UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROFESSOR: DR. ING. ADOLFO BAUCHSPIESS

Classificação de Imagens Codificadas por Cadeias Direcionais Utilizando Redes Neurais Artificiais

Nome: ***Kleyton Cordeiro de Oliveira*** **Matrícula: 94/04643**

Brasília, dezembro de 1999

Agradecimentos

Muitos são os agradecimentos a serem feitos neste momento e me entristece o fato de não saber como me expressar para agradecer como gostaria a todos aqueles que tornaram possível esta conquista. Primeiramente, meus sinceros agradecimentos a Deus, quem realmente realizou todos os trabalhos, e a que devem ser creditados quaisquer méritos que porventura pensem ser meus. Agradeço também a meus pais e meu irmão, que deixaram de pensar neles mesmos, me apoiando durante todo o curso de Engenharia Elétrica e invariavelmente perdendo sono ou momentos de descanso quando não havia necessidade apenas para me acompanhar. Gostaria de agradecer, também, aos professores e funcionários do departamento de Engenharia Elétrica e da UnB, em especial ao professor Dr. Ing. Adolfo Bauchspiess, que demonstrou uma paciência e dedicação como poucas vezes presenciei, acolhendo-me como orientando em três trabalhos e não desanimando mesmo quando os resultados dos trabalhos eram insatisfatórios. Por fim, embora de modo algum menos importante, o meu agradecimento aos meus amigos, grande parte deles colegas engenheiros, que mostraram-se sempre dispostos a animar uns aos outros, que foram sempre compreensivos e que ajudaram, mesmo, na formação final do caráter. Agradecimentos especiais aos meus amigos de Faculdade que tantas vezes viraram noites comigo e que são meus irmãos, dentre os quais cito: Cezar A. A. Carioca, Julio Tatugawa Junior, Graziela Brunalle, Rodrigo Pedroso, Rodrigo Neves, Frank E. G. Amorim, Cristiano J. Miosso, Anderson Nascimento, Mirele Mencari e muitos outros amigos que resultariam em uma lista demasiadamente longa. Muito Obrigado.

Kleyton Cordeiro de Oliveira

Índice Analítico

Índice de Figuras

Capítulo 1

Introdução

Estima-se que cerca de 75% das informações processadas por um ser humano têm origem visual [1]. A importância das informações visuais para os seres humanos é refletida por um interesse crescente nos processos de análise de imagens, tanto sob o aspecto matemático, sendo estudados os aspectos espaciais, como dimensões e orientações, quanto sob o aspecto computacional, sendo estudadas formas de se obter informações diversas, variando desde distâncias de objetos até temperaturas.

Uma das áreas que se ocupam de realizar análise de imagens é a chamada visão computacional, a qual tem por finalidade obter, a partir de uma imagem, informações geométricas, topológicas ou físicas sobre o ambiente representado em uma imagem [2]. As técnicas dessa área recebem grande interesse por parte da robótica, com o objetivo de prover o sentido de visão aos sistemas automatizados.

Dentre as diversas aplicações de processamento e reconhecimento computacional de imagens, encontram-se:

- classificação de cromossomos a partir de imagens de um microscópio;
- análise de imagens de raios-X;
- mapeamento de estruturas subterrâneas;
- reconstrução tridimensional de estruturas espaciais; e
- inspeção de peças em linhas de montagens ou produtos agrícolas em indústrias de processamento de alimentos.

Das aplicações acima citadas, a última serviu de inspiração para o desenvolvimento do presente trabalho.

Quando se trata da classificação de objetos, alguns aspectos referentes aos mesmos, tais como orientação e tamanho, além de aspectos referentes ao ambiente onde se encontra o sistema, tais como grau de luminosidade, que indica se o ambiente está mais ou menos iluminado, ou ruídos em geral introduzidos no sinal de entrada, devem ser considerados. Tais aspectos geram restrições, como, por exemplo, uma quantidade finita de orientações ou faixas de tamanho das imagens permitidas aos objetos sob inspeção.

A robustez, definida no presente trabalho como sendo a capacidade do sistema em reconhecer um dado objeto sob diversas orientações e tamanhos, variações de iluminação do ambiente ou mesmo face a ruídos presentes na imagem, apresenta-se como um fator de grande importância a ser implementado no sistema.

O objetivo deste trabalho foi o desenvolvimento de um sistema robusto de classificação de imagens, segundo a definição de robustez apresentada acima. Conforme será apresentado nos capítulos 2 a 4, a técnica empregada consistiu na obtenção dos coeficientes normalizados das séries de Fourier das imagens codificadas por um método chamado de codificação por cadeias direcionais [3], sendo que estes conceitos serão definidos no texto. Cadeias direcionais dos contornos de objetos permitem transformar a informação da imagem para um domínio unidimensional.

O processo teve continuidade, com a apresentação dos coeficientes das séries de Fourier como pares de treinamento para duas arquiteturas de redes neurais artificiais adotadas neste trabalho.

O Capítulo 2 traz uma apresentação dos conceitos básicos empregados durante o projeto, incluindo as cadeias direcionais na seção 2.1 e uma descrição geral das redes neurais artificiais na seção 2.2. As seções 2.3 e 2.4 focalizam a atenção sobre as duas arquiteturas de redes neurais estudadas neste trabalho.

No Capítulo 3 encontram-se as descrições dos procedimentos experimentais e os recursos empregados neste trabalho, tendo sido apresentada, na seção 3.1, a elaboração da Base de Dados para treinamento e validação do sistema de classificação de imagens, a seção 3.2, por sua vez, explica como foram codificadas as imagens. Os processos de obtenção dos coeficientes de Fourier e de classificação das imagens são apresentados nas seções 3.3 e 3.4.

No Capítulo 4 estão relacionados os resultados experimentais obtidos neste trabalho, sendo apresentados, nas seções 4.1 a 4.3, algumas tabelas e gráficos que ilustram os resultados e as análises feitas sobre os mesmos.

O Capítulo 5 traz as conclusões finais sobre o projeto em si e algumas conclusões gerais acerca dos resultados mostrados no Capítulo 4.

Por fim há, no final deste relatório, três apêndices nos quais encontram-se, respectivamente os resultados gráficos das expansões de todas as imagens da Base de Dados em séries de Fourier, estes mesmos resultados dispostos em tabelas e as listagens dos programas desenvolvidos durante este trabalho de Projeto Final II.

Capítulo 2

Revisão Bibliográfica

2.1. Cadeias Direcionais

2.1.1. Definições

Uma imagem pode ser especificada em um computador segundo três níveis distintos: imagem contínua, imagem discretizada e imagem codificada [2]. Além disso, a discretização de uma imagem pode ocorrer de duas maneiras diferentes: discretização do domínio e discretização do espaço de cor.

Uma forma de se efetuar o reconhecimento de imagens é através da comparação de conjuntos ordenados de caracteres, ou palavras, conhecidas como cadeias direcionais, as quais serão definidas mais adiante. Tais palavras são obtidas a partir da codificação de uma dada figura em um conjunto finito de informações [3]. Nota-se que esta técnica é, essencialmente, uma técnica de compressão de dados, visto que, dada uma determinada imagem, sua representação em cadeias direcionais requer menos espaço para armazenamento das informações necessárias para recuperar os arquivos do que os dados originais.

Em geral, as técnicas de compressão são compostas de 4 etapas [2]:

- uma discretização espacial da imagem;
- a aplicação das transformadas de Fourier no modelo discreto-contínuo resultante da discretização espacial da imagem;
- uma truncagem da imagem com a finalidade de reduzir o número de termos da seqüência de representação da imagem no domínio espectral;

- quantização do modelo discreto-contínuo da imagem;

A codificação por cadeia, por sua vez, utiliza-se de um alfabeto A formado por 8 caracteres A = {0,1,...,7}. As infinitas possibilidades de palavras que podem ser criadas a partir dos caracteres de A geram o espaço A*. Cada caractere do alfabeto A representa um segmento orientado de reta, no espaço $\Re^2$, de comprimento 1 ou $\sqrt{2}$ e de direção múltipla de 45º. Deseja-se obter, através desta definição, uma ferramenta para a codificação das bordas de uma imagem digitalizada. O segmento associado a cada caractere a_i é dado pela Equação (1) [3]:

$$v_i = \left[1 + \frac{\sqrt{2}-1}{2}\left(1-(-1)^{a_i}\right)\right] \angle\left(\frac{\pi}{4}\right) a_i \qquad \textbf{(1)}$$

A Figura 1 apresenta os segmentos orientados e seus respectivos caracteres:

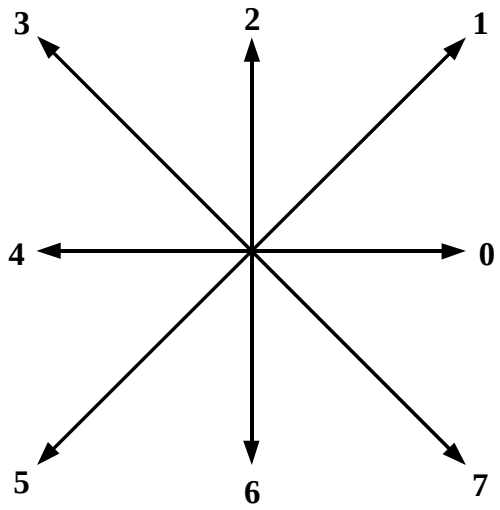


Figura 1 – Segmentos orientados do alfabeto A

Uma palavra, formada por dois ou mais caracteres de A, é denominada um vetor em A*. Define-se, então, comprimento de um vetor no espaço A* como sendo a quantidade de caracteres que compõem uma palavra.

Estabelecidos os conceitos de vetor e comprimento de vetor no espaço A*, define-se uma cadeia direcional $c = c_1c_2...c_K$ como um vetor de comprimento m em A* cujos elementos correspondem a caracteres de A organizados de modo que seus respectivos segmentos orientados estejam conectados de forma a reproduzir, da melhor maneira possível, os contornos de uma dada imagem [3].

Como um exemplo, considere um vetor c = 0022746005443. A imagem associada a c será:

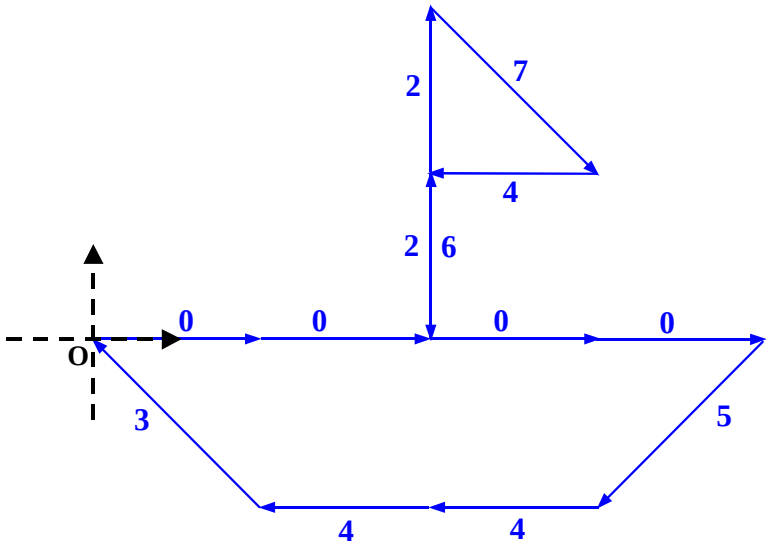


Figura 2 – Imagem associada ao vetor c = 0022746005443

A imagem é formada da seguinte maneira: a origem do segmento v_1, correspondente a c_1, é colocada na origem das coordenadas. O processo prossegue, então, com a origem de v_i sendo conectada à ponta de v_{i-1}, até que seja posicionado o último segmento, v_m, correspondente a c_m.

Nota-se que a imagem associada a *c* é uma imagem linear por partes e que existe apenas uma única imagem associada a uma determinada cadeia direcional. No entanto, uma mesma imagem pode ser representada por diversas cadeias direcionais diferentes, dependendo do ponto da imagem em que se quer iniciar a descrição da mesma.

Define-se o comprimento de arco obtido ao percorrer um segmento de reta associado a um determinado elemento a_i por:

$$\Delta t_i = 1 + \frac{\left(\sqrt{2}-1\right)}{2}\left(1-(-1)^{a_i}\right) \quad \textbf{(2)}$$

Uma vez que uma imagem, segundo a técnica de codificação em cadeias direcionais, é sempre construída a partir da origem dos eixos, pode-se obter o comprimento de arco dos primeiros *p* elementos através da Equação (3):

$$t_p = \sum_{i=1}^{p} \Delta t_i \quad \textbf{(3)}$$

Se $c = c_1c_2...c_K$, for um vetor em A*, então a soma dos comprimentos dos segmentos de reta $v_1,v_2,...v_i$, componentes de *v*, resulta em um comprimento de arco total, T, dado por:

$$\mathrm{T} = \sum_{i=1}^{K}\left[1 + \frac{\sqrt{2}-1}{2}\left(1-(-1)^{c_i}\right)\right] \quad \textbf{(4)}$$

Sendo *v* uma imagem no plano, é natural pensar em uma representação matemática em forma de uma curva fechada *v(t)* em $\Re^2$. De fato, admitindo-se que *t* represente o parâmetro comprimento de arco, pode-se considerar *v* como uma tal curva, apresentando componentes horizontais e verticais *x(t)* e *y(t)*, respectivamente, descritas como funções contínuas de t, com $0 \leq t \leq \mathrm{T}$.

As variações nas componentes horizontais e verticais *x(t)* e *y(t)*, denominadas funções de projeção *x-y*, ou simplesmente projeções *x-y*, causadas por um caractere a_i são calculadas através das relações:

$$\begin{aligned} \Delta x_i &= \operatorname{sgn}(6 - a_i)\operatorname{sgn}(2 - a_i) \\ \Delta y_i &= \operatorname{sgn}(4 - a_i)\operatorname{sgn}(a_i) \end{aligned} \tag{5}$$

onde a função sgn() é dada por:

$$\operatorname{sgn}(\mathrm{z}) = \begin{cases} 1, & \mathrm{z} > 0 \\ 0, & \mathrm{z} = 0 \\ -1, & \mathrm{z} < 0 \end{cases} \tag{6}$$

A exemplo do que ocorre com o comprimento de arco *t*, também é possível obter as projeções *x* e *y* dos primeiros *p* elementos de uma cadeia direcional.

$$\begin{aligned} x_p &= \sum_{i=1}^{p} \Delta x_i \\ y_p &= \sum_{i=1}^{p} \Delta y_i \end{aligned} \tag{7}$$

Os gráficos de $x(t) \times t$ e $y(t) \times t$, projeções x-y da Figura 2, estão ilustrados na Figura 3.

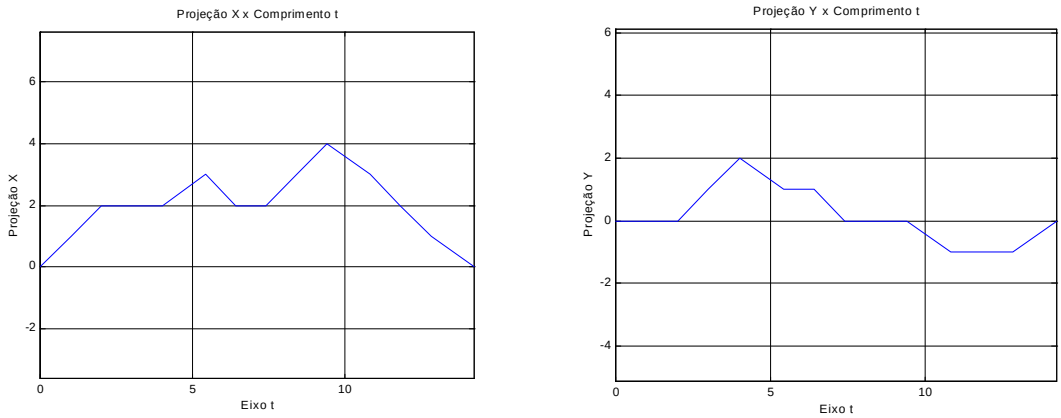


Figura 3 – Projeções x-$y \times t$

Os valores de $x(t)$ e $y(t)$ repetem-se quando a imagem v é percorrida mais de uma vez, ou seja, $x(t)$ e $y(t)$ possuem a propriedade de periodicidade, sendo o comprimento de arco da imagem, T, o período fundamental das funções de projeção x-y. Além de periódicas, as funções de projeção x-y são contínuas. Funções contínuas são limitadas e integráveis. Uma vez que as projeções x-y são integráveis e limitadas, elas são, também, absolutamente integráveis [4]. Portanto, as funções de projeção x-y atendem a todos os requisitos necessários para a sua representação em forma de séries de Fourier. Maiores detalhes e demonstrações matematicamente rigorosas das afirmações utilizadas acima para determinar se as projeções x-y podem ser descritas por séries de Fourier são encontradas em alguns textos citados nas Referências Bibliográficas [4].

Justifica-se a representação das projeções *x-y* em forma de séries de Fourier pelo fato de que, com uma quantidade finita de coeficientes de Fourier, ou harmônicos, é possível reconstruir, de maneira satisfatória, as projeções da imagem original *v*, a qual pode ser constituída de uma quantidade muito grande de segmentos orientados v_i. Desta maneira, um conjunto pré-determinado de coeficientes de Fourier pode ser empregado como um conjunto contendo informações sobre toda a imagem, sendo tais informações mais precisas quanto maior for a quantidade de harmônicos no conjunto.

2.1.2. Formulações Matemáticas

As representações das projeções *x*-*y* em forma de séries de Fourier são descritas por:

$$x(t) = A_0 + \sum_{n=1}^{\infty} A_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + B_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right)$$
$$y(t) = C_0 + \sum_{n=1}^{\infty} C_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + D_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right) \quad \textbf{(8)}$$

onde A_i, B_i, C_i e D_i são os coeficientes não normalizados de Fourier:

$$A_0 = \frac{1}{\mathrm{T}}\int_0^{\mathrm{T}} x(t)dt \qquad C_0 = \frac{1}{\mathrm{T}}\int_0^{\mathrm{T}} y(t)dt$$
$$A_n = \frac{2}{\mathrm{T}}\int_0^{\mathrm{T}} x(t)\cos\left(\frac{2n\pi t}{\mathrm{T}}\right)dt \qquad C_n = \frac{2}{\mathrm{T}}\int_0^{\mathrm{T}} y(t)\cos\left(\frac{2n\pi t}{\mathrm{T}}\right)dt \quad \textbf{(9)}$$
$$B_n = \frac{2}{\mathrm{T}}\int_0^{\mathrm{T}} x(t)\operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right)dt \qquad D_n = \frac{2}{\mathrm{T}}\int_0^{\mathrm{T}} y(t)\operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right)dt$$

A aproximação discreta de uma integral pode ser feita por uma regra conhecida como regra trapezoidal [5], segundo a qual uma integral

$$\Im = \int_0^T e(t)dt$$

.possui uma aproximação

$$\mathrm{I} = \sum_{p=1}^{K} \frac{\left(t_p - t_{p-1}\right)}{2}\left(e_p + e_{p-1}\right)$$

Assim sendo, a aproximação discreta de A_0 é dada por:

$$A_0 = \frac{1}{T}\sum_{p=1}^{K}\frac{\left(t_p - t_{p-1}\right)}{2}\left(x_p + x_{p-1}\right) \qquad \textbf{(10)}$$

A fim de aproveitar os valores já calculados de Δx_p calculados pela Equação (5), pode-se escrever a Equação (10) como:

$$A_0 = \frac{1}{T}\sum_{p=1}^{K}\left(x_p - x_{p-1}\right)\frac{\left(t_p - t_{p-1}\right)}{2} + \left(t_p - t_{p-1}\right)x_{p-1}$$

Pelas Equações (3) e (7), tem-se que

$$x_p - x_{p-1} = \sum_{i=1}^{p}\Delta x_i - \sum_{i=1}^{p-1}\Delta x_i$$

$$x_p - x_{p-1} = \left(\Delta x_p + \sum_{i=1}^{p-1}\Delta x_i\right) - \sum_{i=1}^{p-1}\Delta x_i = \Delta x_p$$

e, analogamente, $t_p - t_{p-1} = \Delta t_p$.

Segue, então, que:

$$A_0 = \frac{1}{T}\sum_{p=1}^{K}\Delta x_p\left(\frac{\left(t_p + t_{p-1}\right)}{2} - \sum_{j=1}^{p-1}\Delta t_j\right) + \left(t_p - t_{p-1}\right)\sum_{j=1}^{p-1}\Delta x_j$$

$$A_0 = \frac{1}{T}\sum_{p=1}^{K}\frac{\Delta x_p}{2}\left(t_p + t_{p-1}\right) + \left(t_p - t_{p-1}\right)\left[\sum_{j=1}^{p-1}\Delta x_j - \frac{\Delta x_p}{\Delta t_p}\sum_{j=1}^{p-1}\Delta t_j\right]$$

Obtém-se, então, a Equação (11):

$$A_0 = \frac{1}{\mathrm{T}}\sum_{p=1}^{K}\frac{\Delta x_p}{2\Delta t_p}\left(t_p^2 - t_{p-1}^2\right) + \xi_p\left(t_p - t_{p-1}\right) \quad \textbf{(11)}$$

onde

$$\xi_p = \sum_{j=1}^{p-1}\Delta x_j - \frac{\Delta x_p}{\Delta t_p}\sum_{j=1}^{p-1}\Delta t_j \quad \textbf{(12)}$$

De maneira análoga, tem-se, para a componente *y(t)*:

$$C_0 = \frac{1}{\mathrm{T}}\sum_{p=1}^{K}\frac{\Delta y_p}{2\Delta t_p}\left(t_p^2 - t_{p-1}^2\right) + \delta_p\left(t_p - t_{p-1}\right) \quad \textbf{(13)}$$

com

$$\delta_p = \sum_{j=1}^{p-1}\Delta y_j - \frac{\Delta y_p}{\Delta t_p}\sum_{j=1}^{p-1}\Delta t_j \quad \textbf{(14)}$$

Os valores de ξ_1 e δ_1, nas Equações (12) e (14), são nulos.

Partindo da Equação (9), tem-se:

$$A_n = \frac{2}{T}\int_0^T x(t)\cos\left(\frac{2n\pi t}{T}\right)dt = \frac{2}{T}\left[\frac{T}{2n\pi}x(t)\operatorname{sen}\left(\frac{2n\pi t}{T}\right)\Big|_0^T - \int_0^T x'(t)\frac{T}{2n\pi}\operatorname{sen}\left(\frac{2n\pi t}{T}\right)dt\right]$$

$$A_n = -\frac{1}{n\pi}\int_0^T x'(t)\operatorname{sen}\left(\frac{2n\pi t}{T}\right)dt = \frac{1}{n\pi}\left[\frac{T}{2n\pi}x'(t)\cos\left(\frac{2n\pi t}{T}\right)\Big|_0^T - \int_0^T x''(t)\frac{T}{2n\pi}\cos\left(\frac{2n\pi t}{T}\right)dt\right]$$

$$A_n = -\frac{T}{2n^2\pi^2}\int_0^T x''(t)\cos\left(\frac{2n\pi t}{T}\right)dt \tag{15}$$

Discretizando a Equação (15), chega-se a:

$$A_n = -\frac{T}{2n^2\pi^2}\sum_{p=1}^{K}\frac{\left(\frac{x_p - x_{p-1}}{t_p - t_{p-1}}\right) - \left(\frac{x_{p-1} - x_{p-2}}{t_{p-1} - t_{p-2}}\right)}{t_p - t_{p-1}}\cos\left(\frac{2n\pi t_p}{T}\right)$$

de onde obtém-se, devido à natureza contínua e linear por partes de *x(t)*, o resultado apresentado na Equação (16).

$$A_n = \frac{\mathrm{T}}{2n^2\pi^2}\sum_{p=1}^{K}\frac{\Delta x_p}{\Delta t_p}\left[\cos\left(\frac{2n\pi t_p}{\mathrm{T}}\right) - \cos\left(\frac{2n\pi t_{p-1}}{\mathrm{T}}\right)\right] \tag{16}$$

Seguindo o mesmo raciocínio, calculam-se os coeficientes B_n pela Equação (17).

$$B_n = \frac{\mathrm{T}}{2n^2\pi^2}\sum_{p=1}^{K}\frac{\Delta x_p}{\Delta t_p}\left[\mathrm{sen}\left(\frac{2n\pi t_p}{\mathrm{T}}\right) - \mathrm{sen}\left(\frac{2n\pi t_{p-1}}{\mathrm{T}}\right)\right] \tag{17}$$

De maneira análoga, calculam-se os coeficientes C_n e D_n, de *y(t)*.

$$C_n = \frac{\mathrm{T}}{2n^2\pi^2}\sum_{p=1}^{K}\frac{\Delta y_p}{\Delta t_p}\left[cos\left(\frac{2n\pi t_p}{\mathrm{T}}\right) - cos\left(\frac{2n\pi t_{p-1}}{\mathrm{T}}\right)\right]$$

$$D_n = \frac{\mathrm{T}}{2n^2\pi^2}\sum_{p=1}^{K}\frac{\Delta y_p}{\Delta t_p}\left[sen\left(\frac{2n\pi t_p}{\mathrm{T}}\right) - sen\left(\frac{2n\pi t_{p-1}}{\mathrm{T}}\right)\right] \tag{18}$$

É uma prática comum adotar um erro máximo admissível, ε_s, e calcular o número de harmônicos, *N*, necessários para que a diferença entre o sinal, *s*, e sua aproximação por séries de Fourier, $\tilde{s}$, seja menor que o erro ε_s [4].

Sejam $\tilde{X}_N(t)$ e $\tilde{Y}_N(t)$ as aproximações por séries de Fourier, com *N* harmônicos, das projeções *x(t)* e *y(t)*, calculadas pela Equação (19).

$$\tilde{X}_N(t) = A_0 + \sum_{n=1}^{N} A_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + B_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right)$$

$$\tilde{Y}_N(t) = C_0 + \sum_{n=1}^{N} C_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + D_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right) \quad \textbf{(19)}$$

A Figura 4 mostra a imagem da Figura 2 e suas aproximações por séries de Fourier para N = 1, 2, 3, 4, 15 e 25 harmônicos.

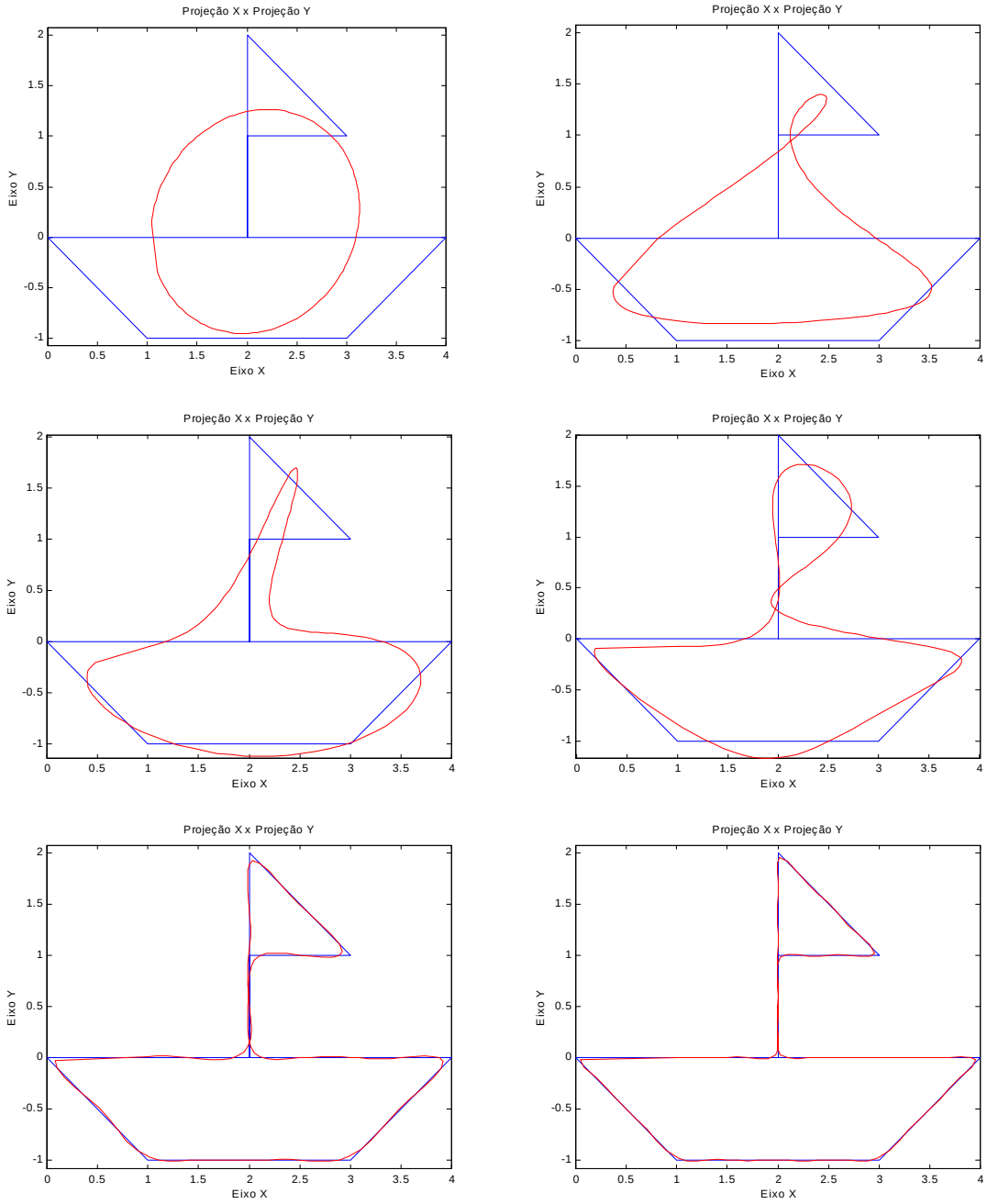


Figura 4 - Imagem gerada por uma cadeia direcional e harmônicos N = 1, 2, 3, 4, 15 e 25

Define-se o erro local ε como:

$$\varepsilon = max\left(\sup_t \left| x(t) - \tilde{X}_N(t) \right|, \sup_t \left| y(t) - \tilde{Y}_N(t) \right| \right) \quad \textbf{(20)}$$

onde $\sup_t(\eta(z))$ denota o supremo de $\eta(z)$.

Define-se, também, um valor de erro máximo, dado por:

$$\varepsilon_{máx} = \frac{\mathrm{T}}{2\pi^2 N} \max\left[V_0^{\mathrm{T}}(x'(t)), V_0^{\mathrm{T}}(y'(t)) \right] \quad \textbf{(21)}$$

onde $V_0^{\mathrm{T}}(x'(t))$ e $V_0^{\mathrm{T}}(y'(t))$ simbolizam as variações totais, ou as somas dos valores, das derivadas *x'(t)* e *y'(t)*. As derivadas das projeções *x*-*y* para o elemento a_i são:

$$\begin{aligned} x'_i &= \frac{\Delta x_i}{\Delta t_i} \\ y'_i &= \frac{\Delta y_i}{\Delta t_i} \end{aligned} \quad \textbf{(22)}$$

Calcula-se, então, os valores $V_0^{\mathrm{T}}(x'(t))$ e $V_0^{\mathrm{T}}(y'(t))$ através da Equação (23).

$$\begin{aligned} V_0^{\mathrm{T}}(x'(t)) &= \sum_2^K \left| x'_i - x'_{i-1} \right| \\ V_0^{\mathrm{T}}(y'(t)) &= \sum_2^K \left| y'_i - y'_{i-1} \right| \end{aligned} \quad \textbf{(23)}$$

Dado o fato de *v(t)*, representar uma curva fechada em $\Re^2$, tem-se *v(0)* = *v(*T*)* e $\varepsilon \leq \varepsilon_{máx}$.

Por fim, para obter uma informação mais precisa sobre a diferença entre a imagem *v* e a imagem reconstruída a partir das aproximações de *x*-*y*, utiliza-se o erro quadrático médio ε_{qm} definido como:

$$\varepsilon_{qm}(N) = \frac{1}{K}\sum_{p=1}^{K}\sqrt{\left(x(t_p) - \tilde{X}_N(t_p)\right)^2 + \left(y(t_p) - \tilde{Y}_N(t_p)\right)^2} \qquad \textbf{(24)}$$

A Figura 5 apresenta uma comparação entre os diversos tipos de erros para a imagem da Figura 2.

Figura 5 - Comparação dos erros de aproximação de *v(t)*

2.1.3. Normalização dos Coeficientes de Fourier

Há, neste ponto, duas observações a serem feitas:

1. uma mesma imagem em $\Re^2$ possui várias representações em A*;
2. uma imagem *v'* obtida a partir de uma rotação, translação ou multiplicação de *v* por um fator possui uma representação *c'* em A* diferente do vetor *c*, correspondente a *v*.

A fim de solucionar estes problemas e obter um mesmo conjunto de coeficientes de Fourier para várias imagens distintas referentes a um mesmo objeto com diferentes orientações, tamanhos ou pontos de origem, recorre-se a uma normalização das aproximações por séries de Fourier pelas chamadas elipses harmônicas, as quais serão definidas mais adiante. A justificativa para a tentativa de se obter uma única representação para um mesmo objeto torna-se evidente pelo fato de significar uma redução na quantidade de padrões a serem armazenados e classificados.

Pode-se reescrever a Equação (19) como:

$$\begin{aligned} \tilde{X}_N(t) &= A_0 + \sum_{n=1}^{N} X_n(t) \\ \tilde{Y}_N(t) &= C_0 + \sum_{n=1}^{N} Y_n(t) \end{aligned} \qquad \textbf{(25)}$$

onde, para $1 \leq n \leq N$, tem-se:

$$\begin{aligned} X_n(t) &= A_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + B_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right) \\ Y_n(t) &= C_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + D_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right) \end{aligned} \qquad \textbf{(26)}$$

Da mesma forma, tem-se $v_n(t) = (X_n(t),Y_n(t))$, para todo n. Para cada n, quando t varia de 0 a T, $v_n(t)$ descreve uma elipse no plano. De fato, partindo da Equação (26) e considerando $A_nD_n - B_nC_n$ diferente de zero, obtém-se o seguinte desenvolvimento:

$$X_n = A_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + B_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right) \qquad \times(C_n)$$

$$Y_n = C_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + D_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right) \qquad \times(-A_n)$$

$$C_nX_n = A_nC_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + B_nC_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right)$$

$$-A_nY_n = -A_nC_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) - A_nD_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right)$$

$$C_nX_n - A_nY_n = (B_nC_n - A_nD_n)\operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right)$$

$$\operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right) = \frac{A_nY_n - C_nX_n}{A_nD_n - B_nC_n}$$

Retornando à Equação (26), tem-se:

$$X_n = A_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + B_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right) \qquad \times(D_n)$$

$$Y_n = C_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + D_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right) \qquad \times(-B_n)$$

$$D_nX_n = A_nD_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) + B_nD_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right)$$

$$-B_nY_n = -B_nC_n \cos\left(\frac{2n\pi t}{\mathrm{T}}\right) - B_nD_n \operatorname{sen}\left(\frac{2n\pi t}{\mathrm{T}}\right)$$

$$D_nX_n - B_nY_n = (A_nD_n - B_nC_n)\cos\left(\frac{2n\pi t}{\mathrm{T}}\right)$$

$$\cos\left(\frac{2n\pi t}{\mathrm{T}}\right) = \frac{D_nX_n - B_nY_n}{A_nD_n - B_nC_n}$$

$$cos^2\left(\frac{2n\pi t}{\mathrm{T}}\right)+sen^2\left(\frac{2n\pi t}{\mathrm{T}}\right)=1$$

$$\frac{\left(D_nX_n-B_nY_n\right)^2+\left(A_nY_n-C_nX_n\right)^2}{\left(A_nD_n-B_nC_n\right)^2}=1$$

Daí resulta em:

$$\frac{\left(C_n^2+D_n^2\right)X_n^2+\left(A_n^2+B_n^2\right)Y_n^2-2\left(A_nC_n+B_nD_n\right)X_nY_n}{\left(A_nD_n-B_nC_n\right)^2}=1 \qquad \textbf{(27)}$$

A (27) é a equação de uma elipse e é válida para todo *n*. Para o caso em que $A_nD_n - B_nC_n = 0$, a (27) representa uma elipse degenerada: uma reta. Vale ressaltar que a mesma elipse $v_n = (X_n, Y_n)$ será encontrada para o n-ésimo harmônico, não importando o ponto da imagem adotada como origem para a codificação em cadeia direcional.

O resultado de uma mudança de ponto de origem da imagem *v* nas formas de ondas das projeções *x-y* é uma mudança de fase, ou seja, se o ponto de origem de *v'* for deslocado λ unidades em relação ao ponto de origem de *v*, os valores das projeções *x-y* serão:

$$X_n\left(t^*+\lambda\right)=A_n\,cos\left(\frac{2n\pi\left(t^*+\lambda\right)}{\mathrm{T}}\right)+B_n\,sen\left(\frac{2n\pi\left(t^*+\lambda\right)}{\mathrm{T}}\right)$$

$$Y_n\left(t^*+\lambda\right)=C_n\,cos\left(\frac{2n\pi\left(t^*+\lambda\right)}{\mathrm{T}}\right)+D_n\,sen\left(\frac{2n\pi\left(t^*+\lambda\right)}{\mathrm{T}}\right) \qquad \textbf{(28)}$$

onde $t = t^* + \lambda$. Expandindo X_n e Y_n e agrupando os termos semelhantes, tem-se:

$$\begin{aligned} X_n^*(t^*) &= A_n^* \cos\left(\frac{2n\pi t^*}{\mathrm{T}}\right) + B_n^* \operatorname{sen}\left(\frac{2n\pi t^*}{\mathrm{T}}\right) \\ Y_n^*(t^*) &= C_n^* \cos\left(\frac{2n\pi t^*}{\mathrm{T}}\right) + D_n^* \operatorname{sen}\left(\frac{2n\pi t^*}{\mathrm{T}}\right) \end{aligned} \quad \textbf{(29)}$$

Com

$$\begin{pmatrix} A_n^* & C_n^* \\ B_n^* & D_n^* \end{pmatrix} = \begin{pmatrix} cos\left(\frac{2n\pi\lambda}{\mathrm{T}}\right) & sen\left(\frac{2n\pi\lambda}{\mathrm{T}}\right) \\ -sen\left(\frac{2n\pi\lambda}{\mathrm{T}}\right) & cos\left(\frac{2n\pi\lambda}{\mathrm{T}}\right) \end{pmatrix} \begin{pmatrix} A_n & C_n \\ B_n & D_n \end{pmatrix} \quad \textbf{(30)}$$

A Equação (30) garante uma normalização com relação ao ponto de origem da imagem *v*, no sentido de que, não importando o ponto de origem da imagem, os coeficientes de Fourier, uma vez aplicada a Equação (30) com deslocamentos λ adequados, serão iguais. O efeito desta normalização, portanto, é fixar um ponto de origem, no espaço, para a imagem em análise.

As projeções de um ponto no espaço $\Re^2$, originalmente em coordenadas X-Y, nos eixos de coordenadas U-V são dadas pela Equação (31).

$$\begin{pmatrix} U \\ V \end{pmatrix} = \begin{pmatrix} cos(\psi) & sen(\psi) \\ -sen(\psi) & cos(\psi) \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} \quad \textbf{(31)}$$

Onde ψ é a diferença angular entre os eixos X-Y e U-V. Para encontrar as projeções de A_n^*, B_n^*, C_n^* e D_n^* nos eixos U-V, aplica-se a Equação (31) em conjunto com a Equação (29). Adotando μ_n e ν_n como as projeções de X_n^* e Y_n^* sobre os eixos U-V, chega-se à Equação (32).

$$\begin{pmatrix} \mu_n \\ \nu_n \end{pmatrix} = \begin{pmatrix} cos(\psi) & sen(\psi) \\ -sen(\psi) & cos(\psi) \end{pmatrix} \begin{pmatrix} X_n^* \\ Y_n^* \end{pmatrix}$$

$$\begin{pmatrix} \mu_n \\ \nu_n \end{pmatrix} = \begin{pmatrix} cos(\psi) & sen(\psi) \\ -sen(\psi) & cos(\psi) \end{pmatrix} \begin{pmatrix} A_n^* & B_n^* \\ C_n^* & D_n^* \end{pmatrix} \begin{pmatrix} cos\left(\frac{2n\pi t^*}{\mathrm{T}}\right) \\ sen\left(\frac{2n\pi t^*}{\mathrm{T}}\right) \end{pmatrix} \quad \textbf{(32)}$$

A partir da Equação (32) são definidos coeficientes de Fourier, A_n^{**}, B_n^{**}, C_n^{**} e D_n^{**}, sobre os eixos U-V:

$$\begin{pmatrix} A_n^{**} & B_n^{**} \\ C_n^{**} & D_n^{**} \end{pmatrix} = \begin{pmatrix} cos(\psi) & sen(\psi) \\ -sen(\psi) & cos(\psi) \end{pmatrix} \begin{pmatrix} A_n^* & B_n^* \\ C_n^* & D_n^* \end{pmatrix} \quad \textbf{(33)}$$

A relação direta entre os coeficientes originais A_n, B_n, C_n, e D_n e os coeficientes após as operações conjugadas de rotação da imagem e de deslocamento de ponto inicial, A_n^{**}, B_n^{**}, C_n^{**} e D_n^{**}, pode ser escrita na forma:

$$\begin{pmatrix} A_n^{**} & B_n^{**} \\ C_n^{**} & D_n^{**} \end{pmatrix} = \begin{pmatrix} cos\,\psi & sen\,\psi \\ -sen\,\psi & cos\,\psi \end{pmatrix} \begin{pmatrix} A_n & B_n \\ C_n & D_n \end{pmatrix} \begin{pmatrix} cos\,\frac{2n\pi\lambda}{\mathrm{T}} & -sen\,\frac{2n\pi\lambda}{\mathrm{T}} \\ sen\,\frac{2n\pi\lambda}{\mathrm{T}} & cos\,\frac{2n\pi\lambda}{\mathrm{T}} \end{pmatrix} \quad \textbf{(34)}$$

O mapeamento $\begin{pmatrix} A_n \\ B_n \\ C_n \\ D_n \end{pmatrix} \rightarrow \begin{pmatrix} A_n^{**} \\ B_n^{**} \\ C_n^{**} \\ D_n^{**} \end{pmatrix}$, denominado Λ, normaliza os coeficientes de Fourier originais com relação a uma rotação da imagem original e de uma mudança de ponto inicial. Resta, no entanto, decidir qual o ponto inicial e a orientação padrão para

gerar representações canônicas das imagens a serem examinadas por cadeias direcionais, ou seja, dadas duas imagens distintas que representam um mesmo objeto, quaisquer que sejam os pontos iniciais ou as orientações das imagens apresentadas, a aplicação de Λ sobre os coeficientes de Fourier deverá resultar em um mesmo conjunto de coeficientes de Fourier, denominado de representação canônica.

Este problema é resolvido tomando-se a elipse dos harmônicos de ordem 1. O valor de λ_1 é tomado como sendo o valor necessário para que o ponto inicial da imagem *v'* esteja sobre o eixo semi-maior da elipse de primeiro harmônico. Este eixo determina, também, a rotação ψ_1 da imagem, de modo que a orientação da imagem seja igual à orientação do eixo. É importante ressaltar que, uma vez que eixo semi-maior cruza a elipse harmônica de ordem 1 em dois pontos distintos, existem sempre duas possibilidades de representações canônicas, distintas por uma rotação de 180º, resultando em dois conjuntos de coeficientes de Fourier com sinais opostos. Como exemplo de uma elipse de primeiro harmônico tem-se, na Figura 4, uma representação de uma imagem e sua reconstituição com apenas o primeiro harmônico. Esta imagem é repetida na Figura 6.

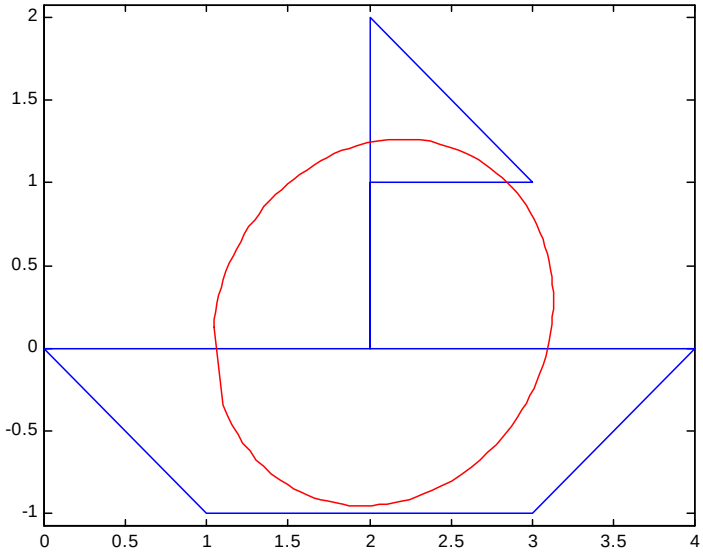


Figura 6 – Imagem linear por partes e sua elipse de primeiro harmônico

Matematicamente, o que foi apresentado é expresso como:

$$\lambda_1 = \frac{\mathrm{T}}{4\pi} arctan\left(\frac{2\left(A_1 B_1 + C_1 D_1\right)}{A_1^2 + C_1^2 - B_1^2 - D_1^2} \right) \quad \textbf{(35)}$$

e

$$\psi_1 = arctan\left(\frac{C_1^*}{A_1^*} \right) \quad \textbf{(36)}$$

A divisão de todos os coeficientes pelo comprimento do eixo semi-maior, $E^* = \sqrt{\left(A_1^*\right)^2 + \left(C_1^*\right)^2}$, gera uma normalização em relação ao tamanho da imagem codificada.

Por fim, para obter uma normalização em relação a uma translação da imagem, basta desconsiderar os componentes DC, para efeitos de comparação.

2.2. Redes Neurais Artificiais

O objetivo ao se trabalhar com inteligência artificial é criar máquinas que executem tarefas que normalmente requerem inteligência humana [6]. Assim, define-se inteligência artificial como um conjunto de métodos, ferramentas e sistemas utilizados para executar tais tarefas [7]. Inteligência, neste contexto compreende as habilidades de aprender, reagir de modo adaptativo, tomar decisões acertadas, comunicar-se através de imagens ou linguagens e de entender.

As diversas abordagens deste tema levaram ao desenvolvimento de várias formas de inteligência artificial. As principais formas de desenvolvimento de inteligência artificial são:

- a obtenção de resultados similares aos adquiridos por inteligência humana através de processos diferentes da maneira de pensar humana;
- a modelagem dos processos de pensamento ou funcionamento fisiológico do cérebro humano para gerar os resultados desejados.

As redes neurais artificiais, ou simplesmente redes neurais, são exemplos de sistemas desenvolvidos tendo em vista a segunda abordagem.

As redes neurais artificiais funcionam através de seus neurônios artificiais, que são unidades de processamento de funcionamento bastante simples e que processam seus dados usando paralelismo lógico combinado com operações seriais. A maneira como funcionam é, em diversos casos, análoga à dos neurônios biológicos [6], [8], [9], tendo sido fortemente inspirada no funcionamento dos neurônios proposto por McCulloch e Pitts em 1943.

A Figura 7 apresenta a constituição básica de um neurônio biológico.

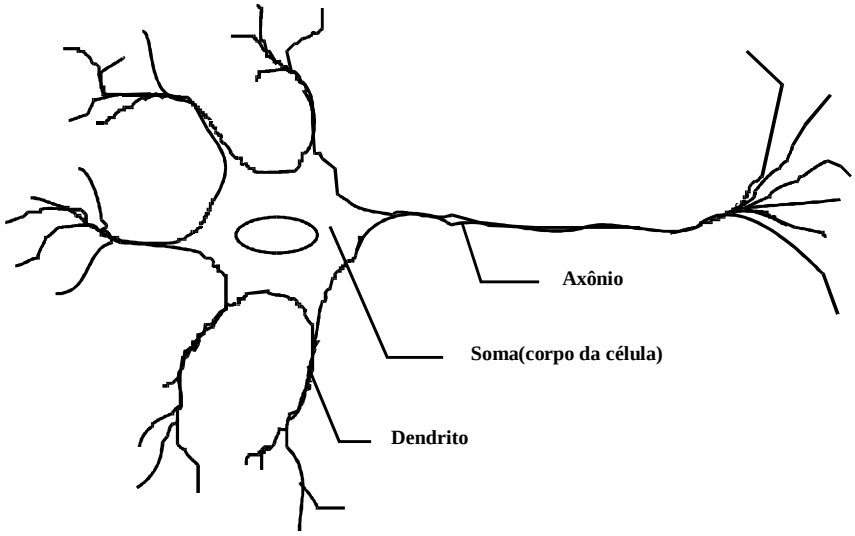


Figura 7 – Representação de um neurônio biológico

O propósito da rede neural é aprender a reconhecer padrões por meio de exemplos. O seu comportamento inteligente vem das interações entre as unidades de processamento da rede. Uma de suas características básicas é o processamento paralelo de informações que é a realização independente de várias tarefas simultâneas.

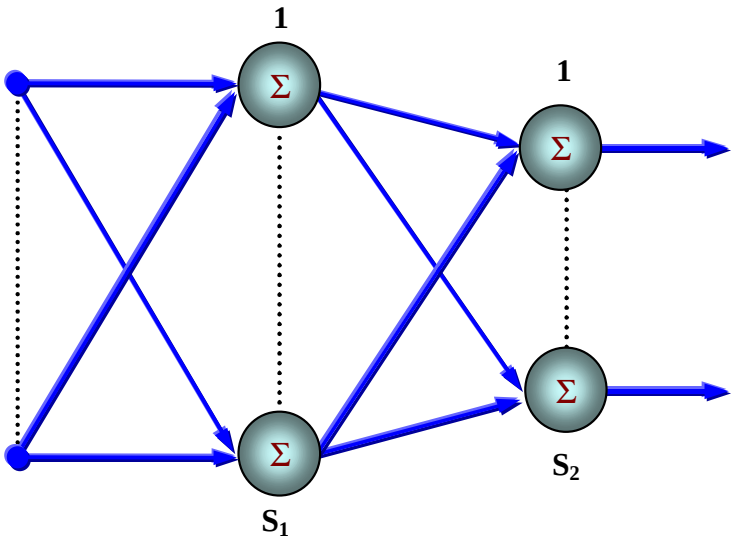


Figura 8 – Exemplo de Rede Neural Artificial (Perceptron Multicamadas)

Na Figura 8, tem-se um exemplo de rede neural, com unidades de processamento que somam os valores de suas entradas ponderadas pelos pesos de cada conexão.

Nas redes neurais artificiais, a informação se encontra nos pesos, ou seja, nos canais de comunicação entre os neurônios.

Existem três características principais que descrevem uma rede neural e que contribuem para sua habilidade funcional [8]:

- Estrutura (arquitetura);
- Dinâmica;
- Aprendizado.

Há três princípios importantes que podem ser encontrados subjacentes à organização estrutural das diferentes áreas do cérebro:

- Camadas de elementos de processamento;
- Colunas de elementos de processamento; e
- Especialização do tecido neural em sistemas específicos e não-específicos.

As arquiteturas neurais estão tipicamente organizadas em camadas. As unidades pertencentes a uma camada estão conectadas às unidades da camada seguinte. Normalmente, as camadas são classificadas em três grupos:

- **Camada de Entrada**: onde os padrões são apresentados à rede;
- **Camadas Intermediárias, Escondidas ou Ocultas:** onde é feita a maior parte do processamento, por intermédio das conexões ponderadas;
- **Camada de Saída:** onde o resultado final é concluído e apresentado.

Diversos processos dinâmicos que ocorrem no sistema neural biológico são integralmente ligados às estruturas destes sistemas:

- Representação distribuída de informação;
- Codificação temporal da informação;
- Regra de inibição; e
- Processamento direto e reverso ou *feedforward* e *feedback*, respectivamente.

No processamento direto, as informações são propagadas da camada de entrada para a camada de saída, enquanto no processamento reverso as informações propagam-se da camada de saída para a de entrada.

As estruturas das redes neurais são agrupadas em três níveis [8]:

- **Micro-estrutura:** define as características de cada neurônio na rede, tais como a função de ativação ou valores de limiar dos neurônios;
- **Meso-estrutura**: define como é organizada a rede: qual o número de camadas da rede, o número de neurônios por camada, os tipos de conexões; e
- **Macro-estrutura**: define como redes diferentes podem ser colocadas juntas para executar diferentes tarefas.

Um neurônio artificial recebe vários sinais de entrada. É, então, computada a média ponderada entre os sinais de entrada (x_i) e os pesos das conexões (w_i). O resultado é aplicado à função de transferência ($f(s)$), de onde se obtém a saída (y).

Pode-se acrescentar uma entrada fictícia de valor +1 e peso w_{0j} de nome viés. Utilizando um valor de viés, é possível deslocar o ponto de disparo da função de transferência, localizado em 0 quando o viés é nulo.

As equações que descrevem os neurônios empregados neste trabalho são dadas por:

$$s = \sum_{i=1}^{n} w_i \cdot x_i + w_0 \quad \textbf{(37)}$$

Onde:

- x_i são as entradas apresentadas ao neurônio;
- w_0 é o valor de viés;
- n é a quantidade de entradas da rede;
- w_i são os pesos das conexões do neurônio às entradas x_i; e
- s é a soma das entradas ponderadas pelos pesos das conexões.

$$y = \gamma \cdot f(\alpha \cdot s) + \delta \quad \textbf{(38)}$$

O parâmetro α multiplicando s é o fator de ganho e os valores γ e δ servem para efetuar uma transformação linear na função de transferência. O resultado da utilização de um valor de viés e dos parâmetros α, δ e γ é uma maior variedade de valores e pontos de operações da saída, y, de um neurônio, sendo que ponto de operação, neste contexto, refere-se ao valor para o qual a função de transferência é ativada.

Para caracterizar grupos de neurônios, devem ser considerados:

- número de camadas da rede;
- número de neurônios por camada;
- tipo de conexões; e
- grau de conectividade entre os elementos de processamento.

Considerando as distinções especificadas, identificam-se cinco classes estruturalmente diferentes de redes [8]:

1. **Multicamadas –** redes *feedforward*;
2. **Camada Simples –** redes conectadas lateralmente;
3. **Bicamadas –** *feedforward* / *feedback*;
4. **Multicamadas –** redes cooperativas; e
5. **Redes Híbridas**.

Com relação à forma de associatividade, existem dois tipos de associação em redes. As redes **auto-associativas** correlacionam um padrão com ele mesmo; as redes **heteroassociativas** associam um padrão com outro. Cada tipo de rede tem estrutura e dinâmica diferentes.

2.3. Redes Perceptron Multicamadas

A propriedade mais importante deste tipo de redes neurais é sua habilidade de aprender de seu ambiente e com isso melhorar seu desempenho. Isto é feito por intermédio de um processo iterativo de ajustes aplicado a seus pesos, o treinamento. O aprendizado ocorre quando a rede neural atinge uma solução generalizada para uma classe de problemas.

Algoritmo de aprendizado é compreendido como um conjunto de regras bem definidas para a determinação dos pesos das conexões dos neurônios de uma rede.

As redes Perceptron Multicamadas, amplamente utilizadas como aproximadoras universais de funções, são uma extensão do Perceptron, a primeira rede neural a surgir, criada por Frank Rosenblatt [8].

O Perceptron utiliza o conceito de neurônio artificial (Figura 9), no qual há várias entradas, cada qual com um peso associado, um valor denominado viés, o qual pode ser considerado como um peso de uma entrada de valor sempre igual a 1 e uma saída, que transmite apenas os valores 0 ou 1. O Perceptron possui o seguinte funcionamento matemático: em primeiro lugar, ele realiza uma soma dos valores nas entradas ponderados pelos pesos de suas respectivas conexões. Em seguida, o valor resultante desta soma é utilizado como valor de entrada da chamada função de ativação, que no caso do Perceptron é a função degrau unitário. Se o valor apresentado à função de ativação for maior que 0, a saída *y* será 1, senão, a saída *y* será 0. Este é um comportamento bastante similar ao do neurônio biológico da Figura 7, que em um determinado instante está ou não disparando.

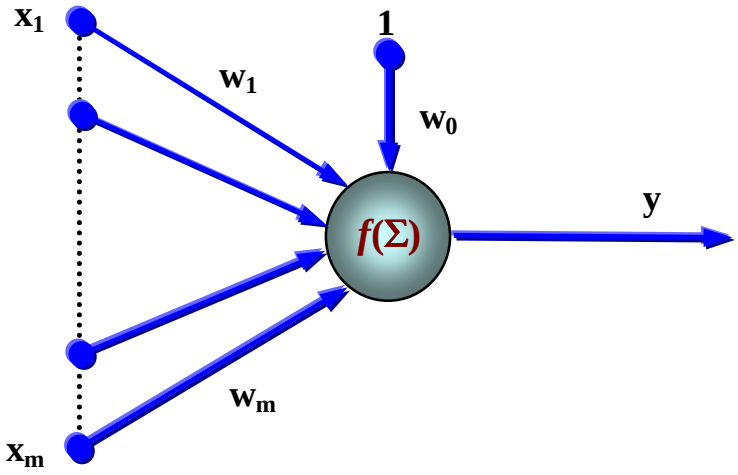


Figura 9 - Perceptron

As redes Perceptron Multicamadas, também chamadas de redes *feedforward*, são redes heteroassociativas, possuem uma ou mais camadas ocultas, uma função de transferência diferente da função degrau unitário, imposta pelo Perceptron primitivo, e dotadas de capacidades mais poderosas e genéricas.

Devido à sua estrutura, as redes Perceptron Multicamadas são capazes de aprender comportamentos que o Perceptron não pode simular.

Os neurônios posicionados na camada oculta são capazes de detectar as correlações de atividades entre diferentes nodos de entrada, o que leva a rede a aprender um conceito. Uma vez que o conceito tenha sido abstraído, as redes Perceptron Multicamadas apresentam uma grande capacidade de generalização, apresentando respostas corretas a padrões complexos que não fizeram parte de seu conjunto de treinamento. Tais características conferem às redes Perceptron Multicamadas um alto grau de robustez, protegendo o sistema de falhas devido a ruídos.

O termo retropropagação refere-se a um método de treinamento empregado para a arquitetura do Perceptron Multicamadas, pelo qual os pesos das conexões são ajustados, também conhecido sob o nome de Regra Delta Generalizada. O algoritmo de

retropropagação exige que a função de ativação dos neurônios seja diferenciável em qualquer valor de seu domínio.

O objetivo de treinar uma rede é ajustar seus pesos de tal maneira que a aplicação de um vetor ou padrão de entrada produza um desejado vetor ou padrão de saída, sendo que o termo vetor, neste contexto, significa apenas uma matriz unidimensional. O treinamento assume que cada vetor de entrada está emparelhado com um vetor de saída, vetores estes chamados de par de treinamento.

Normalmente, a rede é treinada sobre um conjunto de pares de treinamento.

O algoritmo de treinamento da rede neural é o seguinte [8], [9]:

1. Inicializa-se os pesos $w_{ij}^{(k)}$ da rede com valores aleatórios próximos a zero e escolhe-se uma taxa de aprendizagem μ, que controla o grau segundo o qual o gradiente afeta os pesos;

2. Estabelece-se um critério de parada, seja ele um número máximo de iterações, seja um valor de erro admissível. Então, supondo $\left(\vec{x},\vec{d}\right)$ como um par de treinamento, aplica-se o vetor $\vec{x}$ à camada de entrada da rede neural e propagam-se os sinais da rede até a camada de saída, onde o sinal resultante, $\vec{y}$, é comparado ao sinal desejado $\vec{d}$. Calcula-se, então, o erro quadrático ε através da Equação (39). Caso o valor de ε seja menor que um valor de tolerância ε_m, então a rede está treinada. Senão, prossegue-se ao item 3;

$$\varepsilon^2 = \sum_{j=1}^{m}\left(d_j - y_j\right)^2 \qquad \textbf{(39)}$$

onde m é a quantidade de saídas da rede neural.

3. Adota-se *k = última camada*.

4. Para todo o elemento *j* da camada *k*, faz-se:

- Cálculo de $\varepsilon_j^{(k)}$ empregando:

$$\varepsilon_j^{(k)} = \begin{cases} d_j - y_j, & \text{se } k \text{ for última camada} \\ \sum_{i=1}^{N_{k+1}} \left(\delta_{\mathrm{i}}^{(k+1)} w_{ji}^{(k+1)}\right), & \text{se } k \text{ for camada oculta} \end{cases} \quad \textbf{(40)}$$

Onde $\delta_i^{(k+1)}$ é o erro derivativo quadrático associado a cada i-ésimo neurônio da camada *k+1*, e $w_{ji}^{(k+1)}$ é o peso da conexão entre o i-ésimo neurônio da camada *k+1* e o neurônio *j* da camada *k*.

- Cálculo de $\delta_j^{(k)}$ empregando $\delta_j^{(k)} = \varepsilon_j^{(k)} f'\left(s_j^{(k)}\right)$, onde $f'\left(s_j^{(k)}\right)$ é a derivada da função de ativação dos neurônios da camada *k*;

5. O valor de *k* é decrementado de uma unidade e, se *k > 0*, retorna-se ao item 4, caso contrário, passa-se ao item 6;

6. Todos os pesos de conexão da rede são recalculados utilizando a Equação (41).

$$\vec{w}_j^{(k)}(n+1) = \vec{w}_j^{(k)}(n) + 2\mu\delta_j^{(k)}(n)\vec{x}_j^{(k)}(n) \quad \textbf{(41)}$$

Onde μ é a taxa de aprendizagem da rede; *n* indica a iteração corrente e $\vec{x}_j^{(k)}$ é o vetor com todas as entradas associadas ao elemento *j* da camada *k*.

- Por fim, toma-se outro par de treinamento e retorna-se ao item 2.

Durante o treinamento das redes Perceptron Multicamadas, é importante tomar cuidado para evitar o supertreinamento das mesmas, visto que isto pode comprometer a capacidade de generalização das redes [7]. Supertreinamento é um fenômeno que indica que a rede neural adaptou-se, com um erro muito pequeno, a um conjunto de treinamento ruidoso ou que não reflete, satisfatoriamente, o comportamento que se deseja da rede neural. Nestes casos, a rede perde sua capacidade de generalização. Dentre as formas de contornar este problema estão o treinamento da rede com um erro

que não seja demasiadamente pequeno, e a redução da quantidade de neurônios na camada escondida. A determinação da melhor estratégia a ser empregada ou de quanto estipular como erro mínimo ou quantidade de neurônios, no entanto, é feita a partir da experiência do projetista da rede neural.

Embora as redes Perceptron Multicamadas sejam as mais amplamente utilizadas, ainda há alguns problemas relacionados ao comportamento dos padrões que se queiram aprendidos pelas mesmas. Em alguns casos, a rede pode gastar vários dias em treinamento e ainda não aprender o comportamento desejado.

2.4. Redes Neurais de Base Radial

As Redes Neurais de Base Radial possuem uma arquitetura similar à arquitetura do Perceptron Multicamadas, apresentada na Figura 8. O nome Rede Neural de Base Radial justifica-se pelo tipo de funções empregadas como função de ativação na camada escondida, denominadas funções de base radial, que são funções radialmente simétricas e dentre as quais figuram a função gaussiana, função Π e função sino [6], [8].

Uma função de base radial é descrita, em sua forma genérica, pela (42):

$$f(x) = exp\left[\frac{-(x-M)^2}{2\sigma^2}\right] \quad \textbf{(42)}$$

onde os parâmetros *M* e σ significam a média e o desvio padrão da variável *x*.

Para um neurônio de base radial, com o vetor $\vec{x}$ como entrada, a função de ativação será dada pela (43).

$$f(x) = exp\left[-(\vec{x}-M)^t K^{-1}(\vec{x}-M)\right] \quad \textbf{(43)}$$

onde o parâmetros *M* e *K* representam o vetor de médias e a matriz de covariância do neurônio da camada oculta de uma Rede Neural de Base Radial.

Examinando a (43), percebe-se que a função *f(x)* alcança seu valor máximo quando o vetor de entradas, $\vec{x}$, é igual ao vetor de médias, significando que este último representa o centro da região de máxima resposta do neurônio e, ainda, os pesos das conexões, $\vec{w}$,do neurônio. A matriz de covariância, por sua vez, informa a rapidez com que a resposta do neurônio cai a zero à medida em que o vetor de entrada se distancia de *M*. A Figura 10 apresenta o comportamento de *f(x)* acima descrito. Para valores de entradas fora da região de raios concêntricos centrados em *M*, a resposta mostra-se desprezível ou nula. A esta região, denomina-se campo receptivo.

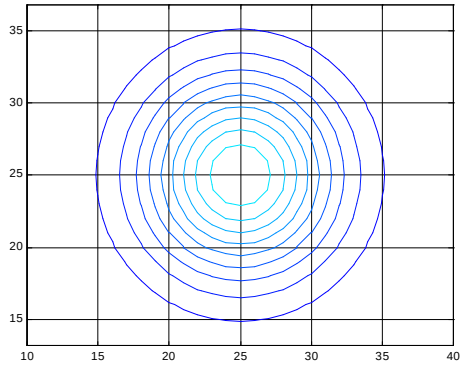


Figura 10 - Curvas de nível circulares de um neurônio de base radial

A Figura 11 ilustra melhor o campo receptivo de um neurônio de base radial. A forma e a abertura das funções de base radial responsáveis pelos campos receptivos é resultado da matriz de covariância *K*.

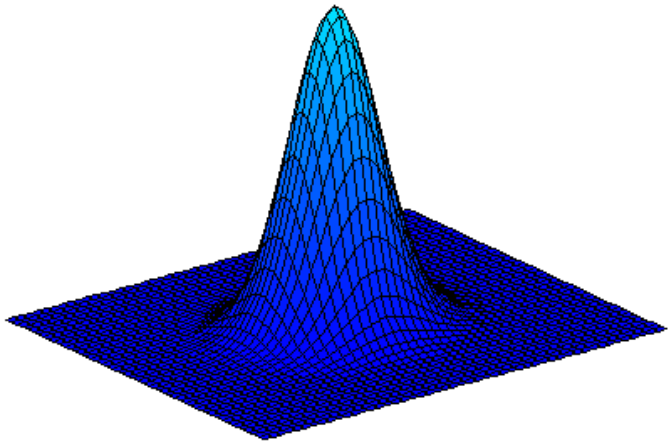


Figura 11 - Campo receptivo de um neurônio de base radial

Biologicamente, a inspiração para as Redes Neurais de Base Radial encontra-se em algumas estruturas corticais e em retinas, onde se tem células altamente sintonizadas em torno de campos receptivos claramente definidos.

É interessante notar que, ao contrário da função sigmóide empregada com freqüência por Redes Perceptron Multicamadas, as funções de base radial são não-monotônicas.

O treinamento de uma Rede Neural de Base Radial consiste, basicamente, de duas etapas [7]:

1. ajuste dos parâmetros das funções de base radial dos neurônios da camada escondida através de métodos estatísticos que envolvem técnicas de agrupamentos;
2. aplicação do algoritmo de "*backpropagation*" para ajuste da camada de saída, cujas funções de ativação são, em geral, funções lineares.

A seguir, descreve-se, de forma mais detalhada, o ajuste dos parâmetros mencionados no item 1:

- Determina-se a posição dos centros de recepção minimizando a distância média entre os exemplos de treinamento e os centros dos campos mais próximos;
- Ajuste dos valores σ_i da matriz *K*, utilizando a distância média em relação aos centros dos *m* campos receptivos mais próximos.

$$\sigma_i = \sqrt{\frac{\sum_{p=1}^{m} \left| c_i - c_{ip} \right|}{m}} \qquad \textbf{(44)}$$

onde c_{ip} é o centro do p-ésimo agrupamento próximo do agrupamento *i*.

Comparativamente às Redes Perceptron Multicamadas, as Redes Neurais de Base Radial apresentam as seguintes características:

- Um neurônio de base radial com uma entrada de dimensão *n* e uma função de transferência descrita pela Equação (43) possui um total de $n+n^2$ parâmetros a serem treinados, em contraposição a um neurônio de uma rede Perceptron em iguais condições, o qual possui um total de *n* ou *n+1* parâmetros a ajustar;
- Apesar da maior quantidade de parâmetros a serem ajustados, as Redes Neurais de Base Radial apresentam um treinamento significativamente mais rápido do que as Redes Perceptron Multicamadas;
- As Redes Neurais de Base Radial demonstram um maior grau de generalização;
- Não possuem problemas de mínimos locais, tão comuns em redes Perceptron Multicamadas.

Vale lembrar que, ao trabalhar com Redes Neurais de Base Radial, tem-se uma certa dificuldade em se determinar a quantidade de neurônios na camada escondida. Pode-se adotar este valor como sendo igual ao comprimento do vetor de entrada, no entanto, isto pode levar a uma rede desnecessariamente grande e, o que é ainda pior, pode impedir a correta aproximação dos mapeamentos que se quer aprendidos.

Capítulo 3

Procedimentos Experimentais

3.1. Base de Dados

Para a execução da fase experimental do projeto, foi criada uma base de dados composta de dez imagens aleatórias, tendo sido utilizadas duas imagens de fotografias e oito imagens de modelos computacionais desenvolvidos em Autocad R.14, em um PC Pentium II 233 MHz, todas gravadas em padrão JPEG (*Joint Photographic Experts Group*). Estas imagens estão dispostas na Figura 12.

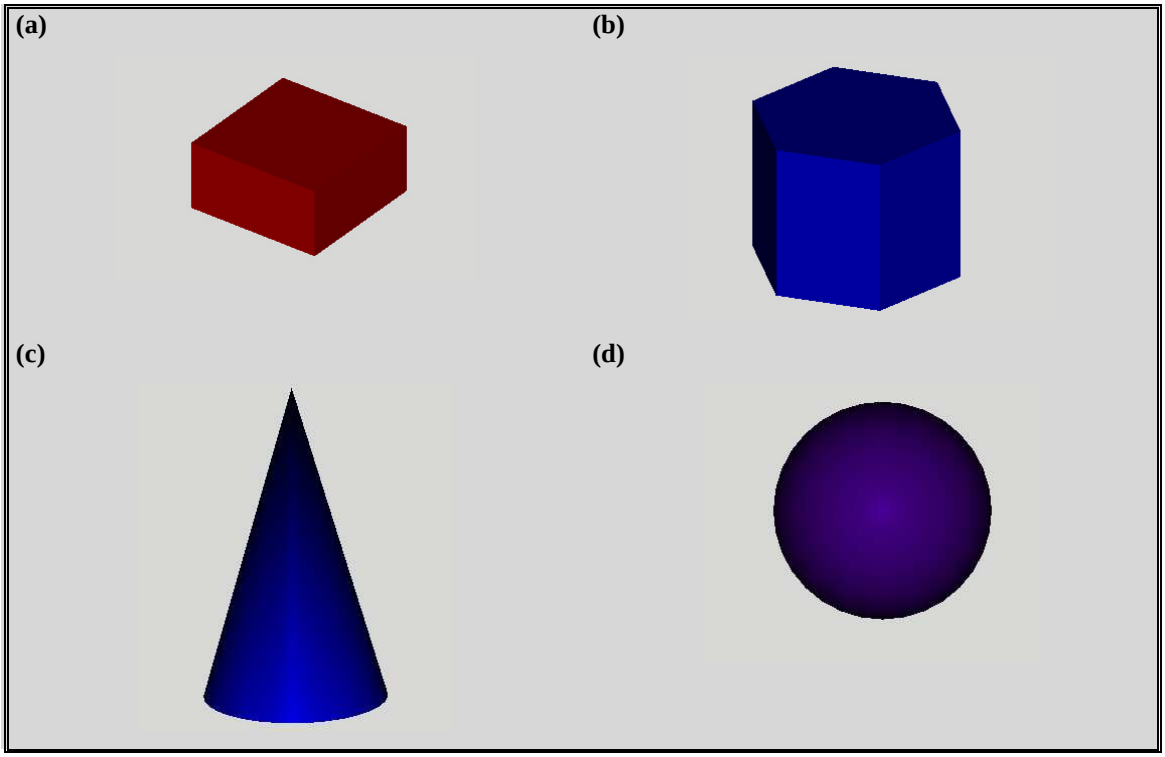


Figura 12 (a) a (d) – Base de dados utilizada para codificação por cadeias direcionais

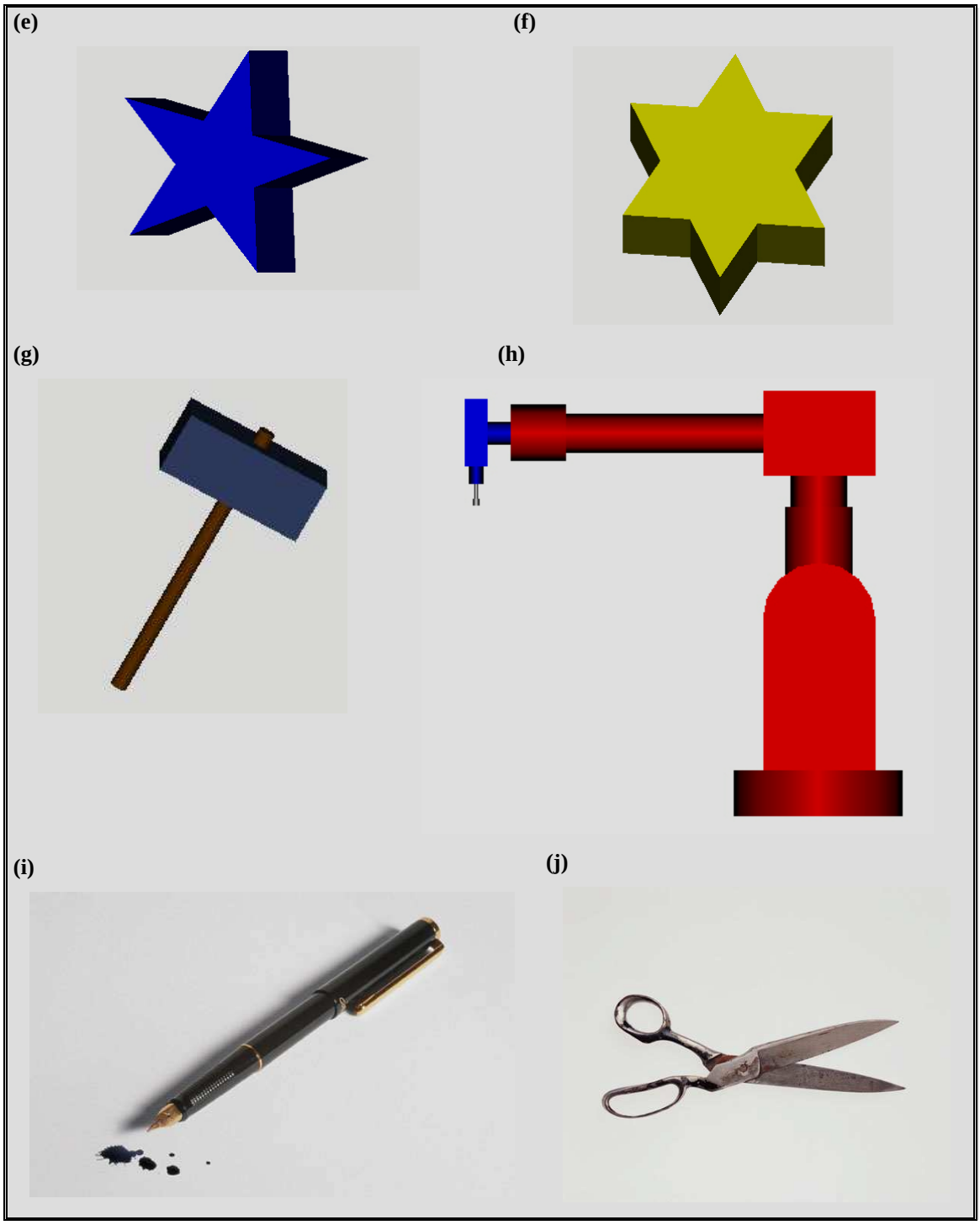


Figura 12 (e) a (j) – Base de dados utilizada para codificação por cadeias direcionais

3.2. Codificação por Cadeias Direcionais

Foi elaborado um programa, em ambiente Matlab 5.0, em um PC Pentium II, 233 MHz, o qual gera, a partir de uma dada imagem, como uma das imagens da Figura 12, por exemplo, uma imagem linear por partes como as descritas na seção 2.1. O interesse, em cada imagem da base de dados, está nas bordas do objeto apresentado em primeiro plano. A partir de tais bordas, foram obtidas imagens lineares por partes semelhantes à imagem apresentada na Figura 2. Em seguida, as imagens lineares por partes foram codificadas em cadeias direcionais.

O processo adotado para a codificação das imagens consiste de duas etapas: uma referente à detecção das bordas e linearização por partes da imagem digitalizada em estudo e a outra que codifica a imagem linear por partes em cadeias direcionais.

Uma imagem digitalizada apresenta-se como uma matriz, cuja dimensão é igual à resolução, em *pixels*, da imagem digitalizada, sendo que a cada *pixel* corresponde um dado valor. Para imagens monocromáticas, este valor será diretamente proporcional à intensidade luminosa representada pelo *pixel*.

De modo a se obter maior facilidade de implementação, este trabalho foi desenvolvido com base em imagens monocromáticas com 256 níveis da chamada "escala de cinza", a qual representa uma discretização da intensidade luminosa da imagem. Segundo a escala de intensidade luminosa empregada, o valor 255 significa a intensidade luminosa máxima, ou branco, enquanto o valor 0 significa o mínimo de intensidade luminosa, ou preto. Com isto, houve a necessidade de transformar as imagens coloridas apresentadas na Figura 12 em imagens monocromáticas.

Em uma imagem baseada em uma foto ou em modelos 3D, é comum que se tenha transições suaves de cores entre o fundo da imagem e o objeto que se deseja focalizar, além de leves variações nos valores dos *pixels*, normalmente imperceptíveis aos olhos humanos. Adotou-se, então, em contrapartida a um valor fixo de intensidade luminosa, um limiar a partir do qual considerava-se ter sido encontrada a borda do objeto que serviria de entrada para as Redes Neurais Artificiais. O ajuste do valor de

limiar permite a inclusão ou exclusão de sombras na região de interesse para codificação.

A etapa de linearização por partes das imagens utiliza-se da natureza geométrica dos *pixels*, os quais apresentam-se como polígonos de uma malha uniforme de retângulos no espaço.

Considere-se uma imagem em que se tem um fundo claro e um contorno escuro qualquer. Neste caso, tomando-se o centro geométrico de um *pixel* escuro como referência, as direções que podem ser seguidas para encontrar o centro geométrico do próximo *pixel* escuro coincidem com as direções do alfabeto A, definido na seção 2.1.

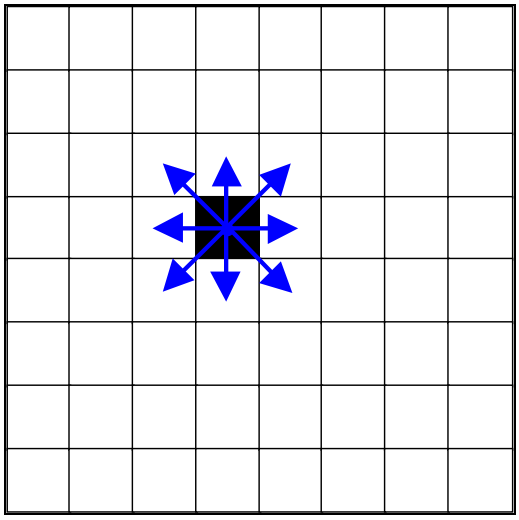


Figura 13 – Possibilidades de variação do pixel *i+1*

A Figura 14 apresenta, apenas ilustrativamente, uma imagem digitalizada, onde cada retângulo verde representa um *pixel*.

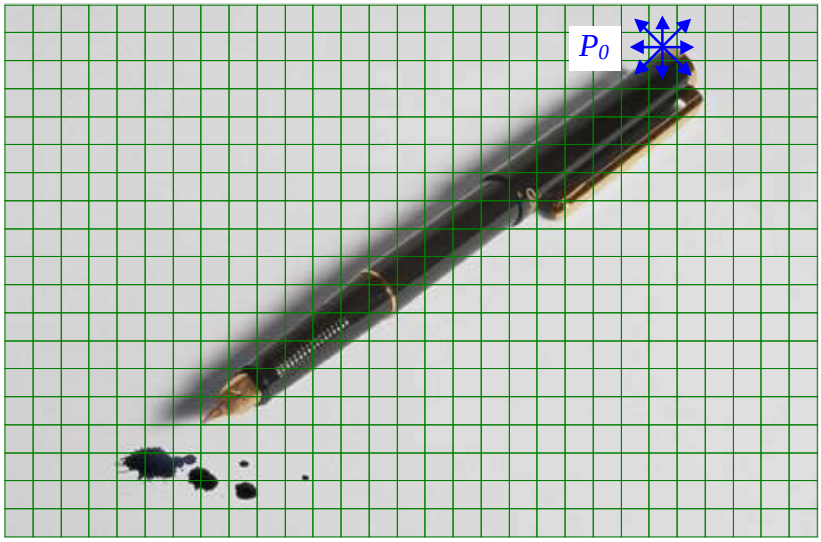


Figura 14 – Representação de uma imagem digitalizada

Observando a Figura 14, nota-se que alguns *pixels* contém informações sobre as bordas de um determinado objeto. Nestes casos, o valor armazenado pelos *pixels* das bordas será a média dos valores dos *pixels* internos ao objeto e dos valores dos *pixels* que compõem o fundo da imagem ponderada pelo grau de representação de cada um destes elementos nos *pixels* em questão.

O algoritmo de detecção de bordas foi implementado em duas etapas:

1. Varredura da imagem em busca do *pixel* P_0 que serviria de ponto de início da codificação:

Uma imagem I de dimensão $n \times m$ sendo n o número de linhas e m o de colunas, tinha todos os *pixels* p_{ij}, ($j = 1,...,m$) da linha i testados com relação ao valor de limiar antes de proceder à varredura da linha seguinte.

2. Determinação das bordas do objeto desejado:

Dado um *pixel* p_{ij}, procurava-se o *pixel* vizinho em sentido horário e segundo um critério de otimização de busca, descrito mais adiante. Caso não fosse encontrado um *pixel*, segundo as especificações de limiar, vizinho ao *pixel* p_{ij}, proceder-se-ia à determinação de um novo ponto de partida.

O critério de otimização de busca empregado neste trabalho baseia-se no conhecimento prévio do *pixel* $P_{k.}$ e nas direções dos caracteres do alfabeto A = {0,1,2,3,4,5,6,7}, definido na seção 2.1, para determinação do próximo *pixel* P_{k+1} da borda do objeto.

Uma vez que o sentido de busca é horário, a seqüência das direções $d = d(a_i)$, segundo os caracteres a_i, é tal que, dada d_0, a direção de a_i, tem-se $d_K = d(a_{i-k})$ até que se chegue a $a_{i-M} = 0$. A partir daí, faz-se $a_{i-(M+1)} = 7$ e o processo prossegue até que d_N seja igual a d_0.

Segundo a convenção adotada acima, a determinação otimizada das bordas do objeto em *I* é realizada da seguinte maneira:

- P_1, o *pixel* de borda vizinho de P_0, é procurado fazendo-se d_0 como sendo a direção de $a_i = 0$ e examinando-se os *pixels* vizinhos a P_0 segundo a seqüência de direções $d(a_i)$ acima estabelecida. A procura por P_1 termina assim que se encontra o primeiro vizinho de P_0 satisfazendo o critério de limiar de luminosidade adotado para a imagem *I*.
- A partir daí, utiliza-se a direção d_f^{i-1}, de P_{i-2} a P_{i-1}, como informação para determinar a direção inicial, d_0^i, de busca do *pixel* P_i. De modo a otimizar a determinação de bordas, evitar problemas que poderiam surgir quando tratando de formas côncavas ou mesmo evitar o retorno constante a *pixels* já armazenados, foi elaborada uma relação entre a direção d_f^{i-1} e a direção d_0^i apresentada na Equação (45). A procura por P_i pára assim que se encontra o primeiro *pixel* satisfazendo ao critério de limiar de luminosidade

$$\begin{pmatrix} d_f^{i-1}(7) \\ d_f^{i-1}(6) \\ d_f^{i-1}(5) \\ d_f^{i-1}(4) \\ d_f^{i-1}(3) \\ d_f^{i-1}(2) \\ d_f^{i-1}(1) \\ d_f^{i-1}(0) \end{pmatrix} \rightarrow \begin{pmatrix} d_0^{i}(2) \\ d_0^{i}(1) \\ d_0^{i}(0) \\ d_0^{i}(7) \\ d_0^{i}(6) \\ d_0^{i}(5) \\ d_0^{i}(4) \\ d_0^{i}(3) \end{pmatrix} \quad \textbf{(45)}$$

O processo de detecção de borda e linearização por partes acima descrito permite que se obtenha, instantaneamente, o código em cadeia direcional, c_i, correspondente ao objeto apresentado na imagem *I*.

3.3. Obtenção dos Coeficientes de Fourier

Uma vez de posse dos vetores em cadeias direcionais das imagens componentes da base de dados, procedeu-se à obtenção dos coeficientes normalizados de Fourier das projeções *x-y* das imagens lineares por parte *v* induzidas por *c*. Vale lembrar que os coeficientes normalizados de Fourier de cada imagem compõem a chamada representação canônica, definida na seção 2.1.

Neste ponto, as relações apresentadas da Equação (1) até a Equação (36) mostraram-se extremamente úteis, tendo sido implementadas em um programa em linguagem própria do ambiente Matlab 5.0, em um PC Pentium II 233MHz.

Como descrito na seção 2.1, a truncagem das séries de Fourier das projeções *x-y* permite que seja armazenada, em um arquivo relativamente pequeno, uma quantidade de informações suficiente para a recuperação satisfatória das imagens que se deseja classificar. Além disso, a normalização torna os coeficientes válidos para qualquer orientação e magnitude das imagens.

É interessante, no que diz respeito à truncagem das séries de Fourier, evitar a armazenagem de uma quantidade muito grande de harmônicos, visto que desta forma não haveria nenhuma vantagem na aplicação deste método. Também deve ser tomado cuidado quanto a uma sub-representação da imagem face a uma quantidade muito pequena de harmônicos.

Em atenção a estes detalhes, os programas de classificação por redes neurais artificiais, descritos na seção 3.4, efetuaram testes comparando o erro obtido na comparação da imagem original e a imagem reconstruída a partir das séries de Fourier truncadas em *N*, onde *N* variou entre 10 e 35 harmônicos. O programa de obtenção das séries de Fourier permitiu, a partir destes testes, notar que uma imagem reconstruída a partir de séries truncadas no 25º harmônico apresentavam erros acumulados desprezíveis e muito próximos aos erros apresentados para o 30º harmônico, tendo sido esta a razão de ser escolhida a truncagem neste harmônico. Vale lembrar que o programa mostrou, também, que abaixo de 10 harmônicos a reconstituição das imagens tornava-se prejudicada.

3.4. Classificação com Redes Neurais Artificiais

A truncagem das séries de Fourier das projeções *x-y* constituiu um avanço no que se refere à representação das imagens, visto as dimensões dos vetores que descrevem as imagens terem sido reduzidas de centenas de elementos, como era o caso das cadeias direcionais, a algumas dezenas de elementos. Restava, porém, determinar a quantidade otimizada de harmônicos a serem utilizados, além de implementar uma rede neural que apresentasse a maior robustez a distorções geradas sobre os coeficientes em questão.

A preocupação quanto a uma robustez das redes neurais artificiais em relação a distorções geradas sobre os coeficientes de Fourier, e não quanto a variações na orientação ou magnitude das imagens colhidas por uma câmera, deve-se ao fato de que, a aplicação do processamento matemático descrito pelas equações da seção 2.1, efetuado sobre imagens distintas entre si por uma operação de rotação ou ampliação resulta em um único conjunto de coeficientes normalizados de Fourier, a chamada representação canônica descrita na seção 2.1. Os ruídos nas imagens, por sua vez, geram pequenas diferenças nas bordas dos objetos, refletidas em variações nas cadeias direcionais e, por fim, nos coeficientes de Fourier.

A tarefa de determinação da quantidade otimizada de coeficientes de Fourier foi executada pelos dois programas de classificação por redes neurais artificiais implementados neste trabalho.

Estes programas são bastante semelhantes entre si, diferindo apenas no tipo de arquitetura empregada: Perceptron Multicamadas ou Rede Neural de Base Radial.

O processo de projeto das redes neurais artificiais ocorreu segundo o descrito a seguir. Primeiramente, os módulos dos coeficientes de Fourier das projeções *x-y*, truncados no harmônico *N*, com *N* variando de 10 a 25 foram organizados em vetores de entrada e de saída desejada para as redes neurais, visto tratar-se de um problema de auto-associatividade [8].A apresentação dos módulos dos coeficientes de Fourier confere a robustez final quanto à orientação da figura, resolvendo o problema das duas possibilidades de orientação de representações canônica de cada imagem. Uma vez que

se tem 4 coeficientes para cada harmônico *N*, os vetores de entrada das redes neurais artificiais tinham dimensão 4*N*. Devido ao fato de terem sido empregadas 10 imagens como Base de Dados, a saída das redes neurais artificiais eram vetores de dimensão 10, sendo que a cada saída era associada uma imagem. A Figura 15 apresenta a topologia das redes neurais empregadas neste trabalho. A imagem cujo neurônio correspondente apresentava maior valor de saída era dita a imagem reconhecida.

Definiu-se um ruído, r, aplicado ao sistema, baseado em um valor de ruído máximo R_{max}, segundo a Equação (46).

$$r = R_{max}(1 - 2rnd(\ )) \quad \textbf{(46)}$$

onde a função rnd() gera valores aleatórios compreendidos entre 0 e 1.

As redes foram treinadas, então, sem ruído algum, a fim de se obter uma primeira aproximação dos pesos das conexões. Em seguida, o grau de dificuldade do mapeamento foi sendo aumentado gradativamente, treinando-se as redes com valores de ruídos máximos, R_{max}, de 5%, 10%,... até 75% dos valores dos coeficientes de Fourier. Os resultados destes treinamentos foram armazenados e os erros de classificação das redes neurais para diferentes valores de *N* foram comparados.

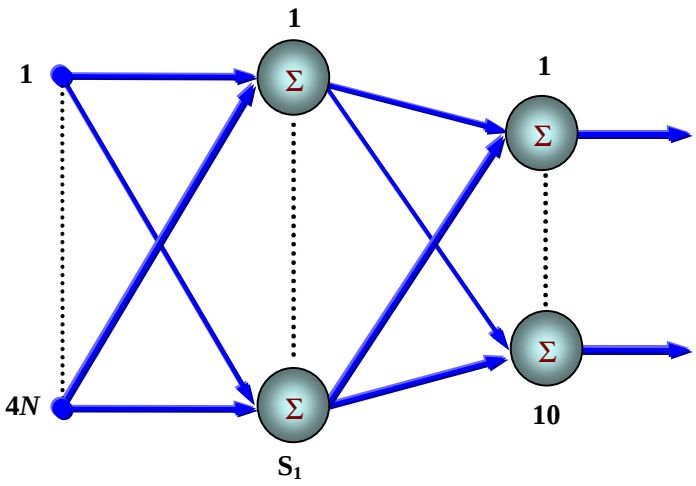


Figura 15 – Topologia das Redes Neurais empregadas neste Trabalho

O valor de S_1 foi estabelecido como 20 neurônios para as Redes Perceptron Multicamadas, tendo sido escolhido este valor devido ao fato de este ser um valor razoável em diversas aplicações. Para as Redes Neurais de Base Radial, permitiu-se que o valor de S_1 variasse até um máximo de 25 neurônios, refletindo a quantidade de épocas de treinamento da rede.

Uma vez treinada cada topologia, apresentaram-se, para fins de avaliação e individualmente, em seqüência aleatória, todas as imagens da Base de Dados como entrada das redes neurais artificiais. O processo foi repetido 500 vezes para cada valor de nível máximo de ruído, R_{max}, o qual variou de 0% a 100%, em passos de 5%, do valor dos coeficientes normalizados de Fourier, tendo sido medidos os tempos totais do processo para as Redes Neurais de Base Radial e para as Redes Perceptron Multicamadas.

Por fim, de posse dos resultados das comparações dos erros de classificação para várias quantidades de harmônicos e das duas arquiteturas diferentes, bastou escolher a rede de melhor desempenho como a rede projetada para o problema de classificação de imagens codificadas por cadeias direcionais.

A Figura 16 apresenta um sumário dos procedimentos adotados.

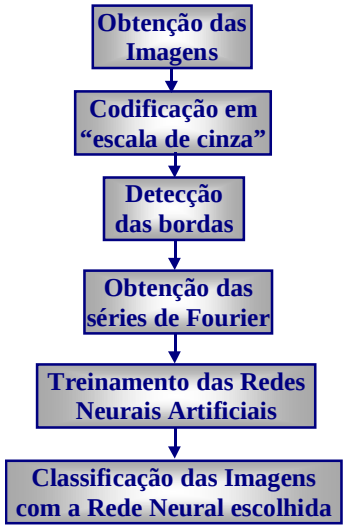


Figura 16 – Procedimentos adotados

Capítulo 4

Resultados

4.1. Resultados das Codificações em Cadeias Direcionais

Primeiramente, como explicado na seção 3.2, foi necessário obter os equivalentes das imagens da Figura 12 em formatação escala de cinza. Os resultados desta etapa são apresentados na Figura 17.

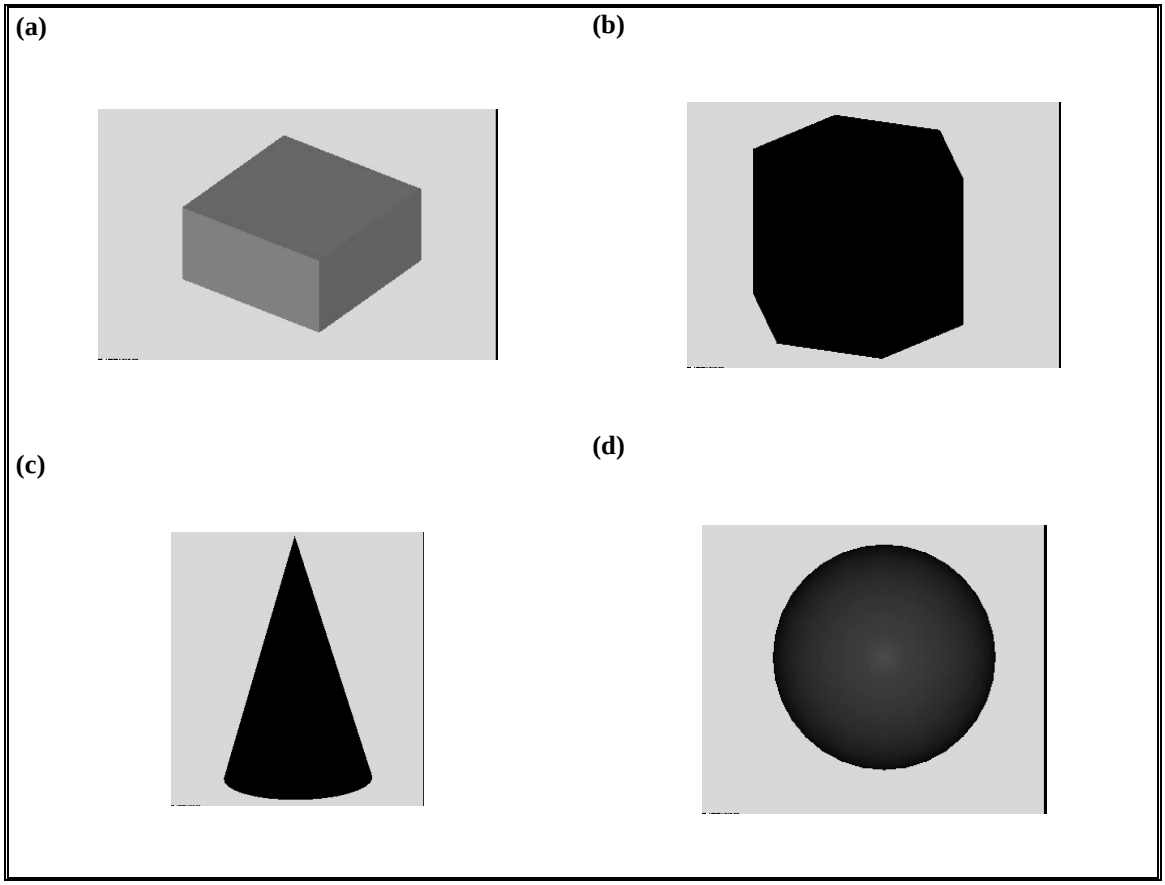


Figura 16 (a) a (d) – Base de Dados representada em escala de cinza

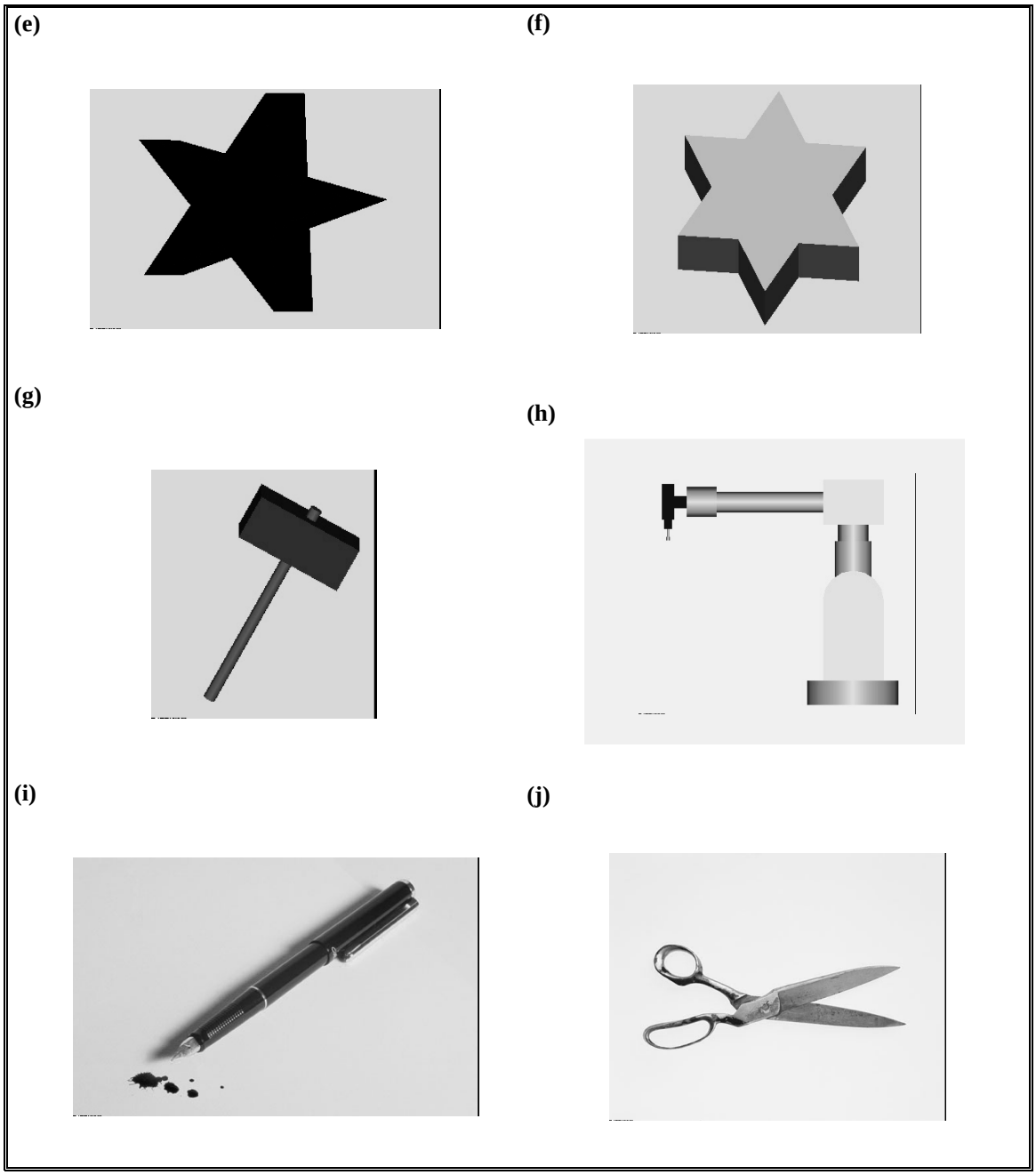


Figura 17 (e) a (j) – Base de Dados representada em escala de cinza

As bordas foram, então, detectadas segundo o critério de otimização de busca apresentado na seção 3.2. Vale lembrar que o processo de detecção de bordas empregado neste trabalho permite a obtenção simultânea das cadeias direcionais que descrevem as imagens lineares por partes representadas pelas bordas dos objetos. Os resultados estão ilustrados na Figura 18.

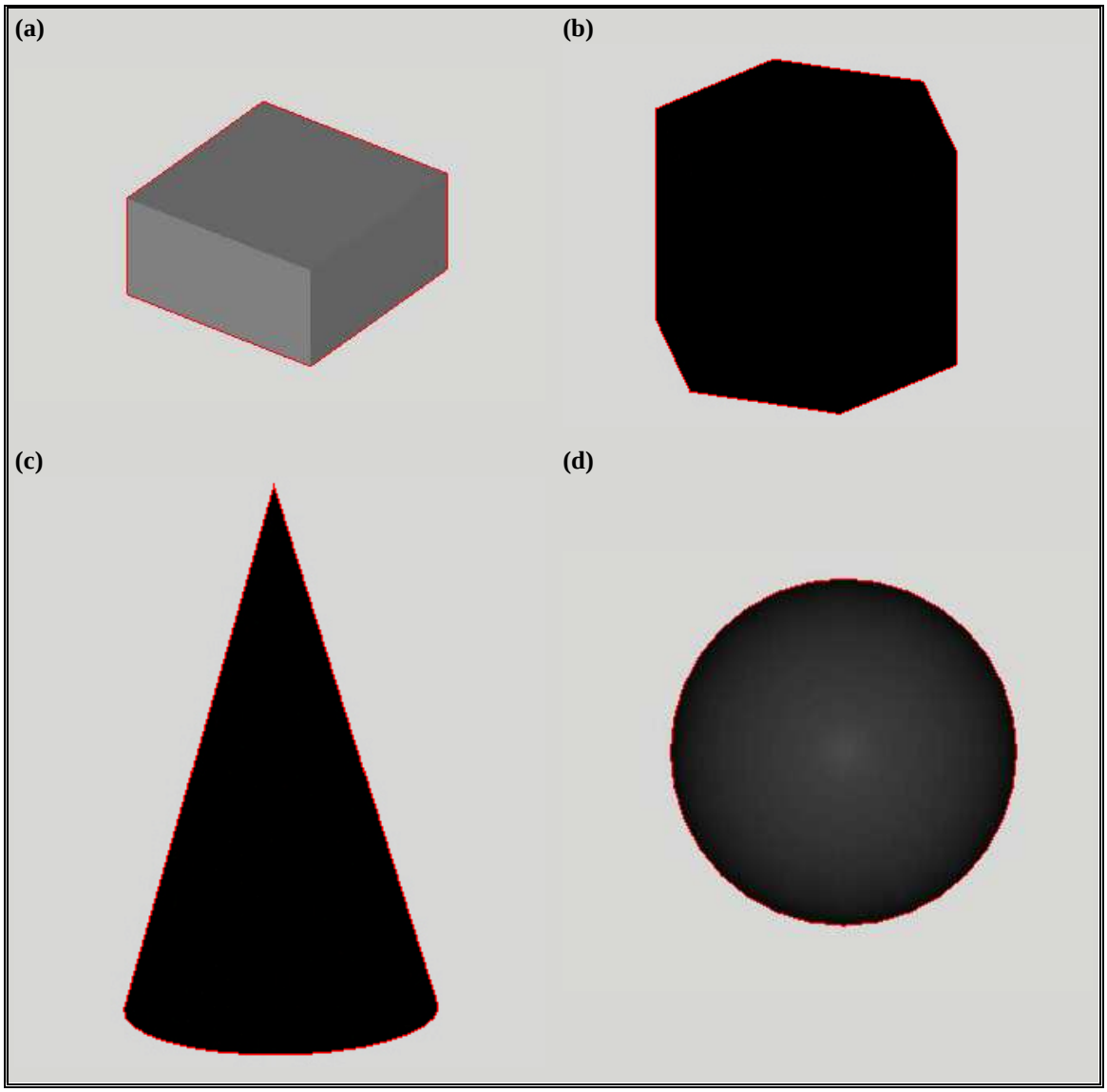


Figura 17 (a) a (d) – Imagens da Base de Dados e as bordas detectadas

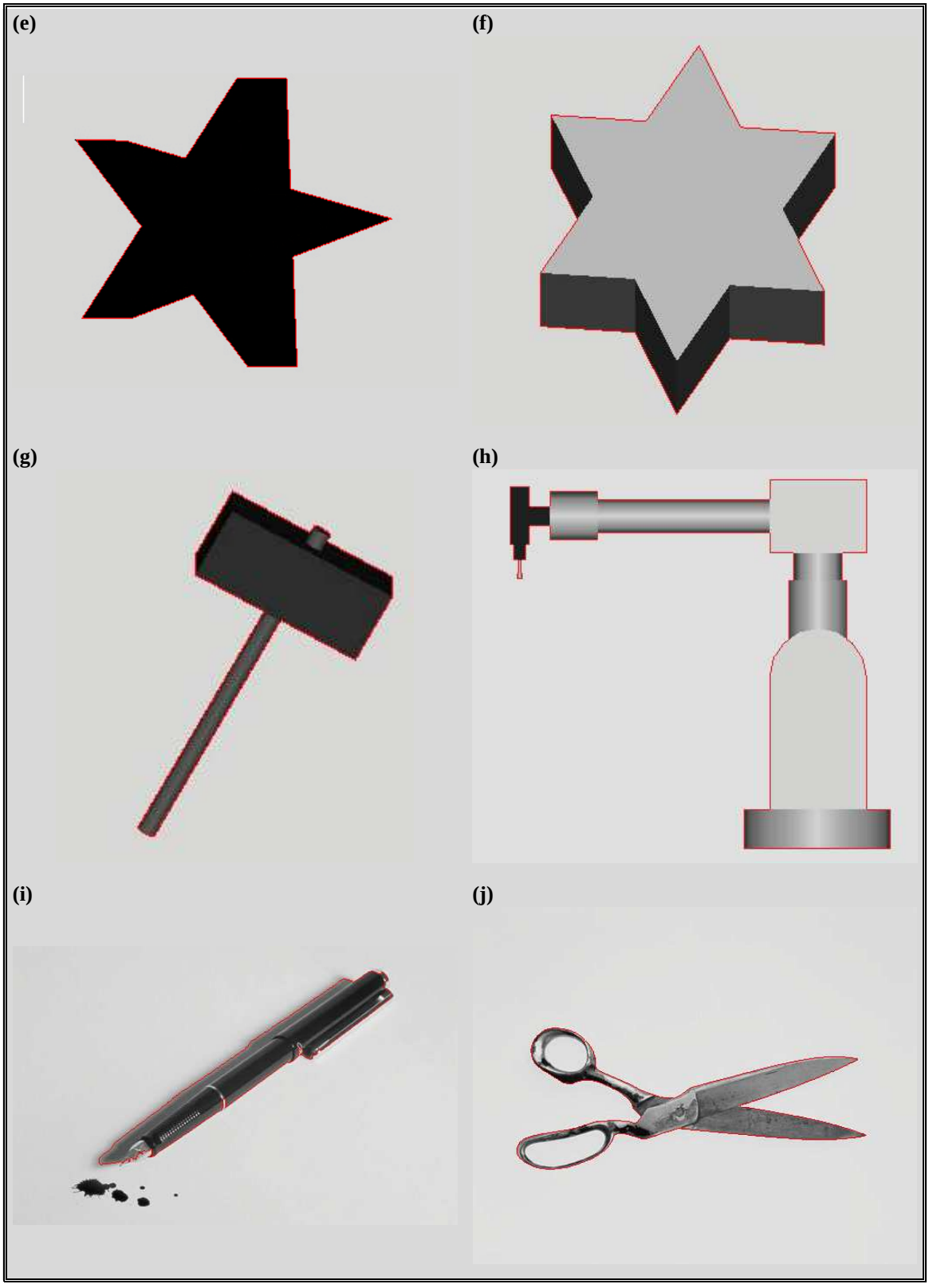


Figura 18 (e) a (j) – Imagens da Base de Dados e as bordas detectadas

Nota-se que as bordas foram detectadas a contento, exceto as bordas da caneta, as quais incluíram a sombra da mesma. Alterando o valor do limiar de luminosidade durante a detecção de borda da mesma foi possível excluir a sombra da região delimitada pela cadeia direcional, no entanto, as partes mais claras da caneta também eram perdidas. Assim sendo, optou-se pela detecção segundo o ilustrado acima.

É interessante notar que, embora não seja perceptível ao olho humano, as bordas dos objetos, apresentadas na Figura 18, são imagens lineares por partes, semelhantes à imagem apresentada na Figura 2. A diferença está apenas na resolução empregada para descrever as imagens, que passou de um código com pouco mais de uma dezena de caracteres na Figura 2 para códigos com várias centenas de caracteres na Figura 17.

4.2. Resultados da Aplicação das Séries de Fourier

A partir das cadeias direcionais da Base de Dados, foram determinadas as projeções *x*-*y* de cada imagem. Obtiveram-se, então as representações canônicas das imagens da Base de Dados, sendo que os valores absolutos dos primeiros 25 harmônicos de $\tilde{X}(t)$ e $\tilde{Y}(t)$ de cada imagem foram colhidos para servir de banco de dados, de onde foram criados os pares de treinamento das redes neurais.

As representações canônicas são conjuntos de coeficientes de Fourier resultantes da aplicação do mapeamento Λ, definido na seção 2.1, sobre os coeficientes de Fourier de duas imagens distintas que representam um mesmo objeto sob diferentes orientações e magnitudes. Os resultados da aplicação deste mapeamento são ilustrados na Figura 19, onde se tem a sobreposição de 8 imagens distintas referentes a um mesmo objeto sobre as respectivas imagens reconstituídas a partir dos coeficientes normalizados de Fourier obtidos para cada imagem. Nota-se que existem apenas 2 representações canônicas possíveis, diferentes por uma rotação de 180º, conforme previsto na seção 2.1. A apresentação dos módulos dos coeficientes de Fourier como entrada das redes neurais, assim como descrito na seção 3.4, garante o reconhecimento destas duas representações possíveis como um único padrão.

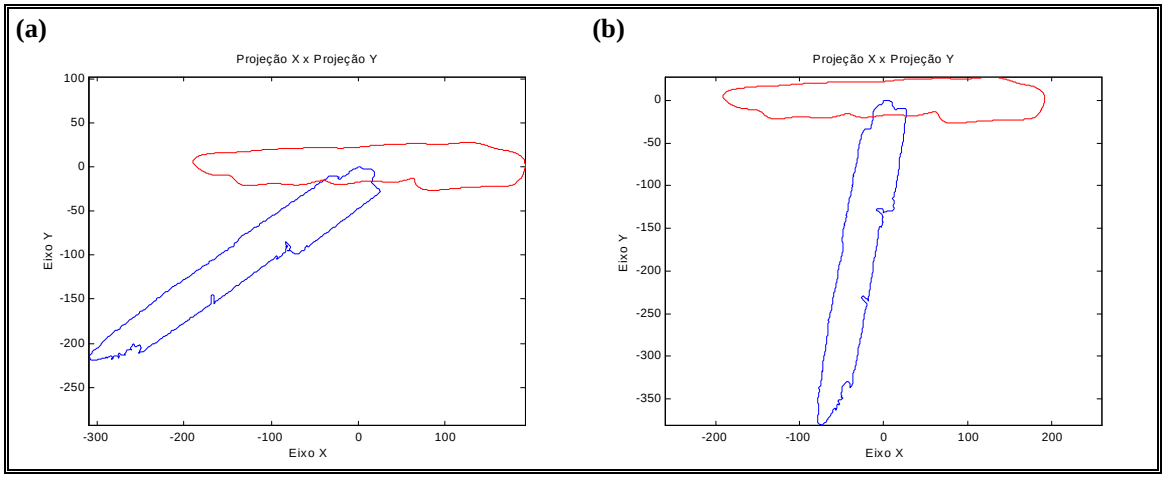


Figura 19 (a) e (b) – Comparações de várias imagens distintas referentes a um mesmo objeto em orientações diferentes com as imagens resultantes das representações canônicas

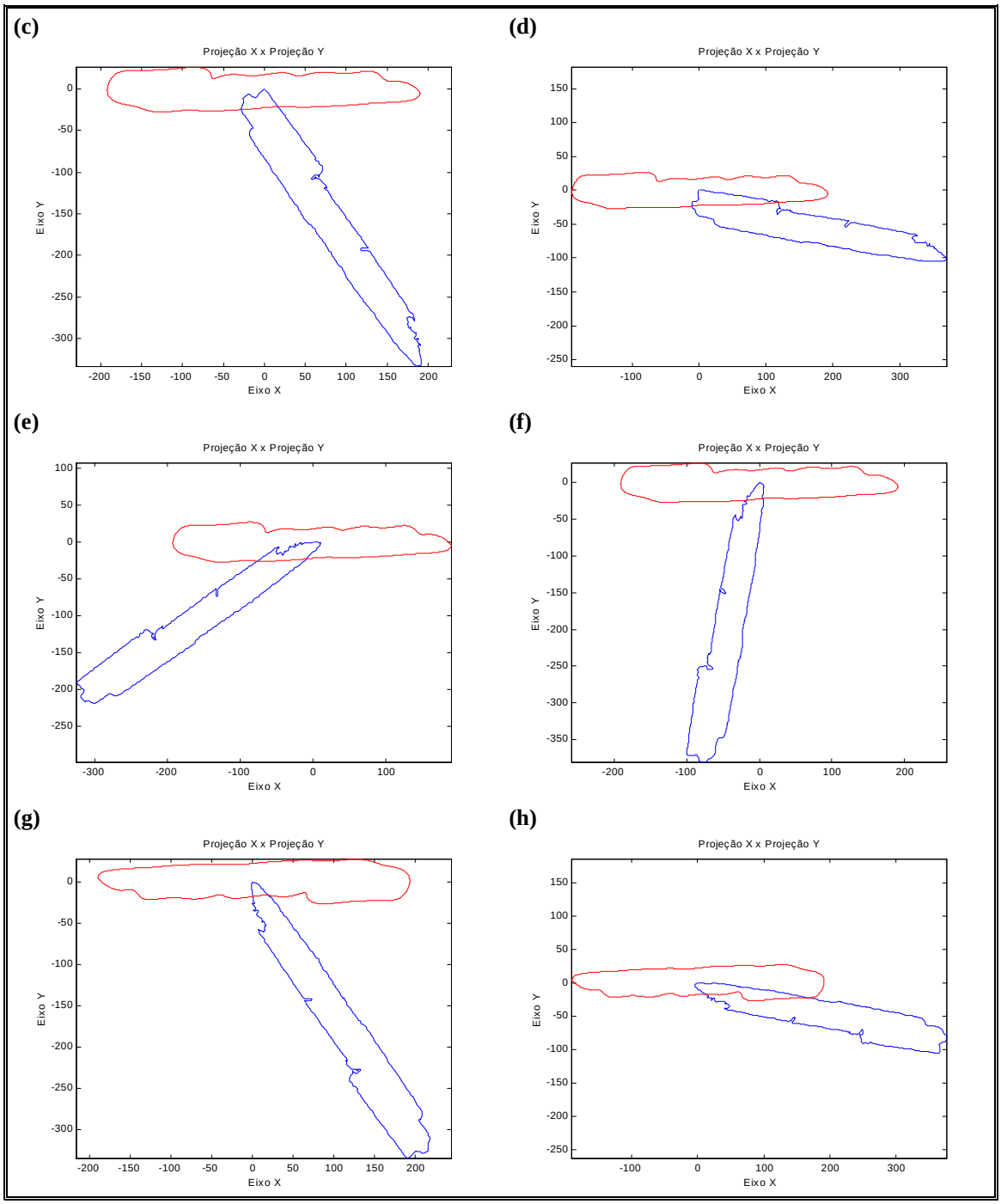


Figura 19 (c) a (h) – Comparações de várias imagens distintas referentes a um mesmo objeto em orientações diferentes com as imagens resultantes das representações canônicas

A quantidade de 25 harmônicos, como sendo uma quantidade suficiente para garantir uma boa reconstrução das imagens, foi determinada a partir de um gráfico de erros médios, apresentado na Figura 20, o qual mostra a média dos erros de reconstituição das imagens dos 10 exemplos da Base de Dados para cada harmônico *N*. Também a partir do mesmo gráfico, pode-se concluir que, para 15 harmônicos, a representação já se mostra razoável, como se pode constatar na Figura 21 e nas figuras do Apêndice A.

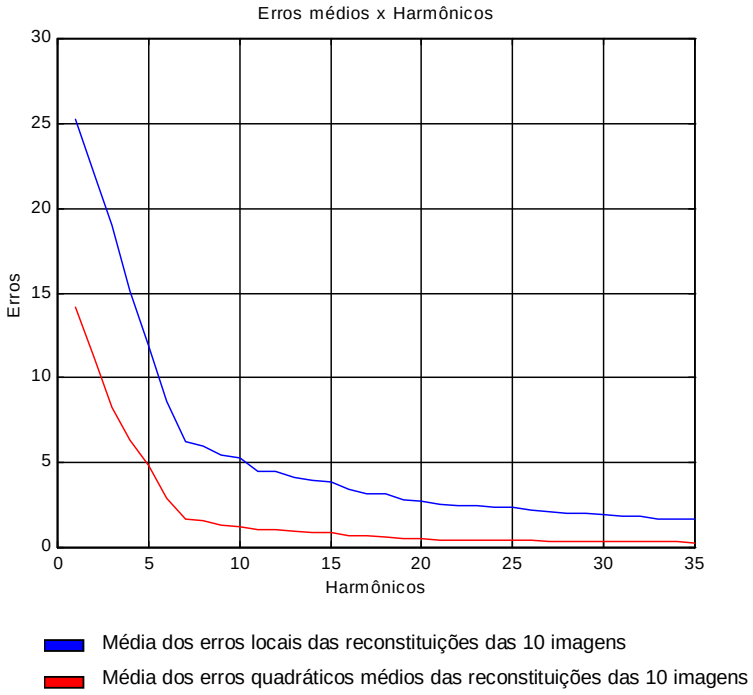


Figura 20 – Médias dos Erros × Harmônicos

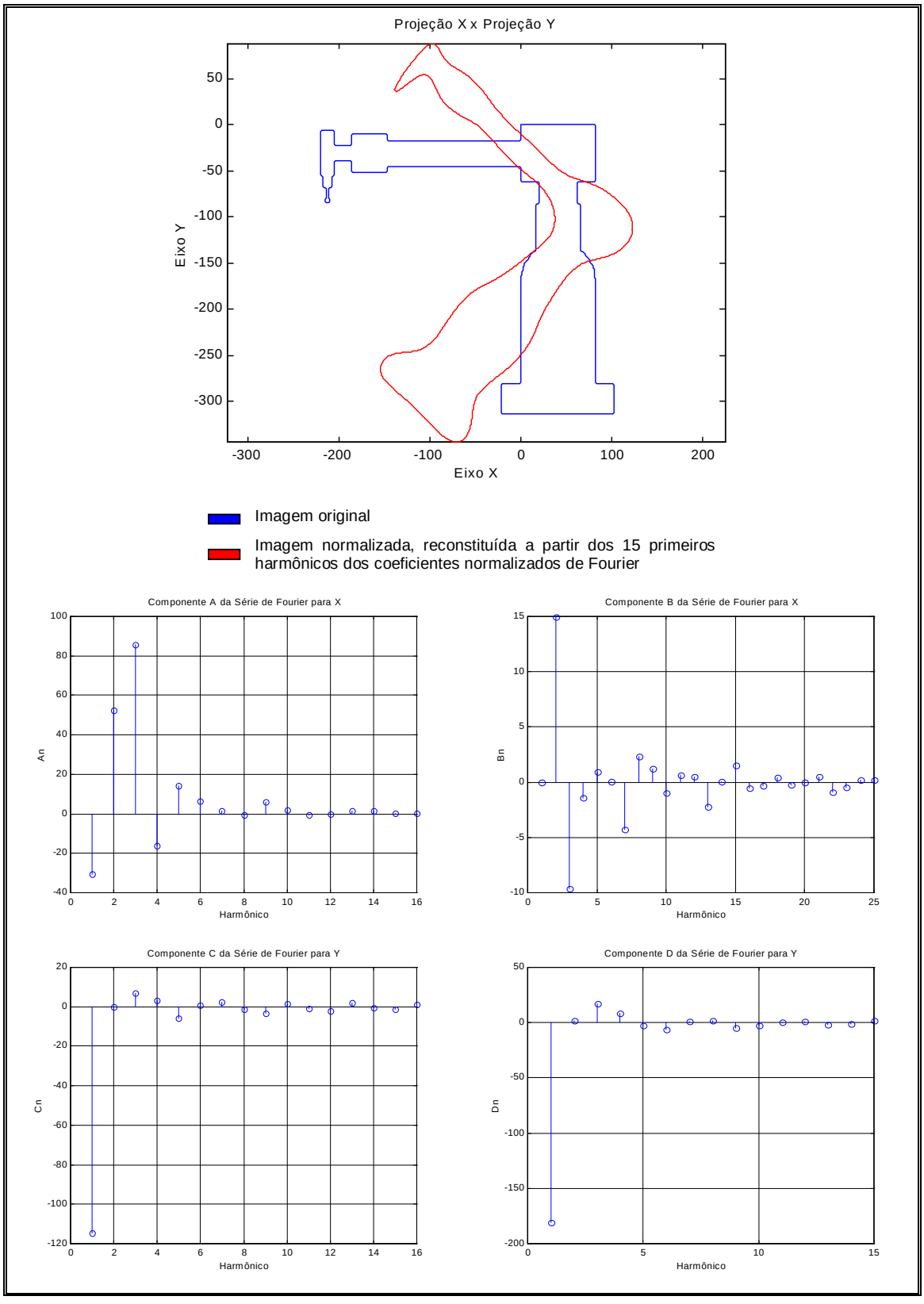


Figura 21 – Robô IRB2000 e seus coeficientes normalizados de Fourier

A Figura 21 e as figuras do Apêndice A mostram as imagens v_i, induzidas pelas cadeias direcionais c_i da Base de Dados e as imagens $\tilde{v}_i$, reconstruções, a partir dos coeficientes normalizados de Fourier, também representados nestas figuras, correspondentes a 15 harmônicos, os quais são utilizados para criar os pares de treinamento das redes neurais artificiais.

Para efeitos de ilustração, as imagens reconstruídas não apresentam a normalização em relação à magnitude, embora tal normalização tenha sido efetuada para montar os vetores de treinamento das redes neurais artificiais.

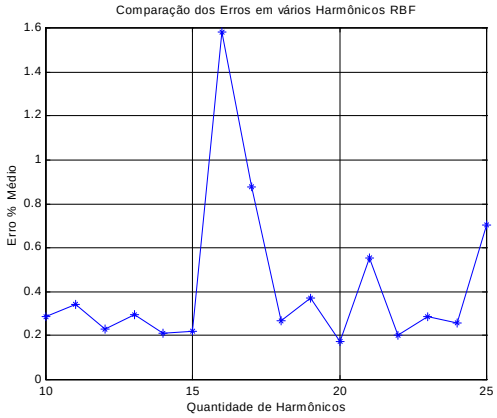
Tendo em vista uma melhor avaliação dos dados apresentados nos gráficos da Figura 21 e nas figuras do Apêndice A, foram montadas tabelas, mostradas no Apêndice B, trazendo as representações canônicas de todas as imagens da Base de Dados.

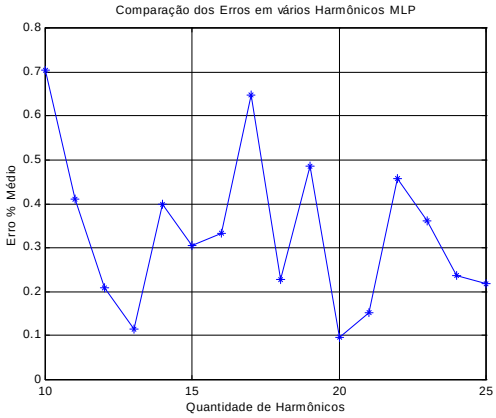
4.3. Resultados das Redes Neurais Artificiais

Cada padrão a ser aprendido pelas redes neurais artificiais era composto dos valores absolutos dos coeficientes A_n, B_n, C_n e D_n de N harmônicos, sendo que N variou de 10 a 25 harmônicos. Isto significa que foram criadas, para cada arquitetura, 16 topologias de redes distintas entre si pela dimensão dos vetores de entrada. Cada uma dessas redes foi treinada 16 vezes, uma vez para cada nível máximo de ruído, R_{max}, o qual variou de 0% a 75% em passos de 5%.

Após o treinamento, cada rede neural foi testada através de 500 simulações para cada nível de ruído máximo, R_{max}, o qual variou de 0 a 100% em passos de 5%. Durante as simulações, as imagens da Base de Dados foram apresentadas individualmente e em seqüência aleatória, de modo que houve variação na quantidade de apresentações de cada imagem às redes neurais, embora, na média as quantidades de apresentações de cada imagem tenham permanecido semelhantes.

As figuras 22 e 23 mostram uma comparação entre os resultados estatísticos obtidos através das Redes Neurais de Base Radial e Redes Perceptron Multicamadas. Isto é, os dados apresentados nestas figuras referem-se à média dos resultados das 500 apresentações de toda a Base de Dados para cada harmônico e cada nível de ruído máximo, R_{max}.

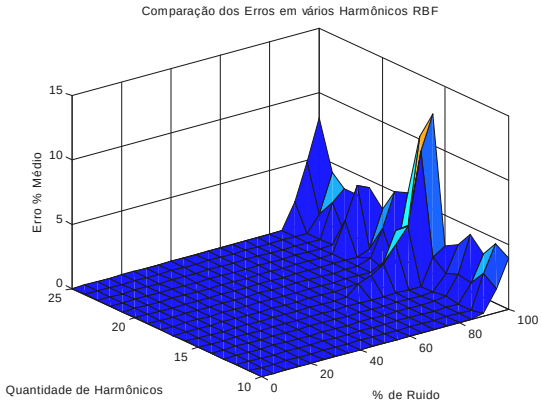


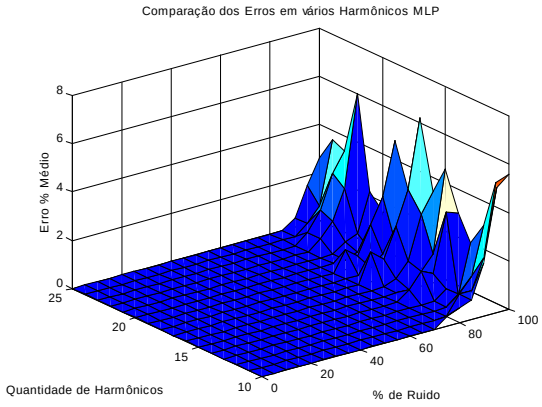


(a) Resultado apresentado pelas Redes Neurais de Base Radial

(b) Resultado apresentado pelas Redes Perceptron Multicamadas

Figura 22 – Comparações dos resultados de Percentual de Erro médio apresentado pelas Redes Neurais de Base Radial (a) e Redes Perceptron Multicamadas (b) em relação à quantidade de harmônicos





(a) Resultado apresentado pelas Redes Neurais de Base Radial

(b) Resultado apresentado pelas Redes Perceptron Multicamadas

Figura 23 – Comparações dos resultados de Percentual de Erro médio apresentado pelas Redes Neurais de Base Radial (a) e Redes Perceptron Multicamadas (b) em relação ao nível máximo de ruído e à quantidade de harmônicos

As tabelas 1 e 2 apresentam, mais detalhadamente, os mesmos resultados das figuras 22 e 23.

Tabela 1 – Comparação dos Erros Percentuais, apresentados pelas Rede Neurais de Base Radial, para vários valores de ruído, com a quantidade de harmônicos variando entre 10 e 25.

Ruído (%)	Harmônicos															
	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0.800	0	0	0	0	0	0	0	0	0
75	0	0	0	0	0	0	1.200	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	1.800	0	0	0	0	0	0	0	0	0
85	0	0	0	0	0	0	3.000	1.200	0	0	0	0	0	0	0	0
90	0.200	0	0	0.200	0	0	4.200	3.400	0	0.400	0	2.400	0	0	0	2.000
95	1.800	2.600	1.400	1.600	1.200	2.000	9.800	3.600	0.200	2.400	0.400	4.800	0.800	2.600	1.200	4.800
100	4.000	4.600	3.400	4.400	3.200	2.600	12.400	10.200	5.400	5.000	3.200	4.400	3.400	3.400	4.200	8.000

Tabela 2 – Comparação dos Erros Percentuais apresentados pelas Rede Perceptron Multicamadas para vários valores de ruído, com a quantidade de harmônicos variando entre 10 e 25.

Ruído (%)	Harmônicos															
	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75	0.400	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
80	0.600	0	0	0	0.400	0	0	1.200	0	0.800	0	0	0	0	0	0
85	0.800	0.800	0	0	1.000	0	0.200	0.800	0.200	0.400	0	0	0.400	0.400	0	0
90	2.200	0.800	0.400	0.400	0.800	0.600	0.800	2.000	0.600	1.600	0.200	0.400	1.000	1.200	0.400	0.400
95	5.200	2.000	1.000	0	3.200	1.200	2.600	3.400	1.000	2.600	0.200	0.600	2.200	2.600	1.000	1.600
100	5.600	5.000	3.000	2.000	3.000	4.600	3.400	6.200	3.000	4.800	1.600	2.200	6.000	3.400	3.600	2.600

Analisando-se os dados apresentados nas figuras 22 e 23, nota-se que os percentuais de Erro médio das Redes Neurais de Base Radial encontram-se, para a maioria dos harmônicos, dentro de uma margem compreendida entre 0,2% e 0,4%, com um erro médio global de 0,31%, semelhante aos resultados obtidos para as Redes Perceptron Multicamadas, cuja margem vai de 0,1% a 0,7% para todos os harmônicos, com um erro médio geral de 0,33%.

Nota-se ainda, pela análise das tabelas 1 e 2 que, para todos os harmônicos, as Redes Neurais de Base Radial, treinadas com 75% de ruído máximo, somente começaram a apresentar algum erro de classificação, cujo máximo chegou a 12,4% com 16 harmônicos, após a apresentação de sinais de entrada com 70% de ruído para o 16º harmônico e após a apresentação de sinais de entrada com 85% de ruído para os demais harmônicos.

As Redes Perceptron Multicamadas, treinadas da mesma forma que as Redes Neurais de Base Radial, apresentaram erros de classificação, cujo máximo foi 5,6% para 10 harmônicos, a partir de 75% de ruído para o 10º harmônico e 80% para a maioria dos harmônicos.

Tabela 3 – Comparação dos Erros Médios devidos a Ruídos

Harmônicos	Erro médio das redes de base radial (%)	Erro médio das redes Perceptrons (%)
10	0,2857	0,7048
11	0,3429	0,4095
12	0,2286	0,2095
13	0,2952	0,1143
14	0,2095	0,4000
15	0,2190	0,3048
16	1,5810	0,3333
17	0,8762	0,6476
18	0,2667	0,2286
19	0,3714	0,4857
20	0,1714	0,0952
21	0,5524	0,1524
22	0,2000	0,4571
23	0,2857	0,3619
24	0,2571	0,2381
25	0,7048	0,2190

Os erros mínimos apresentados pelas redes neurais artificiais foram 0,1714%, para o 20º harmônico na classificação com Redes Neurais de Base Radial, e 0,0952%, também para o 20º harmônico, na classificação com Redes Perceptron Multicamadas.

Vale lembrar, no entanto, que devido à maior homogeneidade das Redes Neurais de Base Radial em torno de um erro médio percentual em torno de 0,31% tornam-na mais confiável no caso da necessidade de variar a quantidade de harmônicos empregados para a classificação das imagens. Além disso, caso seja necessário alterar, para fins de aplicação prática deste trabalho, as imagens da Base de Dados, haverá necessidade de um novo treinamento, e a arquitetura de rede de treinamento mais rápido torna-se preferível. O tempo requerido para treinamento e validação das Redes Neurais de Base Radial foi de aproximadamente 55 minutos, enquanto o tempo necessário para efetuar o mesmo processo com as Redes Perceptron Multicamadas foi de aproximadamente 80 minutos, um aumento de cerca de 45% em relação ao tempo requerido pelas Redes Neurais de Base Radial.

Foi determinada, então, como sendo a rede neural projetada para o sistema de classificação de imagens codificadas por cadeias direcionais, a Rede Neural de Base Radial com 60 entradas, 24 neurônios na camada escondida e 10 neurônios na camada de saída, ilustrada na Figura 24.

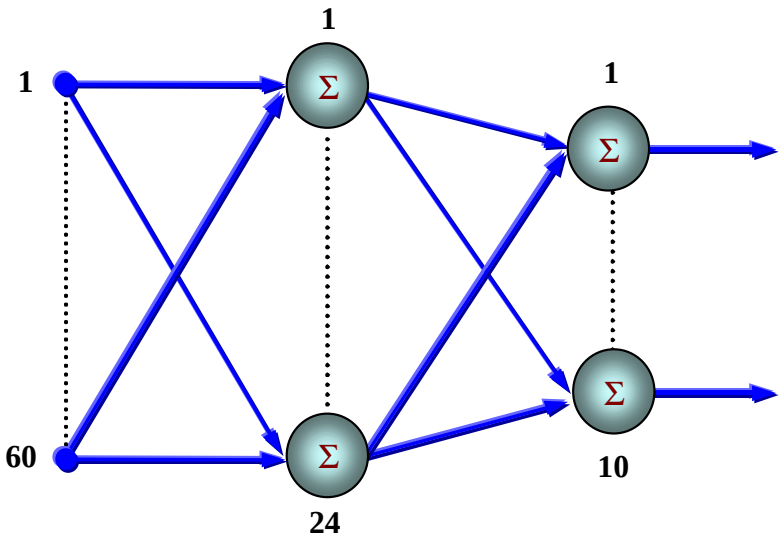


Figura 24 – Topologia da Rede Neural adotada para o Sistema de classificação de imagens codificadas por cadeias direcionais

As figuras 25 a 27 ilustram os resultados de uma classificação de imagem pelo método descrito neste trabalho. Para efetuar esta classificação, foi adotado um ruído de no máximo 15% sobre a imagem a ser classificada, de modo a simular uma possível situação real, como impurezas nas lentes da câmera ou variações de luminosidade, por exemplo. Pelas razões descritas na seção 3.4, tem-se que o ruído gerado sobre os coeficientes de Fourier devido ao ruído imposto à imagem é menor que 15%.

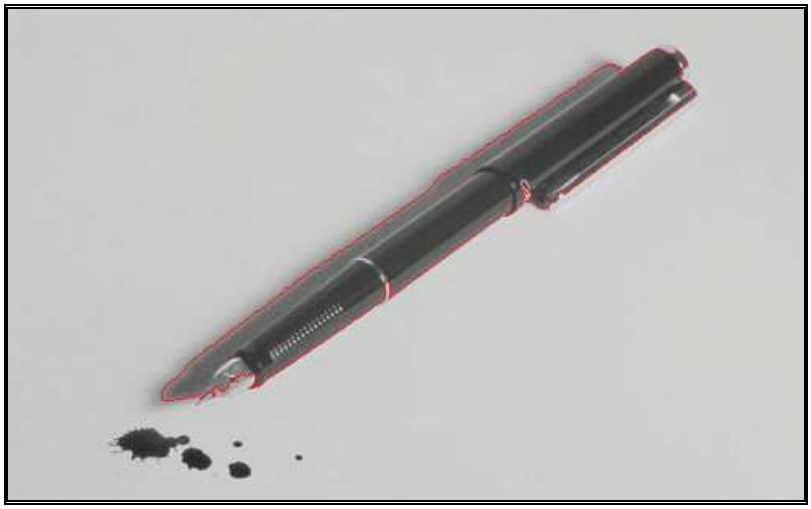


Figura 25 - Imagem com 15% de ruído e a borda detectada

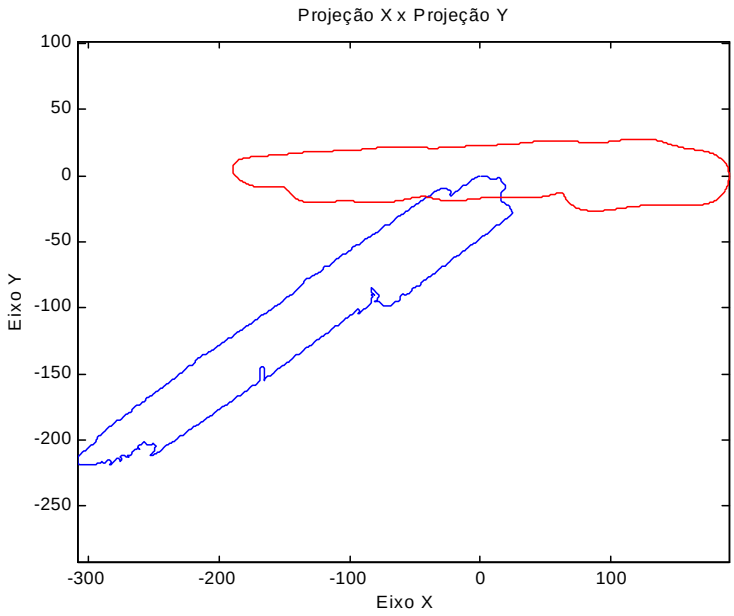


Figura 26 – Imagem codificada e sua representação canônica, ambas com ruído

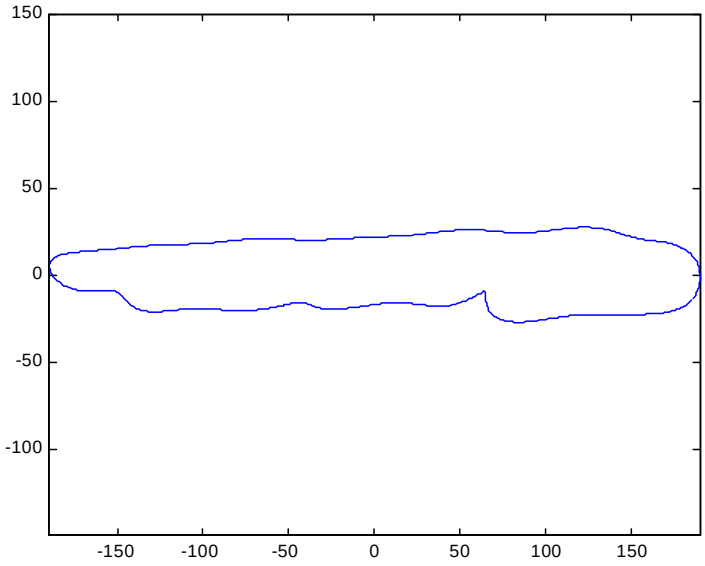


Figura 27 – Imagem correspondente à representação canônica reconhecida a partir dos coeficientes ruidosos

Capítulo 5

Conclusões e Perspectivas

Este trabalho desenvolveu um sistema de classificação de imagens robusto a variações de orientação, tamanho, iluminação e ruído, aliando o uso de redes neurais artificiais ao emprego de um método de compressão de dados baseado nas chamadas cadeias direcionais. Este método mostrou-se relativamente simples e eficiente, sendo indicado para aplicações em que haja interesse em descrever as características de um objeto pelas suas bordas.

As expansões em séries de Fourier, realizadas nas projeções espaciais das imagens induzidas pelas cadeias direcionais, exigem uma truncagem dos harmônicos superiores e geram, conseqüentemente, perda de definição nas regiões onde ocorrem alterações bruscas na forma das bordas. No entanto, estas mesmas operações propiciam independência quanto à magnitude, orientação e ponto de origem da codificação em cadeias direcionais, mostrando-se úteis mesmo em face das limitações constatadas. É importante ressaltar, ainda, que para aplicações em que não há necessidade de alta resolução nas imagens, mas sim a simples classificação de objetos segundo a sua forma, as limitações impostas pela truncagem dos componentes de Fourier exercem pouca influência no resultado final da classificação de padrões.

Em relação aos resultados das redes neurais artificiais, percebe-se que as Redes Neurais de Base Radial são ligeiramente mais adequadas, de um modo geral, para o projeto de um sistema robusto de classificação de imagens codificadas do que as Redes Perceptron Multicamadas, segundo uma série de critérios.

O primeiro fator a favor das Redes Neurais de Base Radial é a maior velocidade de treinamento. Isto significa que, caso seja necessário ensinar novos padrões a uma

rede com esta arquitetura, a resposta será mais rápida do que seria caso a rede possuísse arquitetura Perceptron Multicamadas.

O segundo ponto em que as Redes Neurais de Base Radial mostram-se vantajosas em relação às Redes Perceptron Multicamadas está nos seus erros de classificação.

Por fim, as Redes Neurais treinadas com 75% de ruído somente apresentaram algum erro para entradas com ruído próximos de 75%, significando que as mesmas aprenderam corretamente os padrões que lhes foram apresentados.

Algumas possibilidades de continuidade do trabalho ora apresentado são: a expansão do mesmo no sentido de permitir a determinação de um volume de contorno dos objetos de interesse; uma elaboração mais cuidadosa do algoritmo quanto ao tempo de processamento, de forma que se torne viável a implementação "*on-line"* do algoritmo como um todo em processos de produção; e a determinação das bordas de um objeto não levando em conta apenas a intensidade luminosa dos *pixels*, mas sim as cores dos objetos.

Referências Bibliográficas

[1] HALL, E. L.: *Computer Image Processing and Recognition*. Academic Press, Inc. New York, U.S.A., 1979.

[2] Gomes, J.; Velho, L.: *Computação Gráfica: Imagem*. Instituto de Matemática Pura e Aplicada e Sociedade Brasileira de Matemática – IMPA/SBM, Rio de Janeiro, Brasil, 1994.

[3] DOUGHERTY, R. E.; GIARDINA, R. C.: *Mathematical Methods for Artificial Intelligence and Autonomous Systems.* Prentice-Hall International Editions, Englewood Cliffs, New Jersey, U.S.A., 1988.

[4] FIGUEIREDO, D. G.: *Análise de Fourier e Equações Diferenciais Parciais.* Instituto de Matemática Pura e Aplicada, CNPq, Rio de Janeiro, Brasil, 1977.

[5] FRANKLIN, G. F.; POWELL, J. D.; WORKMAN, M.: *Digital Control of Dynamic Systems. 3rd Edition*. Addison Wesley Longman, Inc. Menlo Park, California, 1997.

[6] ALBUS, J. S.: *Brains, Behavior, and Robotics*. BYTE Publications. Massachusetts, U.S.A., 1981.

[7] KASABOV, N. K.: *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering.* Massachusetts Institute of Technology Press, U.S.A., 1996.

[8] LOESCH, C.; SARI, S.T.: *Redes Neurais Artificiais: Fundamentos e Modelos.* Editora da FURB. Blumenau, Santa Catarina, Brasil, 1996.

[9] KÓVACS, Z. L.: *Redes Neurais Artificiais: Fundamentos e Aplicações*. Editora Acadêmica, Tamboré, SP, Brasil, 1996.

[10] SERRA, J.: *Image Analysis and Mathematical Morphology*. Vol.I. Academic Press, Inc. New York, U.S.A., 1982.

[11] *Neural Network Toolbox User's Guide*. The MathWorks, January 1994.

Apêndice A: Imagens da Base de Dados

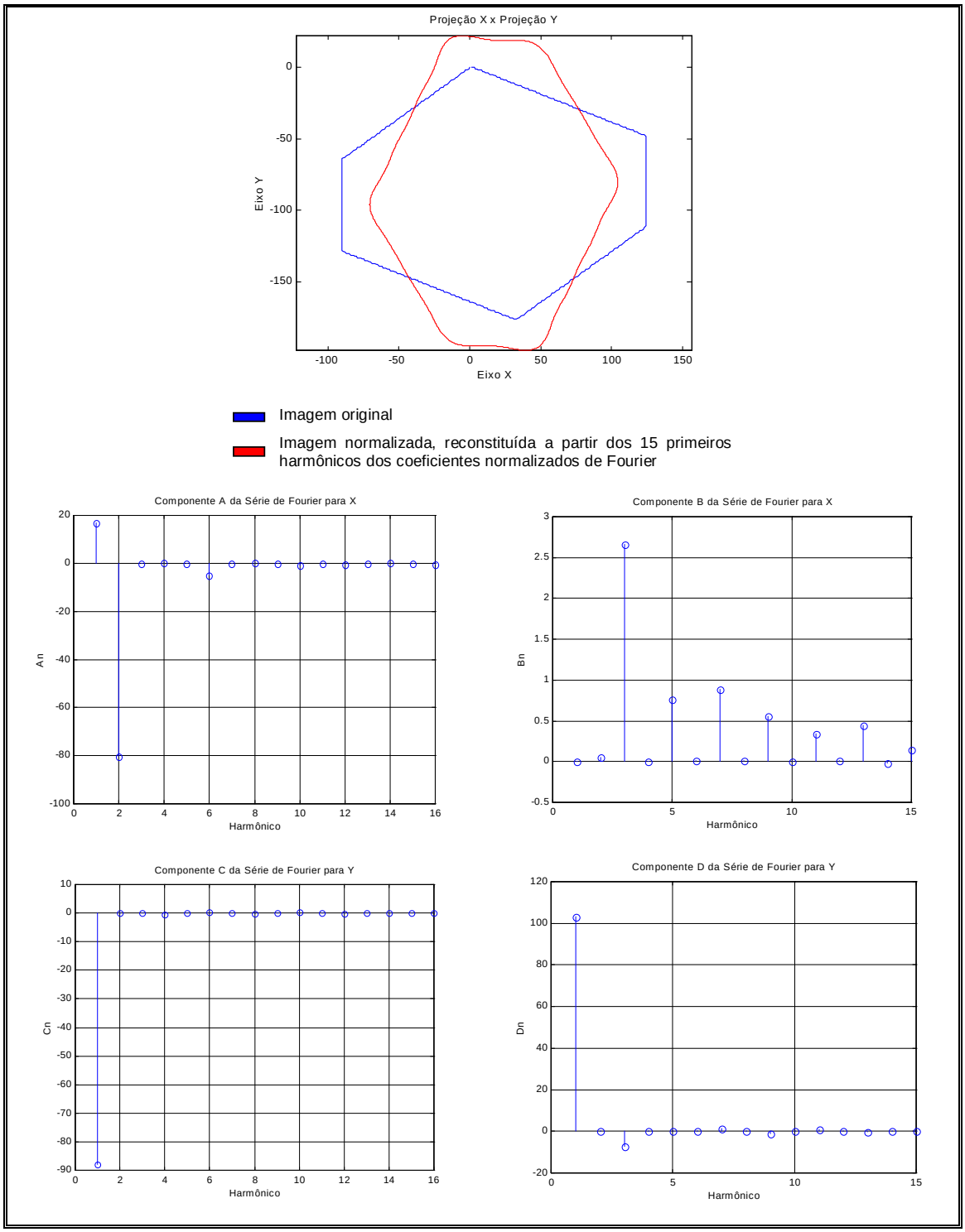


Figura 28 – Prisma de base retangular e seus coeficientes normalizados de Fourier

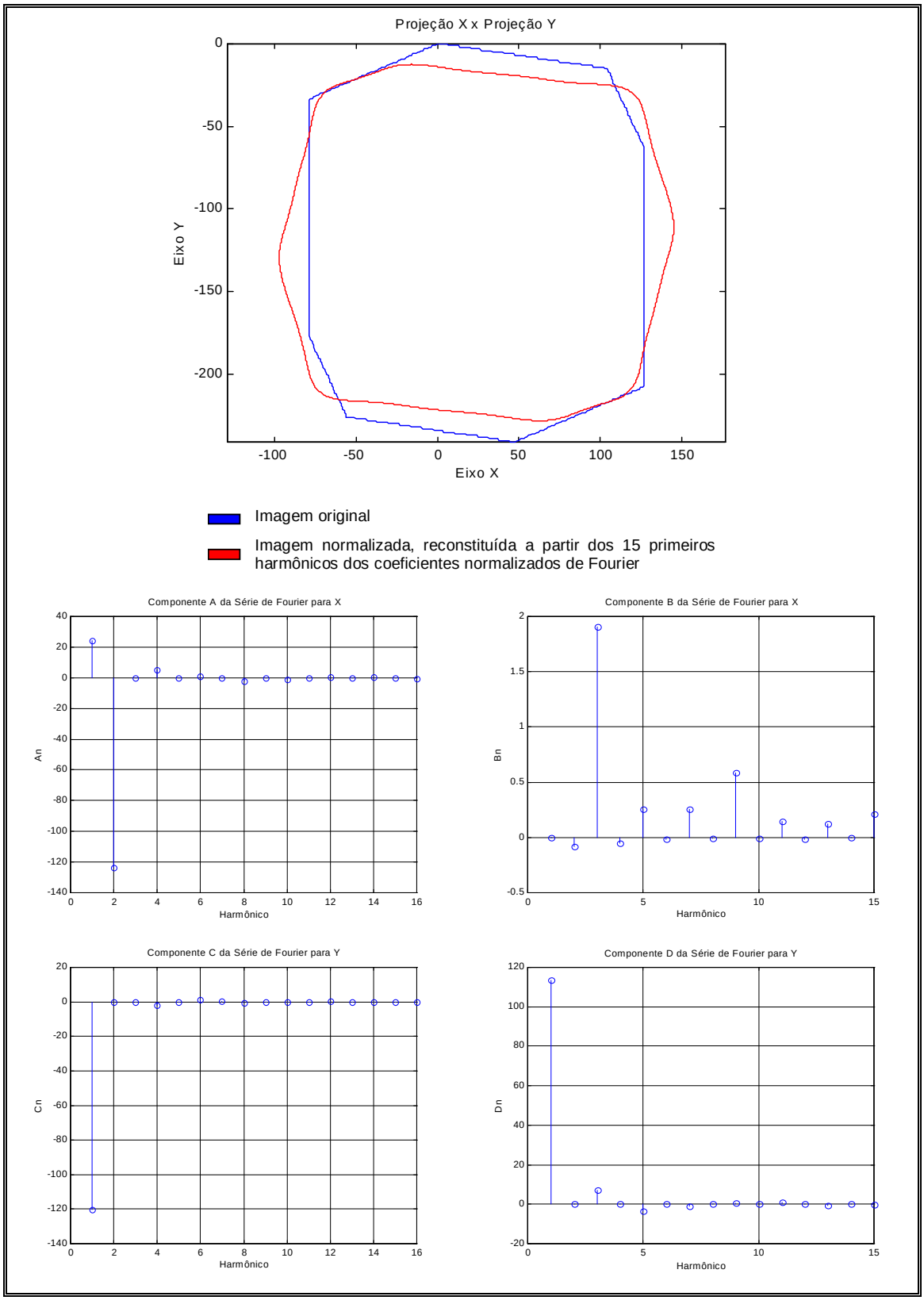


Figura 29 – Prisma de base hexagonal e seus coeficientes normalizados de Fourier

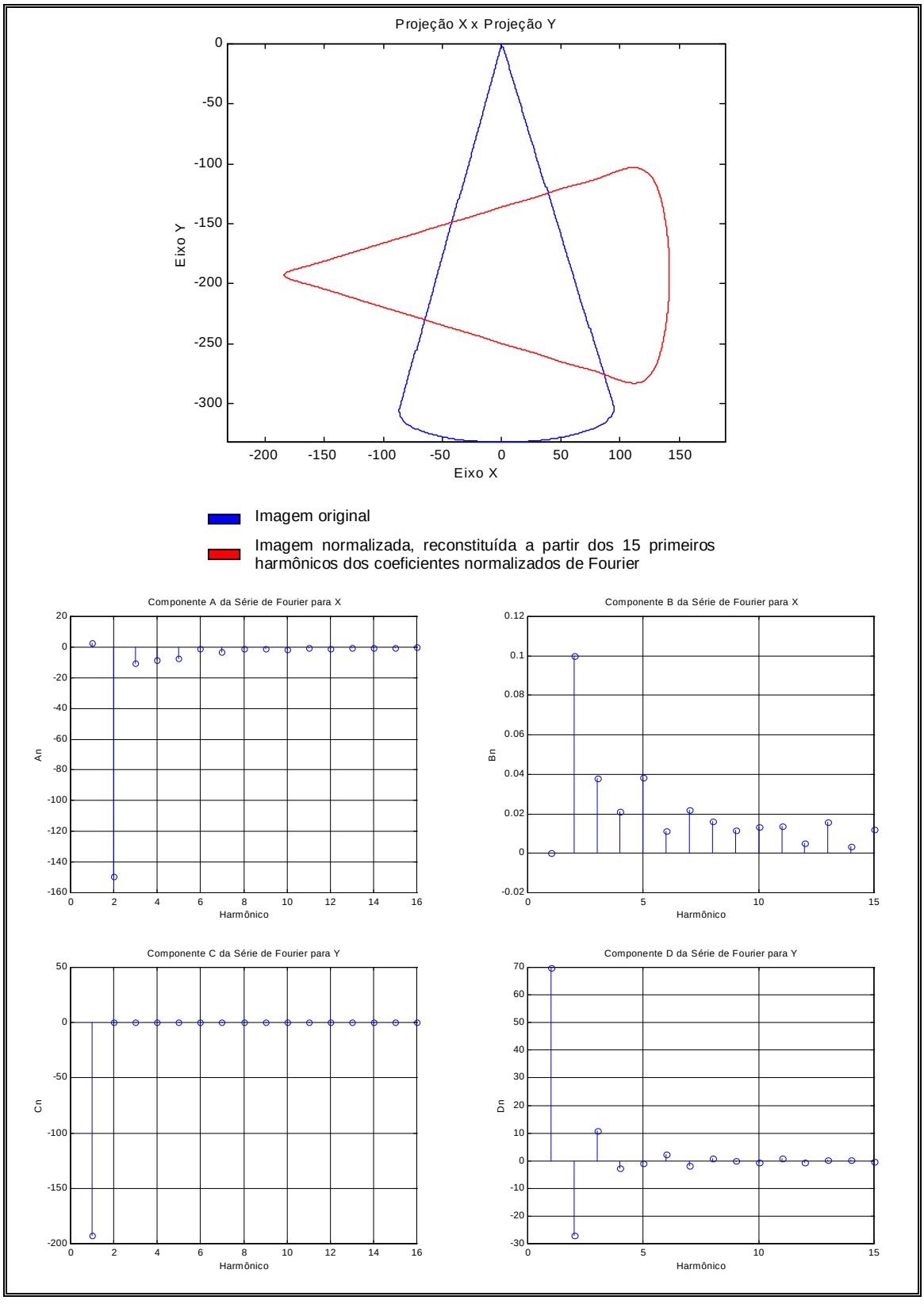


Figura 30 – Cone e seus coeficientes normalizados de Fourier

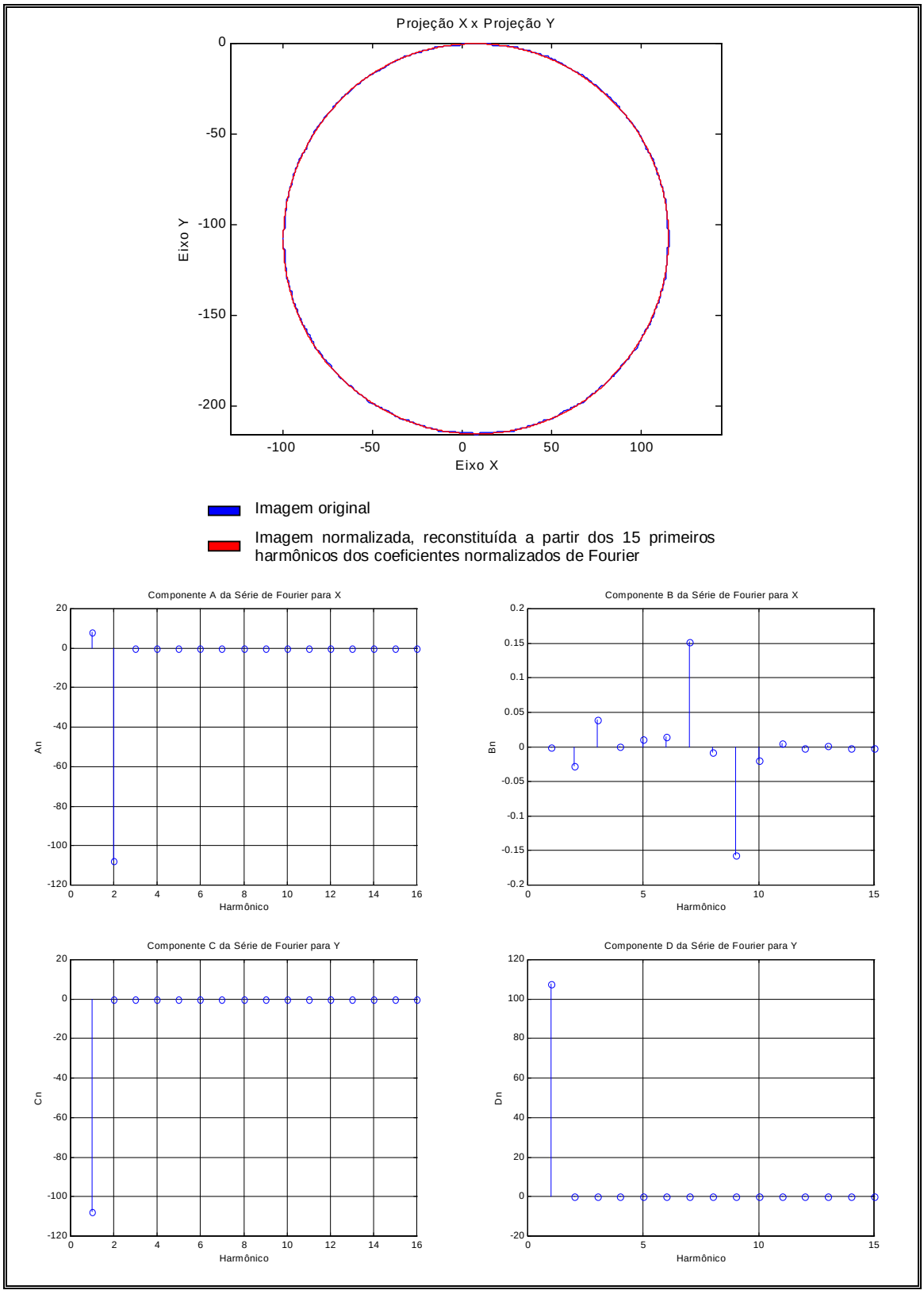


Figura 31 – Esfera e seus coeficientes normalizados de Fourier

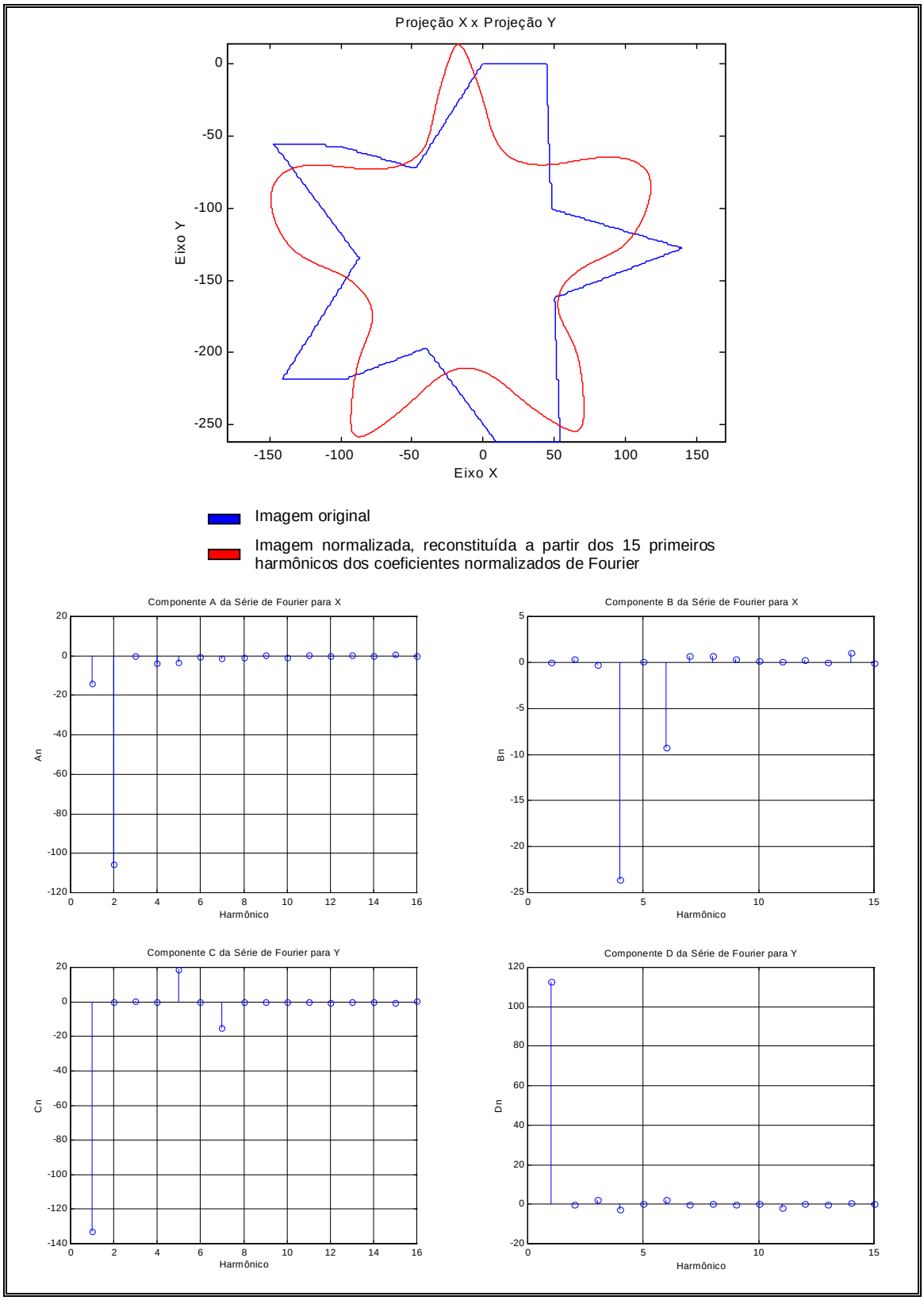


Figura 32 – Estrela 3D e seus coeficientes normalizados de Fourier

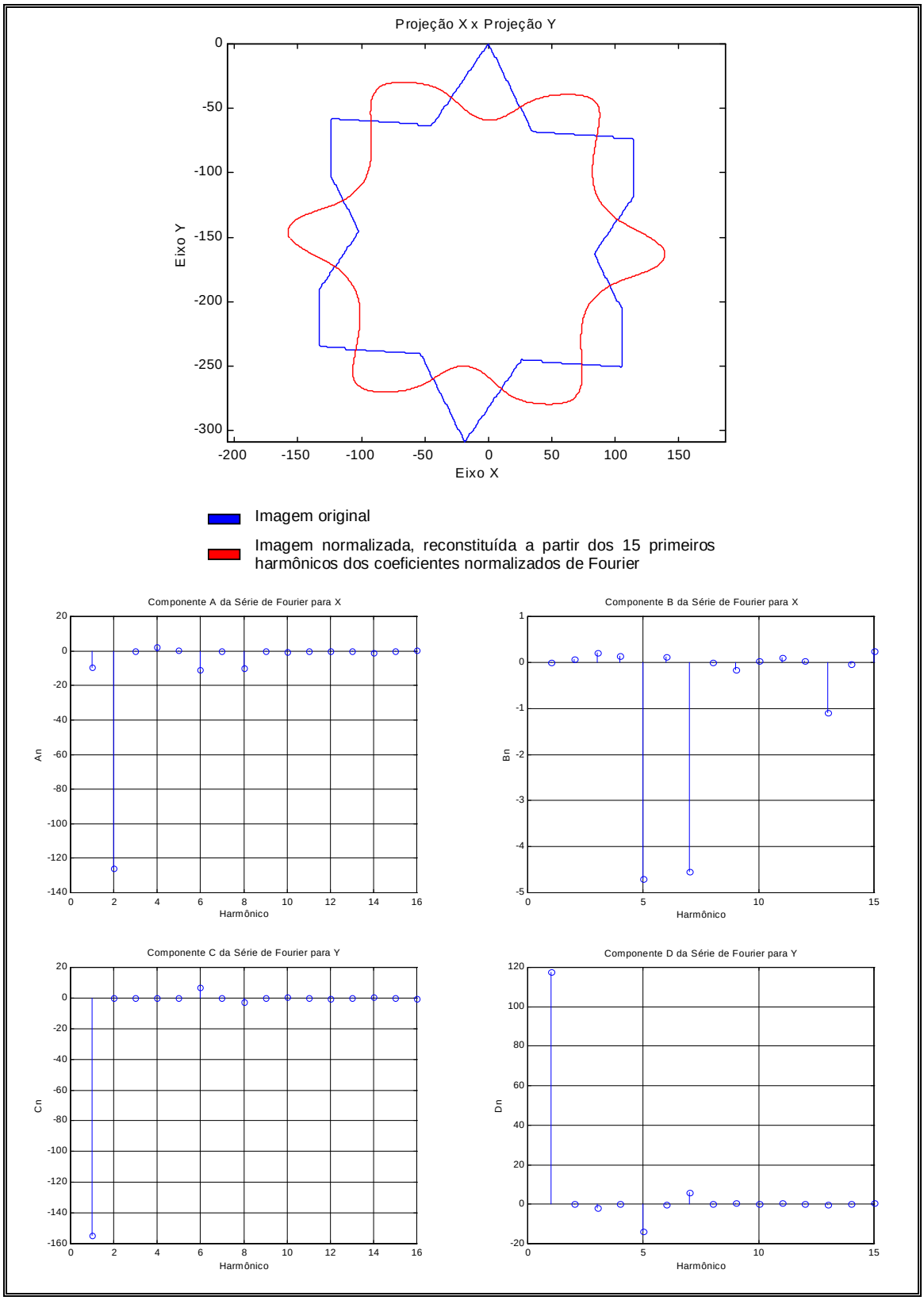


Figura 33 – Estrela de Davi 3D e seus coeficientes normalizados de Fourier

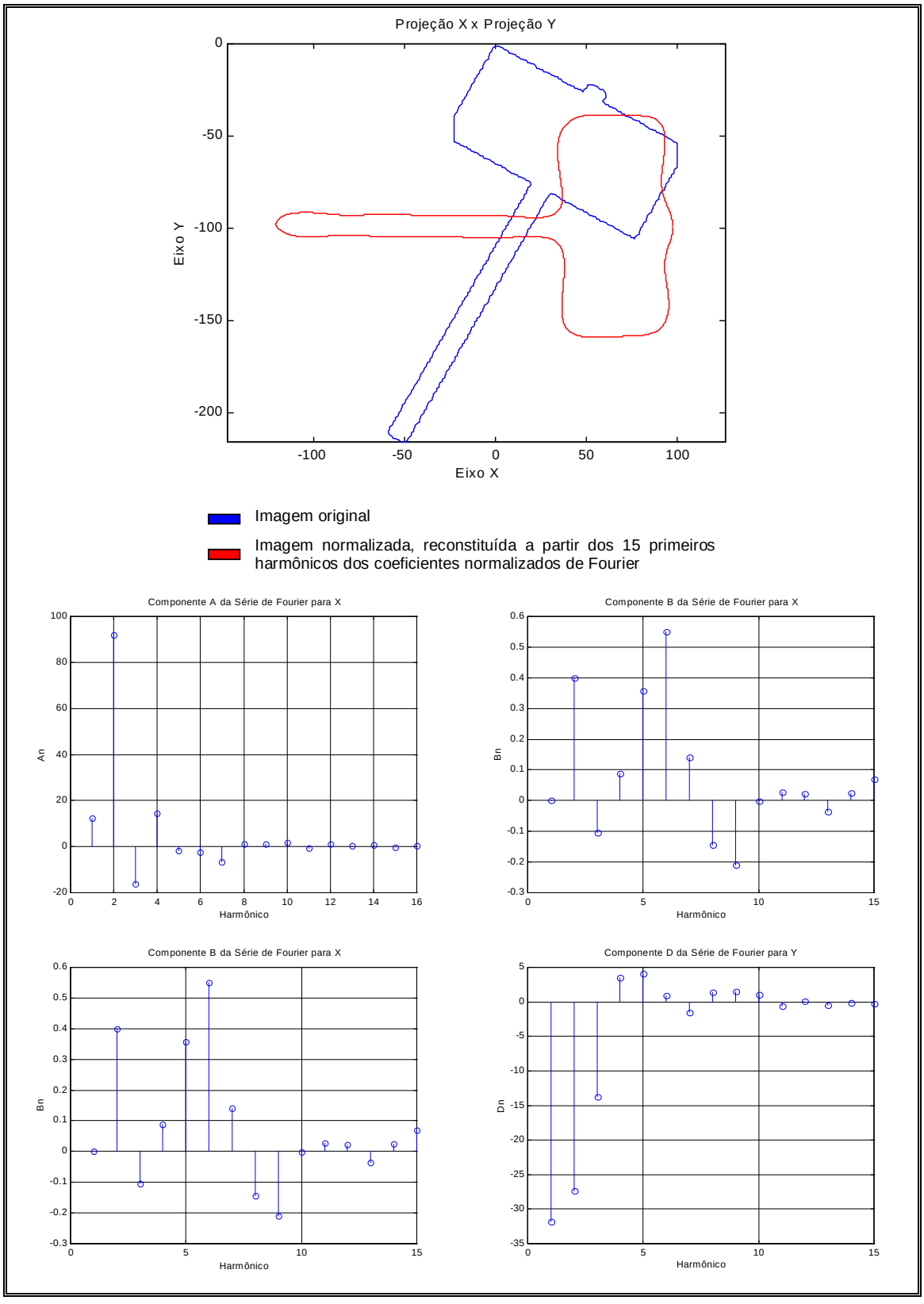


Figura 34 – Martelo e seus coeficientes normalizados de Fourier

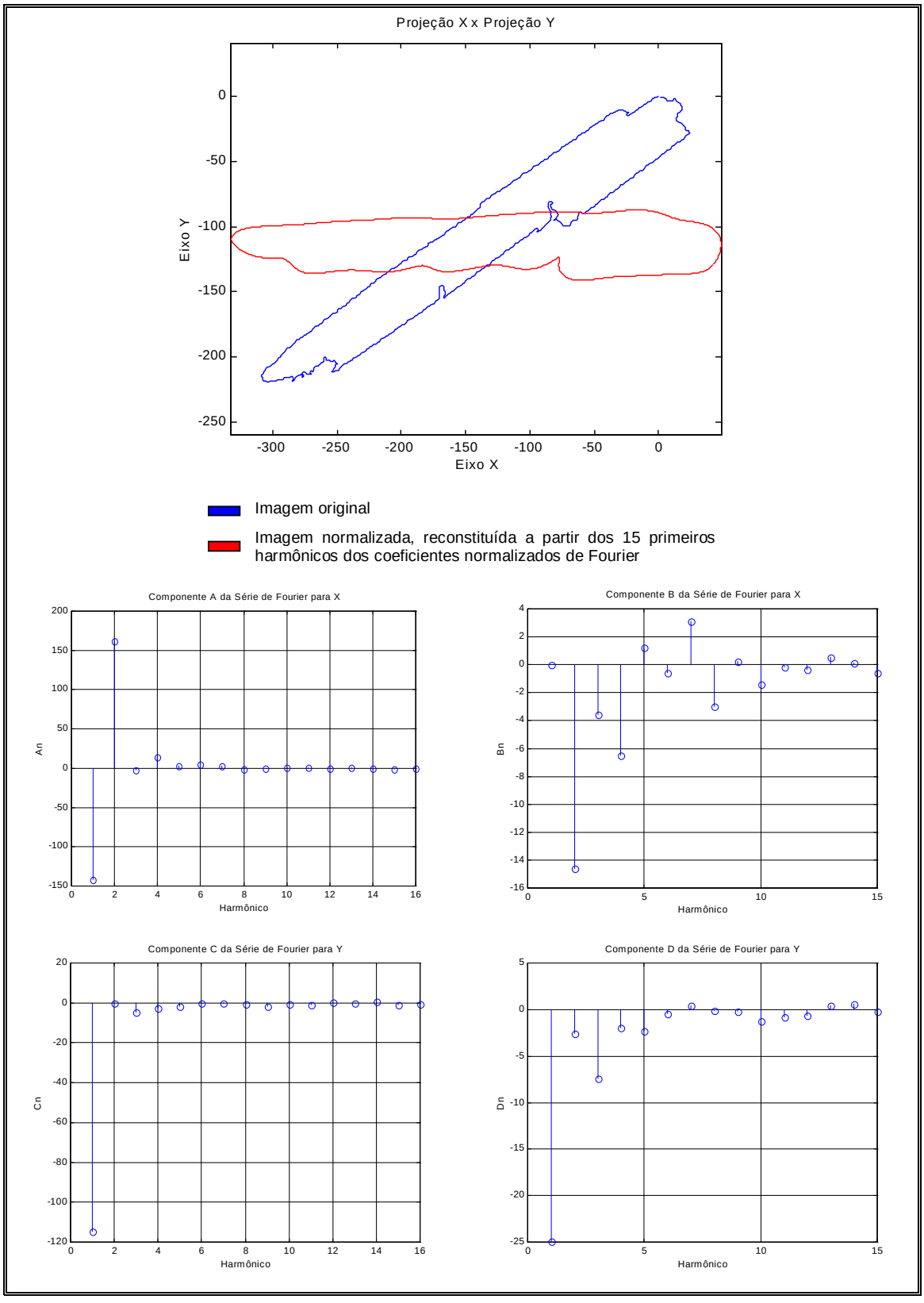


Figura 35 – Caneta e seus coeficientes normalizados de Fourier

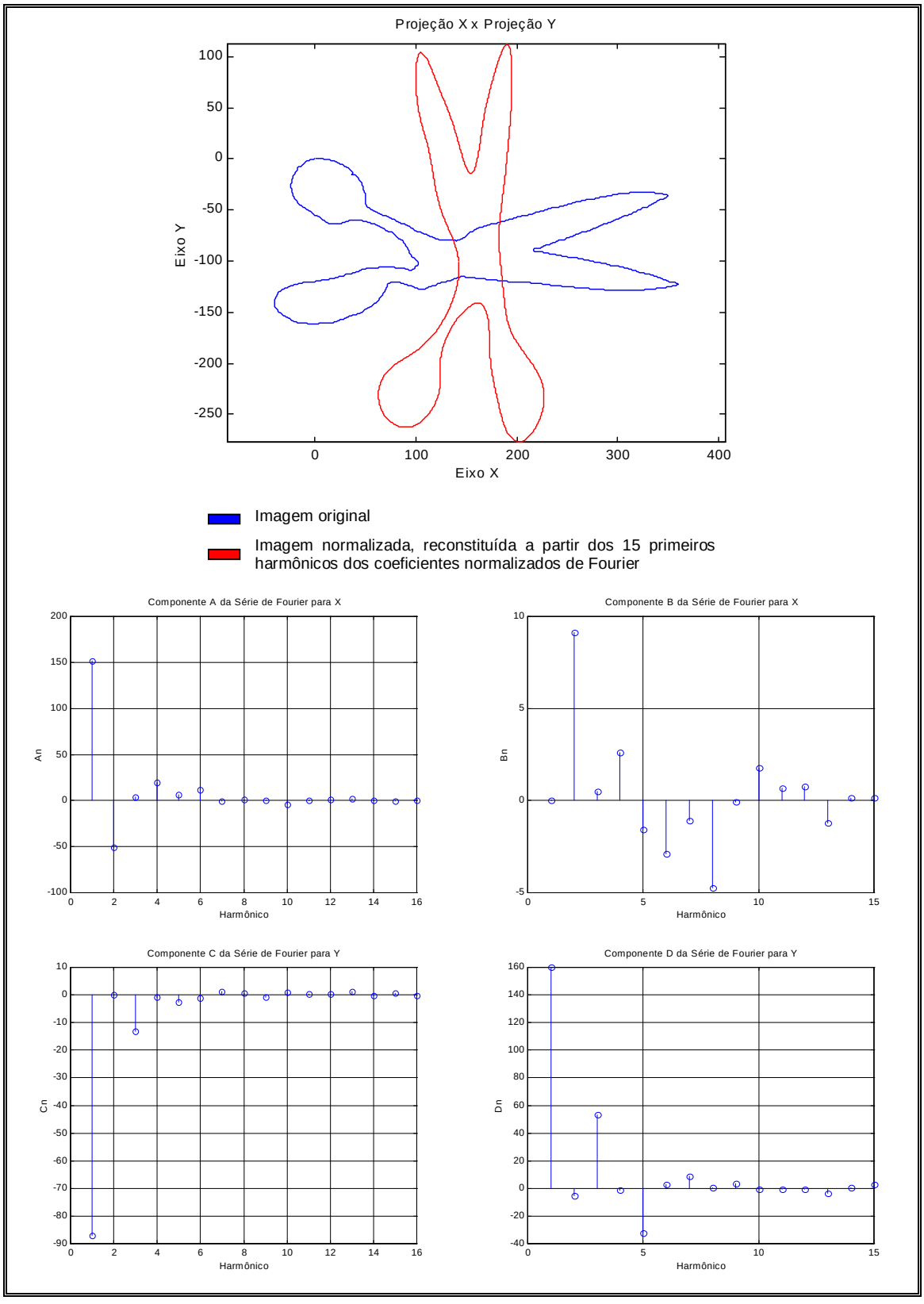


Figura 36 – Tesoura e seus coeficientes normalizados de Fourier

Apêndice B: Coeficientes de Fourier da Base de Dados

Tabela 4 – Coeficientes normalizados de Fourier do prisma de base retangular

N	A_n	B_n	C_n	D_n
1	-80.2254	0.0000	0.0000	103.2473
2	0.0454	0.0536	-0.0641	0.0020
3	0.2030	2.6637	-0.6344	-7.1731
4	-0.0327	0.0072	0.0523	0.0051
5	-4.9491	0.7643	0.4174	0.2265
6	0.0267	0.0113	-0.0347	-0.0030
7	0.2895	0.8805	-0.3888	1.2139
8	-0.0083	0.0110	0.0169	0.0038
9	-0.7774	0.5545	0.3097	-1.1865
10	0.0032	0.0066	-0.0004	0.0014
11	-0.3835	0.3375	-0.1796	0.8641
12	0.0201	0.0130	-0.0050	-0.0006
13	0.2169	0.4424	0.0806	-0.3948
14	-0.0007	-0.0134	0.0171	0.0006
15	-0.3271	0.1451	0.0457	0.0632
16	0.0043	-0.0018	-0.0143	-0.0056
17	0.2201	0.2671	-0.0924	0.1673
18	-0.0041	0.0046	0.0081	0.0033
19	-0.0301	0.1248	0.1369	-0.2374
20	-0.0013	-0.0062	0.0016	-0.0006
21	0.0477	0.0762	-0.1047	0.2203
22	0.0033	-0.0016	-0.0052	0.0022
23	0.1750	0.1528	0.0563	-0.1390
24	0.0064	-0.0042	0.0058	0.0048
25	-0.0329	-0.0011	0.0106	0.0197

Tabela 5 – Coeficientes normalizados de Fourier do prisma de base hexagonal

N	A_n	B_n	C_n	D_n
1	-123.5330	0.0000	0.0000	113.3579
2	-0.0042	-0.0820	-0.0865	0.1056
3	5.3119	1.9037	-2.2358	7.0559
4	0.0039	-0.0495	-0.0323	0.0013
5	1.1803	0.2550	1.4302	-3.4115
6	-0.0175	-0.0130	0.0702	-0.0018
7	-2.2457	0.2567	-0.8308	-1.0757
8	-0.0054	-0.0069	0.0018	0.0297
9	-1.3702	0.5827	-0.0149	0.6234
10	-0.0032	-0.0101	-0.0228	0.0017
11	0.3771	0.1478	0.3035	0.8956
12	-0.0067	-0.0156	0.0064	0.0086
13	0.3584	0.1205	-0.2156	-0.5268
14	-0.0071	0.0021	0.0164	-0.0076
15	-0.4914	0.2101	0.0402	-0.3919
16	-0.0152	0.0031	0.0212	0.0030
17	-0.2955	0.1682	-0.0331	0.4032
18	-0.0102	-0.0138	-0.0111	0.0182
19	0.2034	0.1136	0.0502	0.1217
20	-0.0058	-0.0141	0.0003	-0.0029
21	0.1362	0.0313	0.0473	-0.1327
22	-0.0005	-0.0062	0.0165	0.0029
23	-0.1322	0.1000	-0.1168	-0.1290
24	0.0031	0.0028	0.0010	-0.0010
25	-0.1148	0.1132	0.0651	0.0956

Tabela 6 – Coeficientes normalizados de Fourier do cone

N	A_n	B_n	C_n	D_n
1	-149.3663	0.0000	0.0000	69.7690
2	-10.8578	0.1000	0.1635	-27.0165
3	-8.7319	0.0380	-0.0886	10.6826
4	-7.2302	0.0209	0.0548	-2.7271
5	-1.2140	0.0384	-0.0098	-1.1091
6	-3.4593	0.0110	-0.0053	2.1889
7	-1.2323	0.0219	0.0085	-1.8845
8	-1.1099	0.0161	0.0014	0.8744
9	-1.4025	0.0117	-0.0134	0.0354
10	-0.4003	0.0132	0.0197	-0.6464
11	-0.9157	0.0137	-0.0212	0.7349
12	-0.5093	0.0052	0.0138	-0.5333
13	-0.3712	0.0158	-0.0079	0.1323
14	-0.5861	0.0034	-0.0009	0.1692
15	-0.1993	0.0120	0.0016	-0.3460
16	-0.4014	0.0046	-0.0036	0.2917
17	-0.2763	0.0113	-0.0023	-0.1502
18	-0.1887	-0.0007	0.0031	-0.0547
19	-0.3034	0.0174	-0.0087	0.1658
20	-0.1385	-0.0080	0.0056	-0.2054
21	-0.2040	0.0185	-0.0066	0.1264
22	-0.1848	-0.0072	-0.0057	-0.0223
23	-0.0998	0.0136	-0.0089	-0.0893
24	-0.1933	0.0065	-0.0040	0.1054
25	-0.0909	0.0037	0.0049	-0.1038

Tabela 7 – Coeficientes normalizados de Fourier da esfera

N	A_n	B_n	C_n	D_n
1	-107.6975	0	0	107.7107
2	-0.1608	-0.0277	-0.0309	0.1658
3	0.0350	0.0397	-0.0381	0.0252
4	0.0038	0.0006	0.0018	-0.0021
5	0.0072	0.0116	0.0062	-0.0062
6	0.0025	0.0144	-0.0148	-0.0012
7	-0.0500	0.1525	-0.1441	-0.0476
8	-0.0069	-0.0080	-0.0469	-0.0157
9	0.0460	-0.1565	-0.1677	-0.0493
10	0.0067	-0.0193	-0.0172	-0.0068
11	-0.0027	0.0051	0.0040	0.0017
12	0.0029	-0.0018	0.0017	0.0026
13	-0.0041	0.0021	-0.0042	-0.0078
14	-0.0007	-0.0014	-0.0008	-0.0086
15	0.0006	-0.0014	0.0068	0.0156
16	0.0157	0.0063	0.0142	-0.0252
17	0.0038	-0.0042	0.0000	-0.0156
18	0.0124	0.0096	0.0112	-0.0124
19	-0.0069	0.0040	-0.0090	0.0045
20	-0.0002	-0.0048	0.0040	0.0017
21	0.0028	0.0068	0.0110	0.0052
22	0.0134	-0.0172	0.0112	0.0122
23	-0.0032	0.0002	-0.0103	-0.0126
24	0.0077	-0.0044	0.0073	0.0077
25	0.0257	-0.0220	-0.0172	-0.0201

Tabela 8 – Coeficientes normalizados de Fourier da estrela 3D

N	A_n	B_n	C_n	D_n
1	-105.6886	0.0000	0.0000	112.6627
2	-0.0411	0.3276	0.1793	-0.3240
3	-3.7863	-0.3122	-0.0182	2.2013
4	-3.4149	-23.6241	18.5731	-2.5750
5	-0.5801	0.0311	-0.3471	0.0306
6	-1.5234	-9.2095	-15.0034	1.9830
7	-1.0795	0.6666	-0.1241	-0.4429
8	0.1055	0.6701	-0.1371	0.0700
9	-0.7905	0.3077	0.0093	-0.1677
10	0.2000	0.1191	-0.0806	-0.0173
11	-0.2754	0.0838	-0.6304	-2.0005
12	0.1789	0.2067	-0.1283	0.1204
13	-0.0557	0.0211	-0.0782	-0.4460
14	0.5139	1.0738	-0.7957	0.3284
15	0.0644	-0.1004	0.1128	0.0002
16	-0.0048	-0.1335	2.1785	-0.9449
17	0.1835	-0.2230	0.0172	0.1193
18	-0.3593	-0.8391	-0.2385	0.0559
19	0.3121	-0.1995	-0.0440	-0.0022
20	-0.1450	-0.1519	0.0045	0.0014
21	0.3641	-0.2392	0.2178	0.3182
22	-0.1730	-0.1299	0.0146	-0.0541
23	0.3087	-0.1558	-0.0692	-0.0582
24	-0.2608	-0.2694	0.1414	-0.1234
25	0.1436	0.0490	-0.1185	-0.0940

Tabela 9 – Coeficientes normalizados de Fourier da estrela de Davi 3D

N	A_n	B_n	C_n	D_n
1	-126.1339	0.0000	0.0000	117.6311
2	-0.2001	0.0693	0.0910	0.2906
3	2.1019	0.2007	-0.0760	-1.7480
4	0.0930	0.1448	-0.0737	0.1237
5	-10.9620	-4.7007	6.6110	-13.8318
6	0.0263	0.1138	0.0768	-0.2251
7	-10.2385	-4.5382	-2.6814	5.6803
8	-0.1295	-0.0023	0.0193	0.0469
9	-0.4451	-0.1654	0.2118	0.4640
10	-0.0147	0.0346	-0.0147	0.0066
11	-0.0156	0.1004	-0.4653	0.4179
12	0.0255	0.0256	-0.0607	0.0387
13	-0.9895	-1.1011	0.6399	-0.4429
14	-0.0244	-0.0324	0.0076	0.0144
15	0.0732	0.2517	-0.6484	0.4173
16	-0.0021	0.0249	-0.0728	0.0350
17	-0.1763	-0.4859	0.9142	-0.2846
18	-0.0139	0.0513	0.0190	-0.0328
19	-0.4927	-1.4210	0.2007	-0.0646
20	-0.0285	-0.0162	0.0302	-0.0112
21	-0.1576	-0.5002	-0.6025	0.2550
22	-0.0308	-0.0032	0.0017	0.0000
23	-0.0212	0.0558	-0.1337	-0.0693
24	-0.0089	0.0242	-0.0271	-0.0247
25	0.0070	-0.2014	0.3534	0.0972

Tabela 10 – Coeficientes normalizados de Fourier do martelo

N	A_n	B_n	C_n	D_n
1	92.0771	0.0000	0	-31.8575
2	-16.1885	0.3994	0.6757	-27.3587
3	14.4292	-0.1045	-0.0759	-13.8103
4	-1.6367	0.0876	-0.0167	3.4168
5	-2.2517	0.3574	-0.1307	4.0091
6	-6.7664	0.5509	-0.0812	0.9294
7	1.0890	0.1415	-0.1561	-1.6005
8	1.1417	-0.1435	-0.0039	1.4185
9	1.7929	-0.2101	-0.0380	1.5067
10	-0.5584	-0.0012	-0.0245	1.0238
11	0.9447	0.0261	-0.0774	-0.6235
12	0.3120	0.0210	0.0356	0.0484
13	0.6445	-0.0348	0.0540	-0.4913
14	-0.1412	0.0237	0.0561	-0.1788
15	0.5013	0.0708	-0.0609	-0.3021
16	-0.1642	0.1913	-0.0826	0.6721
17	-0.1631	0.2158	-0.1233	0.2113
18	-0.2658	0.1493	-0.0429	0.1120
19	0.5483	-0.0082	0.0059	-0.2823
20	0.3995	-0.0413	0.1038	0.3190
21	0.2602	0.0104	0.0551	0.0362
22	-0.0273	0.0774	0.0224	0.0237
23	0.2880	0.0444	-0.0352	-0.3012
24	0.1876	0.0185	0.0318	0.0534
25	0.1360	0.0258	0.0180	-0.1503

Tabela 11 – Coeficientes normalizados de Fourier do robô IRB2000

N	A_n	B_n	C_n	D_n
1	52.2169	0.0000	0.0000	-180.6637
2	85.4682	14.9396	6.9287	1.5662
3	-16.3645	-9.6212	3.1071	16.7998
4	14.2401	-1.4240	-6.0507	7.9057
5	6.4500	0.9499	0.6357	-3.0837
6	1.5611	0.0774	2.2878	-6.7055
7	-0.5502	-4.2954	-1.5535	1.2164
8	6.0849	2.3395	-3.2434	1.7866
9	1.8778	1.2467	1.4774	-4.8838
10	-0.5094	-0.9760	-0.9973	-2.7835
11	-0.2229	0.6371	-1.9781	0.4478
12	1.4030	0.4547	1.9804	0.6923
13	1.3851	-2.2296	-0.4560	-1.8257
14	0.3358	0.0253	-1.5184	-1.1670
15	0.0851	1.4807	1.1532	1.6037
16	0.2387	-0.5408	0.7090	0.8404
17	0.9709	-0.2947	-0.1495	0.0070
18	1.1586	0.3813	1.5021	-0.1863
19	-0.5287	-0.2508	0.4585	0.5589
20	-0.6795	-0.0284	-0.2300	0.5680
21	-0.0826	0.4588	1.2821	1.1722
22	1.0169	-0.8996	0.1692	-0.2632
23	0.3050	-0.4734	0.2036	-0.6450
24	0.5061	0.1566	-0.4333	0.5343
25	-0.5125	0.2058	-0.6248	0.3599

Tabela 12 – Coeficientes normalizados de Fourier da caneta

N	A_n	B_n	C_n	D_n
1	161.6161	0.0000	0.0000	-24.8720
2	-1.8167	-14.5858	-4.6629	-2.5698
3	14.6804	-3.5558	-2.5139	-7.4468
4	3.2408	-6.4981	-1.7686	-1.9705
5	5.1952	1.2187	-0.1554	-2.3011
6	2.8906	-0.5726	-0.1391	-0.4277
7	-1.3539	3.1409	-0.3232	0.4130
8	0.0518	-2.9987	-1.8120	-0.0711
9	0.3788	0.2708	-0.4065	-0.2309
10	0.6242	-1.3855	-1.0848	-1.2107
11	0.2119	-0.1672	0.1914	-0.8437
12	0.8579	-0.3390	-0.2103	-0.6507
13	-0.3639	0.5513	0.5002	0.4598
14	-0.6842	0.1197	-1.0310	0.5960
15	-0.6064	-0.6029	-0.6237	-0.2345
16	0.1882	0.6538	-0.6184	-0.1663
17	-0.7724	-0.3576	-0.0897	-0.2608
18	0.6327	0.5084	0.3895	0.0231
19	-0.2678	0.3893	0.4206	0.2316
20	0.1620	0.8428	0.0488	0.5134
21	-0.4983	-0.1188	-0.4881	-0.0680
22	0.4841	0.1488	0.2607	-0.4675
23	-0.0913	0.1705	0.0831	-0.1340
24	0.0949	-0.0461	0.1721	-0.4281
25	0.2923	0.1509	-0.0079	-0.0569

Tabela 13 – Coeficientes normalizados de Fourier da tesoura

N	A_n	B_n	C_n	D_n
1	-50.8211	0.0000	0.0000	159.7450
2	3.4110	9.1335	-13.1583	-5.2108
3	19.7268	0.5173	-0.9208	53.1950
4	6.3649	2.6075	-2.5914	-1.3780
5	11.6211	-1.5621	-1.0236	-32.4650
6	-0.2491	-2.8835	1.0977	2.6168
7	0.8421	-1.0978	0.7368	8.6182
8	0.0100	-4.7419	-0.7962	0.3075
9	-3.8272	-0.0739	0.9225	3.2392
10	0.1292	1.7592	0.1822	-0.9305
11	1.1931	0.6863	0.2357	-0.5777
12	1.7091	0.7726	1.1179	-0.6324
13	0.4866	-1.2273	-0.2863	-3.6593
14	-0.4097	0.1385	0.5137	0.3958
15	0.2035	0.1667	-0.1632	2.5322
16	0.9783	-0.0473	-1.1964	-0.4150
17	0.1262	-0.9231	-0.4220	-0.6601
18	-0.0796	0.7576	0.1380	0.4459
19	0.9083	-0.5101	0.1970	-0.3671
20	-0.2099	-0.0299	0.3408	0.2639
21	0.3263	-0.2209	0.0684	-0.0127
22	-0.2159	-0.1386	-0.0316	0.3256
23	0.3760	-0.1915	0.3698	0.8430
24	-0.3216	-0.1765	-0.4066	-0.3781
25	0.0242	0.1936	0.0921	-0.7782

Apêndice C: Programas Desenvolvidos

Programa 1 – Leitura das Imagens, Conversão para "escala de cinza" e Codificação em Cadeias Direcionais

```
% ------------------------------------------------------------------------------
%           Este Programa Lê e Codifica uma Imagem
% ------------------------------------------------------------------------------

% Lê uma imagem e transforma em arquivos numéricos
% ------------------------------------------------------

clear all, clc

caixa=imread('d:\kleyton\semestre2_99\projfinal\imagens\caixa','jpg');
esfera=imread('d:\kleyton\semestre2_99\projfinal\imagens\esfera','jpg');
cone=imread('d:\kleyton\semestre2_99\projfinal\imagens\cone','jpg');
hex=imread('d:\kleyton\semestre2_99\projfinal\imagens\hex','jpg');
estrela=imread('d:\kleyton\semestre2_99\projfinal\imagens\estrela','jpg');
davi=imread('d:\kleyton\semestre2_99\projfinal\imagens\davi','jpg');
martelo=imread('d:\kleyton\semestre2_99\projfinal\imagens\martelo','jpg');
robo=imread('d:\kleyton\semestre2_99\projfinal\imagens\robo','jpg');
caneta=imread('d:\kleyton\semestre2_99\projfinal\imagens\caneta','jpg');
tesoura=imread('d:\kleyton\semestre2_99\projfinal\imagens\tesoura','jpg');

caixa=caixa(:,:,1);
esfera=esfera(:,:,1);
cone=cone(:,:,1);
hex=hex(:,:,1);
estrela=estrela(:,:,1);
davi=davi(:,:,1);
martelo=martelo(:,:,1);
robo=robo(:,:,1);
caneta=caneta(:,:,1);
tesoura=tesoura(:,:,1);

save d:\kleyton\semestre2_99\projfinal\imagens\caixa caixa
save d:\kleyton\semestre2_99\projfinal\imagens\esfera esfera
save d:\kleyton\semestre2_99\projfinal\imagens\cone cone
save d:\kleyton\semestre2_99\projfinal\imagens\hex hex
save d:\kleyton\semestre2_99\projfinal\imagens\estrela estrela
save d:\kleyton\semestre2_99\projfinal\imagens\davi davi
save d:\kleyton\semestre2_99\projfinal\imagens\martelo martelo
```

```
save d:\kleyton\semestre2_99\projfinal\imagens\robo robo
save d:\kleyton\semestre2_99\projfinal\imagens\caneta caneta
save d:\kleyton\semestre2_99\projfinal\imagens\tesoura tesoura

% Codifica a imagem
% ---------------------

clear all, clc

load('d:\kleyton\semestre2_99\projfinal\imagens\caixa');
load('d:\kleyton\semestre2_99\projfinal\imagens\esfera');
load('d:\kleyton\semestre2_99\projfinal\imagens\cone');
load('d:\kleyton\semestre2_99\projfinal\imagens\hex');
load('d:\kleyton\semestre2_99\projfinal\imagens\estrela');
load('d:\kleyton\semestre2_99\projfinal\imagens\davi');
load('d:\kleyton\semestre2_99\projfinal\imagens\martelo');
load('d:\kleyton\semestre2_99\projfinal\imagens\robo');
load('d:\kleyton\semestre2_99\projfinal\imagens\caneta');
load('d:\kleyton\semestre2_99\projfinal\imagens\tesoura');

imagem=martelo;

cor=190;
l=1;
for i=1:size(imagem,1)
  for j=1:size(imagem,2)
    if imagem(i,j) < cor
      p(l,:)=[i j];
      l=l+1;
      break
    end
  end
end
clear l;

P(1,:)=p(1,:);
k=2;
i=P(1,1); j=P(1,2);

if imagem(i,j+1)<cor
  P(k,:)=[i j+1];
  code(k-1)=0;
elseif imagem(i+1,j+1)<cor
  P(k,:)=[i+1 j+1];
  code(k-1)=7;
elseif imagem(i+1,j)<cor
```

```
    P(k,:)=[i+1 j];
    code(k-1)=6;
  elseif imagem(i+1,j-1)<cor
    P(k,:)=[i+1 j-1];
    code(k-1)=5;
  elseif imagem(i,j-1)<cor
    P(k,:)=[i j-1];
    code(k-1)=4;
  elseif imagem(i-1,j-1)<cor
    P(k,:)=[i-1 j-1];
    code(k-1)=3;
  elseif imagem(i-1,j)<cor
    P(k,:)=[i-1 j];
    code(k-1)=2;
  elseif imagem(i-1,j+1)<cor
    P(k,:)=[i-1 j+1];
    code(k-1)=1;
  end

  while((P(k,1)~=P(1,1)) | (P(k,2)~=P(1,2)))

    k=k+1;
    i=P(k-1,1); j=P(k-1,2);

    if code(k-2)== 0;

      if imagem(i-1,j-1)<cor
        P(k,:)=[i-1 j-1];
        code(k-1)=3;
      elseif imagem(i-1,j)<cor
        P(k,:)=[i-1 j];
        code(k-1)=2;
      elseif imagem(i-1,j+1)<cor
        P(k,:)=[i-1 j+1];
        code(k-1)=1;
      elseif imagem(i,j+1)<cor
        P(k,:)=[i j+1];
        code(k-1)=0;
      elseif imagem(i+1,j+1)<cor
        P(k,:)=[i+1 j+1];
        code(k-1)=7;
      elseif imagem(i+1,j)<cor
        P(k,:)=[i+1 j];
        code(k-1)=6;
      elseif imagem(i+1,j-1)<cor
        P(k,:)=[i+1 j-1];
```

```
            code(k-1)=5;
        elseif imagem(i,j-1)<cor
            P(k,:)=[i j-1];
            code(k-1)=4;
        end

    elseif code(k-2)== 7;

        if imagem(i-1,j)<cor
            P(k,:)=[i-1 j];
            code(k-1)=2;
        elseif imagem(i-1,j+1)<cor
            P(k,:)=[i-1 j+1];
            code(k-1)=1;
        elseif imagem(i,j+1)<cor
            P(k,:)=[i j+1];
            code(k-1)=0;
        elseif imagem(i+1,j+1)<cor
            P(k,:)=[i+1 j+1];
            code(k-1)=7;
        elseif imagem(i+1,j)<cor
            P(k,:)=[i+1 j];
            code(k-1)=6;
        elseif imagem(i+1,j-1)<cor
            P(k,:)=[i+1 j-1];
            code(k-1)=5;
        elseif imagem(i,j-1)<cor
            P(k,:)=[i j-1];
            code(k-1)=4;
        elseif imagem(i-1,j-1)<cor
            P(k,:)=[i-1 j-1];
            code(k-1)=3;
        end

    elseif code(k-2)== 6;

        if imagem(i-1,j+1)<cor
            P(k,:)=[i-1 j+1];
            code(k-1)=1;
        elseif imagem(i,j+1)<cor
            P(k,:)=[i j+1];
            code(k-1)=0;
        elseif imagem(i+1,j+1)<cor
            P(k,:)=[i+1 j+1];
            code(k-1)=7;
        elseif imagem(i+1,j)<cor
```

```
        P(k,:)=[i+1 j];
        code(k-1)=6;
      elseif imagem(i+1,j-1)<cor
        P(k,:)=[i+1 j-1];
        code(k-1)=5;
      elseif imagem(i,j-1)<cor
        P(k,:)=[i j-1];
        code(k-1)=4;
      elseif imagem(i-1,j-1)<cor
        P(k,:)=[i-1 j-1];
        code(k-1)=3;
      elseif imagem(i-1,j)<cor
        P(k,:)=[i-1 j];
        code(k-1)=2;
      end

    elseif code(k-2)== 5;

      if imagem(i,j+1)<cor
        P(k,:)=[i j+1];
        code(k-1)=0;
      elseif imagem(i+1,j+1)<cor
        P(k,:)=[i+1 j+1];
        code(k-1)=7;
      elseif imagem(i+1,j)<cor
        P(k,:)=[i+1 j];
        code(k-1)=6;
      elseif imagem(i+1,j-1)<cor
        P(k,:)=[i+1 j-1];
        code(k-1)=5;
      elseif imagem(i,j-1)<cor
        P(k,:)=[i j-1];
        code(k-1)=4;
      elseif imagem(i-1,j-1)<cor
        P(k,:)=[i-1 j-1];
        code(k-1)=3;
      elseif imagem(i-1,j)<cor
        P(k,:)=[i-1 j];
        code(k-1)=2;
      elseif imagem(i-1,j+1)<cor
        P(k,:)=[i-1 j+1];
        code(k-1)=1;
      end

    elseif code(k-2)== 4;
```

```
        if imagem(i+1,j+1)<cor
          P(k,:)=[i+1 j+1];
          code(k-1)=7;
        elseif imagem(i+1,j)<cor
          P(k,:)=[i+1 j];
          code(k-1)=6;
        elseif imagem(i+1,j-1)<cor
          P(k,:)=[i+1 j-1];
          code(k-1)=5;
        elseif imagem(i,j-1)<cor
          P(k,:)=[i j-1];
          code(k-1)=4;
        elseif imagem(i-1,j-1)<cor
          P(k,:)=[i-1 j-1];
          code(k-1)=3;
        elseif imagem(i-1,j)<cor
          P(k,:)=[i-1 j];
          code(k-1)=2;
        elseif imagem(i-1,j+1)<cor
          P(k,:)=[i-1 j+1];
          code(k-1)=1;
        elseif imagem(i,j+1)<cor
          P(k,:)=[i j+1];
          code(k-1)=0;
        end

      elseif code(k-2)== 3;

        if imagem(i+1,j)<cor
          P(k,:)=[i+1 j];
          code(k-1)=6;
        elseif imagem(i+1,j-1)<cor
          P(k,:)=[i+1 j-1];
          code(k-1)=5;
        elseif imagem(i,j-1)<cor
          P(k,:)=[i j-1];
          code(k-1)=4;
        elseif imagem(i-1,j-1)<cor
          P(k,:)=[i-1 j-1];
          code(k-1)=3;
        elseif imagem(i-1,j)<cor
          P(k,:)=[i-1 j];
          code(k-1)=2;
        elseif imagem(i-1,j+1)<cor
          P(k,:)=[i-1 j+1];
          code(k-1)=1;
```

```
    elseif imagem(i,j+1)<cor
      P(k,:)=[i j+1];
      code(k-1)=0;
    elseif imagem(i+1,j+1)<cor
      P(k,:)=[i+1 j+1];
      code(k-1)=7;
    end

  elseif code(k-2)== 2;

    if imagem(i+1,j-1)<cor
      P(k,:)=[i+1 j-1];
      code(k-1)=5;
    elseif imagem(i,j-1)<cor
      P(k,:)=[i j-1];
      code(k-1)=4;
    elseif imagem(i-1,j-1)<cor
      P(k,:)=[i-1 j-1];
      code(k-1)=3;
    elseif imagem(i-1,j)<cor
      P(k,:)=[i-1 j];
      code(k-1)=2;
    elseif imagem(i-1,j+1)<cor
      P(k,:)=[i-1 j+1];
      code(k-1)=1;
    elseif imagem(i,j+1)<cor
      P(k,:)=[i j+1];
      code(k-1)=0;
    elseif imagem(i+1,j+1)<cor
      P(k,:)=[i+1 j+1];
      code(k-1)=7;
    elseif imagem(i+1,j)<cor
      P(k,:)=[i+1 j];
      code(k-1)=6;
    end

  elseif code(k-2)== 1;

    if imagem(i,j-1)<cor
      P(k,:)=[i j-1];
      code(k-1)=4;
    elseif imagem(i-1,j-1)<cor
      P(k,:)=[i-1 j-1];
      code(k-1)=3;
    elseif imagem(i-1,j)<cor
      P(k,:)=[i-1 j];
```

```
            code(k-1)=2;
        elseif imagem(i-1,j+1)<cor
            P(k,:)=[i-1 j+1];
            code(k-1)=1;
        elseif imagem(i,j+1)<cor
            P(k,:)=[i j+1];
            code(k-1)=0;
        elseif imagem(i+1,j+1)<cor
            P(k,:)=[i+1 j+1];
            code(k-1)=7;
        elseif imagem(i+1,j)<cor
            P(k,:)=[i+1 j];
            code(k-1)=6;
        elseif imagem(i+1,j-1)<cor
            P(k,:)=[i+1 j-1];
            code(k-1)=5;
        end

    end

end

delete(figure(1))
figure(1)
imshow(imagem);
set(gca,'XTickLabel',''); set(gca,'YTickLabel','');
set(gca,'XTick',1:10:size(imagem,2)); set(gca,'YTick',1:10:size(imagem,1))
set(gca,'GridLineStyle','-')
set(gca,'XColor','green');set(gca,'YColor','green')
grid on
%pause
set(gca,'Visible','off');
%pause
hold on;
set(gca,'XTick',[]); set(gca,'YTick',[]);
set(gca,'LineWidth',8);
for l=1:length(P)
   plot(P(l,2),P(l,1),'r-');
end
hold off;

%save d:\kleyton\semestre2_99\projfinal\imagens\code_caixa code
%save d:\kleyton\semestre2_99\projfinal\imagens\code_esfera code
%save d:\kleyton\semestre2_99\projfinal\imagens\code_cone code
%save d:\kleyton\semestre2_99\projfinal\imagens\code_hex code
%save d:\kleyton\semestre2_99\projfinal\imagens\code_estrela code
```

```
%save d:\kleyton\semestre2_99\projfinal\imagens\code_davi code
%save d:\kleyton\semestre2_99\projfinal\imagens\code_martelo code
%save d:\kleyton\semestre2_99\projfinal\imagens\code_robo code
%save d:\kleyton\semestre2_99\projfinal\imagens\code_caneta code
%save d:\kleyton\semestre2_99\projfinal\imagens\code_tesoura code
```

Programa 2 – Obtenção dos Coeficientes Normalizados de Fourier

```
% ------------------------------------------------------------------------------------
%           Criação dos Bancos de Dados para Treinamento das RNAs
% ------------------------------------------------------------------------------------

clear all, clc, for s=1:6, delete(figure(s)), end;

path(path,'d:\kleyton\Semestre2_99\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Programas');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Imagens');

% ----------------------------------------------------
%         Recuperação dos Códigos
% ----------------------------------------------------

load('d:\kleyton\semestre2_99\projfinal\imagens\code_caixa');
%load('d:\kleyton\semestre2_99\projfinal\imagens\code_esfera');
%load('d:\kleyton\semestre2_99\projfinal\imagens\code_cone');
%load('d:\kleyton\semestre2_99\projfinal\imagens\code_hex');
%load('d:\kleyton\semestre2_99\projfinal\imagens\code_estrela');
%load('d:\kleyton\semestre2_99\projfinal\imagens\code_davi');
%load('d:\kleyton\semestre2_99\projfinal\imagens\code_martelo');
%load('d:\kleyton\semestre2_99\projfinal\imagens\code_robo');
%load('d:\kleyton\semestre2_99\projfinal\imagens\code_caneta');
%load('d:\kleyton\semestre2_99\projfinal\imagens\code_tesoura');

% ----------------------------------------------------
%         Entrada dos códigos
% ----------------------------------------------------

c = code;
K=length(c);

% ----------------------------------------------------
%         Definição do comprimento de arco
```

```
% ------------------------------------------------

T=0;

for i=1:K

  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));

  deltat(i)= 1+((sqrt(2)-1)/2)*(1-(-1)^(c(i)));
  deltax(i)=sign(6-c(i))*sign(2-c(i));
  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

    t(p)=t(p)+deltat(i);
    x(p)=x(p)+deltax(i);
    y(p)=y(p)+deltay(i);

  end

end

clear i, clear p;

% ------------------------------------------------
%       Determinação dos Coeficientes de Fourier
% ------------------------------------------------
m=5;          % Número máximo de harmônicos

for N=1:m;    % Número de harmônicos

  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

A(n)=( T/(2*(n^2)*(pi^2)) )*An;

B(n)=( T/(2*(n^2)*(pi^2)) )*Bn;

C(n)=( T/(2*(n^2)*(pi^2)) )*Cn;

D(n)=( T/(2*(n^2)*(pi^2)) )*Dn;

l=(T/(4*pi))*atan(2*(A(1)*B(1)+C(1)*D(1))/(A(1)^2+C(1)^2-B(1)^2-D(1)^2));

Tempor=[cos((2*pi*n*l)/T) sin((2*pi*n*l)/T);-sin((2*pi*n*l)/T) cos((2*pi*n*l)/T)]*[A(n) C(n);B(n) D(n)];

Ax(n)=Tempor(1,1); Cx(n)=Tempor(1,2); Bx(n)=Tempor(2,1); Dx(n)=Tempor(2,2);

clear Tempor;

psi=atan(Cx(1)/Ax(1));

Ex=sqrt((Ax(1))^2+(Cx(1))^2);

Tempor2=[cos(psi) sin(psi);-sin(psi) cos(psi)]*[Ax(n) Bx(n);Cx(n) Dx(n)];

Axx(n)=Tempor2(1,1); Bxx(n)=Tempor2(1,2); Cxx(n)=Tempor2(2,1); Dxx(n)=Tempor2(2,2);

clear Tempor2;

end

clear An, clear Bn, clear Cn, clear Dn, clear p, clear n, clear i;


% 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 j=1:p-1
    zetap1=zetap1+deltax(j);
    zetap2=zetap2+deltat(j);
    deltap1=deltap1+deltay(j);
    deltap2=deltap2+deltat(j);
  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, clear zetap1, clear zetap2, clear delta, clear deltap1, clear deltap2;

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

clear a0, clear c0, clear p;

t1=[];

for i=2:K
  tt=t1;
  t1=[tt(1:length(tt)-1),t(i-1):(t(i)-t(i-1))/10:t(i)];
end

clear tt, clear i;

for p=1:length(t1)

  X(p)=A0/Ex;
  Y(p)=C0/Ex;

  for n=1:N
```

```
        X(p)=X(p)+( Axx(n)*cos((2*n*pi*t1(p))/T) + Bxx(n)*sin((2*n*pi*t1(p))/T) )/Ex;
        Y(p)=Y(p)+( Cxx(n)*cos((2*n*pi*t1(p))/T) + Dxx(n)*sin((2*n*pi*t1(p))/T) )/Ex;
      end

    end

    % -----------------------------------------------------
    %             Definição do erro
    % -----------------------------------------------------

    s=1:K;
    err(N)=max( max(abs(x(s)-Ex*X(10*(s-1)+1)),abs(y(s)-Ex*Y(10*(s-1)+1))) );

    erqm(N)=sum(sqrt((x(s)-Ex*X(10*(s-1)+1)).^2+(y(s)-Ex*Y(10*(s-1)+1)).^2))/K;

    clear s;

    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);

  end

  % -----------------------------------------------------
  %                  Desenho
  % -----------------------------------------------------

  figure(1)

  %plot(x,y,'b-',X*Ex,Y*Ex,'r-')
  %axis('equal'), xlabel('Eixo X'), ylabel('Eixo Y'), title('Projeção X x Projeção Y')

  %figure(2)

  %hold on
  plot([1:m],err,'r',[1:m],erqm,'b'); %,[1:m],er,'r');
  xlabel('Quantidade de Harmônicos N'), ylabel('Erro'), title('Erro versus quantidade de harmônicos'), grid
  %plot([1:m],err,'r--',[1:m],erqm,'r');
  %hold off

  %figure(3)

  %stem([1:length(Axx)+1],[A0,Axx]);
```

```
%xlabel('Harmônico'), ylabel('An'), title('Componente A da Série de Fourier para X'), grid

%figure(4)

%stem([1:length(Bxx)],Bxx,'b');
%xlabel('Harmônico'), ylabel('Bn'), title('Componente B da Série de Fourier para X'), grid

%figure(5)

%stem([1:length(Cxx)+1],[C0,Cxx],'b');
%xlabel('Harmônico'), ylabel('Cn'), title('Componente C da Série de Fourier para Y'), grid

%figure(6)

%stem([1:length(Dxx)],Dxx,'b');
%xlabel('Harmônico'), ylabel('Dn'), title('Componente D da Série de Fourier para Y'), grid

%clc, [erqm' err']

% ------------------------------------------------------
%                Salva os Resultados
% ------------------------------------------------------

err_cx=err; erqm_cx=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_caixa err_cx erqm_cx
%err_ef=err; erqm_ef=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_esfera err_ef erqm_ef
%err_co=err; erqm_co=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_cone err_co erqm_co
%err_hex=err; erqm_hex=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_hex err_hex erqm_hex
%err_st=err; erqm_st=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_estrela err_st erqm_st
%err_dv=err; erqm_dv=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_davi err_dv erqm_dv
%err_mt=err; erqm_mt=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_martelo err_mt erqm_mt
%err_rb=err; erqm_rb=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_robo err_rb erqm_rb
%err_ca=err; erqm_ca=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_caneta err_ca erqm_ca
%err_ts=err; erqm_ts=erqm; save d:\kleyton\semestre2_99\projfinal\relatórios\erro_tesoura err_ts erqm_ts


%save d:\kleyton\semestre2_99\projfinal\imagens\banco_caixa A0 Axx Bxx C0 Cxx Dxx x y X Y
%save d:\kleyton\semestre2_99\projfinal\imagens\banco_esfera A0 Axx Bxx C0 Cxx Dxx x y X Y
%save d:\kleyton\semestre2_99\projfinal\imagens\banco_cone A0 Axx Bxx C0 Cxx Dxx x y X Y
%save d:\kleyton\semestre2_99\projfinal\imagens\banco_hex A0 Axx Bxx C0 Cxx Dxx x y X Y
%save d:\kleyton\semestre2_99\projfinal\imagens\banco_estrela A0 Axx Bxx C0 Cxx Dxx x y X Y
%save d:\kleyton\semestre2_99\projfinal\imagens\banco_davi A0 Axx Bxx C0 Cxx Dxx x y X Y
%save d:\kleyton\semestre2_99\projfinal\imagens\banco_martelo A0 Axx Bxx C0 Cxx Dxx x y X Y
%save d:\kleyton\semestre2_99\projfinal\imagens\banco_robo A0 Axx Bxx C0 Cxx Dxx x y X Y
%save d:\kleyton\semestre2_99\projfinal\imagens\banco_caneta A0 Axx Bxx C0 Cxx Dxx x y X Y
%save d:\kleyton\semestre2_99\projfinal\imagens\banco_tesoura A0 Axx Bxx C0 Cxx Dxx x y X Y
```

Programa 3 – Treinamento e Validação relativos às Redes Perceptron Multicamadas

```
% ------------------------------------------------------------------
%           Treinamento da Rede Neural Backpropagation
% ------------------------------------------------------------------

for i=1:25
  delete(figure(i));
end
clear all, clc;

inicio=clock;
path(path,'d:\kleyton\Semestre2_99\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Programas');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Imagens');

% ------------------------------------------------
%         Criação da Rede
% ------------------------------------------------

df = 250;                   % Epochs between updating display
me = 5000;                      % Maximum number of epochs to train
eg = 2e-3;                  % Sum-squared error goal
lr = 0.1;                   % Learning rate
lri = 1.05;                 % Learning rate increase
lrd = 0.7;                  % Learning rate decrease
mc = 0.01;                  % Momentum constant
mer = 1.04;                     % Maximum error ratio

tp=[df,me,eg,lr,lri,lrd,mc,mer];

S1=20; S2=10;

F1='purelin';F2='purelin';

t=eye(10);
t=[t t t t];
t=[t t t t t];

% ------------------------------------------------
%         Preparação dos Dados
% ------------------------------------------------

Ruido=75/100;
```

```
for M=1:16

  N = M+9;        % Quantidade de harmônicos utilizados

  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_caixa');
  p(:,1)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_esfera');
  p(:,2)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_cone');
  p(:,3)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_hex');
  p(:,4)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_estrela');
  p(:,5)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_davi');
  p(:,6)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_martelo');
  p(:,7)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_robo');
  p(:,8)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_caneta');
  p(:,9)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
  load('d:\kleyton\semestre2_99\projfinal\imagens\banco_tesoura');
  p(:,10)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';

  % ------------------------------------------------
  %         Treinamento da Rede
  % ------------------------------------------------

  p1=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];
  p2=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];
  p3=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];
  p4=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];
  p5=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];

  p=[p1 p2 p3 p4 p5]; P=abs(p);

  [w1,b1,w2,b2]=initff(P,S1,F1,t,F2);

  w1=w1/10000;
  b1=b1/10000;
  w2=w2/10000;
  b2=b2/10000;

  figure(1)
  [w1,b1,w2,b2,te,tr]=trainbpx(w1,b1,F1,w2,b2,F2,P,t,tp);
```

```
  eval(strcat('W',num2str(M),'1=w1;')); eval(strcat('B',num2str(M),'1=b1;'));
  eval(strcat('W',num2str(M),'2=w2;')); eval(strcat('B',num2str(M),'2=b2;'));

  % ------------------------------------------------
  %         Teste da Rede
  % ------------------------------------------------

  z=500;

  for R=1:21
    ruido(R)=5*(R-1);
    erro(M,R)=analiseff(p,w1,b1,F1,w2,b2,F2,'simuff',ruido(R),z,N);
  end

  figure(M+1)
  plot(ruido,erro(M,:),'b',ruido,erro(M,:),'b*');
  xlabel('% de Ruído'); ylabel('% de Erro');
  str=strcat('Classificações erradas x Percentagem de ruído para ',num2str(N),' harmônicos');
  title(str)
  grid on;
  clear p;

  % ------------------------------------------------
  %         Salva os Resultados da Rede
  % ------------------------------------------------

  eval(strcat('save d:\kleyton\semestre2_99\projfinal\relatórios\pesobp',num2str(M),' W',...
    num2str(M),'1 B',num2str(M),'1 W',num2str(M),'2 B',num2str(M),'2;'));

end

Comp=sum(erro')/length(erro);
figure(18)
surf(ruido,[10:25],erro); grid on;
xlabel('% de Ruido'); ylabel('Quantidade de Harmônicos'); zlabel('Erro % Médio');
title('Comparação dos Erros em vários Harmônicos')

figure(19)
plot([10:25],Comp,'b.-')
grid on; xlabel('Quantidade de Harmônicos'); ylabel('Erro % Médio');
title('Comparação dos Erros Médios em vários Harmônicos')
[min(Comp(2:16)),find(Comp==min(Comp(2:16)))+9]

termino=clock;
```

```
tempo=termino-inicio
save d:\kleyton\semestre2_99\projfinal\relatórios\resultbp tr erro tempo
```

Programa 4 – Simulação das Redes Perceptron Multicamadas

```
function incor=analiseff(p,w1,b1,F1,w2,b2,F2,f,ruido,z,N)

% analise(p,w1,b1,w2,b2,f,ruido,z,N)

% p     - matriz 48x10 de padrões possíveis

% wi,bi - Pesos da Rede Neural

% f     - Função para simular a Rede Neural

% ruido - Porcentagem de Ruido

% z     - número de amostras

% N     - número de harmônicos utilizados

% incor - numero de digitos classificaos incorretamente

% ------------------------------------------------------------------------
%       Análise da Quantidade de Classificações Incorretas
% ------------------------------------------------------------------------

path(path,'d:\kleyton\Semestre2_99\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Programas');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Imagens');

% ------------------------------------------------
%         Teste da Rede
% ------------------------------------------------

incor=0;

for j=1:z

   q=1+round(9*rand);
   eval(strcat('a = ',f,'((1+(ruido/100)*(1-2*rand))*abs(p(:,q)),w1,b1,F1,w2,b2,F2);')
   a = full(compet(a));
   a=find(a);
   incor=incor+(q~=a);
```

```
end

incor=incor*100/z;
```

Programa 5 – Treinamento e Validação das Redes Neurais de Base Radial

```
% ------------------------------------------------------------------
%          Treinamento da Rede Neural de Base Radial
% ------------------------------------------------------------------

for i=1:25
  delete(figure(i));
end
clear all, clc;

inicio=clock;
path(path,'d:\kleyton\Semestre2_99\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Programas');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Imagens');

% ------------------------------------------------
%         Criação da Rede
% ------------------------------------------------

df = 10;                    % Epochs between updating display
mn = 25;                    % Maximum number of neurons
eg = 2e-4;                      % Sum-squared error goal
sc = 1500.0;                    % Spread

dp=[df mn eg sc];


t=eye(10);
t=[t t t t];
t=[t t t t t];

% ------------------------------------------------
%         Preparação dos Dados
% ------------------------------------------------

Ruido=75/100;

For M=1:16
```

```
N = M+9;        % Quantidade de harmônicos utilizados

Load('d:\kleyton\semestre2_99\projfinal\imagens\banco_caixa');
p(:,1)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
load('d:\kleyton\semestre2_99\projfinal\imagens\banco_esfera');
p(:,2)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
load('d:\kleyton\semestre2_99\projfinal\imagens\banco_cone');
p(:,3)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
load('d:\kleyton\semestre2_99\projfinal\imagens\banco_hex');
p(:,4)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
load('d:\kleyton\semestre2_99\projfinal\imagens\banco_estrela');
p(:,5)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
load('d:\kleyton\semestre2_99\projfinal\imagens\banco_davi');
p(:,6)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
load('d:\kleyton\semestre2_99\projfinal\imagens\banco_martelo');
p(:,7)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
load('d:\kleyton\semestre2_99\projfinal\imagens\banco_robo');
p(:,8)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
load('d:\kleyton\semestre2_99\projfinal\imagens\banco_caneta');
p(:,9)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';
load('d:\kleyton\semestre2_99\projfinal\imagens\banco_tesoura');
p(:,10)=[Axx(1:N),Bxx(1:N),Cxx(1:N),Dxx(1:N)]';

% ------------------------------------------------
%       Treinamento da Rede
% ------------------------------------------------

p1=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];
p2=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];
p3=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];
p4=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];
p5=[p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p (1+Ruido*(1-2*rand))*p];

p=[p1 p2 p3 p4 p5]; P=abs(p);

figure(1)
[w1,b1,w2,b2,nr,dr]=solverb(P,t,dp);

eval(strcat('W',num2str(M),'1=w1;'));   eval(strcat('B',num2str(M),'1=b1;'));
eval(strcat('W',num2str(M),'2=w2;'));   eval(strcat('B',num2str(M),'2=b2;'));

NR(M)=nr

% ------------------------------------------------
%       Teste da Rede
```

```
    % ------------------------------------------------

    z=500;

    for R=1:21
      ruido(R)=5*(R-1);
      erro(M,R)=analise(p,w1,b1,w2,b2,'simurb',ruido(R),z,N);
    end

    figure(M+1)
    plot(ruido,erro(M,:),'b',ruido,erro(M,:),'b*');
    xlabel('% de Ruído'); ylabel('% de Erro');
    str=strcat('Classificações erradas x Percentagem de ruído para ',num2str(N),' harmônicos');
    title(str)
    grid on;
    clear p;

    % ------------------------------------------------
    %         Salva os Resultados da Rede
    % ------------------------------------------------

    eval(strcat('save d:\kleyton\semestre2_99\projfinal\relatórios\pesorb',num2str(M),' W',...
      num2str(M),'1 B',num2str(M),'1 W',num2str(M),'2 B',num2str(M),'2;'));

  end

  Comp=sum(erro')/length(erro);
  Figure(18)
  surf(ruido,[10:25],erro); grid on;
  xlabel('% de Ruido'); ylabel('Quantidade de Harmônicos'); zlabel('Erro % Médio');
  title('Comparação dos Erros em vários Harmônicos')

  figure(19)
  plot([10:25],Comp,'b.-')
  grid on; xlabel('Quantidade de Harmônicos'); ylabel('Erro % Médio');
  title('Comparação dos Erros Médios em vários Harmônicos')
  [min(Comp(2:16)),find(Comp==min(Comp(2:16)))+9]


  Termino=clock;

  tempo=termino-inicio
  save d:\kleyton\semestre2_99\projfinal\relatórios\resultrb NR erro tempo
```

Programa 6 – Simulação das Redes Neurais de Base Radial

```
function incor=analise(p,w1,b1,w2,b2,f,ruido,z,N)

% analise(p,w1,b1,w2,b2,f,ruido,z,N)

% p     - matriz 48x10 de padrões possíveis

% wi,bi - Pesos da Rede Neural

% f     - Função para simular a Rede Neural

% ruido - Porcentagem de Ruido

% z     - número de amostras

% N     - número de harmônicos utilizados

% incor - numero de digitos classificaos incorretamente

% ------------------------------------------------------------------------
%       Análise da Quantidade de Classificações Incorretas
% ------------------------------------------------------------------------

path(path,'d:\kleyton\Semestre2_99\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Programas');
path(path,'d:\kleyton\Semestre2_99\ProjFinal\Imagens');

% ------------------------------------------------------
%         Teste da Rede
% ------------------------------------------------------

incor=0;

for j=1:z

   q=1+round(9*rand);
   eval(strcat('a = ',f,'((1+(ruido/100)*(1-2*rand))*abs(p(:,q)),w1,b1,w2,b2);'))
   a = full(compet(a));
   a=find(a);

   incor=incor+(q~=a);
end
incor=incor*100/z;
```