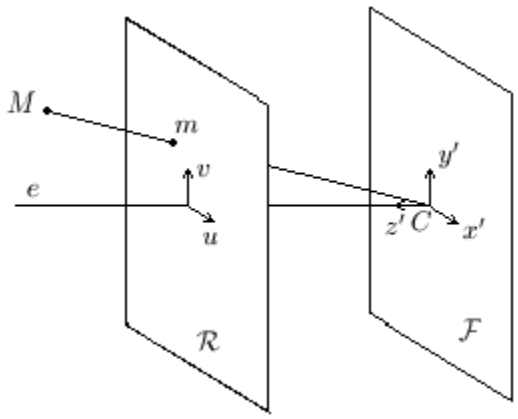


Figura 2.12: Projeção do ponto M no plano de retina.

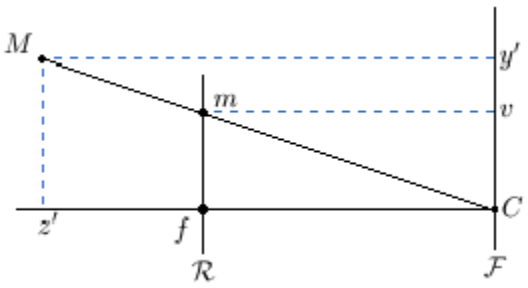


Figura 2.13: Projeção do raio de luz proveniente do ponto M no plano $y'z'$.

$$v = \frac{f \cdot y'}{z'} \tag{2.13}$$

$$u = \frac{f \cdot x'}{z'} \tag{2.14}$$

Colocando em forma de matriz, tem-se (2.15).

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z'} \cdot \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{2.15}$$

Para encontrar as coordenadas i e j, em pixels, sabendo a distância focal em milímetros, deve-se conhecer o tamanho do pixel, tanto na direção u quanto na direção v, e as coordenadas do ponto principal. A Figura 2.14 mostra o plano de retina. A região amarela é a imagem obtida com a câmera.

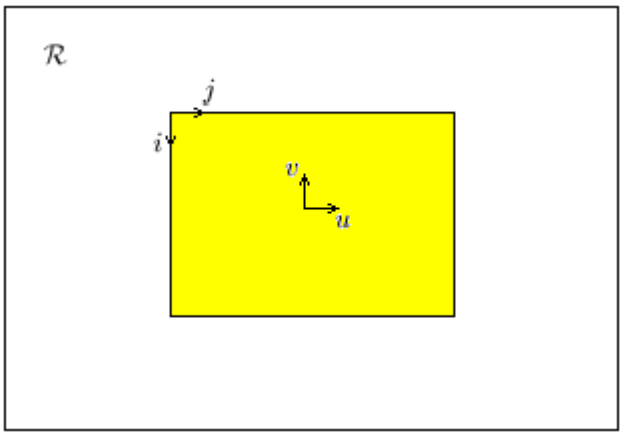


Figura 2.14: Sistemas de coordenadas uv e ij em relação à imagem.

Sendo, dv o tamanho do pixel na direção v e du o tamanho do pixel na direção u, as coordenadas do pixel i e j são dadas pelas equações (2.16) e (2.17), onde i_0 e j_0 são as coordenadas do ponto principal.

$$i = -\frac{v}{dv} + i_0 \tag{2.16}$$

$$j = \frac{u}{du} + j_0 \tag{2.17}$$

Em forma de matriz, tem-se (2.18).

$$\begin{bmatrix} j \\ i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{du} & 0 & j_0 \\ 0 & -\frac{1}{dv} & i_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.18}$$

Substituindo o resultado de (2.15) em (2.18), obtém-se (2.20).

$$\begin{bmatrix} j \\ i \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{du} & 0 & j_0 \\ 0 & -\frac{1}{dv} & i_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \frac{1}{z'} \cdot \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{2.19}$$

$$\begin{bmatrix} j \\ i \\ 1 \end{bmatrix} = \frac{1}{z'} \cdot \begin{bmatrix} \frac{f}{du} & 0 & j_0 & 0 \\ 0 & -\frac{f}{dv} & i_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{2.20}$$

Se $S = z'$, tem-se (2.21).

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} \frac{f}{du} & 0 & j_0 & 0 \\ 0 & -\frac{f}{dv} & i_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{2.21}$$

A matriz dos parâmetros intrínsecos K, que estabelece a relação mostrada em (2.22), é definida por (2.23).

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = K \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{2.22}$$

$$K = \begin{bmatrix} \frac{f}{du} & 0 & j_0 & 0 \\ 0 & -\frac{f}{dv} & i_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.23}$$

2.3.1.2 Parâmetros extrínsecos

Os parâmetros extrínsecos determinam a posição da câmera em relação ao sistema de coordenadas externo. A Figura 2.15 mostra o vetor T que representa a posição do centro óptico C. Pode-se observar também que os eixos do sistema de coordenadas de câmera sofreram uma rotação R em relação ao sistema de coordenadas externo. Ambos T e R estão em coordenadas de mundo.

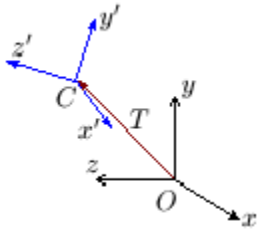


Figura 2.15: Posição do sistema de coordenadas de câmera em relação ao sistema externo.

Dado um ponto M com coordenadas externas (x, y, z) e coordenadas de câmera (x', y', z'), é possível estabelecer a relação (2.24).

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \tag{2.24}$$

Para encontrar (x', y', z') em função de (x, y, z), basta inverter a matriz, como mostrado em (2.25).

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.25}$$

A matriz dos parâmetros extrínsecos L, que estabelece a relação mostrada em (2.26), é definida por (2.27).

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = L \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.26}$$

$$L = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}^{-1} \tag{2.27}$$

2.3.1.3 Matriz de projeção perspectiva

Substituindo o vetor encontrado em (2.26) na equação (2.21), encontra-se (2.28).

$$\begin{bmatrix} U \\ V \\ S \end{bmatrix} = \begin{bmatrix} \frac{f}{du} & 0 & j_0 & 0 \\ 0 & -\frac{f}{dv} & i_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.28}$$

Observando (2.10), deduz-se (2.29).

$$P = K \cdot L \tag{2.29}$$

Substituindo os valores de K e L, encontra-se a matriz P em (2.30).

$$P = \begin{bmatrix} \frac{f}{du} & 0 & j_0 & 0 \\ 0 & -\frac{f}{dv} & i_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}^{-1} \tag{2.30}$$

Se o sistema de coordenadas externo for coincidente com o sistema de coordenadas de câmera, A matriz de dos parâmetros extrínsecos será igual à matriz identidade 4×4, e, portanto, $P = K$.

2.4 SISTEMA ESTÉREO

O imageamento estéreo requer a obtenção de duas vistas separadas de um mesmo objeto. Sabendo as características das câmeras, bem como sua posição, é possível fazer uma reconstrução 3D. Para isso, é preciso fazer uma equivalência de pontos nas duas imagens obtidas. Isso significa buscar o pixel de uma imagem que é correspondente a determinado pixel de outra imagem. De posse das características das duas câmeras, se é sabido qual o ponto equivalente na câmera direita de um ponto na câmera esquerda, é possível encontrar a posição deste ponto no espaço.

As características da câmera podem ser dadas pela matriz de calibração. Seja P a matriz de calibração definida em (2.39). Esta, junto com as coordenadas do pixel, permitem a construção de um sistema com três variáveis (as coordenadas x, y e z) e duas equações, observadas em (2.39). Utilizando-se duas câmeras, passa-se a ter três variáveis e quatro equações. A solução deste sistema sobredeterminado são as coordenadas 3D do pixel [Rodrigues & de Almeida Mencari, 1999].

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \tag{2.31}$$

$$U = p_{11} \cdot x + p_{12} \cdot y + p_{13} \cdot z + p_{14} \tag{2.32}$$

$$V = p_{21} \cdot x + p_{22} \cdot y + p_{23} \cdot z + p_{24} \tag{2.33}$$

$$S = p_{31} \cdot x + p_{32} \cdot y + p_{33} \cdot z + p_{34} \tag{2.34}$$

$$i = \frac{p_{21} \cdot x + p_{22} \cdot y + p_{23} \cdot z + p_{24}}{p_{31} \cdot x + p_{32} \cdot y + p_{33} \cdot z + p_{34}} \tag{2.35}$$

$$(p_{21} - i \cdot p_{31})x + (p_{22} - i \cdot p_{32})y + (p_{23} - i \cdot p_{33})z = i \cdot p_{34} - p_{24} \tag{2.36}$$

$$j = \frac{p_{11} \cdot x + p_{12} \cdot y + p_{13} \cdot z + p_{14}}{p_{31} \cdot x + p_{32} \cdot y + p_{33} \cdot z + p_{34}} \tag{2.37}$$

$$(p_{11} - j \cdot p_{31})x + (p_{12} - j \cdot p_{32})y + (p_{13} - j \cdot p_{33})z = j \cdot p_{34} - p_{14} \tag{2.38}$$

$$\begin{cases} (p_{21} - i \cdot p_{31})x + (p_{22} - i \cdot p_{32})y + (p_{23} - i \cdot p_{33})z = i \cdot p_{34} - p_{24} \\ (p_{11} - j \cdot p_{31})x + (p_{12} - j \cdot p_{32})y + (p_{13} - j \cdot p_{33})z = j \cdot p_{34} - p_{14} \end{cases} \tag{2.39}$$

2.4.1 Correspondência

Um sistema estéreo é composto por duas imagens: E e D. $E(i, j)$ é função de intensidade luminosa no pixel (i, j) da imagem da esquerda. $D(i, j)$ é função de intensidade luminosa no pixel (i, j) da imagem da direita. Um desafio da visão computacional é encontrar a correspondência dos pontos de E com os pontos de D.

A correspondência é feita analisando similaridades nas vizinhanças dos pixels das duas imagens. Retirando uma janela de cada imagem, pode-se aplicar uma função de correlação. Quanto maior a correlação, maiores as chances de a correspondência estar correta. Exemplos de função de correlação são a soma das diferenças absolutas (SAD), a soma do quadrado das diferenças (SSD) e correlação cruzada normalizada (NCC), mostradas em (2.40), (2.41) e (2.42), onde A e B são as dimensões da janela. A função $\mu(I_{i,j})$ calcula a média aritmética das intensidades luminosas da janela de dimensões A e B, cujo canto superior é o ponto (i, j) da imagem I [de Oliveira, 2006].

$$SAD(i_L, j_L, i_R, j_R) = \sum_{u=0}^{A-1} \sum_{v=0}^{B-1} |E(i_L + u, j_L + v) - D(i_R + u, j_R + v)| \tag{2.40}$$

$$SSD(i_L, j_L, i_R, j_R) = \sum_{u=0}^{A-1} \sum_{v=0}^{B-1} \left(E(i_L + u, j_L + v) - D(i_R + u, j_R + v)\right)^2 \tag{2.41}$$

$$NCC(i_L, j_L, i_R, j_R) = \frac{\sum_{u=0}^{A-1} \sum_{v=0}^{B-1} \left(E(i_L + u, j_L + v) - \mu(E_{i_L, j_L})\right) \cdot \left(D(i_R + u, j_R + v) - \mu(D_{i_R, j_R})\right)}{\left(\sum_{u=0}^{A-1} \sum_{v=0}^{B-1} \left(E(i_L + u, j_L + v) - \mu(E_{i_L, j_L})\right)^2 \cdot \sum_{u=0}^{A-1} \sum_{v=0}^{B-1} \left(D(i_R + u, j_R + v) - \mu(D_{i_R, j_R})\right)^2\right)^{\frac{1}{2}}} \tag{2.42}$$

Em imagens binarizadas, a função de correlação pode ser muito mais simples. Isso porque cada pixel pode possuir apenas 2 valores (0 ou 1). Como as imagens utilizadas neste trabalho eram em preto e branco (binárias), utilizou-se uma função de correlação baseada na função lógica XNOR. A Tabela 2.2 mostra o funcionamento da função.

Tabela 2.2: Tabela verdade da função XNOR

A	B	A $\odot B$
0	0	1
0	1	0
1	0	0
1	1	1

A função de correlação é mostrada em (2.43).

$$COR(i_L, j_L, i_R, j_R) = \sum_{u=0}^{A-1} \sum_{v=0}^{B-1} \left(E(i_L + u, j_L + v) \odot D(i_R + u, j_R + v)\right) \tag{2.43}$$

Essa função apresentaria o mesmo resultado da soma das diferenças absolutas. Entretanto, nesta última, quanto menor o resultado, maior a correlação. Com o XNOR, a correlação aumenta com o resultado da função. Utilizando essa função simples, o esforço computacional foi reduzido bastante.

2.4.2 Geometria epipolar

Com a geometria epipolar, é possível restringir a busca pela correspondência a uma única linha, em vez de toda a imagem.

Sejam m_L e m_R as projeções de um ponto M em relação a duas câmeras, esquerda e direita, cujos centros ópticos são C_L e C_R, respectivamente. Estes cinco pontos pertencem ao plano epipolar, que pode ser observado em amarelo na Figura 2.16, onde $\mathcal{R}_L$ é o plano de retina da câmera da esquerda e $\mathcal{R}_R$ é o plano de retina da câmera da direita. Linha epipolar é a intersecção do plano epipolar e o plano de retina. Os pontos e_L e e_R são, respectivamente, a projeção do ponto C_R na câmera da esquerda, e a projeção do ponto C_L na câmera da direita. Eles são o ponto de intersecção de todas as linhas epipolares e são chamados de epipolos [de Araújo, 2010].

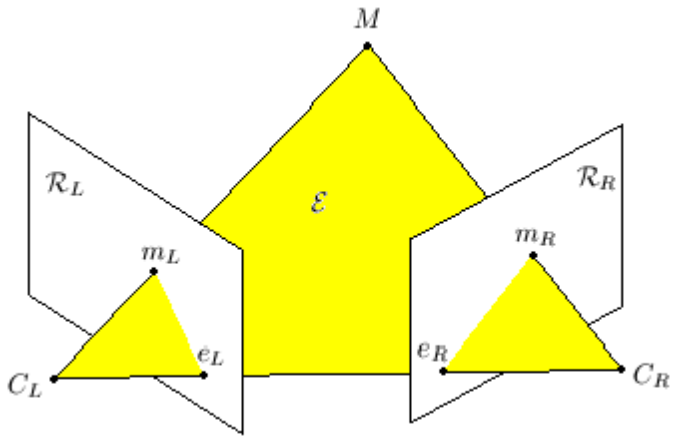


Figura 2.16: Plano epipolar (E).

Dado o ponto m_L, o ponto m_R estará obrigatoriamente na linha epipolar da imagem da direita. Esta linha é a intersecção do plano epipolar (definido pelos pontos m_L, C_L e C_R) com o plano de retina da imagem da direita. Pode-se observar na Figura 2.17 que o ponto m_L, da imagem da esquerda, refere-se a algum ponto da reta r. A projeção da reta r (definida pelos pontos M_1, M_2, M_3 e M_4) na imagem da direita é a linha epipolar p (definida pelos pontos m_1, m_2, m_3 e m_4). Qualquer ponto presente na reta r terá sua projeção na linha p. Logo, a busca pelo ponto correspondente a m_L precisa ser feita apenas na linha p [Faugeras, 1996].

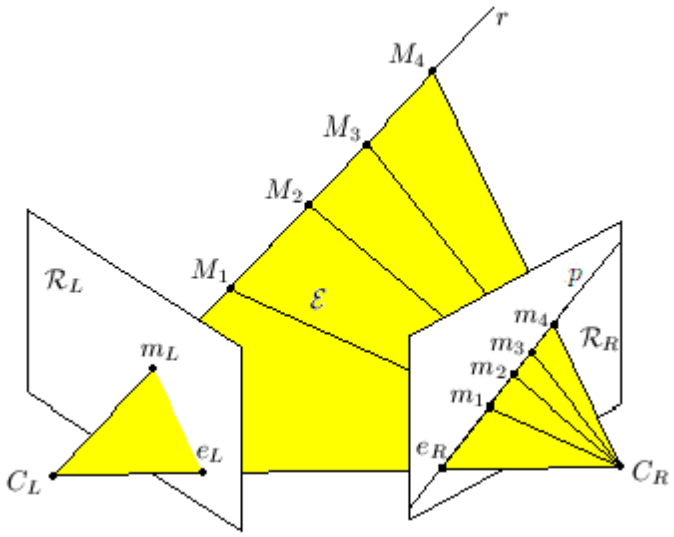


Figura 2.17: Linha epipolar (p).

2.4.3 Retificação

Teoricamente, as duas câmeras podem ocupar qualquer posição. Entretanto, os cálculos são consideravelmente reduzidos na configuração canônica. Nesta configuração, as duas câmeras são postas na mesma linha e com eixos ópticos paralelos, como ilustrado na Figura 2.18. Dessa forma, os epipolos são deslocados para o infinito, e as linhas epipolares serão todas paralelas e horizontais, o que torna a busca mais fácil e menos custosa de ser implementada em software.

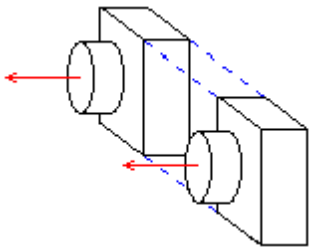


Figura 2.18: Câmeras em posição canônica.

Na retificação, os pontos presentes na coordenada i da imagem da direita terão seus correspondentes na coordenada i da imagem da esquerda (Figura 2.19). Para isso, aplica-se uma transformação em cada imagem do sistema estéreo. Cada câmera, então, passa a ter uma nova matriz de calibração, que se adequa à nova imagem gerada. São as matrizes de calibração retificadoras.

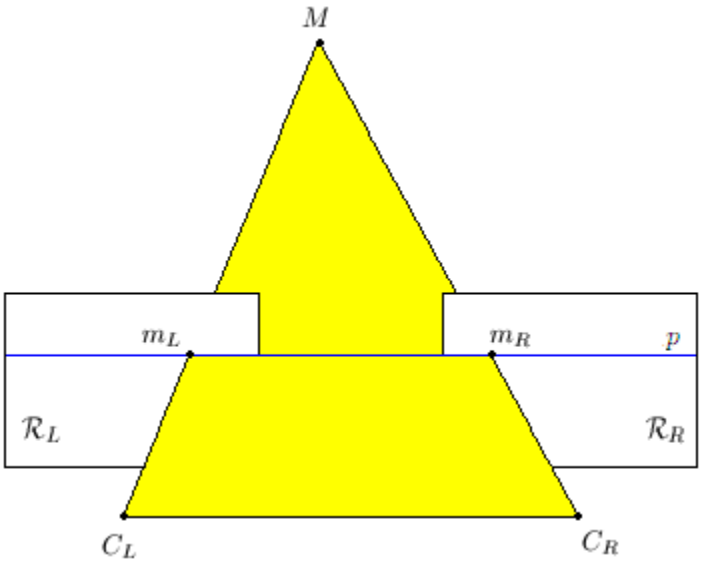


Figura 2.19: Esquema de imagens retificadas: As projeções m_L e m_R estão na mesma linha epipolar p (horizontal)

2.4.4 Reconstrução 3D

Um ponto m de uma imagem é a projeção de um ponto M no espaço. A partir de uma única imagem, é impossível determinar as coordenadas do ponto M. Tudo que se sabe é que ele está situado na reta que une m ao centro óptico da câmera que originou a imagem. Para descobrir as coordenadas exatas do ponto M, é preciso que uma segunda câmera forneça outra imagem do mesmo ponto. A segunda imagem fornece outra reta para a localização do ponto M. Este estará na interseccção das duas retas.

Para reconstituir um ponto no espaço, é preciso saber sua projeção nas duas imagens e as matrizes de calibração das duas câmeras.

Seja M um ponto com coordenadas (x, y, z). (i_L, j_L) e (i_R, j_R) são as coordenadas do pixel na imagem da imagem da esquerda e na imagem da direita, respectivamente. P_L e P_R são as matrizes de calibração da câmera da esquerda e da câmera da direita, respectivamente. P_L e P_R são definidas em (2.44) e (2.45).

$$P_L = \begin{bmatrix} l_{11} & l_{12} & l_{13} & l_{14} \\ l_{21} & l_{22} & l_{23} & l_{24} \\ l_{31} & l_{32} & l_{33} & l_{34} \end{bmatrix} \tag{2.44}$$

$$P_R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} \\ r_{21} & r_{22} & r_{23} & r_{24} \\ r_{31} & r_{32} & r_{33} & r_{34} \end{bmatrix} \tag{2.45}$$

A partir das equações de (2.39), pode-se montar o sistema (2.46).

$$\begin{cases} (l_{11} - j_L \cdot l_{31})x + (l_{12} - j_L \cdot l_{32})y + (l_{13} - j_L \cdot l_{33})z = j_L \cdot l_{34} - l_{14} \\ (l_{21} - i_L \cdot l_{31})x + (l_{22} - i_L \cdot l_{32})y + (l_{23} - i_L \cdot l_{33})z = i_L \cdot l_{34} - l_{24} \\ (r_{11} - j_R \cdot r_{31})x + (r_{12} - j_R \cdot r_{32})y + (r_{13} - j_R \cdot r_{33})z = j_R \cdot r_{34} - r_{14} \\ (r_{21} - i_R \cdot r_{31})x + (r_{22} - i_R \cdot r_{32})y + (r_{23} - i_R \cdot r_{33})z = i_R \cdot r_{34} - r_{24} \end{cases} \tag{2.46}$$

Considerando (2.47) e (2.48), pode-se colocar o sistema em forma de matriz, de acordo com (2.49).

$$F = \begin{bmatrix} (l_{11} - j_L \cdot l_{31}) & (l_{12} - j_L \cdot l_{32}) & (l_{13} - j_L \cdot l_{33}) \\ (l_{21} - i_L \cdot l_{31}) & (l_{22} - i_L \cdot l_{32}) & (l_{23} - i_L \cdot l_{33}) \\ (r_{11} - j_R \cdot r_{31}) & (r_{12} - j_R \cdot r_{32}) & (r_{13} - j_R \cdot r_{33}) \\ (r_{21} - i_R \cdot r_{31}) & (r_{22} - i_R \cdot r_{32}) & (r_{23} - i_R \cdot r_{33}) \end{bmatrix} \tag{2.47}$$

$$G = \begin{bmatrix} j_L \cdot l_{34} - l_{14} \\ i_L \cdot l_{34} - l_{24} \\ j_R \cdot r_{34} - r_{14} \\ i_R \cdot r_{34} - r_{24} \end{bmatrix} \tag{2.48}$$

$$F \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = G \tag{2.49}$$

A solução fornece (2.50).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = F^{-1} \cdot G \tag{2.50}$$

Se as imagens estiverem retificadas, P_L e P_R são as matrizes de calibração retificadoras, e a matriz de rotação R é a matriz identidade, pois não há rotação entre as câmeras na configuração canônica. $\left(R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right)$.

2.5 REDES NEURAIS ARTIFICIAIS

A inteligência artificial tem como objetivo simular a capacidade racional de resolver problemas. Os sistemas inteligentes têm como característica a capacidade de aprender e adaptar-se.

As redes neurais artificiais são um método de resolver problemas de inteligência artificial que tem como base a fisiologia do sistema nervoso. Uma rede neural pode ter centenas de unidades de processamento. Cada uma dessas unidades é chamada de neurônio artificial. Um neurônio artificial recebe e transmite informações, assim como um neurônio natural.

O neurofisiologista Warrem MacCulloch criou um modelo para o neurônio biológico, no qual este era representado por um circuito com entradas binárias produzindo uma saída também binária. As entradas eram submetidas a uma soma ponderada, criando uma entrada efetiva. O valor da saída dependeria de a entrada efetiva atingir ou não um certo valor (Figura 2.20).

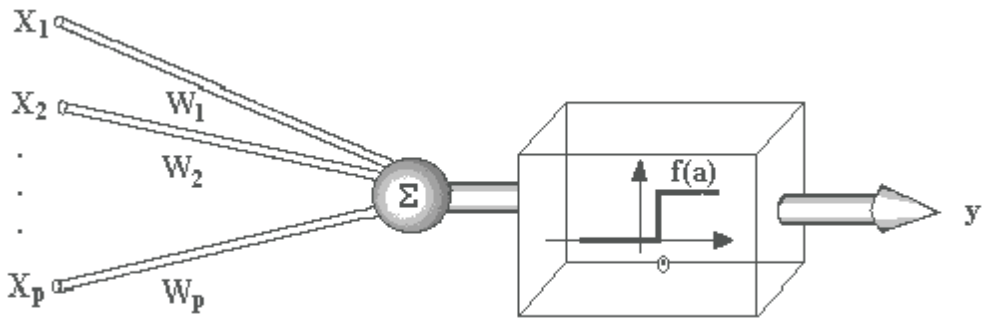


Figura 2.20: Modelo do neurônio de MacCulloch [net, n.d.].

Um neurônio é frequentemente modelado segundo a Figura 2.21.

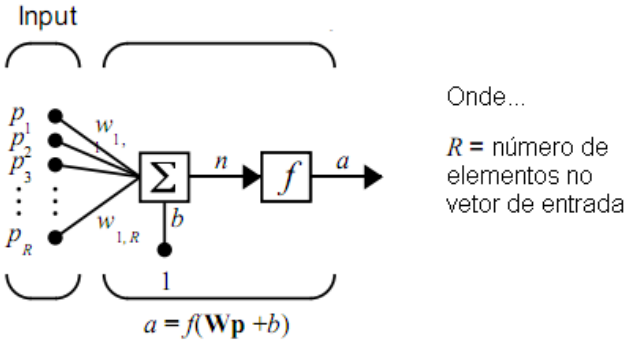


Figura 2.21: Modelo de neurônio [NNT, 2006].

O vetor $p = [p_1 \ \ p_2 \ \ p_3 \ \ ... \ \ p_R]^T$ é a entrada do sistema. $W = [w_{1,1} \ \ w_{1,2} \ \ w_{1,3} \ \ ... \ \ w_{1,R}]$ são os pesos de cada entrada. À soma de todas as entradas com seus respectivos pesos, é adicionado um valor de ajuste b, para formar, assim, o elemento n, descrito em (2.51). Este é o parâmetro da função de ativação f, que fornece a saída a [NNT, 2006].

$$n = W \cdot p + b \tag{2.51}$$

As funções de ativação $a = f(n)$ mais comuns são essas mostradas na Figura 2.22.

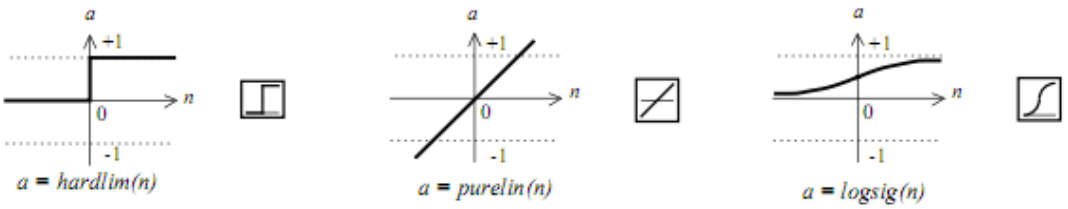


Figura 2.22: Exemplos de funções de ativação [NNT, 2006].

Uma rede neural é um conjunto de neurônios interligados através de conexões sinápticas. Ela geralmente é composta por três tipos de neurônios. Há os neurônios de entrada, que recebem estímulo diretamente do meio externo. Há os neurônios de saída, que enviam sinais para o meio externo. E existem ainda os neurônios internos, cujas entradas e a saída não estão ligadas ao meio externo, mas conectadas a outros neurônios. Esses neurônios são grupados em camadas como pode ser visto na Figura 2.23. Uma rede neural com essa topologia e sem realimentação é do tipo *Feedforward* Multicamadas. E este será o tipo utilizado neste trabalho.

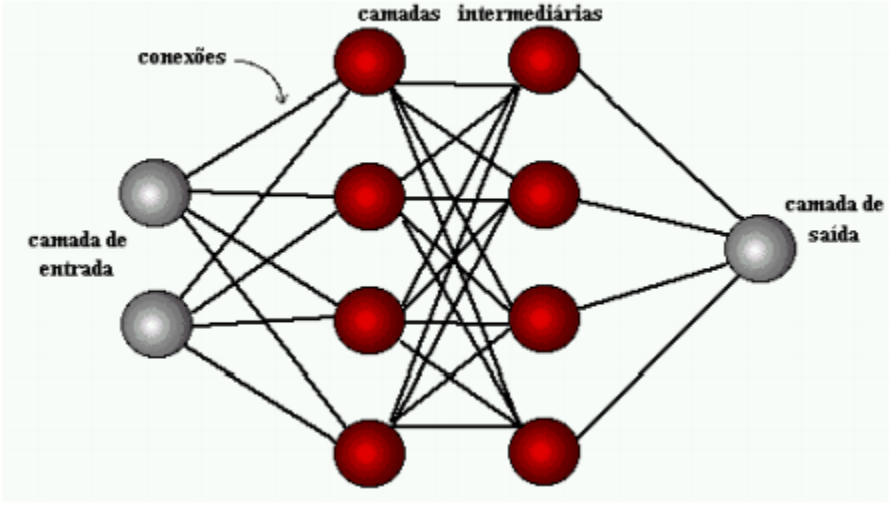


Figura 2.23: Camadas de uma rede neural [net, n.d.].

Um único neurônio não é capaz de aprender. Para isso, é necessário uma rede com vários neurônios. Os neurônios internos têm uma importância crucial, pois permitem solucionar problemas de alta complexidade. Uma das propriedades mais interessantes das redes neurais é a capacidade de aprender e, assim, melhorar seu desempenho.

Para uma rede neural funcionar apropriadamente, ela precisa de treinamento. O treinamento é feito através da apresentação de exemplos reais e de um processo iterativo de ajustes aplicado aos seus pesos. Os pesos das conexões vão sendo ajustados de acordo com os

exemplos apresentados. Existem vários tipos de algoritmos de aprendizado específicos para determinados modelos de redes neurais. Estes algoritmos diferem principalmente pelo modo como os pesos são modificados.

Uma rede neural pode relacionar-se com o ambiente segundo os seguintes paradigmas de aprendizado:

- Aprendizado Supervisionado: quando é utilizado um agente externo que indica à rede a resposta desejada para o padrão de entrada;
- Aprendizado Não Supervisionado: quando não existe uma agente externo indicando a resposta desejada para os padrões de entrada;
- Reforço: quando um crítico externo avalia a resposta fornecida pela rede.

Denomina-se ciclo a apresentação de todos os pares de entrada e saída do conjunto de treinamento no processo de aprendizado. A correção dos pesos em um ciclo pode ser executado de duas maneiras:

- Modo Padrão: A correção dos pesos acontece a cada apresentação à rede de um exemplo do conjunto de treinamento. Cada correção de pesos baseia-se somente no erro do exemplo apresentado naquela iteração.
- Modo Batch: Apenas uma correção é feita por ciclo. Todos os exemplos do conjunto de treinamento são apresentados à rede e é calculado um erro médio para a realização as correções dos pesos.

O algoritmo Backpropagation é um dos mais utilizados para aprendizagem. Nesse treinamento, valores aleatórios são atribuídos aos pesos. Um padrão é apresentado à camada de entrada da rede e uma resposta é produzida pela camada de saída. A saída obtida é comparada à saída desejada e o erro é calculado. Se o erro estiver acima dos limites aceitáveis, ele é propagado a partir da camada de saída até a camada de entrada, e os pesos das conexões das unidades das camadas internas vão sendo modificados [net, n.d.].

3 PROCEDIMENTOS

Neste capítulo são explicados os procedimentos executados ao longo do projeto:

- Calibração da câmeras.
- Aquisição das imagens.
- Segmentação (detecção de bordas).
- Retificação, correspondência e reconstrução 3D (visão estéreo).
- Reprojeção dos pontos.
- Treinamento e teste da rede neural artificial.

3.1 APARATO EXPERIMENTAL

No Laboratório de Automação e Robótica da Universidade de Brasília (LARA), há uma amostra do cabo condutor da linha de transmissão, com dois espaçadores (Figura 3.1). Para a realização do trabalho, tem-se dois computadores a disposição: um Intel Core i7 64 bits de 2,67 GHz com memória RAM de 8GB, e outro Intel Pentium 4 32 bits de 3,06GHz com memória RAM de 1GB. Ambos com sistema operacional Windows XP Professional.

Figura 3.1: Amostra de cabo presente no LARA.

Há também duas câmeras da *Videre Design* modelo STH-MDCS3-VARX, que, juntas, formam o sistema estéreo (Figura 3.2). A comunicação é feita através de uma placa *Firewire*

(IEEE 1394) instalada na máquina com Core i7. *Firewire* é uma interface serial parecida com USB. Uma das principais diferenças é a velocidade de transferência. Enquanto o USB transfere dados a, no máximo, 12 Mbps, o *Firewire* transfere a 400 Mbps. A interface 1394, portanto, é mais adequada quando se lida com grandes quantidades de dados, como é o caso de imagens de alta resolução. Cada câmera utiliza uma porta *Firewire*.

Figura 3.2: Câmera STH-MDCS3-VARX.

Para os testes no laboratório, as câmeras são fixadas a um suporte. As imagens são adquiridas com o auxílio do programa *SRI Small Vision System* (Figura 3.3), que pode ser obtido no site do fabricante http://www.videredesign.com/.

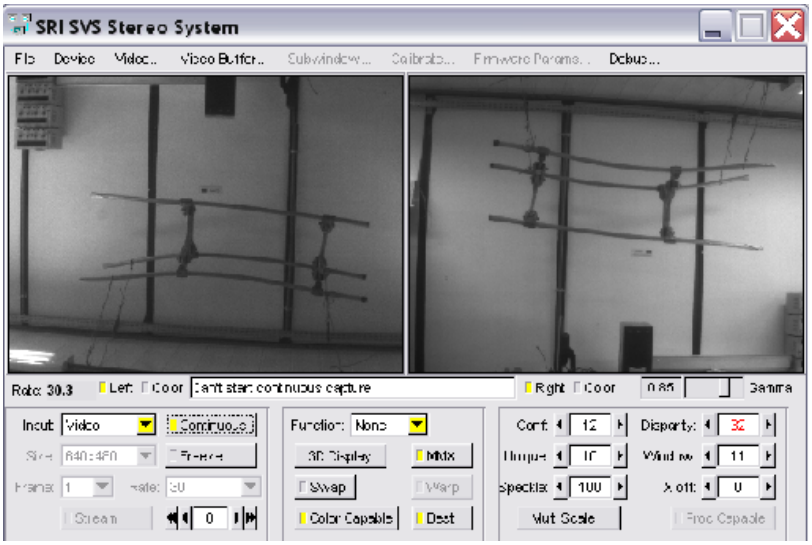


Figura 3.3: Interface do software SRI Small Vision System.

Pode-se observar que uma das câmeras está invertida em relação à outra. Isso não é, de fato, um problema, pois a imagem pode ser ajustada via *software* através de uma rotação de 180°.

A maior parte do trabalho foi realizado com o MATLAB, uma linguagem de alto nível e ambiente interativo que permite criar programas de forma mais rápida que a maioria das linguagens [Matworks, 2011].

3.2 CALIBRAÇÃO

A primeira tarefa a ser realizada é a calibração das câmeras. Para isso, foi utilizada a *toolbox Camera Calibration* para o MATLAB da Caltech (*California Institute of Technology*). Com essa *toolbox* também foi possível realizar a retificação.

Utilizou-se o software SVS (Figura 3.3) para colher 16 pares de imagens estéreo. As imagens eram constituídas de fotos de um tabuleiro (Figura 3.4) no qual os vértices são facilmente identificados. A resolução utilizada foi de 1280x960 (size), com FPS de 7,5 (rate). As imagens foram salvas no formato Bitmap Image (.bmp). Todas foram salvas com o nome seguindo o padrão "imageLX", para as imagens da esquerda, ou "imageRX", para as imagens da direita, onde X é um número de 1 a 16. Estas imagens foram colocadas na mesma pasta da *toolbox* "*toolbox_calib*". O exemplo mostrado em [Caltech, 2010] mostra como utilizar essa ferramenta para a calibração de cada câmera. As duas câmeras foram calibradas separadamente e depois realizou-se a calibração estéreo, para encontrar os parâmetros que montam as matrizes de calibração retificadoras.

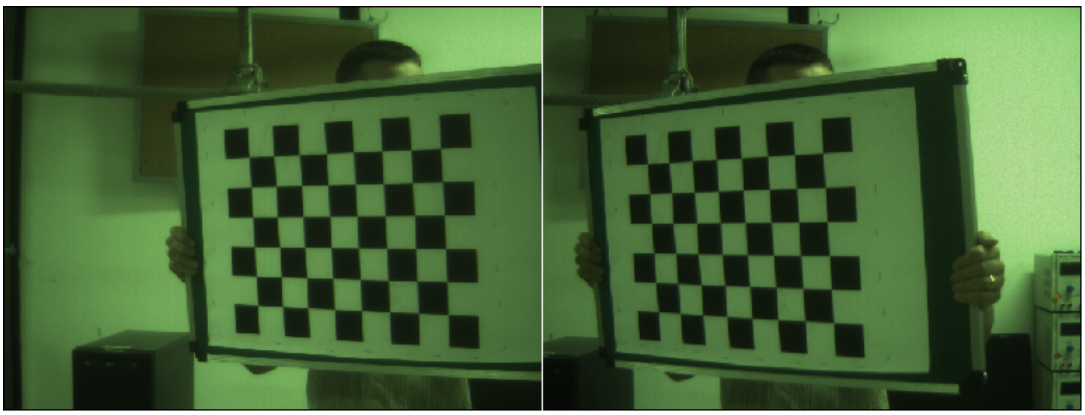


Figura 3.4: Imagens do tabuleiro usado na calibração.

A fim de facilitar os cálculos, todas as coordenadas terão como base a câmera da esquerda, ou seja, o sistema de coordenadas externo será coincidente com sistema de coordenadas da câmera da esquerda. O centro ótico da câmera esquerda é a origem do sistema cartesiano. A coordenada z coincide com o eixo focal da câmera da esquerda e o eixo x é paralelo ao eixo u da imagem da câmera da esquerda. A Figura 3.5 ilustra essa situação.

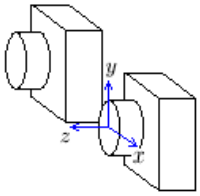


Figura 3.5: Posição do sistema de coordenadas de mundo.

3.3 AQUISIÇÃO DE IMAGENS

Em seguida, foi realizada a aquisição de imagens. Foram obtidas imagens de diferentes posições, tanto de angulação como de distância. As coordenadas em si não são importantes. O que realmente interessa é o formato da imagem. Foram tiradas 69 pares de fotos de garras ruins, com distância variando de 1m a 1,5m e angulação de 0 a 20°, e 80 pares de fotos de garra presa corretamente, com distância de 1m a 1,4m e angulação de 0 a 45°.

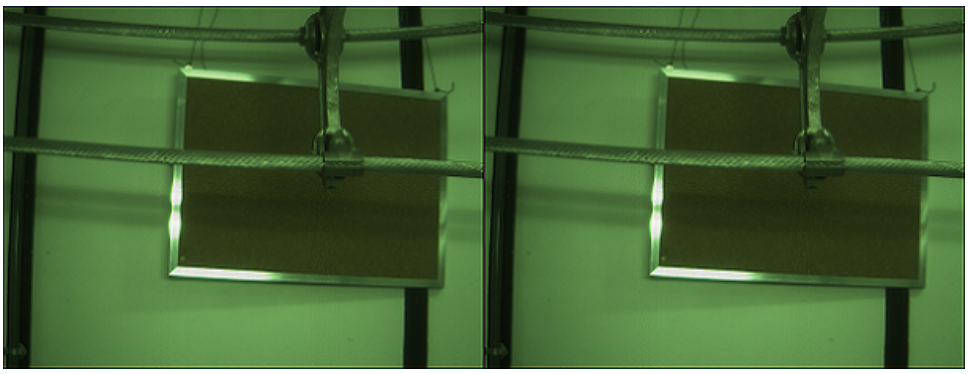


Figura 3.6: Exemplo de imagens adquiridas com as câmeras.

3.4 DETECÇÃO DE BORDAS

Depois de obtidas as imagens, foi feita a segmentação do cabo. Uma questão crucial para a segmentação é o problema do plano de fundo. Por trás do cabo pode haver qualquer tipo de objeto, cujas cores podem assemelhar-se à malha do cabo. Por isso, a segmentação é uma das tarefas mais complexas. Entretanto, o trabalho de [Levin *et al.*, 2006] foi de grande valia para a resolução desse problema. Com ele, pôde-se fazer a segmentação do cabo a partir da derivada de uma função de custo baseada na suavidade das cores de *foreground* e *background*. Na expressão resultante, é possível eliminar analiticamente as cores do *foreground* e *background* para obter uma função de custo quadrática em *alpha*.

Para permitir uma eficiente detecção de bordas, foi feita a seleção de uma área de interesse. Essa seleção foi feita manualmente. O usuário seleciona a pequena região entre as

garras que será a máscara de foreground central. As demais máscaras são formadas, dentro de uma região de interesse de dimensões 251×321 (Figura fig:cap5:mask). A partir dessas máscaras, foi possível a obtenção da imagem α do cabo. Algumas operações morfológicas foram usadas para obter um contorno mais preciso. A questão da iluminação, que não era muito eficiente dentro do laboratório, foi crucial nesta etapa.

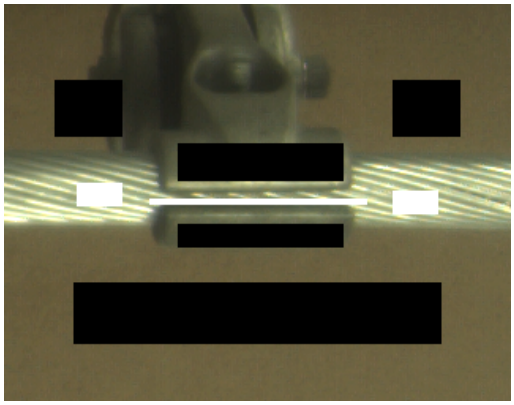


Figura 3.7: Máscaras.

Com a detecção de bordas, obteve-se uma imagem binarizada. Foi criada uma outra imagem binarizada preenchida com zeros, com as mesmas dimensões da imagem original (1280x960). A imagem com as bordas foi sobreposta a essa segunda imagem binarizada na mesma posição em que ela foi retirada da imagem original. Assim, obteve-se uma imagem binarizada com o tamanho da imagem original e com as bordas detectadas. Esse procedimento foi feito para facilitar a retificação das imagens estéreo.

3.5 VISÃO ESTÉREO

A retificação é feita utilizando a *toolbox* "*Camera Calibration*" do MATLAB. Antes de fazer a retificação, as imagens do tabuleiro, dentro da pasta "*toolbox_calib*", foram substituídas pelas imagens da borda do cabo. São as imagens resultantes do processo anterior de detecção de bordas.

A correspondência é outra questão complexa, pois depende da qualidade da detecção de bordas. O par de imagens retificadas é passado como parâmetro no programa 'corresp.m', em anexo, para que seja criada uma matriz de correspondência. Esta matriz está na forma $[\mathbf{Y}\ \mathbf{X_L}\ \mathbf{Y_R}]$, onde cada linha possui a coordenada i, igual para as duas imagens, a coordenada j_L, do pixel da esquerda, e a coordenada j_R, do pixel da direita, respectivamente. O pixel (i, j_L) da imagem da esquerda corresponde ao pixel (i, j_R) da imagem da direita. Os pixels da imagem foram percorridos na ordem em que formam a borda, em vez de percorrer linha por linha. Isso facilitou as etapas posteriores em que se pretendia refazer a borda apenas ligando os pontos. O programa utiliza a função descrita em (2.43) para fazer a correlação.

Correlação abaixo de 75% são desprezadas. Também é desprezado o caso em que o pixel da direita possui máxima correlação com mais de um pixel da imagem da esquerda.

O programa 'reconsti.m', utiliza a fórmula (2.49) para calcular as coordenadas dos pontos no espaço.

3.6 REPROJEÇÃO

De posse das coordenadas dos pontos, procurou-se uma forma mais simples e eficaz de analisar esses pontos. Dessa forma, optou-se por reprojetar os pontos em um plano. Assim, a análise se restringe a duas dimensões, e não três.

Sabendo a posição da garra, pode-se aplicar uma transformação em cada ponto. Uma matriz de projeção perspectiva é calculada, como se uma outra câmera fosse colocada numa posição mais adequada para conseguir uma imagem mais fácil de ser processada. Essa câmera deverá ter sua posição recalculada para cada par de imagens, de forma a estar sempre à mesma distância do cabo e de frente para a garra. Assim as imagens sempre teriam o mesmo padrão, independente da posição das câmeras reais. A Figura 3.8 ilustra essa idéia. As câmeras do sistema estéreo são mostradas em verde e a câmera virtual é mostrada em azul.

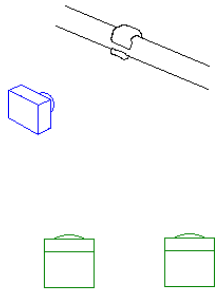


Figura 3.8: Câmera virtual (mostrada em azul).

De posse das coordenadas tridimensionais dos pontos, através de uma aproximação por mínimos quadrados, é calculado o plano que busca englobar todos os pontos. Depois, calcula-se um um vetor perpendicular ao plano, com origem no ponto médio P_M de todos os pontos, que aponte para a posição da câmera virtual. Esse vetor é usado para calcular a matriz de rotação desta câmera. A função 'calculo_camera_virtual.m' calcula T e R para a construção da matriz dos parâmetros extrínsecos da câmera virtual. A distância do centro focal ao ponto P_M é de aproximadamente 200mm. Os parâmetros intrínsecos são definidos como $f/du = f/dv = 1000$ e $i_0 = j_0 = 500$. Assim, a matriz de projeção perspectiva é

dada por:

$$P = \begin{bmatrix} 1000 & 0 & 500 & 0 \\ 0 & -1000 & 500 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}^{-1}$$

Uma nova imagem é formada usando essa transformação. Foram calculados os coeficientes de Fourier da imagem final. O programa 'ccf.m' tem como entrada a imagem binária da borda e retorna o código de cadeia. O programa 'coeficientes.m' retorna os coeficientes de Fourier para 25 hamônicos em função do código de cadeia do respectivo contorno.

3.7 TREINAMENTO DA REDE NEURAL

A entrada da rede neural são os coeficientes de Fourier. Os coeficientes A_0 e C_0 são desprezados, pois o resultado deve ser invariante à translação. Os demais coeficientes são reagrupados para construir uma matriz coluna na forma $[A_n\ B_n\ Cn\ Dn]^T$. A entrada da rede neural é uma matriz na forma $100 \times N$, onde N é a quantidade de imagens. O *target* é definido como 1 para garra presa e 0 para garra defeituosa. O programa 'treinamento.m' separa automaticamente os dados de treinamento e de testes e faz o treinamento da rede.

A rede foi criada com duas camadas. Baseado na observação experimental, utilizou-se 20 neurônios na primeira camada e 15 na segunda. A função de ativação utilizada foi a tangente hiperbólica (tansig). O número máximo de ciclos selecionado foi de 1000, mas o treinamento sempre parava em torno de 13.

Existem vários algoritmos de *Backpropagation*. Para o aprendizado da rede, foi escolhido o algoritmo quasi-Newton (trainbfg) [NNT, 2006].

Ao todo, foram usadas 119 imagens para treinar e testar a rede neural, sendo 80% para treino e 20% para validação. O conjunto consistia de 69 imagens de garras ruins e 50 imagens de garras boas. As fotografias adquiridas a partir de uma angulação maior que 20° não foram utilizadas nesta etapa, pois a detecção de bordas não foi satisfatória.

4 RESULTADOS

Neste capítulo são apresentados os resultados obtidos com os procedimentos.

4.1 CALIBRAÇÃO

A toolbox "*Camera Calibration*" do MATLAB fornece as características da câmera, que podem equivaler aos parâmetros intrínsecos de acordo com a Tabela 4.1.

Tabela 4.1: Equivalência dos parâmetros intrínsecos no MATLAB.

Parâmetro	Notação no MATLAB
f/du	fc(1)
f/dv	fc(2)
j_0	cc(1)
i_0	cc(2)

A calibração individual das câmeras fornece o resultado mostrado nas Tabelas 4.2 e 4.2.

Tabela 4.2: Calibração individual da câmera esquerda.

Parâmetro	Valor
fc	[2283.27186 2288.18143] $\pm$ [16.96283 16.43303]
cc	[607.25195 418.14516] $\pm$ [30.21853 22.79467]
alpha_c	[0.00000] $\pm$ [0.00000] => angle of pixel axes = 90.00000 $\pm$ 0.00000 degrees
kc	[-0.39898 -0.11584 0.00317 0.00318 0.00000]
err	[0.36679 0.33445]

Tabela 4.3: Calibração individual da câmera direita.

Parâmetro	Valor
fc	[2309.28563 2315.42982] ± [15.79085 15.76480]
cc	[625.45216 415.30385] ± [29.05202 23.08584]
alpha_c	[0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees
kc	[-0.42271 0.50382 -0.00196 -0.00096 0.00000]
err	[0.34700 0.33963]

A calibração estéreo fornece os dados da Tabela 4.4.

Tabela 4.4: Calibração estéreo.

Parâmetro	Valor
fc_left	[2286.92038 2294.76115] ± [10.80618 10.51902]
cc_left	[623.10028 432.23779] ± [31.75856 24.32184]
alpha_c_left	[0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees
kc_left	[-0.38921 -0.28200 0.00163 0.00234 0.00000]
fc_right	[2318.20664 2326.87757] ± [10.22096 10.25832]
cc_right	[632.93017 391.44766] ± [29.19063 23.98784]
alpha_c_right	[0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees
kc_right	[-0.38858 0.26980 -0.00242 -0.00058 0.00000]
om	[-0.02511 -0.01165 -0.00784] ± [0.01418 0.01773 0.00085]
T	[-243.22407 -0.79661 5.18889] ± [0.42066 0.43409 4.72103]

A matriz de rotação referente ao vetor 'om' é dada por R = $\begin{bmatrix} 0.9999 & 0.0080 & -0.0115 \\ -0.0077 & 0.9997 & 0.0252 \\ 0.0117 & -0.0251 & 0.9996 \end{bmatrix}$.

Pode-se agora, calcular as matrizes de calibração das duas câmeras. A matriz dos parâmetros intrínsecos é dada por (4.1).

$$K = \begin{bmatrix} \mathrm{fc}(1) & 0 & \mathrm{cc}(1) & 0 \\ 0 & -\mathrm{fc}(2) & \mathrm{cc}(2) & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{4.1}$$

Logo, os valores de K para as duas câmeras serão:

$$K_L = \begin{bmatrix} 2286.92038 & 0 & 623.10028 & 0 \\ 0 & -2294.76115 & 432.23779 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

e

$$K_R = \begin{bmatrix} 2318.20664 & 0 & 632.93017 & 0 \\ 0 & -2326.87757 & 391.44766 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

A matriz dos parâmetros extrínsecos depende da posição das câmeras. Como as coordenadas de mundo são coincidentes com as coordenadas da câmera esquerda, a matriz dos parâmetros extrínsecos desta será a matriz identidade. Logo:

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

e

$$L_R = \begin{bmatrix} 0.9999 & 0.0080 & -0.0115 & -243.22407 \\ -0.0077 & 0.9997 & 0.0252 & -0.79661 \\ 0.0117 & -0.0251 & 0.9996 & 5.18889 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Como $P = K \cdot L$, tem-se:

$$P_L = \begin{bmatrix} 2286.92038 & 0 & 623.10028 & 0 \\ 0 & -2294.76115 & 432.23779 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P_R = 1.0e+005 \cdot \begin{bmatrix} 0.0233 & 0.0000 & 0.0061 & -5.6056 \\ 0.0002 & -0.0234 & 0.0033 & 0.0388 \\ 0.0000 & -0.0000 & 0.0000 & 0.0001 \end{bmatrix}$$

A Tabela 4.5 mostra os resultados após a retificação. A matriz de rotação 'R_new' é a matriz identidade 3×3.

Tabela 4.5: Calibração estéreo retificada.

Parâmetro	Valor
fc_left	[2186 2186]
cc_left_new	[700.5677 408.9473]
fc_right_new	[2186 2186]
cc_right_new	[682.3864 408.9473]
om_new	[0 0 0]
T_new	[-243.2807 0.0000 -0.0000]

Os valores de K e L são, então, reescritos para as imagens retificadas:

$$K_L = \begin{bmatrix} 2186 & 0 & 700.5677 & 0 \\ 0 & -2186 & 408.9473 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$K_R = \begin{bmatrix} 2186 & 0 & 682.3864 & 0 \\ 0 & -2186 & 408.9473 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

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

$$L_R = \begin{bmatrix} 1 & 0 & 0 & -243.2807 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Logo:

$$P_L = \begin{bmatrix} 2186 & 0 & 700.5677 & 0 \\ 0 & -2186 & 408.9473 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$P_R = 1.0e+005 \cdot \begin{bmatrix} 0.0219 & 0 & 0.0068 & -5.3181 \\ 0 & -0.0219 & 0.0041 & -0.0000 \\ 0 & 0 & 0.0000 & -0.0000 \end{bmatrix}$$

4.2 DETECÇÃO DE BORDAS E RETIFICAÇÃO

Depois de selecionada a região de interesse (ROI), são aplicadas as máscaras. A preta é um *background* e a branca é um *foreground*. A Figuras 4.1 e 4.2 mostram regiões de

interesse. Nas Figuras 4.3 e 4.4, as mesmas regiões com as máscaras.

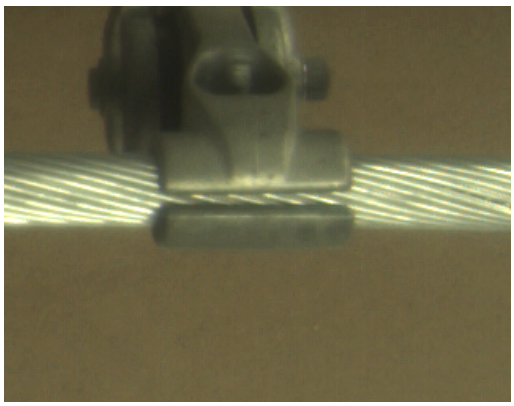


Figura 4.1: ROI (garra sem defeitos).

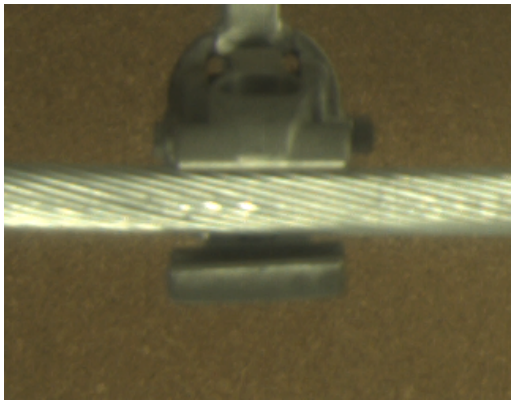


Figura 4.2: ROI (garra com defeito).

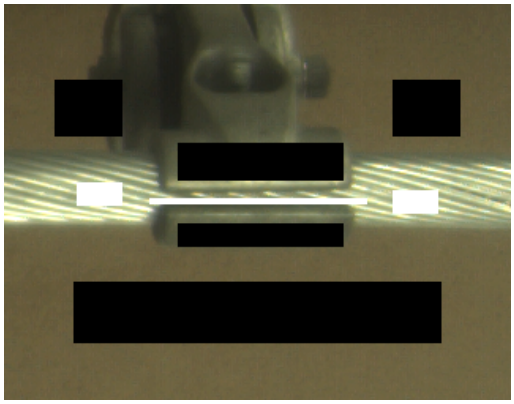


Figura 4.3: Máscaras (garra sem defeitos).

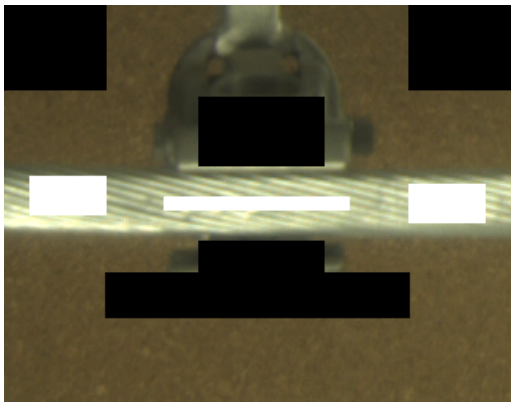


Figura 4.4: Máscaras (garra com defeito).

As Figuras 4.5 e 4.6 mostram as bordas detectadas. Depois que o tamanho das imagens foi restituido, o resultado é visto nas Figuras 4.7 e 4.8.

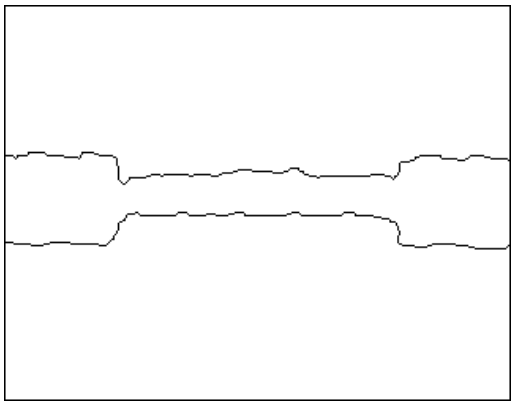


Figura 4.5: Bordas (garra sem defeitos).

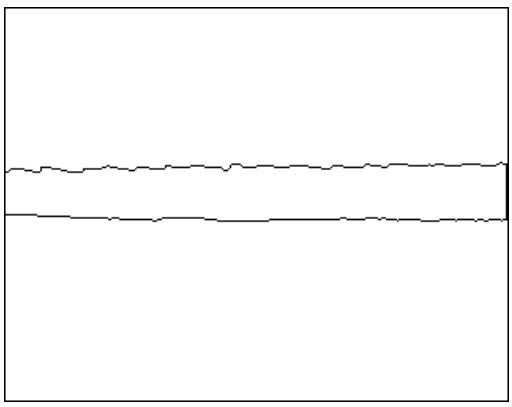


Figura 4.6: Bordas (garra com defeito).

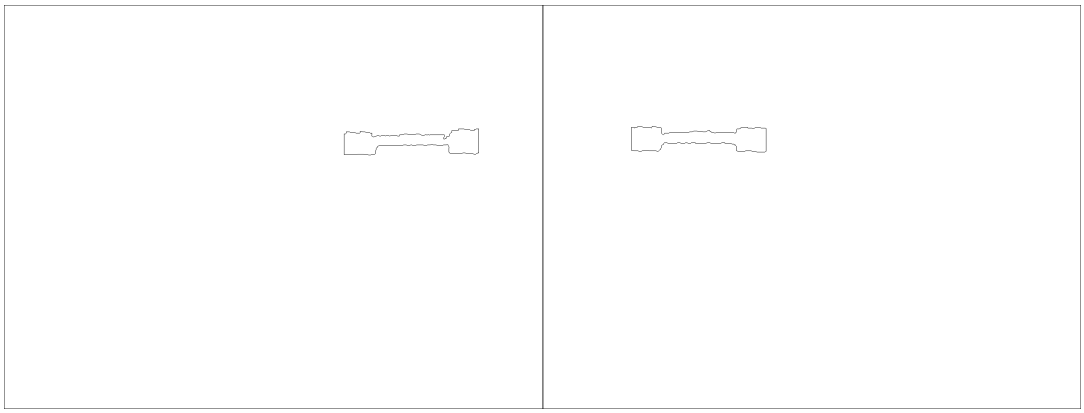


Figura 4.7: Bordas com imagem em tamanho original (garra sem defeitos).

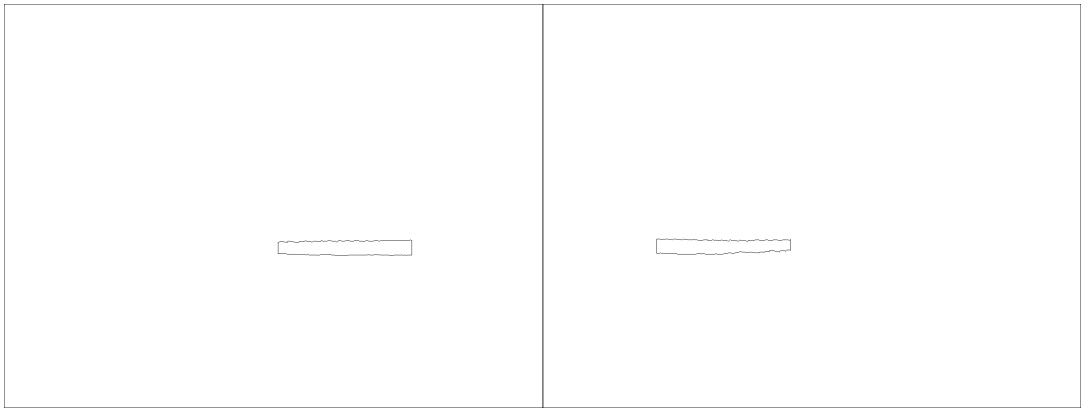


Figura 4.8: Bordas com imagem em tamanho original (garra com defeito).

As Figuras 4.9 e 4.10 mostram as imagens retificadas.

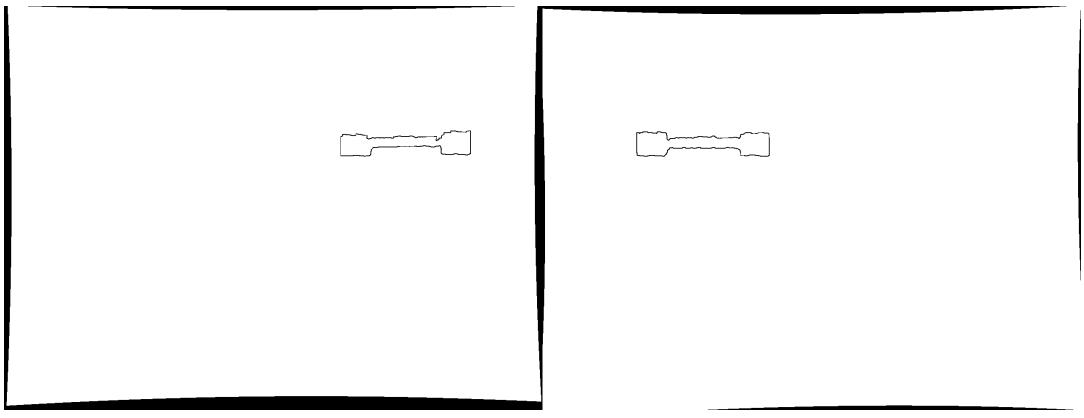


Figura 4.9: Imagens retificadas (garra sem defeitos).

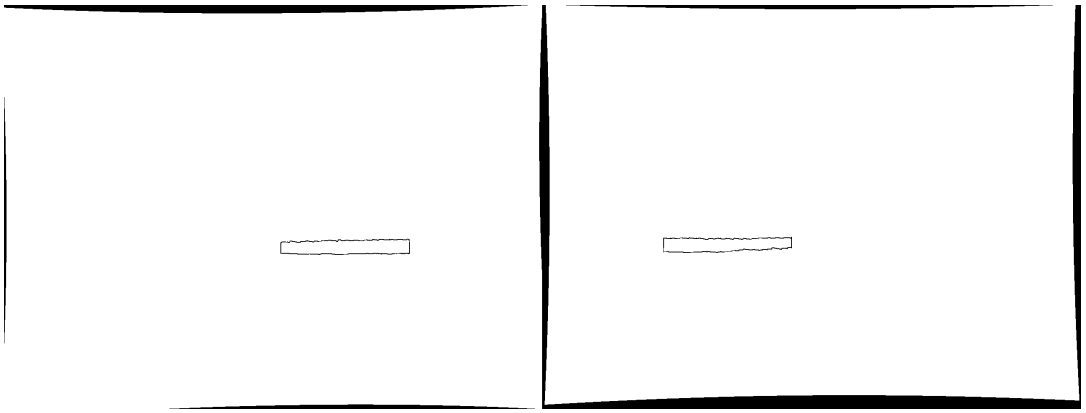


Figura 4.10: Imagens retificadas (garra com defeito).

Fotos tiradas com uma angulação muito grande não tiveram uma boa detecção de bordas (Figura 4.11) .

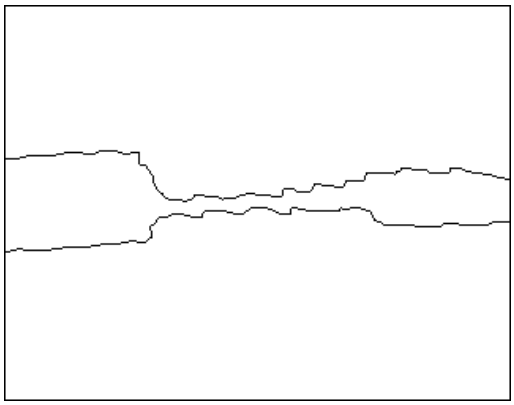


Figura 4.11: Detecção de bordas para uma imagem com angulação de 30°).

4.3 CORRESPONDÊNCIA

Como as imagens estão retificadas, as coordenadas i de dois pontos correspondentes nas duas imagens é a mesma. A Tabela 4.6 mostra um resultado de correspondência para um par de imagens estéreo.

Tabela 4.6: Matriz de correspondência.

i	j_L	j_R	i	j_L	j_R	i	j_L	j_R	i	j_L	j_R
310	862	295	310	863	296	310	864	297	311	864	297
312	864	297	313	864	297	314	864	297	315	864	297
316	864	297	317	865	298	317	870	302	317	871	302
317	872	302	316	873	303	316	874	303	315	878	309
315	879	310	315	880	310	316	881	314	316	882	314
316	883	314	315	884	314	315	885	314	315	886	314
315	887	314	315	888	314	315	889	314	315	890	323
315	891	324	315	892	325	315	894	324	315	895	325
315	896	325	315	897	325	315	898	325	315	899	325
315	900	325	315	901	323	315	903	324	316	904	325
315	908	341	315	918	341	315	919	341	315	920	341
315	921	342	316	923	344	316	925	355	315	926	359
314	926	359	313	929	362	312	931	362	312	933	366
312	934	367	312	935	368	312	936	368	311	937	368
311	938	368	311	939	368	311	940	368	312	945	371
312	946	371	312	947	380	312	948	380	312	949	380
312	950	380	312	951	380	312	952	380	312	953	380
312	954	380	312	955	380	312	956	380	312	957	380
312	958	380	312	959	380	312	960	393	311	964	397
311	965	398	311	966	399	311	967	400	311	968	393
311	969	393	311	970	393	311	971	393	311	972	393
315	979	400	315	980	413	314	982	403	313	992	419
313	993	420	313	994	426	313	995	427	313	996	427
313	997	427	312	1023	453	312	1024	453	312	1025	453
316	1038	467	316	1039	468	315	1040	470	314	1040	470
311	1040	471	310	1042	472	308	1044	475	308	1045	476
307	1046	477	303	1049	482	303	1050	483	303	1051	483
303	1052	483	302	1062	495	301	1062	495	354	1077	506
354	1076	506	354	1075	506	354	1074	506	356	1052	473
356	1051	472	356	1050	471	355	1042	471	354	1041	471
354	1040	471	353	1040	471	352	1039	471	349	1039	470
348	1039	470	347	1039	471	346	1039	471	345	1039	471
343	1039	470	342	1039	470	341	1039	469	340	1039	469
337	1027	456	337	1026	456	337	1014	437	337	1013	437
337	1012	437	337	1011	437	337	990	411	337	989	411
337	988	411	337	981	402	337	950	376	337	949	376

i	j_L	j_R	i	j_L	j_R	i	j_L	j_R	i	j_L	j_R
337	948	376	337	947	376	337	946	376	337	938	366
337	937	366	337	936	366	339	890	311	339	879	303
340	878	302	341	876	300	342	876	300	343	875	298
344	875	298	345	874	298	346	874	298	347	874	298
348	874	298	349	873	298	350	873	297	351	873	297
352	873	296	354	873	295	355	873	295	356	872	295
359	869	290	359	868	290	359	867	290	360	858	290
360	857	290	360	856	289	359	853	285	359	852	285

4.4 RECONSTITUIÇÃO 3D E REPROJEÇÃO

Para a correspondência da Tabela 4.6, encontrou-se os pontos plotados na Figura 4.12.

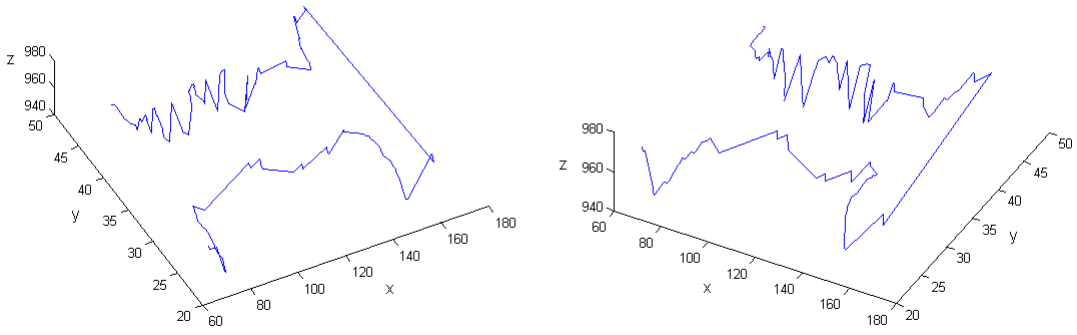


Figura 4.12: Plot 3D.

O plano que melhor se encaixa nos pontos amostrados é dado pela equação $-0.0243x - 0.2362y + 0.9714z - 921.9954 = 0$. Os parâmetros extrínsecos da câmera virtual foram calculados como:

$$T = \begin{bmatrix} 113.4501 \\ 84.0351 \\ 766.4802 \end{bmatrix}$$

e

$$K_L = \begin{bmatrix} 0.9997 & -0.0059 & -0.0243 \\ 0 & 0.9717 & -0.2362 \\ 0.0250 & 0.2361 & 0.9714 \end{bmatrix}$$

A reprojeção destes pontos resultou na imagem observada na Figura 4.13.

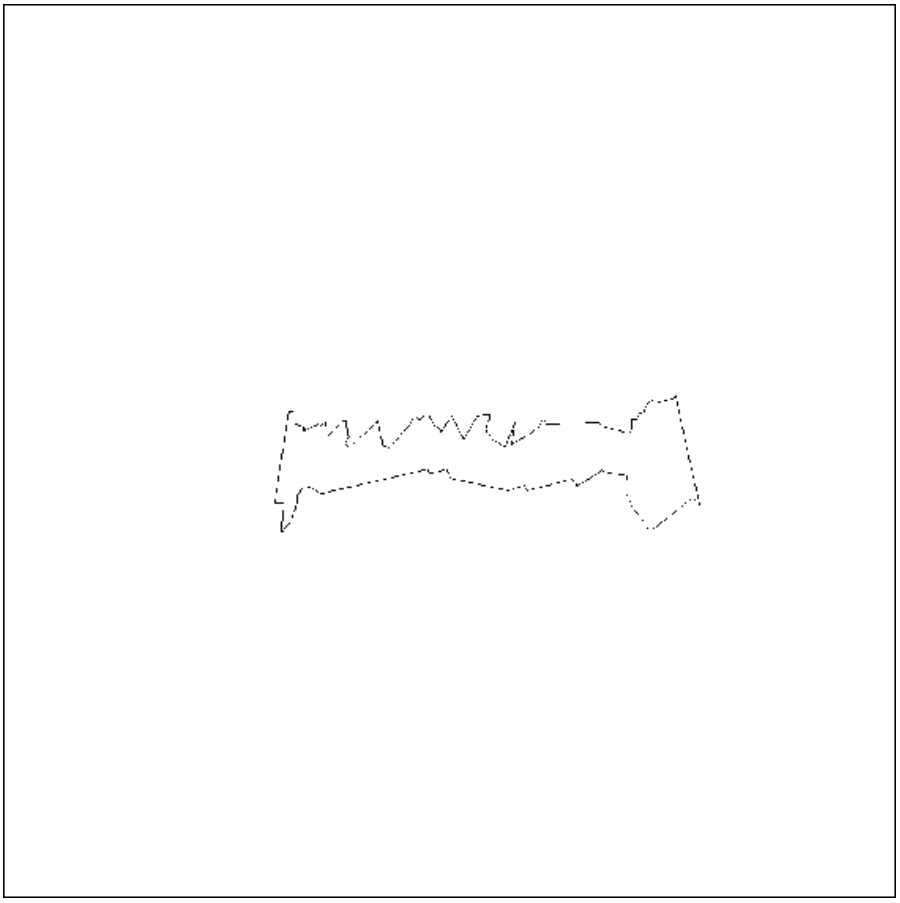


Figura 4.13: Reprojeção dos pontos.

O grande desvio de alguns pontos fez com que o plano que atravessa os pontos (calculado por mínimos quadrados) não fosse o mais adequado. O gráfico da Figura 4.14 é outra vista dos pontos dos gráficos da Figura 4.13. Ele mostra uma projeção mais adequada do que a que foi calculada.

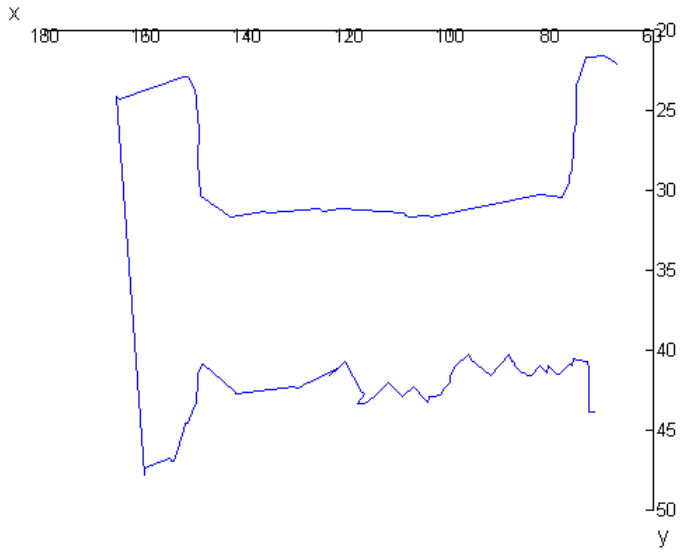


Figura 4.14: Outra vista dos gráficos da Figura 4.13.

Um outro par de imagens resultou na imagem observada na Figura 4.15, que é bem mais fiel ao original.

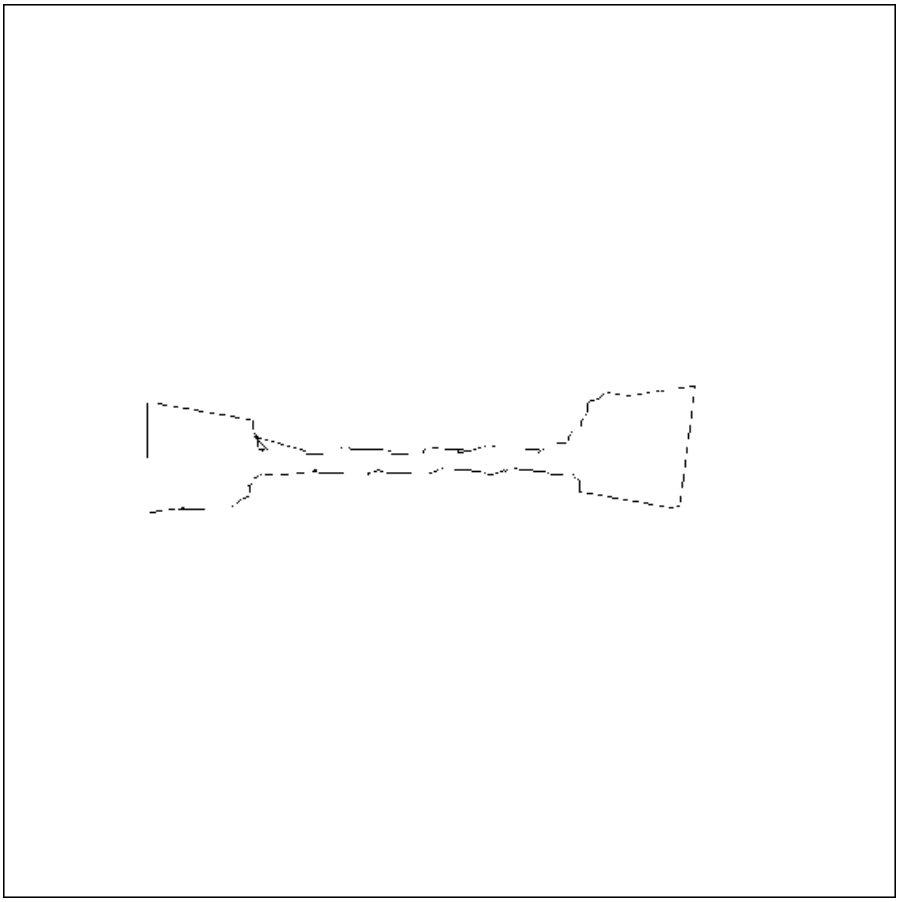


Figura 4.15: Reprojeção de pontos.

4.5 REDE NEURAL

Os erros provenientes da detecção de bordas, juntamente com a parte da correspondência, foram visivelmente acumulados na projeção da câmera virtual. Apesar disso, os testes com redes neurais tiveram um índice de acerto de cerca de 80%, como pode ser visto na Tabela 4.7, onde os erros estão marcados em vermelho.

Tabela 4.7: Resultados da rede neural.

Par de imagens	valor desejado	valor obtido
1	0	-0.0826
2	0	1.0836
3	0	0.1379
4	0	0.7555
5	0	0.4782
6	0	0.2842
7	0	-0.0679
8	0	0.3851
9	0	0.4068
10	0	0.1260
11	0	0.3242
12	1	0.5910
13	1	0.7374
14	1	0.8811
15	1	0.5276
16	1	0.5901
17	0	0.3256
18	1	0.8123
19	1	0.5545
20	1	0.6804
21	1	0.4063
22	1	0.5474
23	0	0.8353

A Figura 4.16 mostra a reconstituição e a reprojeção dos resultados errados.

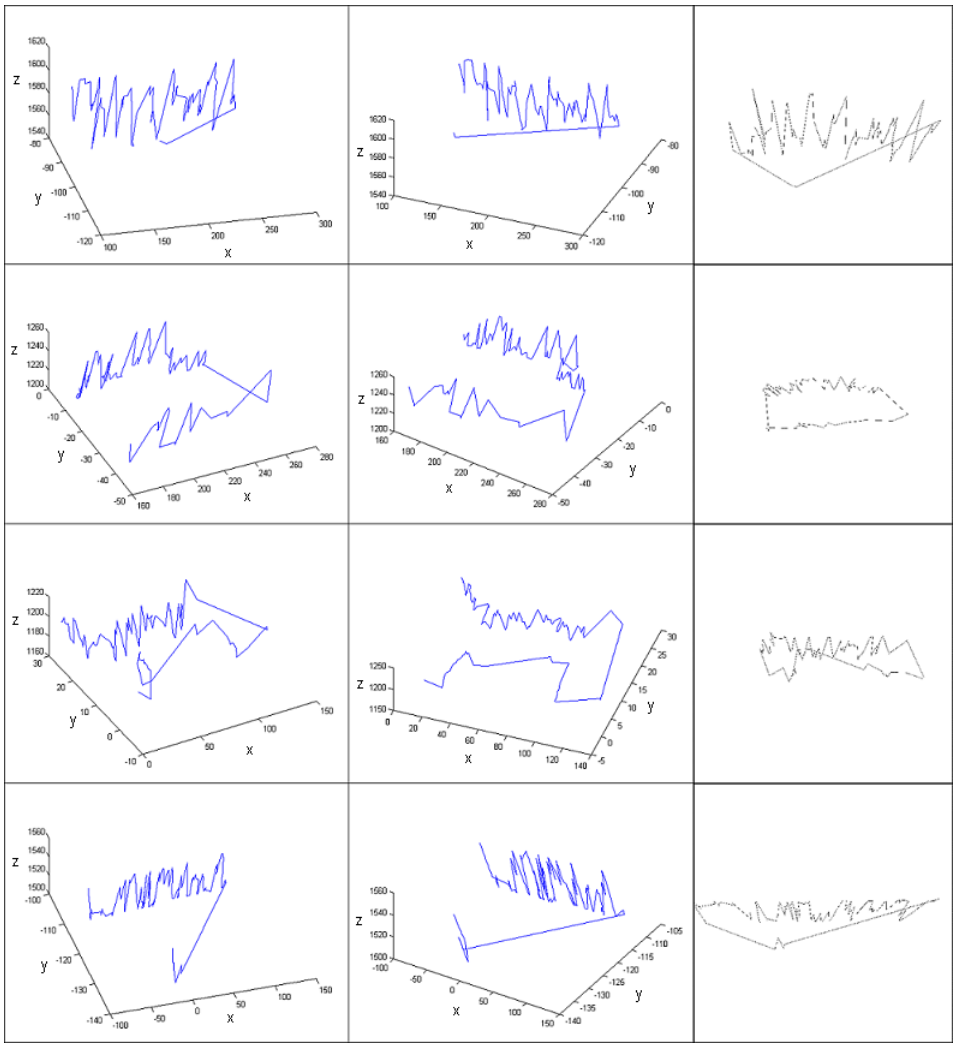


Figura 4.16: Cada linha mostra duas vistas da reconstituição dos pontos e a reprojeção dos pares de imagens 2, 4, 21 e 23, respectivamente.

Pode-se observar que a principal fonte de erro é o cálculo errado do vetor normal à garra, feito na etapa de reprojeção. Isso ocorre devido à aproximação incorreta do plano que contém a garra. O erro da correspondência de pontos resultou em uma descontinuidade dos pontos adjacentes que formam o contorno da garra. Isso, aliado ao fato de a metade superior ter mais pontos correlacionados do que a metade inferior, motivou a inadequação do método dos mínimos quadrados no cálculo do plano que possui a garra.

5 CONCLUSÕES

O presente trabalho teve como objetivo o desenvolvimento de um sistema de reconhecimento de falhas em linhas de transmissão. A inspeção será feita utilizando visão estéreo e redes neurais artificiais. Espera-se que todo o processamento possa ser feito automaticamente e de forma imediata, enquanto o helicóptero (VANT) está em pleno voo. O sistema que foi desenvolvido não é autônomo e necessita de intervenção humana para a seleção das máscaras. Mas é um passo para que se atinja o objetivo.

A iluminação foi um problema constante no desenvolvimento deste projeto. Devido a essa questão, a detecção de bordas não foi precisa o suficiente para se conseguir um resultado robusto. Além disso, a câmera não permitia a obtenção de uma imagem de boa qualidade. Pode-se notar que o tom de verde é bastante forte. Uma imagem com melhor definição de cores forneceria um resultado mais confiável. O resultado precário neste quesito levou a uma correspondência também pouco eficiente. Mesmo que as bordas estivessem perfeitamente definidas, a correspondência ainda seria uma questão complicada. O padrão das bordas permite que um ponto tenha uma alta correlação com vários outros. Como a imagem processada é binária, a frequência de altas correlações aumenta.

Além disso, os pontos detectados nas bordas de uma imagem não possuem necessariamente um correspondente na outra imagem. Pode ocorrer de esse ponto ter sido uma oclusão na imagem original (não binária). Isso porque a própria garra esconde pontos do cabo. Alguns pontos aparecem numa imagem, mas não aparecem na outra. Todo o procedimento foi realizado acreditando-se que a rede neural seria robusta o suficiente para ignorar pequenos erros.

As tarefas realizadas foram: calibração das câmeras, aquisição das imagens, segmentação do cabo, correspondência, reconstrução 3D, reprojeção dos pontos, determinação dos coeficientes de Fourier e treinamento da rede neural. Ao final, de um total de 23 pares de imagens de garras boas e ruins, houve erro de análise em apenas 4. A segmentação foi uma das tarefas mais árduas, juntamente com a correspondência.

Ainda são necessárias melhorias nos algoritmos de detecção de borda e de correspondência. Uma alternativa seria construir tridimensionalmente todo o cabo com a garra, reprojetá-lo e, a partir daí, segmentar o cabo ou a garra. Ou mesmo reconstruir o cabo com a garra, rotacionar o sistema e utilizar as coordenadas para treinar a rede neural. Entretanto, estas soluções demandariam um maior custo computacional. Futuros projetos podem envolver: embarcar o software no VANT, verificar outros componentes da linha de transmissão e fazer medições de campo elétrico e ruído elétrico. Estas são algumas tarefas que certamente auxiliariam na inspeção de linhas de transmissão.

[net, n.d.] *Homepage de Redes Neurais*. website: http://www.din.uem.br/ia/neurais/. Acesso em 7 de fevereiro de 2011.

[NNT, 2006] 2006. *Neural Network Toolbox User's Guide (NNT)*. The Mathworks, Inc.

[Beckmann, 2008] Beckmann, Ener Diniz. 2008. *Arquiteturas de Instrumentação e Controle para um Protótipo de Robô Aéreo Baseado em um Helimodelo*. Dissertação de Mestrado, Engenharia Elétrica. Brasília: Universidade de Brasília.

[Caltech, 2010] Caltech. 2010. *Camera Calibration Toolbox for Matlab*. Disponível em http://www.vision.caltech.edu/bouguetj/calib_doc/. Acesso em 5 de julho de 2011.

[Chuang *et al.*, 2001] Chuang, Y.Y., Curless, B., Salesin, D.H., & Szeliski, R. 2001. *A Bayesian Approach to Digital Matting*. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).

[de Araújo, 2010] de Araújo, Allan David Guarcia. 2010. *Uma proposição para o cálculo de mapas de disparidade de imagens estéreo usando um interpolador neural baseado em funções de base radial*. Dissertação de Mestrado, Engenharia Elétrica. Natal: Universidade Federal do Rio Grande do Norte.

[de Oliveira & de Araújo Dias, 2007] de Oliveira, Flávio Augusto Rodrigues, & de Araújo Dias, Alexandre Guedes. 2007. *Verificação de Falhas em Linhas de Transmissão por Redes Neurais Artificiais a partir dos Espectros de Frequência de Imagens*. Trabalho de Graduação, Engenharia de Controle e Automação. Brasília: Universidade de Brasília.

[de Oliveira, 2006] de Oliveira, Marco Antonio Floriano. 2006. *Correlacionamento Estéreo de Complexidade Linear Baseado em Indexação de Regiões*. Dissertação de Mestrado, Ciência da Computação. Florianópolis: Universidade Federal de Santa Catarina.

[Faugeras, 1996] Faugeras, Oliver. 1996. *Three-Dimensional Computer Vision - A Geometric Viewpoint*. Massachusetts: The MIT Press.

[Giardina & Dougherty, 1988] Giardina, Charles R., & Dougherty, Edward R. 1988. *Mathematical Methods for Artificial Intelligence and Autonomous Systems*. Englewood Cliffs, New Jersey, U.S.A.: Prentice Hall.

[Gonzalez & Woods, 2000] Gonzalez, Rafael C., & Woods, Richard E. 2000. *Processamento de Imagens Digitais*. São Paulo: Editora Blucher.

[Leal, 2010] Leal, Milton. 2010. *Cteep quer disputar leilão da linha da usina de Belo Monte*. Jornal da Energia. São Paulo, 17 de maio de 2010. Acesso em 7 de fevereiro de 2011. <http://www.jornaldaenergia.com.br/ler_noticia.php?id_noticia=3452&id_tipo=2&id_secao=11&id_pai=0&titulo_info=Cteep%20quer%20disputar%20leil%E3o%20da%20linha%20da%20usina%20de%20Be... >.

[Levin *et al.*, 2006] Levin, A., Lischinski, D., & Weiss, Y. 2006 (Junho). *A Closed Form Solution to Natural Image Matting*. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), New York.

[Li *et al.*, 2004] Li, Y., Sun, J., Tang, C.-K., & Shum, H.-Y. 2004. *Lazy Snapping*. Vol. 23. ACM Trans. Graphics.

[Matworks, 2011] Matworks. 2011. *MATLAB - The Language Of Technical Computing*. Disponível em http://www.mathworks.com/products/matlab/. Acesso em 5 de julho de 2011.

[Oliveira, 2009] Oliveira, Renata Francine. 2009. *Linhas de Transmissão*. Trabalho de Graduação, Engenharia Industrial Elétrica/Eletrotécnica. Curitiba: Universidade Tecnológica Federal do Paraná.

[Rodrigues & de Almeida Mencari, 1999] Rodrigues, Cristiano Jacques Miosso, & de Almeida Mencari, Mirele. 1999. *Sistema de Visão Estéreo para Formas Poliméricas*. Trabalho de Graduação, Engenharia Elétrica. Brasília: Universidade de Brasília.

[Rother *et al.*, 2004] Rother, C., Kolmogorov, V., & Blake, A. 2004. *"Grabcut": Interactive Foreground Extraction Using Iterated Graph Cuts*. Vol. 23. ACM Trans. Graphics.

ANEXOS

Programas utilizados na detecção de bordas:

```
% go.m Script principal
% Adaptado do trabalho de [de Oliveira & de Araújo Dias, 2007]
% Inicia a execução dos scripts de detecção de borda.

clear all, clc
close all

% Localização do diretório que contém as imagens

fileFolder  = fullfile('D:','TG\imagens')

%% Criação das listas com os nomes dos arquivos
dirOutputL = dir(fullfile(fileFolder,'imL*.bmp'));
dirOutputR = dir(fullfile(fileFolder,'imR*.bmp'));

% Vetores contendo os nomes dos arquivos
fileNamesL = {dirOutputL.name}';
fileNamesR = {dirOutputR.name}';

%% Loop principal para aquisição das bordas do cabo.
for k = 1:(length(fileNamesL))
    clc;
    disp('Iteração: ')
    disp(k);disp('No total de: '),disp(length(fileNamesL)),
    disp('-----------------------------------------------------');
    %% Obtenção da cadeia codificada

     datL = processFile(fileFolder,fileNamesL{k});
     datR = processFile(fileFolder,fileNamesR{k});

     %ROI
     cadeiaL(k).imagem=datL.imagem;
     cadeiaR(k).imagem=datR.imagem;

     %imagem com as máscaras
     cadeiaL(k).mask=datL.mask;
     cadeiaR(k).mask=datR.mask;
```

```
      %% funções utilizadas de fato
      %coordenadas da Área de Interesse na imagem.
      cadeiaL(k).coord_AI=datL.coord_AI;
      cadeiaR(k).coord_AI=datR.coord_AI;

      % Recebe as imagens com deteção de bordas.
      cadeiaL(k).bordas=datL.perimetro;
      cadeiaR(k).bordas=datR.perimetro;

      %%

%       %% funções não utilizadas provenientes do código
%       %% original
%      % Cadeia codificada.
%      cadeiaL(k).fcc = datL.cadeia.fcc;
%      cadeiaR(k).fcc = datR.cadeia.fcc;
%
%      % Cadeia normalizada para rotação.
%      cadeiaL(k).diff = datL.cadeia.diff;
%      cadeiaR(k).diff = datR.cadeia.diff;
%
%      % Cadeia normalizada quanto ao ponto de partida.
%      cadeiaL(k).mm = datL.cadeia.mm;
%      cadeiaR(k).mm = datR.cadeia.mm;
%
%      % Cadeia normalizada para rotação e ponto de partida.
%      cadeiaL(k).diffmm = datL.cadeia.diffmm;
%      cadeiaR(k).diffmm = datR.cadeia.diffmm;
%
%      % Ponto de partida.
%      cadeiaL(k).x0y0 = datL.cadeia.x0y0;
%      cadeiaR(k).x0y0 = datR.cadeia.x0y0;

  save cadeia;
end

%% recupera o tamanho original da imagem
image_recovering(cadeiaL,cadeiaR);
```

```
function data = processFile(fileFolder,imageFile)
%   Realiza a leitura da imagem, a obtenção da região de
%   interesse (ROI), a aplicação das restrições (máscaras)
%   e a execução do algoritmo de processamento das imagens.
```

```
%
%   Recebe o local (fileFolder) e o nome da imagem (imageFile)
%   e retorna na estrutura DATA a região de interesse, o resultado
%   da segmentação sobreposto e a cadeia codificada do contorno.

imageFilePath = fullfile(fileFolder,imageFile);

% Leitura do arquivo
A = imread(imageFilePath);

%% Dimensões da região de interesse
imageW=321;
imageH=251;
%%
imshow(A);
%Obtenção da região de interesse.
%Deve ser selecionado o pequeno espaço entre as garras.
%[xmin,ymin,width,height]
rect=getrect;
rect=round(rect);

%Coordenadas do Quadrado;
xmin=rect(1);
xmax=rect(1)+rect(3);
ymin=rect(2);
ymax=rect(2)+rect(4);

%coordenadas do centro da figura;
xc=(xmin+xmax)/2;
yc=(ymin+ymax)/2;

%Area de Interesse
AI=[xc-160, yc-125,imageW-1,imageH-1];
%Coordenadas da Área de Interesse na imagem plena.
coord_AI=[AI(1), AI(1)+AI(3)+1, AI(2), AI(2)+AI(4)+1];

% Pega a região de interesse (ROI) no centro da imagem
Icrop = imcrop(A,AI);

% Camadas separadas
iR = Icrop(:,:,1);
iG = Icrop(:,:,2);
iB = Icrop(:,:,3);
```

```
%% Foreground
Imaskf=zeros(imageH,imageW);
%rectângulo do cabo
imin = round((imageH-rect(4))/2+2);
imax = round((imageH+rect(4))/2-2);
jmin = round((imageW-rect(3))/2.2);
jmax = round((imageW+rect(3))/1.9);
for i=max(1,imin):min(imageH,imax)
    for j=max(1,jmin):min(imageW,jmax)
        Imaskf(i,j)=1;
    end
end
%Retangulos do cabo à esquerda
imin = round((imageH-2*rect(4))/2-5);
imax = round((imageH+2*rect(4))/2-5);
jmin = round((imageW-rect(3))/2-8*rect(4));
jmax = round((imageW-rect(3))/2-4*rect(4));
for i=max(1,imin):min(imageH,imax)
    for j=max(1,jmin):min(imageW,jmax)
        Imaskf(i,j)=1;
    end
end
%Retângulos do cabo à direita
imin = round((imageH-2*rect(4))/2);
imax = round((imageH+2*rect(4))/2);
jmin = round((imageW+rect(3))/2+4*rect(4));
jmax = round((imageW+rect(3))/2+8*rect(4));
for i=max(1,imin):min(imageH,imax)
    for j=max(1,jmin):min(imageW,jmax)
        Imaskf(i,j)=1;
    end
end
%% Background
Imaskb=zeros(imageH,imageW);
%Retângulo de baixo
imin = round((imageH+rect(4)+rect(3))/2-10);
imax = round((imageH+rect(4))/2+rect(3)-30);
jmin = round((imageW-2*rect(3))/2);
jmax = round((imageW+2*rect(3))/2);
for i=max(1,imin):min(imageH,imax)
    for j=max(1,jmin):min(imageW,jmax)
        Imaskb(i,j)=1;
```

```
    end
end

%Retângulo de cima da garra
imin = round((imageH-12*rect(4))/2+5);
imax = round((imageH-4*rect(4))/2);
jmin = round((imageW-rect(3)+rect(4))/2+5);
jmax = round((imageW+rect(3)-rect(4))/2);
for i=max(1,imin):min(imageH,imax)
    for j=max(1,jmin):min(imageW,jmax)
        Imaskb(i,j)=1;
    end
end

%Retângulo logo abaixo da garra
imin = round((imageH+4*rect(4))/2);%era 2
imax = round((imageH+8*rect(4))/2);%era 2
jmin = round((imageW-rect(3)+rect(4))/2+5);
jmax = round((imageW+rect(3)-rect(4))/2);
for i=max(1,imin):min(imageH,imax)
    for j=max(1,jmin):min(imageW,jmax)
        Imaskb(i,j)=1;
    end
end

%quadrados à esquerda
imin = round((imageH-22*rect(4))/2);
imax = round((imageH-12*rect(4))/2);
jmin = round((imageW-rect(3))/2-10*rect(4));
jmax = round((imageW-rect(3))/2-4*rect(4));
for i=max(1,imin):min(imageH,imax)
    for j=max(1,jmin):min(imageW,jmax)
        Imaskb(i,j)=1;
    end
end
%quadrados à direita
imin = round((imageH-22*rect(4))/2);
imax = round((imageH-12*rect(4))/2);
jmin = round((imageW+rect(3))/2+4*rect(4));
jmax = round((imageW+rect(3))/2+10*rect(4));
for i=max(1,imin):min(imageH,imax)
    for j=max(1,jmin):min(imageW,jmax)
        Imaskb(i,j)=1;
```

```
    end
end
%%
Imaskf=logical(Imaskf);
Imaskb=logical(Imaskb);
%--------------------------------------------------------------------
% Geração da imagem para processamento (adição das máscaras).
iR(Imaskf) = 255;
iG(Imaskf) = 255;
iB(Imaskf) = 255;

iR(Imaskb) = 0;
iG(Imaskb) = 0;
iB(Imaskb) = 0;

% Concatenção das camadas RGB para início do processamento.
imagem = cat(3, iR, iG, iB);

% Script para processamento de imagens.
executa;

% Resultados.
data.coord_AI=coord_AI;%Coordenadas da Região de Interesse.
data.res = result;   % Imagem com cabo segmentado sobreposto.
% data.cadeia = ccf(im_final); % Estrutura com as cadeias.
data.perimetro=perimetro;
data.imagem=Icrop;
data.mask=imagem;
```

```
% executa.m
% Após a obtenção da imagem alpha, realiza operação de limiarização
% e uma série de operações morfológicas para obtenção de uma imagem
% binária contendo o cabo segmentado.

% Parâmetros para algoritmo de obtenção do "alpha matting"
win_size = 2;
levels_num = 4;
active_levels_num = 3;
thr_alpha = 0.3;%
epsilon = 0.1^5;
%sig=0.1^5;

% Execução do algoritmo para obtenção da imagem alpha (Levin et al.)
```

```
runMatting2

% Binarização do alpha matting gerado.
level = graythresh(alpha);
bw = im2bw(alpha, level);

% Conecta pixels desconectados 5x (conectividade 4).
bw = bwmorph(bw, 'bridge', 10);

% Remove de componentes desconectados < 70 pixels (conectividade 8).
BWopen = bwareaopen(bw,70);

% Realiza operação morfológica de fechamento ("Closing").
BWclose = bwmorph(BWopen,'close');

% Realiza operação de dilatação a partir dos elementos estruturais
% gerados.
se90 = strel('line', 3, 90);
se0 = strel('line', 3, 0);
im_dilate = imdilate(BWclose, [se90 se0]);

% Preenche buracos na imagem binária
im_fill = imfill(im_dilate, 'holes');

% Suavisação das bordas do cabo pela através de operações
% de erosão e dilatação.
seL1 = strel('line',5,0);
seL2 = strel('line',5,90);
im_final = imerode(im_fill,[seL1 seL2]);
im_final = imdilate(im_final,[seL1 seL2]);

% Obtenção do contorno do cabo segmentado sobreposto na imagem
% inicial com o uso da função bwperim().
perimetro = bwperim(im_final);

result = im_final; %
```

```
%runMatting2.m
% Script principal criado por Levin et al. para a geração do alpha
% matting a partir de uma abordagem analítica (Closed Form Solution)
% do problema, contendo algumas modificações.
%
```

```
% Análise dos parâmetros de entrada
if (~exist('thr_alpha','var'))
  thr_alpha=[];
end
if (~exist('epsilon','var'))
  epsilon=[];
end
if (~exist('win_size','var'))
  win_size=[];
end

if (~exist('levels_num','var'))
  levels_num=1;
end
if (~exist('active_levels_num','var'))
  active_levels_num=1;
end

% Leitura da imagem a ser processada
I=double(Icrop)/255;
%I=double(imread(img_name))/255;

% Leitura da imagem com a máscara de foreground e background
mI=double(imagem)/255;
%mI=double(imread(scribs_img_name))/255;
%mI=double(scribs_img_name)/255;

consts_map=sum(abs(I-mI),3)>0.001;
if (size(I,3)==3)
  consts_vals=rgb2gray(mI).*consts_map;
end
if (size(I,3)==1)
  consts_vals=mI.*consts_map;
end

% Execução do algoritmo
alpha=solveAlphaC2F(I,consts_map,consts_vals,levels_num, ...
                    active_levels_num,thr_alpha,epsilon,win_size);

%figure, imshow(alpha);
drawnow;
%[F,B]=solveFB(I,alpha);
```

```
function image_recovering(cadeiaL,cadeiaR)
%% transforma matrizes de imagens em imagens.

%%
%A imagem da Área de interesse precisa ter sua dimensão readequada
%à dimensão original da Imagem. para isso feitos 4 inserções de
%matriz de zeros, são ele:
% * insercao_sup;
% * insercao_inf;
% * lateral_dir;
% * lateral_esq;
%%
for i=1:(length(cadeiaL))
    %-------------------------------------------------------------
    %Imagem esquerda:
    imL=cadeiaL(i).bordas;%Recebe a imagem esquerda;

    %Obtebção das coordenadas da Área de Interesse.
    xmin=round(cadeiaL(i).coord_AI(1));
    xmax=round(cadeiaL(i).coord_AI(2));
    ymin=round(cadeiaL(i).coord_AI(3));
    ymax=round(cadeiaL(i).coord_AI(4));

    lateral_dir=zeros(ymax-ymin,1280-xmax);%inserção à direita.
    lateral_esq=zeros(ymax-ymin,xmin);%inserção à esquerda.

    %concatenação lateral.
    concatenacao_lateral=[lateral_esq imL lateral_dir];

    insercao_sup=zeros(ymin,1280);%inserção da parte superior;
    insercao_inf=zeros(960-ymax,1280);%inserção da parte iferior;

    %concatenação vertical:
    concatenacao_superior=[insercao_sup; concatenacao_lateral; ...
        insercao_inf];

    imL=concatenacao_superior;

    %Salva a imagem recomposta.
    imwrite(imL,['imageL' num2str(i) '.bmp']);
    %-------------------------------------------------------------
    %Imagem direita:
```

```
    imR=cadeiaR(i).bordas;%Recebe a imagem direita;

    %Obtebção das coordenadas da Área de Interesse.
    xmin=round(cadeiaR(i).coord_AI(1));
    xmax=round(cadeiaR(i).coord_AI(2));
    ymin=round(cadeiaR(i).coord_AI(3));
    ymax=round(cadeiaR(i).coord_AI(4));

    lateral_dir=zeros(ymax-ymin,1280-xmax);%inserção à direita.
    lateral_esq=zeros(ymax-ymin,xmin);%inserção à esquerda.

    %concatenação lateral.
    concatenacao_lateral=[lateral_esq imR lateral_dir];

    insercao_sup=zeros(ymin,1280);%inserção da parte superior;
    insercao_inf=zeros(960-ymax,1280);%inserção da parte iferior;

    %concatenação vertical:
    concatenacao_superior=[insercao_sup; concatenacao_lateral; ...
        insercao_inf];

    imR=concatenacao_superior;

    %Salva a imagem recomposta.
    imwrite(imR,['imageR' num2str(i) '.bmp']);
    %-------------------------------------------------------------
end
```

Programas utilizados na reconstrução 3D e reprojeção:

```
% armazena.m
% Armazena os dados na estrutura 'data':
%     data.matching=matriz de correspondencia;
%     data.xyz=coordenadas 3D;
%     data.final_image=imagem reprojetada;
%     data.cadeia=estrutura com o código de cadeia;
%     data.coef=coeficientes de Fourier;
%     data.net_input=coeficintes na forma de entrada da rede neural;
%     data.net_target = saida desejada na rede neural;

% Entradas:
% matriz de calibração da câmera esquerda 'PL'
% matriz de calibração da câmera direita 'PR'
% vetor 'ok', onde ok(i)=1 para garra i ok e
```

```
%     ok(i)=0 para garra i com defeito

fileFolder  = fullfile('D:','TG\toolbox_calib');
dirOutputL = dir(fullfile(fileFolder,'imageL_rectified*.bmp'));
fileNamesL = {dirOutputL.name}';

net_P = [];
net_T = [];

for i=1:length(fileNamesL)

    net_target = ok(i);

    disp('Iteração: ')
    disp(i);disp('No total de: '),disp(length(fileNamesL)),
    disp('----------------------------------------------------');

    nomeL = ['imageL_rectified' int2str(i) '.bmp'];
    nomeR = ['imageR_rectified' int2str(i) '.bmp'];
    correspondencia=corresp3(nomeL,nomeR);
    ij_imagl = [correspondencia(:,1) correspondencia(:,2)];
    ij_imagr = [correspondencia(:,1) correspondencia(:,3)];
    xyz_cena = reconsti(ij_imagl,PL,ij_imagr,PR,0,0);
    pontos = -xyz_cena';
    pontos = [pontos;ones(1,size(pontos,2))];
    [T R] = posicao_camera_virtual(-xyz_cena);
    P=[1000 0 500 0;0 -1000 500 0;0 0 1 0]*inv([R T;0 0 0 1]);
    pixels = P*pontos;
    pixels(1,:)= round(pixels(1,:)./pixels(3,:));
    pixels(2,:)= round(pixels(2,:)./pixels(3,:));
    pixels(3,:)= [];
    quant=size(pixels,2);
    pixels=[pixels pixels(:,1)];
    imagem = logical(zeros(1000,1000));
    for j=1:quant
        imagem=segmento(imagem,pixels(2,j),pixels(1,j), ...
            pixels(2,j+1),pixels(1,j+1));
    end

    cadeia = ccf(imagem);
    coef=coeficientes(cadeia.mm);
    net_input=[coef.A';coef.B';coef.C';coef.D'];
```

```
    data(i).matching=correspondencia;
    data(i).xyz=xyz_cena;
    data(i).final_image=imagem;
    data(i).cadeia=cadeia;
    data(i).coef=coef;
    data(i).net_input = net_input;
    data(i).net_target = net_target;

    net_P=[net_P net_input];
    net_T=[net_T net_target];
end
```

```
function result= corresp3(nomeL,nomeR)

% nomeL = 'imageL_rectified1.bmp';
% nomeR = 'imageR_rectified1.bmp';

imL=imread(nomeL);
imR=imread(nomeR);

%Elimina as bordas brancas (0) oriundas da rectificação.
imL=border_destructor(imL);
imR=border_destructor(imR);

%bordas na ordem correta
bordasL_cell = bwboundaries(imL,'noholes');
% bordasR_cell = bwboundaries(imR,'noholes');
bordasL = bordasL_cell{celula(bordasL_cell)};

[il,jl]=find(imL==1);%indices i e j para a imagem left.
[ir,jr]=find(imR==1);%indices i e j para a imagem right.

il_min=min(il);%Ranges de linha na imagem Esquerda
il_max=max(il);

ir_min=min(ir);%Ranges de linha na imagem direita
ir_max=max(ir);

jl_min=min(jl);%Ranges de coluna na imagem Esquerda
jl_max=max(jl);

jr_min=min(jr);%Ranges de coluna na imagem direita
jr_max=max(jr);
```

```
%correções
jl_min=jl_min+round((jl_max-jl_min)/10);
jl_max=jl_max-round((jl_max-jl_min)/10);

jr_min=jr_min+round((jr_max-jr_min)/10);
jr_max=jr_max-round((jr_max-jr_min)/10);

% desvios
dsup = abs(il_max - ir_max);
dinf = abs(il_min - ir_min);
desvio = min(5,max(dsup,dinf));
ddir = abs(jl_max - jr_max);
desq = abs(jl_min - jr_min);
dispari = (ddir+desq)/2;

% valores de disparidade aceitáveis
minsoma = dispari - (jr_max - jr_min)/32;
maxsoma = dispari + (jr_max - jr_min)/32;

% %janela
dx = 15;
dy = 6;

index = 1;

for n=1:size(bordasL,1)
 subL = imL(bordasL(n,1)-ceil(dy/2)+1:bordasL(n,1)+floor(dy/2), ...
     bordasL(n,2)-ceil(dx/2)+1:bordasL(n,2)+floor(dx/2));
 indice1=1;
 for icorresp=(bordasL(n,1)-desvio):(bordasL(n,1)+desvio)
  indice2=1;
  for jcorresp=jr_min:jr_max
   if ((imR(icorresp,jcorresp)~=0))
    if ((imR(icorresp,jcorresp)~=0)&&((bordasL(n,2)-maxsoma)< ...
            jcorresp)&&(jcorresp<bordasL(n,2)-minsoma))
     subR = imR((icorresp-ceil(dy/2)+1):(icorresp+floor(dy/2)), ...
         (jcorresp-ceil(dx/2)+1):(jcorresp+floor(dx/2)));
     correlacao = ~xor(subL,subR);
%       correlacao = correlacao + 2*subL.*subR;
     correl(indice1,indice2) = sum(correlacao(:));
     if(bordasL(n,2)<jcorresp)
      correl(indice1,indice2) = 0;
```

```
      end
     end
    end
    indice2=indice2+1;
   end
   indice1=indice1+1;
  end
  if(exist('correl','var'))
   maxcorrel = max(correl(:));
   [a,b] = find(correl==maxcorrel);
   if (length(b)<2 && maxcorrel>0.75*dx*dy)
    for c=1:length(a)
     Y(index) = bordasL(n,1);
     XL(index) = bordasL(n,2);
     XR(index) = jr_min+b(c)-1;
     index=index+1;
    end
   end
   clear correl;
  end
end
result=[Y' XL' XR'];
```

```
function image_out=border_destructor(image_in)
%% Destrói as bordas oriundas da rectificação.

%tamanho da borda a preencher (em pixels):
borda=50;

%borda superior;
for i=1:borda
    for j=1:size(image_in,2)
        image_in(i,j)=0;
    end
end

%borda inferior;
for i=(size(image_in,1)-borda):size(image_in,1)
    for j=1:size(image_in,2)
        image_in(i,j)=0;
    end
end

```

```
%borda esquerda;
for i=1:size(image_in,1)
    for j=1:borda
        image_in(i,j)=0;
    end
end

%borda esquerda;
for i=1:size(image_in,1)
    for j=(size(image_in,2)-borda):size(image_in,2)
        image_in(i,j)=0;
    end
end

image_out=logical(image_in);
```

```
function result=celula(c)
%Retorna o índice da célula de maior vetor
valor=max(cellfun('length', c));
for i=1:size(c,1)
    if(length(c{i})==valor)
        result=i;
    end
end
```

```
function xyz_cena=reconsti(ij_im1,P1,ij_im2,P2,k_linhas,k_colunas)
%Retirado do trabalho de [Rodrigues & de Almeida Mencari, 1999]
%xyz_cena=reconsti(ij_imag1,P1,ij_imag2,P2,k_linhas,k_colunas)
%
%Efetua o calculo das tres  coordenadas dos pontos de uma cena
%arbitraria, a  partir  das  posicoes  dos pixels correspondentes
%em duas imagens  distintas da  mesma. Cada  linha  em ij_imagl e
%em ij_imagr  deve  corresponder 'as  coordenadas i e j  do pixel
%de um dos pontos a serem reconstituidos. P1 e P2 sao as matrizes
%de transformacao perspectiva das cameras que  geraram  cada  uma
%das imagens. A matriz xyz_cena retornada pela funcao contem
%em cada linha as coordenadas x,y,z de um unico ponto.
%k_linhas e k_colunas:
% valores somados às posições de cada pixel das imagens retificadas
% calculadas, com o objetivo de evitar índices nulos  ou  negativos
% na representação matricial (estes  valores  serão  subtraídos
% dos  índices i,j durante o processo de  reconstituição  da cena a
% partir das imagens stereo).
```

```
ij_imagl=[ij_im1(:,1)-k_linhas ij_im1(:,2)-k_colunas];
ij_imagr=[ij_im2(:,1)-k_linhas ij_im2(:,2)-k_colunas];

k=size(ij_imagl);

if((k(1,2)~=2)|(k(1,1)~=size(ij_imagr,1)))
   disp('Entradas invalidas');
else
   k=k(1,1);
   xyz_cena=zeros(k,3);
   for j=1:k
      A(1,1:3)=P1(1,1:3)-ij_imagl(j,2)*P1(3,1:3);
      A(2,1:3)=P1(2,1:3)-ij_imagl(j,1)*P1(3,1:3);
      A(3,1:3)=P2(1,1:3)-ij_imagr(j,2)*P2(3,1:3);
      A(4,1:3)=P2(2,1:3)-ij_imagr(j,1)*P2(3,1:3);

      y(1,1:1)=-P1(1,4)+ij_imagl(j,2)*P1(3,4);
      y(2,1:1)=-P1(2,4)+ij_imagl(j,1)*P1(3,4);
      y(3,1:1)=-P2(1,4)+ij_imagr(j,2)*P2(3,4);
      y(4,1:1)=-P2(2,4)+ij_imagr(j,1)*P2(3,4);

      xyz_cena(j,1:3)=((inv((A')*A))*A'*y)';
   end
end
```

```
function [T R] = posicao_camera_virtual(P)

% fornece a posição da câmera virtual.
% P = [x y z] são as coordenadas dos pontos no espaço
% x = [x1 x2 x3 x4...]';
% y = [y1 y2 y3 y4...]';
% z = [z1 z2 z3 z4...]'.

distancia_camera = 200;

[a b c d] = fitplano(P);%plano que passa pelos pontos

%rotação em torno de x
alfa = asin(b);
Rx = [1 0 0; 0 cos(alfa) sin(alfa); 0 -sin(alfa) cos(alfa)];

%rotação em torno de y
```

```
beta = asin(a/cos(alfa));
Ry = [cos(beta) 0 sin(beta); 0 1 0; -sin(beta) 0 cos(beta)];

Pm = sum(P)/size(P,1);%ponto médio de todos os pontos
R = Ry*Rx;%matriz de rotação
T = Pm' - distancia_camera*[a; b; c];%matriz de translação
```

```
function [a b c d] = fitplano(P)
% calcula o plano ax+by+cz+d=0 que melhor se encaixa
% nos pontos P = [x y z], usando mínimos quadrados.

norma = 1;

R=ones(size(P,1),1);
result = inv(P'*P)*(P')*R;
a = result(1);
b = result(2);
c = result(3);

%normalização
modulo = (a^2+b^2+c^2)^(0.5);
a = norma*a/modulo;
b = norma*b/modulo;
c = norma*c/modulo;
d = -norma/modulo;
```

```
function Im=segmento(I,y1,x1,y2,x2)
%Desenha o segmento de reta AB
%Im = Imagem I com o segmento AB
%onde A = (x1,y1) e B = (x2,y2)

Im=I;
if(x1==x2 && x1>0 && x1<=size(I,2))
    if(y1<y2)
        Im(max(1,y1):min(y2,size(I,1)),x1)=1;
    else
        Im(max(1,y2):min(y1,size(I,1)),x1)=1;
    end
    return
end
if(y1==y2 && y1>0 && y1<=size(I,1))
    if(x1<x2)
        Im(y1,max(1,x1):min(x2,size(I,2)))=1;
```

```
    else
        Im(y1,max(1,x2):min(x1,size(I,2)))=1;
    end
    return
end
if((abs(x1-x2)>abs(y1-y2)))
    if(x1<x2)
        for x=x1:x2
            y = round(y1+(y1-y2)*(x-x1)/(x1-x2));
            if(x>0 && y>0 && x<=size(I,2) && y<=size(I,1))
                Im(y,x)=1;
            end
        end
    else
        for x=x2:x1
            y = round(y1+(y1-y2)*(x-x1)/(x1-x2));
            if(x>0 && y>0 && x<=size(I,2) && y<=size(I,1))
                Im(y,x)=1;
            end
        end
    end
else
    if(y1<y2)
        for y=y1:y2
            x = round(x1+(x1-x2)*(y-y1)/(y1-y2));
            if(x>0 && y>0 && x<=size(I,2) && y<=size(I,1))
                Im(y,x)=1;
            end
        end
    else
        for y=y2:y1
            x = round(x1+(x1-x2)*(y-y1)/(y1-y2));
            if(x>0 && y>0 && x<=size(I,2) && y<=size(I,1))
                Im(y,x)=1;
            end
        end
    end
end
```

```
function c = ccf(im_final)
%ccf.m Gera a cadeia codificada
%
%   C = ccf(IM_FINAL)
```

```
%   Recebe a imagem binária IM_FINAL com o cabo segmentado, obtém
%   seu contorno e retorna a estrutura C contendo as cadeias.

% A partir da imagem recebida, obtém as coordenadas de cada ponto
% dos contornos dos objetos na imagem por meio da função
% bwboundaries() da toolbox e em seguida o maior dos contornos
% obtidos, o cabo.
%B = boundaries(im_final);   % Coordenadas das fronteiras
[B,L] = bwboundaries(im_final,'noholes');
%B Possui as fronteiras de cada objeto constante na imagem.
%L possui a própria imagem.
d = cellfun('length', B);
[max_d, k] = max(d);
b = B{k};   % Maior fronteira

% Gera a partir das coordenadas de cada ponto da fronteira do cabo,
% obtidas pela função bwboundaries(), a imagem binária do contorno.
[M N] = size(im_final);
bim = bound2im(b, M, N, min(b(:, 1)), min(b(:, 2)));
%figure, imshow(bim); title('bim');

% Obtenção das cadeias codificadas da imagem binária b com o
% contorno do cabo segmentado.
c = fchcode(b);
```

```
function B = bound2im(b,M,N,x0,y0)
%BOUND2IM converte um contorno em uma imagem.
%B = BOUND2IM(b) converte b, um array np por 2 ou 2 por np que
%representa as coordenadas inteiras de um contorno, em uma imagem
%binaria com 1s nos lugares definidos pelo contorno e 0s para o
%fundo.

%B = BOUND2IM(b,M,N) traça um contorno aproximadamente centrado
%numa imagem M por N. Se alguma parte do contorno estiver fora do
%retangulo M por N, é emitida uma mensagem de erro.

%B = BOUND2IM(B,M,N,X0,Y0) traça um contorno numa imagem de tamanho
%M por N, com o ponto de mais acima do contorno localizado em X0 e
%o ponto mais a esquerda localizado em Y0. Se o contorno deslocado
%estiver fora do retangulo M por N, é emitida uma mensagem de erro.
%X0 e Y0 devem ser  inteiros positivos.
[np,nc] = size(b);
if np < nc
```

```
    b = b';%Para converter para o tamanho np por 2.
    [np, nc] = size(b);
end

% Tenha certeza que as coordenadas sejam inteiras.
x = round(b(:,1));
y = round(b(:,2));

%Configure o tamanho padrao dos parametros.

x = x - min(x) + 1;
y = y - min(y) + 1;
B = false(max(x),max(y));
C = max(x) - min(x) + 1;
D = max(y) - min(y) + 1;

if nargin == 1
    %Use os valores padroes configurados
elseif nargin == 3
    if C > M || D > N
        error(' O contorno esta fora da regiao M por N')
    end
 % O tamanho da imagem sera M por N.
 % Configure os parametros para isso.

 B = false(M,N);
 % Distribua linhas extras entre o topo e a parte de baixo.
 NR = round((M-C)/2);
 NC = round((N-D)/2);% O mesmo para as colunas
 x = x + NR; % Faça a compensação do contorno para a nova posição
 y = y + NC;
elseif nargin == 5
    if x0 < 0 || y0 < 0
        error('x0 e y0 devem ser inteiros positivos')
    end
    x = x + round(x0) - 1;
    y = y + round(y0) - 1;

    C = C + x0 - 1;
    D = D + y0 - 1;
    if C > M || D > N
        error(' O contorno deslocado esta fora da regiao M por N')
    end
```

```
    B = false(M,N);
else
    error('Numero incorreto de entradas')
end
B(sub2ind(size(B),x,y)) = true;
```

```
function coef = coeficientes(cadeia)
%coeficientes.m
%   Realiza a leitura da imagem, a obtenção da região de interesse
%   (ROI) localizada no centro da imagem, a aplicação das
%   restrições (máscaras) e a execução do algoritmo de
%   processamento das imagens.
%
%   COEF = coeficientes(CADEIA)
%   Recebe a estrutura CADEIA contendo as cadeias codificadas e
%   retorna na estrutura COEF os descritores de Fourier. Calcula
%   também os erros máximo, local e quadrático médio da
%   reconstrução das imagens pelos descritores a partir de uma
%   aproximação para um certo número de harmônicos.

% Alocação da cadeia codificada
c = cadeia;
% Comprimento da cadeia codificada
K = length(c);

T=0;
for i=1:K

   % Comprimento total do arco
   % Sendo v(i) o segmento orientado associado ao caractere c(i) da
   % cadeia codificada, em que |v(i)| =
   % (1+((sqrt(2)-1)/2)*(1-(-1)^(c(i)))), o comprimento do arco é
   % calculado
   T=T+(1+((sqrt(2)-1)/2)*(1-(-1)^(c(i))));

%    v(i)=(1+((sqrt(2)-1)/2)*(1-(-1)^(c(i))))* ...
%        (cos(pi*c(i)/4)+j*sin(pi*c(i)/4));

   % Comprimento de cada segmento
   deltat(i)= 1+((sqrt(2)-1)/2)*(1-(-1)^(c(i)));
   % Comprimento de cada segmento na projeção x
   deltax(i)=sign(6-c(i))*sign(2-c(i));
   % Comprimento de cada segmento na projeção y
```

```
   deltay(i)=sign(4-c(i))*sign(c(i));

end

clear i;

for p=1:K

   t(p)=0;
   x(p)=0;
   y(p)=0;

   for i=1:p

      % Comprimento do arco acumulado
      t(p)=t(p)+deltat(i);
      % Soma dos comprimentos das projeções x dos segmentos
      x(p)=x(p)+deltax(i);
      % Soma dos comprimentos das projeções y dos segmentos
      y(p)=y(p)+deltay(i);

   end

end
%-------------------------------------------------------------------
clear i, clear p;

% Determinação dos coeficientes de Fourier
% Número de harmônicos a considerar
m=25;

%loop que varia o nř de harmônicas.
for N=1:m;

   for n=1:N

      An=(deltax(1)/deltat(1))*(cos(2*n*pi*t(1)/T)-1);
      Bn=(deltax(1)/deltat(1))*sin(2*n*pi*t(1)/T);
      Cn=(deltay(1)/deltat(1))*(cos(2*n*pi*t(1)/T)-1);
      Dn=(deltay(1)/deltat(1))*sin(2*n*pi*t(1)/T);

      for p=2:K

```

```
        An=An+(deltax(p)/deltat(p))*( cos(2*n*pi*t(p)/T)- ...
            cos(2*n*pi*t(p-1)/T) );
        Bn=Bn+(deltax(p)/deltat(p))*( sin(2*n*pi*t(p)/T)- ...
            sin(2*n*pi*t(p-1)/T) );
        Cn=Cn+(deltay(p)/deltat(p))*( cos(2*n*pi*t(p)/T)- ...
            cos(2*n*pi*t(p-1)/T) );
        Dn=Dn+(deltay(p)/deltat(p))*( sin(2*n*pi*t(p)/T)- ...
            sin(2*n*pi*t(p-1)/T) );

    end

    % Coeficientes An
    A(n)=( T/(2*(n^2)*(pi^2)) )*An;
    % Coeficientes Bn
    B(n)=( T/(2*(n^2)*(pi^2)) )*Bn;
    % Coeficientes Cn
    C(n)=( T/(2*(n^2)*(pi^2)) )*Cn;
    % Coeficientes Dn
    D(n)=( T/(2*(n^2)*(pi^2)) )*Dn;

end

clear An Bn Cn Dn p n i;

% Salva os coeficientes na estrutura coef
coef.A = A;
coef.B = B;
coef.C = C;
coef.D = D;
%-----------------------------------------------------------------
% Cálculo dos componenetes DC
zeta(1)=0; delta(1)=0;

a0=( deltax(1)/(2*deltat(1)) )*(t(1)^2)+zeta(1)*t(1);
c0=( deltay(1)/(2*deltat(1)) )*(t(1)^2)+delta(1)*t(1);

for p=2:K

    zetap1=0;
    zetap2=0;
    deltap1=0;
    deltap2=0;
```

```
        for g=1:p-1

            zetap1=zetap1+deltax(g);
            zetap2=zetap2+deltat(g);
            deltap1=deltap1+deltay(g);
            deltap2=deltap2+deltat(g);

        end

        zeta(p)=zetap1-(deltax(p)/deltat(p))*zetap2;
        delta(p)=deltap1-(deltay(p)/deltat(p))*deltap2;
        a0=a0+( deltax(p)/(2*deltat(p)) )*( (t(p)^2)-t(p-1)^2 )+ ...
            zeta(p)*( t(p)-t(p-1) );
        c0=c0+( deltay(p)/(2*deltat(p)) )*( (t(p)^2)-t(p-1)^2 )+ ...
            delta(p)*( t(p)-t(p-1) );

    end

    clear zeta zetap1 zetap2 delta deltap1 deltap2;

    % Coeficiente A0
    A0=a0/T;
    % Coeficiente C0
    C0=c0/T;

    % Salva os coeficientes DC na estrutura coef
    coef.A0 = A0;
    coef.C0 = C0;

    clear a0, clear c0, clear p;

    %t1=t(1):(t(K)-t(1))/(10*K):t(K);
    %for p=1:10*K+1

        %X(p)=A0;
        %Y(p)=C0;

        %for n=1:N
%
%           X(p)=X(p)+( A(n)*cos((2*n*pi*t1(p))/T) + ...
%               B(n)*sin((2*n*pi*t1(p))/T) );
%           Y(p)=Y(p)+( C(n)*cos((2*n*pi*t1(p))/T) + ...
%               D(n)*sin((2*n*pi*t1(p))/T) );
```

```
%
    %end

%end

% Definição do erro
% Erro quadrático médio e erro local
% Criam-se as matrizes com os tamanhos necessarios para se
% calcular os erros.

eraux =zeros(K,N);
errox =zeros(K,N);
erroy =zeros(K,N);
erqm = zeros(1,N);
err = zeros(1,N);
Xav = zeros(1,K);
Yav = zeros(1,K);

for p=1:K

    % Coeficientes Dc
    Xav(p)= A0;
    Yav(p)= C0;

    for n=1:N %Executa os somatórios dos 'n' harmônicos.

        % As matrizes de erro guardarão as operações para cada
        % ponto associado ao numero de harmônicos.
        Xav(p)= Xav(p)+ (A(n)*cos((2*n*pi*t(p))/T) + ...
            B(n)*sin((2*n*pi*t(p))/T));
        Yav(p)= Yav(p)+ (C(n)*cos((2*n*pi*t(p))/T) + ...
            D(n)*sin((2*n*pi*t(p))/T));
        eraux(p,n) = sqrt((x(p)-Xav(p)).^2+(y(p)-Yav(p)).^2);
        errox(p,n) = (abs(x(p) - Xav(p)));
        erroy(p,n) = (abs(y(p) - Yav(p)));

    end

end

% Loop para computar o erro local
for n=1:N
```

```
        err(n) = max(max(errox(:,n)),max(erroy(:,n)));

    end

    % Loop duplo para computar o erro quadrático médio
    for n=1:N

        erqm(n) = 0;

        for p=1:K

            erqm(n) = erqm(n)+ eraux(p,n);

        end

        erqm(n)=erqm(n)/K;

    end

    % Erro máximo
    V0x=0;V0y=0;

    for i=2:K

       V0x=V0x+abs((deltax(i)/deltat(i))-(deltax(i-1)/deltat(i-1)));
       V0y=V0y+abs((deltay(i)/deltat(i))-(deltay(i-1)/deltat(i-1)));

    end

    er(N)=(T/(2*(pi^2)*N))*max(V0x,V0y);

% Fim do for principal
end
```

Programas utilizados no treinamento da rede neural:

```
% treinamento.m
% Separação dos dados de treinamento e de validação da rede neural
% e treinamento.

% inicializa dados de validação
entrada=[];
saida_desejada=[];
```

```
% inicializa dados de treinamento
net_P=[];
net_T=[];

%separação de dados
for i=1:length(data)
    if(mod(i,5)==0)
        entrada=[entrada data(i).net_input];
        saida_desejada=[saida_desejada data(i).net_target];
    else
        net_P=[net_P data(i).net_input];
        net_T=[net_T data(i).net_target];
    end
end

%treinamento
net = lab3(net_P,net_T);
```

```
function net=lab3(P,T)
%-----------------------------------------------------------------
%Treinamento da Rede Neural

%%
% [P,T] são os vetores de entrada e o target para treinamento
% P é um vetor 100 x N em que cada coluna está na forma [A B C D]';
%OBS: Os níveis cc [A0, B0] estão sendo desconsiderados

    %-------------------------------------------------------------
    %       NEWFF(P,T,S,TF,BTF,BLF,PF,IPF,OPF,DDF) takes,
    %       P  - RxQ1 matrix of Q1 representative R-element input
    %            vectors.
    %       T  - SNxQ2 matrix of Q2 representative SN-element
    %            target vectors.
    %       Si  - Sizes of N-1 hidden layers, S1 to S(N-1),
    %            default = [].
    %           (Output layer size SN is determined from T.)
    %       TFi - Transfer function of ith layer. Default is
    %           'tansig' forhidden layers, and 'purelin' for
    %           output layer.
    %       BTF - Backprop network training function, default =
    %            'trainlm'.
    %       BLF - Backprop weight/bias learning function,
    %            default = 'learngdm'.
```

```
    %         PF  - Performance function, default = 'mse'.
    %         IPF - Row cell array of input processing functions.
    %               Default is
    %                {'fixunknowns','remconstantrows','mapminmax'}.
    %         OPF - Row cell array of output processing functions.
    %               Default is {'remconstantrows','mapminmax'}.
    %         DDF - Data division function, default = 'dividerand';
    %-------------------------------------------------------------
    net = newff(P,T,[25 20],({'tansig','tansig'}),'trainbfg', ...
        'learngdm','mse');

%      net.layers{1}.transferFcn='purelin';
%      net.layers{2}.transferFcn= 'purelin';%'tansig';
%      net.performFcn='mse';
%      net.trainfcn='trainbfg';
%      %net.trainfcn='trainlm';

     net.trainParam.epochs = 1000;  %número de ciclos
     net.trainParam.goal= 1e-10;    %parâmetro de parada
     net.trainParam.lr=0.001;       %taxa de aprendizagem

     net = train(net,P,T);
%%
save net;

% y=sim(net,P(:,11));
% disp(y);
%-------------------------------------------------------------------
```