SHEILA SOUZA

Sistema de reconhecimento de caracteres numéricos manuscritos baseado nas redes neurais artificiais paraconsistentes

Dissertação apresentada à Faculdade de Medicina da Universidade de São Paulo, para obtenção do título de Mestre em Ciências

Programa de: Fisiopatologia Experimental
Orientador: Prof. Dr. Eduardo Massad

(Versão corrigida. Resolução CoPGr 5890, de 20 de dezembro de 2010.
A versão original está disponível na Biblioteca FMUSP)

São Paulo
2013

Dados Internacionais de Catalogação na Publicação (CIP)
Preparada pela Biblioteca da
Faculdade de Medicina da Universidade de São Paulo

©reprodução autorizada pelo autor

Souza, Sheila
Sistema de reconhecimento de caracteres numéricos manuscritos baseado nas redes neurais artificiais paraconsistentes / Sheila Souza. -- São Paulo, 2013.
Dissertação(mestrado)--Faculdade de Medicina da Universidade de São Paulo.
Programa de Fisiopatologia Experimental.
Orientador: Eduardo Massad.

Descritores: 1.Lógica 2.Inteligência artificial 3.Reconhecimento automatizado de padrão 4.Escrita manual 5.Redes neurais (computação) 6.Processamento de imagem assistida por computador 7.Sistemas de computação

USP/FM/DBD-331/13

Dedico este trabalho aos meus pais com amor, admiração e gratidão pela compreensão, carinho, presença e incansável apoio que tem me oferecido por toda minha vida.

AGRADECIMENTOS

Nesta oportunidade desejo expressar gratidão às seguintes pessoas que contribuíram para a feitura desta dissertação:

Inicialmente a minha família por todo apoio, carinho e compreensão.

Aos colegas Cauê Thenório, David Rios, Eduardo Sorpreso, Euridan Costa, Flávio Cordeiro, Henzo Otsuki, Juca Machado, Marcio Barreto, Mario Mania, Priscila Ghirardello, Rafaela Netto, Renato Bonadio, Thiago Malaquias, Thiago Padula e Viviane Rodrigues pela disponibilidade dos caracteres manuscritos.

Aos amigos e colegas Prof. Adenilso Josias Fogaça e Prof. Dr. João Inácio da Silva Filho pelo apoio.

Ao Prof. Euro de Barros Couto Júnior pela amizade, apoio, esclarecimentos e ensinamentos da curva ROC que foram úteis para o trabalho.

Ao Prof. Dr. Cláudio Rodrigo Torres pela amizade, apoio, insights e discussões importantes para a utilização das redes neurais artificiais paraconsistentes.

Aos integrantes e participantes do Grupo de Lógica e Teoria da Ciência do IEA-USP que contribuíram com sugestões de aperfeiçoamento para o meu trabalho.

Aos funcionários da FMUSP, Iracene Aparecida Boccia, Suely Campos Cardoso, Tânia Souza, Valéria de Vilhena Lombarde e Vilma Sanches pelo solícito atendimento que sempre me dispensou.

Ao meu orientador Prof. Dr. Eduardo Massad pela sugestão do tema e encorajamento para a confecção deste trabalho.

Ao Prof. Dr. Jair Minoro Abe expresso meus especiais agradecimentos. Sua orientação segura, discussões inovadoras, acompanhamento do meu trabalho, amizade e estímulo constante fez com que pudesse terminar o meu projeto, bem como a confiança em mim depositada e o reconhecimento do meu esforço.

A todos que, diretamente ou indiretamente, contribuíram para a realização deste trabalho.

"Mais do que máquinas, precisamos de humanidade.
Mais do que inteligência, precisamos de afeição e doçura"
Charles Chaplin

Esta dissertação ou tese está de acordo com as seguintes normas, em vigor no momento desta publicação:

Referências: adaptado de *International Committee of Medical Journals Editors* (Vancouver). Universidade de São Paulo. Faculdade de Medicina. Divisão de Biblioteca e Documentação. *Guia de apresentação de dissertações, teses e monografias*. Elaborado por Anneliese Carneiro da Cunha, Maria Julia de A. L. Freddi, Maria F. Crestana, Marinalva de Souza Aragão, Suely Campos Cardoso, Valéria Vilhena. 3a ed. São Paulo: Divisão de Biblioteca e Documentação; 2011.

Abreviaturas dos títulos dos periódicos de acordo com *List of Journals Indexed in Index Medicus*.

SUMÁRIO

Lista de abreviaturas, símbolos e siglas
Lista de figuras, tabelas
Resumo
Summary

1 INTRODUÇÃO 1
1.1 Objetivo 1
1.2 Inteligência Artificial (IA) 2
1.2.1 Utilização da IA na automatização do processo de reconhecimento de padrões 3
1.3 Reconhecimento de caracteres numéricos 3
1.4 Grafismo 5
1.5 Reconhecimento de padrões na área médica 6

2 REVISÃO DA LITERATURA 8
2.1 Rede Neural Artificial 8
2.2 Lógica Paraconsistente 11
2.2.1 Lógicas Paraconsistentes Anotadas Qτ 13
2.2.2 Lógica Paraconsistente Anotada Evidencial Eτ 14
2.3 Rede Neural Artificial Paraconsistente 19

3 MÉTODOS 23
3.1 Mapeamento da imagem em graus de evidência 27
3.2 Extração de características 35
3.3 Reconhecimento da Imagem 46

4 RESULTADOS 52
5 DISCUSSÃO 59
6 CONCLUSÃO 64
7 REFERÊNCIAS 65
APÊNDICES
Apêndice A - Testes realizados com caracteres do tipo CMC-7
Apêndice B - Testes realizados com caracteres do tipo numérico manuscrito
Apêndice C - Testes realizados com caracteres do tipo alfabético
Apêndice D - Curva ROC
Apêndice E - Algoritmos das células neurais artificiais paraconsistentes utilizados no desenvolvimento do sistema
Apêndice F - Algoritmo do Nó de Análise Paraconsistente e Algoritmo $ParaExtr_{ctr}$
Apêndice G - Código-fonte do Sistema Computacional Paraconsistente

LISTA DE FIGURAS

Figura 1 - Diagrama das abordagens do Grafismo, Grafoscopia e Grafologia 6
Figura 2 - Neurônio biológico 8
Figura 3 - Rede Neural Artificial – esquema geral 10
Figura 4 - Modelo simplificado de neurônio artificial de McCulloch e Pitts 11
Figura 5 - Representação do reticulado associado à Lógica Paraconsistente Anotada Evidência Eτ 16
Figura 6 - Representação do Reticulado τ com os valores-limite e dos graus de certeza, incerteza e estados de decisão 18
Figura 7 - Representação de um sistema de análise paraconsistente 19
Figura 8 - Célula Neural Artificial Paraconsistente analítica real (CNAPar) 21
Figura 9 - Célula Neural Artificial Paraconsistente de detecção de igualdade (CNAPdi) 21
Figura 10 - Células Neurais Artificiais Paraconsistentes de conexão lógica de maximização e de minimização (CNAPcls Max e CNAPcls Min) 22
Figura 11 - Célula Neural Artificial Paraconsistente de passagem (CNAPpa) 22
Figura 12 - Estrutura dos campos de um cheque bancário brasileiro 24
Figura 13 - Exemplo de linha MICR CMC-7 24
Figura 14 - Fases para o processo de reconhecimento 25
Figura 15 - Exemplo dos tratamentos adotados na fase de pré-processamento da imagem 26
Figura 16 - Arquitetura do Sistema Computacional Paraconsistente 26
Figura 17 - Tipos de mapeamento da imagem 28
Figura 18 - Mapeamento da borda externa da imagem em graus de evidência 30
Figura 19 - Cortes diagonais internos superior e inferior 31
Figura 20 - Arquitetura da Unidade das Linhas Verticais e Horizontais (ULVH) . 37
Figura 21 - Arquitetura da Unidade das Células da Primeira Camada (UCPC) 38
Figura 22 - Arquitetura da Unidade das Células das Camadas Finais (UCCF) 39
Figura 23 - Exemplos dos casos especiais tratados pela UCCF 39
Figura 24 - Exemplo de segmentos de reta verticais e horizontais obtidos de uma imagem 41
Figura 25 - Exemplo de duas UCPC referente ao lado esquerdo e topo da imagem com os valores obtidos nas camadas de uma ULVH 43
Figura 26 - Exemplo dos histogramas obtidos no processo de extração de características 45
Figura 27 - Conjunto de caracteres dos tipos CMC-7 e numérico manuscrito adotados com caracteres padrão 46
Figura 28 - Processo de reconhecimento das características do caractere apresentado 47

Figura 29 - Arquitetura da Unidade de Agrupamento do Reconhecimento de Linhas (UARL) 47
Figura 30 - Arquitetura da Unidade de Reconhecimento de Padrões (URP) 48
Figura 31 - Representação simbólica de um Nó de Análise Paraconsistente (NAP) 49
Figura 32 - Algoritmo $ParaExtr_{ctr}$ 49
Figura 33 - Esquematização do algoritmo $ParaExtr_{ctr}$..... 51
Figura 34 - Visão geral do Sistema Computacional Paraconsistente 59

LISTA DE TABELAS

Tabela 1 - Testes realizados pelo sistema configurado com as características "Borda Externa", "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" para o reconhecimentos de caracteres do tipo CMC-7 52

Tabela 2 - Testes realizados pelo sistema configurado com as características "Borda Externa", "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" para o reconhecimentos de caracteres do tipo Manuscrito 53

Tabela 3 - Testes realizados pelo sistema configurado com as características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo CMC-7 54

Tabela 4 - Testes realizados pelo sistema configurado com as características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo CMC-7 55

Tabela 5 - Testes realizados pelo sistema configurado com as características "Histograma: Borda Externa", "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo numérico manuscrito 55

Tabela 6 - Testes realizados pelo sistema configurado com os picos superiores e inferiores das características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo CMC-7 56

Tabela 7 - Testes realizados pelo sistema configurado com os picos superiores e inferiores das características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo CMC-7 57

Tabela 8 - Testes realizados pelo sistema configurado com os picos superiores e inferiores das características "Histograma: Borda Externa", "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo numérico manuscrito 57

Tabela 9 - Possíveis resultados de um teste 62

LISTA DE ABREVIATURAS

GeC	Grau de evidência do caractere apresentado
GeCaractere	Lista de graus de evidência do caractere apresentado
GeIntervalo	Lista de graus de evidência resultante do cálculo do intervalo entre as listas GePadrão e GeCaractere
GeP	Grau de evidência do caractere padrão
GePadrão	Lista de graus de evidência do caractere padrão
$ParaExtr_{ctr}$	Algoritmo Paraconsistente Extrator de Efeitos da Contradição

LISTA DE SÍMBOLOS

AR	Saída de uma CNAP Analítica Real (Grau de evidência resultante real)
C	Complemento (1 - μ_{1B})
DI	Saída de uma CNAP de Detecção de Igualdade (0 ou 1)
Ft_c	Fator de Tolerância à Certeza
Ft_{ct}	Fator de Tolerância à Contradição
G_{ce}	Grau de certeza
G_{in}	Grau de incerteza
$QV \rightarrow T$	Quase-verdadeiro tendendo ao Inconsistente
$QV \rightarrow \perp$	Quase-verdadeiro tendendo ao Paracompleto
$QF \rightarrow T$	Quase-falso tendendo ao Inconsistente
$QF \rightarrow \perp$	Quase-falso tendendo ao Paracompleto
$QT \rightarrow V$	Quase-inconsistente tendendo ao Verdadeiro
$QT \rightarrow F$	Quase-inconsistente tendendo ao Falso
$Q\perp \rightarrow V$	Quase-paracompleto tendendo ao Verdadeiro
$Q\perp \rightarrow F$	Quase-paracompleto tendendo ao Falso
V_{icc}	Valor inferior de controle de certeza
V_{icin}	Valor inferior de controle de incerteza
V_{scc}	Valor superior de controle de certeza
V_{scin}	Valor superior de controle de incerteza
λ	Evidência desfavorável ou contrária
μ	Evidência favorável
μ_{1A}	Primeira entrada da CNAP
μ_{1B}	Segunda entrada da CNAP
μ_{1Max}	Valor máximo da entrada da CNAP
μ_{1Min}	Valor mínimo da entrada da CNAP
μ_E	Grau de Evidência Resultante
μ_{ER}	Grau de Evidência Resultante Real
μ_{max}	Grau de evidência de maior valor
μ_{min}	Grau de evidência de menor valor
φ	Intervalo de Evidência Resultante
φ_E	Intervalo de Evidência

LISTA DE SIGLAS

CMC-7	Caracteres Magnéticos Codificado em 7 Barras
CNAP	Célula Neural Artificial Paraconsistente
CNAPa	Célula Neural Artificial Paraconsistente Analítica
CNAPar	Célula Neural Artificial Paraconsistente Analítica Real
CNAPcls	Célula Neural Artificial Paraconsistente de Conexão Lógica Simples de Maximização ou Minimização
CNAPdi	Célula Neural Artificial Paraconsistente de Detecção de Igualdade
CNAPpa	Célula Neural Artificial Paraconsistente de Passagem
HC-FMUSP	Hospital da Clínicas da Faculdade de Medicina da Universidade de São Paulo
IA	Inteligência Artificial
ICR	Intelligent Character Recognition
INCOR	Instituto do Coração
Lógica Eτ	Lógica Paraconsistente Anotada Evidencial Eτ
MICR	Magnetic Ink Character Recognition
NAP	Nó de Análise Paraconsistente
OCR	Optical Character Recognition
$ParaExtr_{ctr}$	Algoritmo Paraconsistente Extrator de Efeitos da Contradição
PEP	Prontuário Eletrônico do Paciente
RNA	Rede Neural Artificial
RNAP	Rede Neural Artificial Paraconsistente
UAPParaExtr	Unidade de Análise Paraconsistente $ParaExtr_{ctr}$
UARL	Unidade de Agrupamento do Reconhecimento das Linhas
UCCF	Unidade das Células das Camadas Finais
UCPC	Unidade das Células da Primeira Camada
ULVH	Unidade das Linhas Verticais e Horizontais
UNAP	Unidade Neural Artificial Paraconsistente
URP	Unidade de Reconhecimento de Padrões

RESUMO

Souza S. *Sistema de reconhecimento de caracteres numéricos manuscritos baseado nas redes neurais artificiais paraconsistentes* [Dissertação]. São Paulo: Faculdade de Medicina, Universidade de São Paulo; 2013.

O reconhecimento de padrões por computador é uma das mais importantes ferramentas da Inteligência Artificial presente em inúmeras áreas do conhecimento com aplicações em diversos setores, incluindo o reconhecimento de caracteres. O objetivo da dissertação se concentra na investigação de um processo computacional automatizado - Sistema Computacional Paraconsistente - capaz de reconhecer Caracteres Numéricos Manuscritos e Caracteres Magnéticos Codificados em 7 Barras utilizados em cheques bancários brasileiros, fornecendo uma fundamentação técnica para reconhecer documentos e imagens digitalizadas e, também, sinais biológicos. Embora haja vários estudos em reconhecimento de caracteres, optou-se pelo estudo desse tema devido à sua intrínseca importância e constante desenvolvimento, além de possibilitar adaptações para fazer o reconhecimento de diferentes tipos de sinais como, por exemplo, sinais biológicos. A metodologia adotada para essa tarefa se baseia nas Redes Neurais Artificiais Paraconsistentes por se tratar de uma ferramenta com capacidade de trabalhar com dados imprecisos, inconsistentes e paracompletos sem o perigo de trivialização. O processo de reconhecimento desse sistema é realizado a partir de algumas características do caractere previamente selecionadas com base em algumas técnicas do Grafismo e realiza-se a análise dessas características bem como o reconhecimento do caractere através das Redes Neurais Artificiais Paraconsistentes O sistema foi construído para reconhecer caracteres numéricos com um padrão previamente definido, onde adotou-se os Caracteres Magnéticos Codificados em 7 Barras utilizados em cheques bancários e, posteriormente, o sistema foi aperfeiçoado para fazer o reconhecimento de Caracteres Numéricos Manuscritos. Para a validação do estudo proposto apresentou-se dados reais, a saber, lotes de cheques e caracteres numéricos manuscritos digitalizados onde o sistema apresentou 97,85% de acertos para os Caracteres Magnéticos Codificados em 7 Barras e 91,62% de acertos para Caracteres Numéricos Manuscritos. O resultado obtido demonstra que o sistema é robusto o suficiente e pode servir de estudo para a análise de sinais em áreas correlatas com nível de precisão semelhante.

Descritores: Lógica; Inteligência artificial; Reconhecimento automatizado de padrão; Escrita manual; Redes neurais (computação); Processamento de imagem assistida por computador; Sistemas de computação.

ABSTRACT

Souza S. *Handwritten numeric character recognition system based on paraconsistent artificial neural network* [Dissertation]. São Paulo: Faculdade de Medicina, Universidade de São Paulo; 2013.

Computer pattern recognition is one of the most important Artificial Intelligence tools present in numerous knowledge areas with applications in several themes, including the character recognition. The aim of this dissertation is the investigation of an automated computational process - Paraconsistent Computational System - able to recognize Handwritten Numeric Characters and Magnetic Ink Character Recognition used on Brazilian bank checks furnishing a technical basis to recognize digital documents, digital images and biological signals. Although there are several studies on character recognition, it was chosen to study this theme due to its intrinsic importance and constant improvement, besides enabling adjustments to the recognition of different kinds of signals such as, biological signals. The methodology employed for the task is based on Paraconsistent Artificial Neural Networks for being a tool with the ability to work with imprecise, inconsistent and paracomplete data without trivialization. The recognition process of this system is performed from some previously selected character features based on some Graphics techniques and, it performs the analysis of these features as well as the character recognition are performed through the Paraconsistent Artificial Neural Networks. The system was built to recognize numeric characters with a previously defined pattern where it was chosen the Magnetic Ink Character Recognition used on Brazilian bank checks and then the system was improved to recognize handwritten numeric characters. It was presented real data as checks' batches and scanned handwritten numeric characters to validate the proposed study and the system reached 97.85% hits for Magnetic Ink Character Recognition and 91.62% hits for Handwritten Numeric Characters. The obtained result demonstrates that the system is robust enough for signal analysis study in correlated areas with similar precision level.

Descriptors: Logic; Artificial intelligence; Pattern recognition, automated; Handwriting; Neural networks (computer); Image processing, computer-assisted; Computer systems.

1 INTRODUÇÃO

Neste trabalho apresenta-se um sistema computacional automatizado para reconhecimento de caracteres numéricos manuscritos e caracteres numéricos com um padrão previamente determinado denominado Sistema Computacional Paraconsitente.

O reconhecimento é realizado a partir de algumas características previamente selecionadas de cada caractere com base em algumas técnicas do Grafismo como valores curvilíneos, altura, largura e direção.

Dentre as várias técnicas da Inteligência Artificial optamos por trabalhar com a Rede Neural Artificial Paraconsistente (RNAP) pela sua capacidade de tratar dados imprecisos, inconsistentes e paracompletos.

Nessa pesquisa sobre reconhecimento de padrões optou-se pelo estudo de reconhecimento de caracteres visto que, embora haja vários estudos nesse ramo, essa área encontra-se em constante desenvolvimento, além de possibilitar adaptações para fazer o reconhecimento de diferentes tipos de sinais em diversas áreas de conhecimento.

1.1 Objetivo

O objetivo desta dissertação se concentra na investigação de um processo computacional automatizado de reconhecimento de caracteres numéricos manuscritos, caracteres com um padrão previamente determinado como Caracteres Magnéticos Codificados em 7 Barras (CMC-7) utilizados em cheques bancários e, impressos digitalizados. Esse estudo possui relevância imediata no estudo de sistemas computacionais de análise e reconhecimento de informações, relevantes na área médica e sinais biológicos.

Segundo Dedgaonkar et al. (2012) existem vários métodos disponíveis para reconhecimento de caracteres como: Clustering (agrupamento automático de dado conforme grau de semelhança), Extração de Características, Pattern Matching (análise da forma do caractere e comparação de suas características) e Rede Neural Artificial (RNA).

Neste trabalho optou-se pela utilização de uma RNA especial, a RNAP, por ser uma rede capaz de tratar dados imprecisos, contraditórios e paracompletos sem o perigo de trivialização. Também a escolha por tal rede reside no fato de as RNAPs terem sido empregadas com êxito em algumas aplicações com as características apontadas (Lopes, 2009; Mário, 2006; Prado, 1996; Prado, 2007; Amaral, 2013).

Para a finalidade do trabalho, utilizou-se uma arquitetura conveniente assentada nas RNAPs e, apresenta-se aplicação com dados reais para sua validação.

1.2 Inteligência Artificial (IA)

As técnicas de IA podem ser definidas como métodos que visam explorar o conhecimento a ser representado de modo que: capture generalizações; seja compreendido por pessoas que necessitam supri-lo; seja facilmente modificável; seja utilizado em várias situações, mesmo quando não totalmente preciso ou completo e; seja utilizado para superar seu próprio volume ao ajudar estreitar a faixa de possibilidades. (Rich; Knight, 1993)

Um sistema de IA possui três componentes fundamentais: representação, que armazena o conhecimento; raciocínio, que aplica o conhecimento armazenado em resolução de problemas e; aprendizagem, que a partir de experiências adquiri novos conhecimentos (Kato et al., 1999; Luger, 2004; De Mendonça, 2008).

Alguns problemas que podem ser destacados dentro do escopo da IA são: jogos, provas de teoremas, resolução de problemas gerais, percepção (visão e fala), compreensão da linguagem natural, resolução de problemas especializados (matemática simbólica, diagnose médica, análise química, projeto de engenharia) etc. (Rich; Knight, 1993).

Os métodos utilizados pela IA, usualmente denominados “inteligentes”, são amplamente utilizados em reconhecimento de padrões (Silva, 2006).

1.2.1 Utilização da IA na automatização do processo de reconhecimento de padrões

Para os propósitos do estudo a seguir, um padrão é uma entidade que possui características simbólicas ou numéricas que podem ser extraídas (Silva, 2006).

O objetivo das técnicas computacionais de reconhecimento de padrões é agrupar padrões que compartilham determinadas semelhanças de uma maneira eficiente (Silva, 2006).

A naturalidade inerente do ser humano nas habilidades de reconhecimento de padrões como som, formas, imagens, além de outras, abstraem as complexidades cognitivas que estão por trás delas. Ao tentar reproduzir tais habilidades artificialmente em um computador essas complexidades se tornam evidentes desafiando assim a comunidade científica dessa área (Silva, 2006).

Segundo Silva (2006), o reconhecimento de padrões por computador é uma das mais importantes ferramentas da IA que está presente em inúmeras áreas do conhecimento e encontra aplicações diretas em visão computacional, reconhecimento de faces, identificação de digitais, reconhecimento de caracteres, entre outras.

1.3 Reconhecimento de caracteres numéricos

O reconhecimento de caracteres, uma das mais conhecidas e exploradas modalidades de reconhecimento de padrões, consiste em características extraídas de um conjunto de caracteres e seu objetivo é tentar reproduzir a capacidade humana de ler textos (Mantas, 1986; Silva et al., 2001; Ferrara, 2003).

Essa tecnologia pode ser dividida em dois tipos: OCR (Optical Character Recognition) e ICR (Intelligent Character Recognition). (Duneau, 1994; Hertz et al., 1991).

OCR é uma tecnologia que faz o reconhecimento óptico de caracteres, o que permite o reconhecimento de caracteres de texto em imagens (De Mendonça, 2008).

A funcionalidade e o ramo de atividade de um sistema OCR podem ser identificados pelo seu algoritmo. Segundo De Mendonça (2008) e Mantas (1986) algumas funcionalidades são:

- Edição de texto impresso;
- Tradução de texto;
- Segmentação de caracteres e extração de características;
- Reconhecimento de textos manuscritos.

ICR é um tipo de OCR avançado que faz uso de RNAs para o reconhecimento de caracteres impressos, diferentes formas de escritas manuscritas e código de barras. É utilizado para reconhecer dados de formulários impressos ou desenhados através de ferramentas de desenho (De Mendonça, 2008).

O funcionamento básico dos sistemas que utilizam essas tecnologias é descrito a seguir: receber a imagem; processar de forma que possa entendê-la; comparar com um padrão e; aplicar um processo de reconhecimento. Assim, tenta-se criar dispositivos e algoritmos capazes de localizar objetos, classificar padrões e detectar relações entre eles (Ferrara, 2003).

O objetivo da técnica de reconhecimento dos caracteres é classificar as palavras a partir das informações geradas pela extração das suas características. Em sistemas de reconhecimento de padrões a maior dificuldade está em determinar o conjunto de características passíveis de extração devido à interferência de erros e/ou ruídos (Ferrara, 2003).

O desempenho de um sistema automático de reconhecimento depende fundamentalmente da qualidade dos documentos nas suas formas original e digital. Após a captura e digitalização da imagem alguns processos como realces, remoção de linhas, de sublinhados e de ruídos, além de outros, são necessários para compensar uma qualidade nos originais e nas imagens digitalizadas. No tratamento da imagem e do texto, os problemas geralmente relacionados são: ruído, distorção, variação de estilo, translação, escala, rotação, texturas e traços (Silva et al., 2001; De Mendonça, 2008; Suen et al., 1980; Engelbrecht, 2007).

Para o reconhecimento de caracteres utilizam-se técnicas de reconhecimento de padrões, entre elas, as mais comuns são: estatísticas (teoria de decisão), sintáticas (estruturais) e baseadas em RNAs (Ferrara, 2003).

Dentre as etapas do processo de reconhecimento a segmentação de caracteres depende da forma, padrão e características de como as palavras foram escritas. Para a

segmentação de caracteres não conectados, o método baseado em histogramas de projeção é bastante utilizado (Duda; Hart, 1973; De Mendonça, 2008).

Para reconhecer letras e números, manuscritos e cursivos são utilizadas técnicas de RNA, que se preocupam mais em reconhecer elementos genéricos de palavras e padrões de “loops” do que na classificação individual dos caracteres. (Suen et al., 1980; De Mendonça, 2008).

Nesse projeto optou-se pelas RNAPs para o processo de reconhecimento de caracteres numéricos por ter a capacidade de trabalhar com dados imprecisos, inconsistentes e paracompletos (Abe, 1992). Por se tratar de caracteres numéricos digitalizados, eles estão sujeitos a diversos tipos de ruídos que, possivelmente, podem gerar dados inconsistentes. Assim, espera-se que as RNAPs tenham um melhor desempenho do que as RNAs usuais.

Por se tratar de uma nova teoria escrevemos no capítulo 2 um resumo dos principais conceitos da RNAP, desde a sua fundamentação originada das Lógicas Paraconsistentes Anotadas.

1.4 Grafismo

O Grafismo possui vários tratamentos, dentre eles, a Grafoscopia e a Grafologia. Tais tratamentos são importantes para a definição dos laudos dos peritos da área criminalista, que são responsáveis pela análise confiável de assinaturas manuscritas (Justino, 2001, 2002; Santos, 2004; Gomide, 2005).

A Grafoscopia, desenvolvida para esclarecer questões criminais, é conceituada como uma área responsável pela verificação da autenticidade de um documento que determina as características gráficas na elaboração do mesmo (Gomide, 2005).

A Grafologia permite diferenciar um autor do outro através das características que determinam os aspectos psíquicos do autor.

As técnicas utilizadas para exame das identidades gráficas da Grafoscopia e Grafologia podem ser visualizadas na Figura 1.

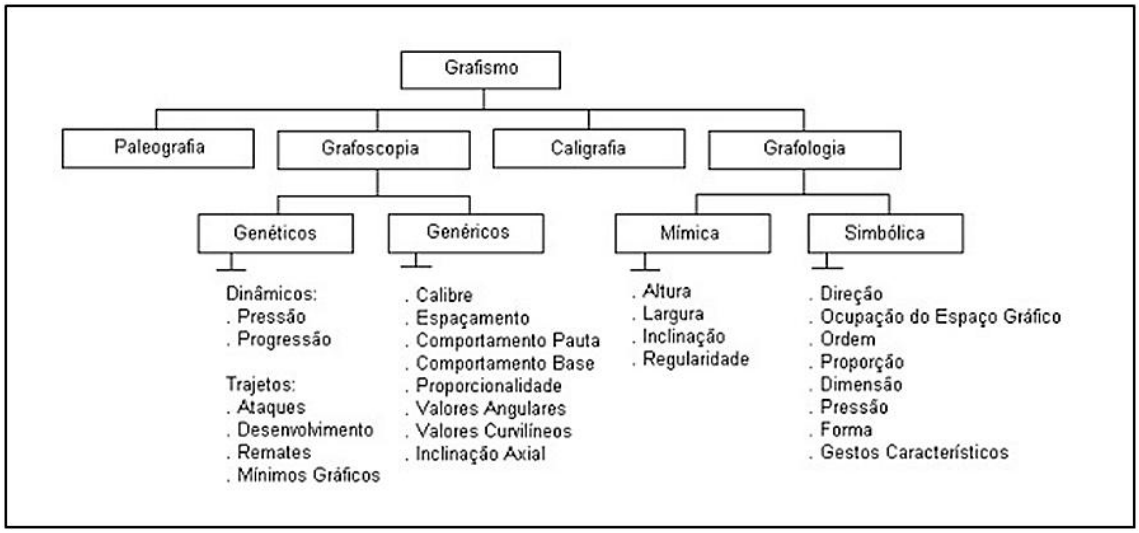


Figura 1 - Diagrama das abordagens do Grafismo, Grafoscopia e Grafologia (Santos, 2004)

Os sistemas de reconhecimento de assinatura, por exemplo, tem evoluído através do estudo e simulação dos critérios adotados por peritos da área criminalista utilizando técnicas de computação e processamento de imagens.

Para a realização deste trabalho o processo de extração de características do caractere numérico, a princípio, será realizado com base nos fundamentos dessa técnica e, que será utilizada de modo a ser tratada pelas RNAPs.

1.5 Reconhecimento de padrões na área médica

Atualmente o Prontuário Eletrônico do Paciente (PEP) tem se apresentado como uma importante ferramenta que aperfeiçoou o cadastro do paciente, a leitura e interpretação dos pedidos de exames, agilizou a conduta do médico, além de reduzir o volume de papeis e ligações telefônicas entre médicos e laboratoristas para esclarecer dúvidas (Massad et al., 2003; Jatene et al., 2012).

A técnica de Reconhecimento de Padrões pode ser aplicada em alguns ramos da área médica como sistemas administrativos de relatórios médicos por código de barras, onde um paciente pode ser identificado pelo código de barras. Ultimamente há alguns estudos para implantação do PEP em dispositivos móveis com o intuito de minimizar o

número de erros referentes à interpretação equivocada de prescrição médica e tornar as informações completas do paciente mais acessíveis, de forma ágil e atualizada, aos profissionais da saúde (Murakami et al., 2004; Martha et al., 2006). Uma das soluções estudadas pelo Instituto do Coração (INCOR), por exemplo, utiliza códigos de barra para identificar os leitos e os pacientes e, através de dispositivos móveis com aplicativos capazes de ler códigos de barras, os médicos e enfermeiras podem rapidamente identificar o paciente e ter acesso ao prontuário completo do mesmo (Murakami et al., 2004). Nesse caso o estudo proposto nesse trabalho tem aplicação no processo de reconhecimento do código de barras.

Segundo Lira (apud Massad et al., 2003) os novos Sistemas de Informação Laboratorial utilizam um protocolo específico nos tubos de coleta para fazer a devida identificação do mesmo e, posteriormente, faz a leitura desse protocolo para identificar o material coletado nos demais processos para elaboração do laudo do exame. Esse poderia ser um outro ramo de aplicação do estudo proposto para o reconhecimento desse protocolo específico.

Segundo Tachinardi e Furuie (apud Massad et al., 2003) documentos manuscritos são arquivados no PEP como imagens digitalizadas. O reconhecimento da informação contida nesses manuscritos também seria uma aplicação do estudo proposto nesse trabalho.

Aplicações futuras podem ser direcionadas para compressão, encriptação e decriptação de dados. Compressão devido ao alto volume dados armazenados no PEP diariamente e, encriptação e decriptação devido a necessidade de manter a segurança e o sigilo das informações conforme necessidade.

2 REVISÃO DA LITERATURA

2.1 Rede Neural Artificial

O cérebro humano possui aproximadamente 10 bilhões de neurônios formando uma gigantesca rede neural capaz de processar informações através das conexões existentes entre os neurônios, conhecidas como sinapses. Cada neurônio realiza entre mil a dez mil sinapses.

Grosso modo, um neurônio biológico compõe-se por um corpo celular denominado soma e por diversas ramificações. Essas ramificações, conhecidas como dendritos, conduzem os sinais das extremidades para o corpo. Existe também uma ramificação, geralmente única, denominada axônio, responsável por conduzir os sinais para outros neurônios através das sinapses (Figura 2) (Haykin, 1994).

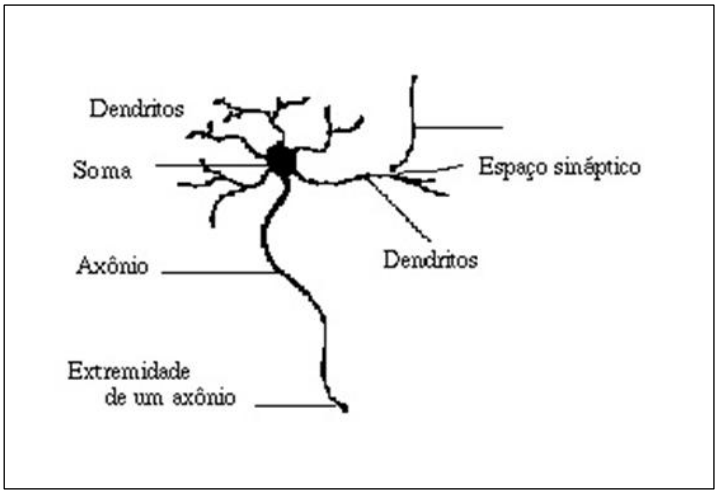


Figura 2 - Neurônio biológico

A teoria das RNAs são paradigmas computacionais baseados em modelos matemáticos que a despeito das computações tradicionais possui uma estrutura e operação que, grosso modo, tentam “imitar” o funcionamento do neurônio biológico humano.

Tais redes também são conhecidas como sistemas conexionistas, sistemas paralelos distribuídos ou sistemas adaptativos, pois são compostos por uma série de elementos de processamento interconectados que operam em paralelo.

Um dos objetivos originais das RNAs é entender e modelar as características funcionais e propriedades computacionais do cérebro quando executa processos cognitivos tais como percepção sensorial, categorização de conceitos, associação de conceitos e aprendizado. Entretanto, hodiernamente um grande esforço está direcionado nas aplicações que envolvem reconhecimento e classificação de padrões, compressão de dados e otimização.

Uma RNA genérica pode ser definida como um sistema computacional consistindo de um conjunto de elementos de processamento altamente interconectados, denominados neurônios artificiais, o qual processa uma informação como resposta a um estímulo externo. Um neurônio artificial é uma representação simplificada que emula a integração de sinais e dispara no limiar comportamento dos neurônios biológicos por meio de equações matemáticas.

Como em sua contraparte biológica, neurônios artificiais são limitados conjuntamente por conexões que determinam o fluxo de informações entre os neurônios. Estímulos são transmitidos de um processamento a outro via sinapses ou interconexões, que podem ser excitatórios ou inibitivos. Se o *input* a um neurônio é excitatório, é mais provável que este neurônio transmita um sinal excitatório aos outros neurônios a ele conectados. Um *input* inibitório provavelmente se propagará como inibitório (Haykin, 1994).

RNAs dispõem-se tipicamente em “camadas” (Figura 3). Cada camada em uma rede neural constitui um arranjo de elementos processadores ou neurônios. Informações fluem através de cada elemento na forma de *input-output*. Em outras palavras, cada elemento recebe um sinal de *input*, manipula-o e encaminha um sinal de output a outros elementos conectados na camada adjacente.

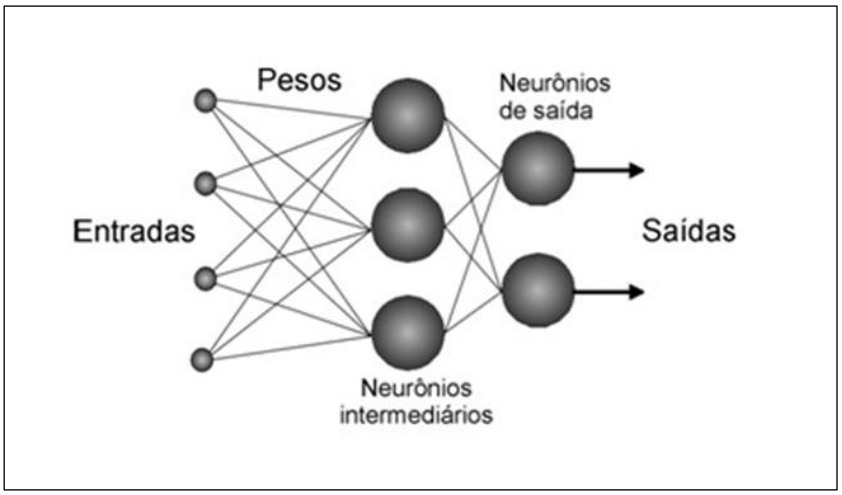


Figura 3 - Rede neural artificial – esquema geral

Assim como o cérebro humano é capaz de aprender e tomar decisões baseadas na aprendizagem, uma RNA permite que um sistema de processamento seja capaz de armazenar conhecimento baseado em experiências e disponibilizar este conhecimento para aplicação.

Esse comportamento inteligente advém das interações entre as unidades de processamento da rede que, em sua constituição, possui várias unidades de processamento com uma série de neurônios artificiais conectados entre si.

Sendo assim, a capacidade de aprendizado, ou seja, a capacidade de auto ajustar-se na tentativa de reconhecer padrões a partir das informações fornecidas, destaca-se talvez como a maior vantagem apresentada pelas RNAs.

O primeiro modelo de neurônio artificial (Figura 4) foi proposto por McCulloch e Pitts (1943). Ele constitui-se um modelo “simples” e a intenção deles era de se obter um modelo teórico do neurônio biológico, considerando-o basicamente como um circuito binário. Hoje sabe-se que tal teoria é extremamente poderosa e provou-se ser equivalente à máquina de Turing, por exemplo (Arbib, 1987).

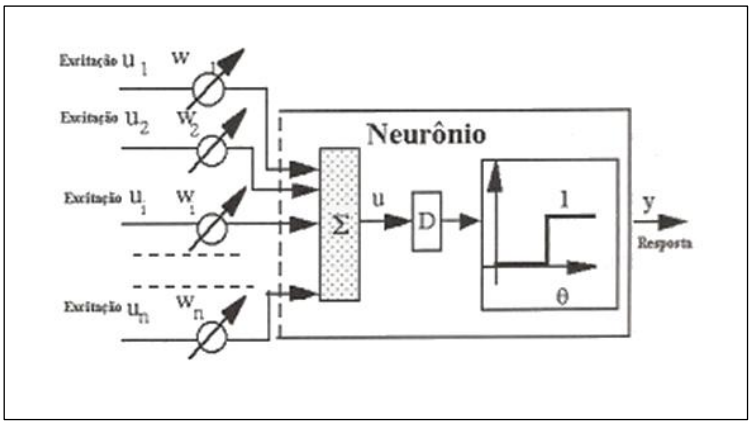


Figura 4 – Modelo simplificado de neurônio artificial de McCulloch e Pitts (1943)

Uma outra teoria, hoje clássica, o “Perceptron”, foi inicialmente utilizado em reconhecimento de caracteres. Como no caso anterior, observamos que o perceptron também faz uso de neurônios binários. Independentemente, porém, semelhante a esses estudos, Widrow e Lehr (1990) propuseram o ADALINE (ADAptive LInear NEuron). Para formar uma RNA, Widrow construiu o MADALINE, que grosso modo considera como regra o voto de “maioria”. Minsky e Papert (1988) investigaram pormenorizadamente o Perceptron usando argumentos geométricos e lógicos.

2.2 Lógica Paraconsistente

Como este trabalho apoia-se fortemente em conceitos da Lógica Paraconsistente, a seguir apresenta-se um resumo da mesma, com base em Abe (1992).

A Lógica Paraconsistente teve como precursores o lógico russo N. A. Vasil'ev e o lógico polonês J. Łukasiewics. Ambos, em 1910, independentemente, publicaram trabalhos nos quais tratavam da possibilidade de uma lógica que não eliminasse, *ab initio*, as contradições. Todavia, os trabalhos desses autores, no tocante à paraconsistência, se restringiram à Lógica Aristotélica tradicional. Somente em 1948 e 1954 que o lógico polonês S. Jaśkowski e o lógico brasileiro N.C.A. da Costa, respectivamente, embora independentemente, construíram a Lógica Paraconsistente. Também, independentemente dos trabalhos de Da Costa, o lógico D. Nelson sugeriu em 1959 uma Lógica Paraconsistente como uma versão de seu sistema conhecido como Lógicas Construtivas com negação forte.

Jaśkowski formalizou um cálculo proposicional paraconsistente denominado *Cálculo Proposicional Discursivo* (ou *Discussivo*), ao passo que Da Costa desenvolveu várias lógicas paraconsistentes contendo todos os níveis lógicos. Seus sistemas são conhecidos como sistemas C_n ($1 \leq n \leq \omega$). Da Costa desenvolveu cálculos proposicionais, cálculo de predicados, lógicas de ordem superior (na forma de teoria de conjuntos) e cálculo de descrições.

Seja *T* uma teoria fundada sobre uma lógica *L*, e suponha-se que a linguagem de *T* e de *L* contenha um símbolo para a negação — se houver mais de uma negação, uma delas deve ser escolhida, pelas suas características lógico formais. *T* diz-se *inconsistente* se ela possuir teoremas contraditórios; isto é, um é a negação do outro; caso contrário, *T* diz-se *consistente*. *T* diz-se *trivial* se todas as fórmulas de *L* — ou todas as fórmulas fechadas de *L* — forem teoremas de *T*; em hipótese contrária, *T* diz-se *não-trivial*.

Analogamente, a mesma definição aplica-se a sistemas de proposições, conjunto de informações, etc. (levando-se em conta, naturalmente, o conjunto de suas consequências).

Na Lógica Clássica e em muitas categorias de lógica, a consistência desempenha papel deveras importante. Com efeito, na maioria dos sistemas lógicos usuais, uma teoria *T* é trivial, então *T* é inconsistente e reciprocamente.

Uma lógica *L* chama-se *Paraconsistente* se puder servir de base para teorias inconsistentes, mas não-triviais.

Outro conceito significativo para o que se segue é o de Lógica Paracompleta. Uma lógica *L* chama-se *Paracompleta* se ela puder ser a lógica subjacente a teorias nas quais se infringe a lei do terceiro excluído na seguinte forma: de duas proposições contraditórias, uma delas é verdadeira. De modo preciso, uma lógica se diz paracompleta se nela existirem sistemas não-triviais maximais aos quais não pertencem uma dada fórmula e sua negação.

Finalmente, uma lógica *L* denomina-se *Não-Alética* se *L* for Paraconsistente e Paracompleta.

2.2.1 Lógicas Paraconsistentes Anotadas Qτ

As Lógicas Anotadas são uma família de Lógicas Não-clássicas que foram inicialmente empregadas em programação lógica e subsequentemente estudadas por vários autores: Da Costa, Akama, Abe, entre outros. Abe fez um estudo detalhado de seus fundamentos, bem como prosseguiu nas aplicações, tendo divulgado através de vários colaboradores.

As Lógicas Anotadas constituem, em geral, Lógicas Paraconsistentes, Paracompletas e Não-aléticas, dependendo do reticulado associado que veremos com mais detalhes adiante.

As Lógicas Anotadas, desde a sua descoberta, têm sido aplicadas em inúmeras direções: por volta de 1993, Abe juntamente com seus discípulos implementou a linguagem de programação paraconsistente *Paralog* independentemente de Subrahmanian. Tais ideias aplicaram-se na construção de um protótipo e especificação de uma arquitetura baseada na Lógica Paraconsistente Anotada que integra vários sistemas computacionais – planejadores, base de dados, sistemas de visão, etc. de uma célula de manufatura e representação de conhecimento por *Frames*, permitindo representar inconsistências e exceções. Em circuitos digitais, obteve-se a implementação das portas lógicas *Complement*, *And* e *Or*. Tais circuitos permitem sinais "conflitantes" implementados em sua estrutura de modo não-trivial. Acredita-se que a contribuição dos circuitos elétricos paraconsistentes seja pioneira na área dos circuitos elétricos, abrindo-se novas vias de investigações. Nas pesquisas ainda sobre *hardware*, foi a edificação do analisador lógico – Para-analisador - que permite tratar conceitos de incerteza, inconsistência e paracompleteza. Também foram construídos controladores lógicos baseados nas Lógicas Anotadas – *Paracontrol*, simuladores lógicos – *Parasim*, tratamento de sinais – *Parasônico*. Como materialização dos conceitos implementados construiu-se o primeiro robô paraconsistente com o *hardware* paraconsistente: a robô *Emmy*. O projeto teve sequência com o protótipo II da *Emmy* e *Emmy III* que incorporaram diversas melhorias. Um outro robô paraconsistente construído com o software baseado na Lógica Anotada denominou-se *Sofya* e muitos outros protótipos subsequentes foram construídos: Amanda, etc.

Os sistemas anotados também abarcam aspectos dos conceitos envolvidos em raciocínio não-monotônico, *defesiable*, default e deôntico.

Versões de Lógicas Anotadas também envolvem muitos aspectos das Lógicas *Fuzzy*. Isto pode ser visto sob vários ângulos. A Teoria Anotada de Conjuntos engloba *in totun* a Teoria de Conjuntos *Fuzzy*. Versões axiomatizadas de algumas formulações da Teoria *Fuzzy* foram obtidas. Foi erigido o controlador híbrido *Para-fuzzy* que une características das Lógicas Anotadas e *Fuzzy*. Adotando-se um reticulado particular, o Para-analisador anteriormente citado foi utilizado em Engenharia de Produção como ferramenta de teoria da decisão nos processos decisórios, sendo aplicado em diversos temas da área. Finalmente, aspectos algébricos também foram investigados por Abe e outras algebrizações interessantes têm sido estudadas por vários autores.

A Rede Neural Artificial Paraconsistente foi construída tendo por base essa lógica. Os textos sobre Lógica Paraconsistente e Lógica Paraconsistente Anotada Qτ são baseados em Abe (1992, 2001) e Da Costa et al. (1999).

2.2.2 Lógica Paraconsistente Anotada Evidencial Eτ

A Lógica Paraconsistente Anotada Evidencial Eτ (Lógica Eτ)**,** possui a propriedade de acolher os conceitos de incerteza, inconsistência e paracompleteza em sua estrutura linguística, permitindo atingir, capturar e refletir melhor as nuances da realidade (Abe, 2006).

Uma proposição atômica da linguagem da Lógica Eτ pode ser representada por $p_{(\mu,\ \lambda)}$, onde μ e λ são elementos do intervalo [0, 1]. (Abe 1992)

Dentre várias leituras intuitivas, $p_{(\mu,\lambda)}$, pode ser lida como: μ é a evidência favorável expressa pela proposição *p* e λ a evidência desfavorável ou contrária expressa pela proposição *p*. Por exemplo:

p: "Pedro está com febre"
μ = 0,3
λ = 0,8

Intuitivamente, $p_{(0,3;\ 0,8)}$ pode ser lida

"Pedro está com febre com evidência favorável de 30% e evidência desfavorável (ou contrária) de 80%"

Uma outra leitura da anotação (μ, λ) é dada no exemplo a seguir:

p: "Há fantasmas no antigo Edifício Joelma"
$\mu = 0{,}6$
$\lambda = 0{,}9$
"Há fantasmas no antigo Edifício Joelma com crença favorável de 60% e crença desfavorável (ou contrária) de 90%"

Os conectivos da Lógica Eτ são como usualmente: negação, conjunção, disjunção, implicação e bi-implicação (Abe, 1992).

A negação da Lógica Eτ possui propriedades interessantes. Isso pode ser ilustrada através de um exemplo. Dada uma proposição $p_{(1{,}0;\ 0{,}0)}$ temos uma evidência favorável total à proposição *p* e uma evidência contrária nula, dando uma leitura intuitiva que *p* é verdadeira. Logo, a negação de $p_{(1{,}0;\ 0{,}0)}$ é a mesma proposição com os graus de evidência invertidos $p_{(0{,}0;\ 1{,}0)}$, ou seja, o grau de evidência favorável de $\neg p_{(1{,}0;\ 0{,}0)}$ é o grau de evidência contrária de $p_{(1{,}0;\ 0{,}0)}$ e o grau de evidência contrária de $\neg p_{(1{,}0;\ 0{,}0)}$ é o grau de evidência favorável de $p_{(1{,}0;\ 0{,}0)}$. Assim, temos um operador natural definido sobre τ que desempenha o papel da negação do conectivo da lógica anotada: $\sim$: $|\tau| \rightarrow |\tau|$, $\sim(\mu, \lambda) = (\lambda, \mu)$.

Os demais conectivos são exemplificados a seguir, onde $p_{(\mu,\lambda)}$ e $q_{(\theta,\ \gamma)}$ são proposições atômicas:

Conjunção: $p_{(\mu,\lambda)} \wedge q_{(\theta,\ \gamma)}$

Disjunção: $p_{(\mu,\lambda)} \vee q_{(\theta,\ \gamma)}$

Implicação: $p_{(\mu,\lambda)} \rightarrow q_{(\theta,\ \gamma)}$

Bi-implicação: $p_{(\mu,\lambda)} \leftrightarrow q_{(\theta,\ \gamma)}$

Introduz-se algumas terminologias. O par (μ, λ) denomina-se constante de anotação. Tal par é um elemento de $[0, 1]\times[0, 1]$ que algumas vezes indicamos por $[0, 1]^2$, onde [0, 1] indica o intervalo fechado real unitário. Mune-se esse conjunto de uma relação de ordem assim definida: $(\mu_1, \lambda_1) \leq (\mu_2, \lambda_2) \Leftrightarrow \mu_1 \leq \mu_2$ e $\lambda_2 \leq \lambda_1$ (onde a segunda ordem é a ordem usual de números reais).

Propriedades:

1. $\forall \mu, \lambda \in \tau$, $(\mu, \lambda) \leq (\mu, \lambda)$ (reflexividade)

2. $\forall \mu_1, \lambda_1, \mu_2, \lambda_2 \in \tau$, $(\mu_1, \lambda_1) \leq (\mu_2, \lambda_2)$ e $(\mu_2, \lambda_2) \leq (\mu_1, \lambda_1)$, implicam $(\mu_1, \lambda_1) = (\mu_2, \lambda_2)$ (anti-simetria)
3. $\forall \mu_1, \lambda_1, \mu_2, \lambda_2, \mu_3, \lambda_3 \in \tau$, $(\mu_1, \lambda_1) \leq (\mu_2, \lambda_2)$ e $(\mu_2, \lambda_2) \leq (\mu_3, \lambda_3)$, implicam $(\mu_1, \lambda_1) \leq (\mu_3, \lambda_3)$ (transitividade)
4. $\forall \mu_1, \lambda_1, \mu_2, \lambda_2 \in \tau$, existe o supremo de $\{(\mu_1, \lambda_1), (\mu_2, \lambda_2)\}$ indicado por $(\mu_1, \lambda_1) \vee (\mu_2, \lambda_2) = (\text{Máx}\{\mu_1, \mu_2\}, \text{Mín}\{\lambda_1, \lambda_2\})$
5. $\forall \mu_1, \lambda_1, \mu_2, \lambda_2 \in \tau$, existe o ínfimo de $\{(\mu_1, \lambda_1), (\mu_2, \lambda_2)\}$ indicado por $(\mu_1, \lambda_1) \wedge (\mu_2, \lambda_2) = (\text{Mín}\{\mu_1, \mu_2\}, \text{Máx}\{\lambda_1, \lambda_2\})$
6. $\forall \mu, \lambda \in \tau$, $(0, 1) \leq (\mu, \lambda) \leq (1, 0)$

O quadrado unitário $[0, 1]\times[0, 1]$ com a relação de ordem constitui um reticulado que simbolizamos por $<\tau, \leq>$ ou simplesmente por $\in \tau$. (Figura 5)

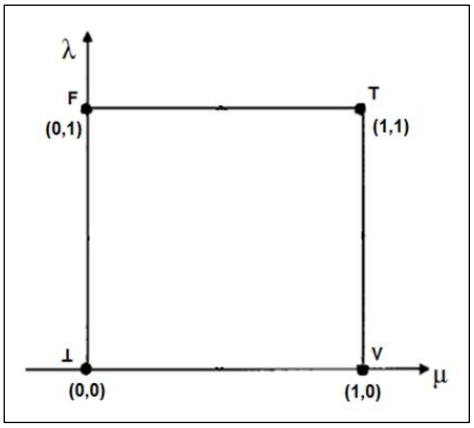


Figura 5 - Representação do reticulado associado à Lógica Paraconsistente Anotada Evidencial $E\tau$ onde: μ - evidência favorável, λ - evidência desfavorável ou contrária, V - verdadeiro, F - falso, T - inconsistente, $\perp$ - paracompleto

Na Figura 6 apresenta-se o reticulado τ acrescentados dos valores-limite, grau de certeza (G_{ce}) e grau de incerteza (G_{in}).

O G_{ce} que varia continuamente desde a falsidade até a veracidade é definido pela fórmula: $G_{ce}(\mu, \lambda) = \mu + \lambda - 1$

O G_{in} que varia continuamente desde a inconsistência até a paracompleteza é definido pela fórmula: $G_{in}(\mu, \lambda) = \mu - \lambda$

Os valores-limites V_{scin}, V_{icc}, V_{icin} e V_{scc} permitem determinar regiões no reticulado que comporão os estados lógicos resultantes. A partir dessas regiões delimitadas pode-se relacionar estados lógicos resultantes que serão obtidos pelas interpolações dos Graus

de Certeza (G_{ce}) e de Incerteza (G_{in}). Assim, para cada ponto de interpolação haverá uma região única delimitada no reticulado equivalente a um estado lógico resultante da análise.

- Valor superior de controle de incerteza (V_{scin})
- Valor inferior de controle de certeza (V_{icc})
- Valor inferior de controle de incerteza (V_{icin})
- Valor superior de controle de certeza (V_{scc})

Além dos quatro estados extremos descritos anteriormente consideramos mais oito estados não-extremos conforme a Figura 6:

- Quase-verdadeiro tendendo ao Inconsistente: $QV \rightarrow T$
- Quase-verdadeiro tendendo ao Paracompleto: $QV \rightarrow \perp$
- Quase-falso tendendo ao Inconsistente: $QF \rightarrow T$
- Quase-falso tendendo ao Paracompleto: $QF \rightarrow \perp$
- Quase-inconsistente tendendo ao Verdadeiro: $QT \rightarrow V$
- Quase-inconsistente tendendo ao Falso: $QT \rightarrow F$
- Quase-paracompleto tendendo ao Verdadeiro: $Q\perp \rightarrow V$
- Quase-paracompleto tendendo ao Falso: $Q\perp \rightarrow F$

Nesse reticulado podem ser consideradas as operações NOT, Maximização e Minimização. (Abe et al., 2011)

A operação NOT é definida por:

$$\sim(\mu, \lambda) = (\lambda, \mu).$$

A operação de Maximização, indicada por OR, é definida por:

$$(\mu_1, \lambda_1) \text{ OR } (\mu_2, \lambda_2) = (\text{Máx}\{\mu_1, \mu_2\}, \text{Mín}\{\lambda_1, \lambda_2\})$$

A operação de Minimização, indicada por AND, é definida por:

$$(\mu_1, \lambda_1) \text{ AND } (\mu_2, \lambda_2) = (\text{Mín}\{\mu_1, \mu_2\}, \text{Máx}\{\lambda_1, \lambda_2\})$$

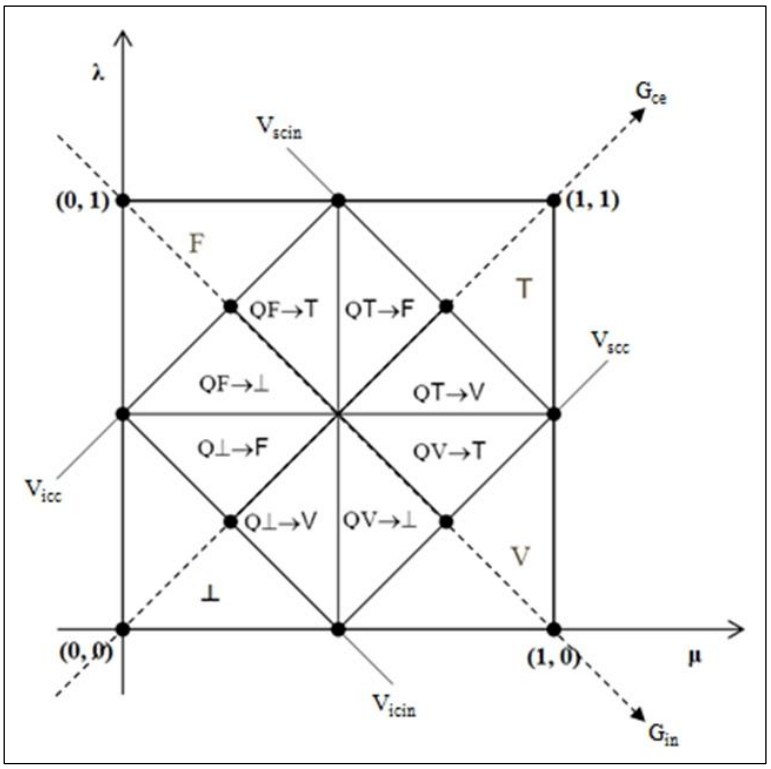


Figura 6 - Representação do Reticulado τ com os valores-limite e dos graus de certeza, incerteza e estados de decisão, onde:
μ - evidência favorável; λ - evidência desfavorável ou contrária; G_{ce} - Grau de certeza, G_{in} - Grau de incerteza, V - verdadeiro, F - falso, T - inconsistente, ⊥ - paracompleto, V_{scc} - Valor superior de controle de certeza, V_{scin} - Valor superior de controle de incerteza, V_{icc} - Valor inferior de controle de certeza, V_{icin} - Valor inferior de controle de incerteza, QF→T - Quase-falso tendendo ao Inconsistente, QF→⊥ - Quase-falso tendendo ao Paracompleto, Q⊥→F - Quase-paracompleto tendendo ao Falso, Q⊥→V - Quase-paracompleto tendendo ao Verdadeiro, QT→V - Quase inconsistente tendendo ao Verdadeiro, QT→ F - Quase-inconsistente tendendo ao Falso, QV→⊥ - Quase-verdadeiro tendendo ao Paracompleto, QV→T - Quase-verdadeiro tendendo ao Inconsistente

Apresentam-se na Figura 7 as três etapas de uma análise com base na lógica paraconsistente realizada por um Sistema de Controle Paraconsistente, ou seja, um sistema de controle com base na lógica paraconsistente, que usa o algoritmo Para-Analisador. As três etapas são: entrada das informações, processamento e conclusão. (Abe, 1992)

As informações de entrada são dois valores variáveis contínuos e independentes: o grau de evidência favorável e o grau de evidência desfavorável, ambos, valores reais entre 0 e 1.

No processamento, o sistema procura tomar decisões através de análises que utilizam o algoritmo Para-Analisador para que a "intensidade" das contradições diminua. Como saída são determinados os estados lógicos representados pelas 12

regiões do reticulado e são calculados os Graus de Certeza e de Incerteza, que determinam a certeza sobre a proposição e se existe ou não contradição.

Na conclusão, o sistema gera uma decisão com base em um dos doze estados lógicos, obtido pela comparação dos valores de controle e os Graus de Certeza e de Incerteza.

Assim, o algoritmo Para-Analisador pode ser considerado um Sistema de Análise Paraconsistente que permite criar, através da linguagem de programação ou por hardware, sistemas capazes de tratar incertezas, inconsistências e paracompletezas (Abe, 1992).

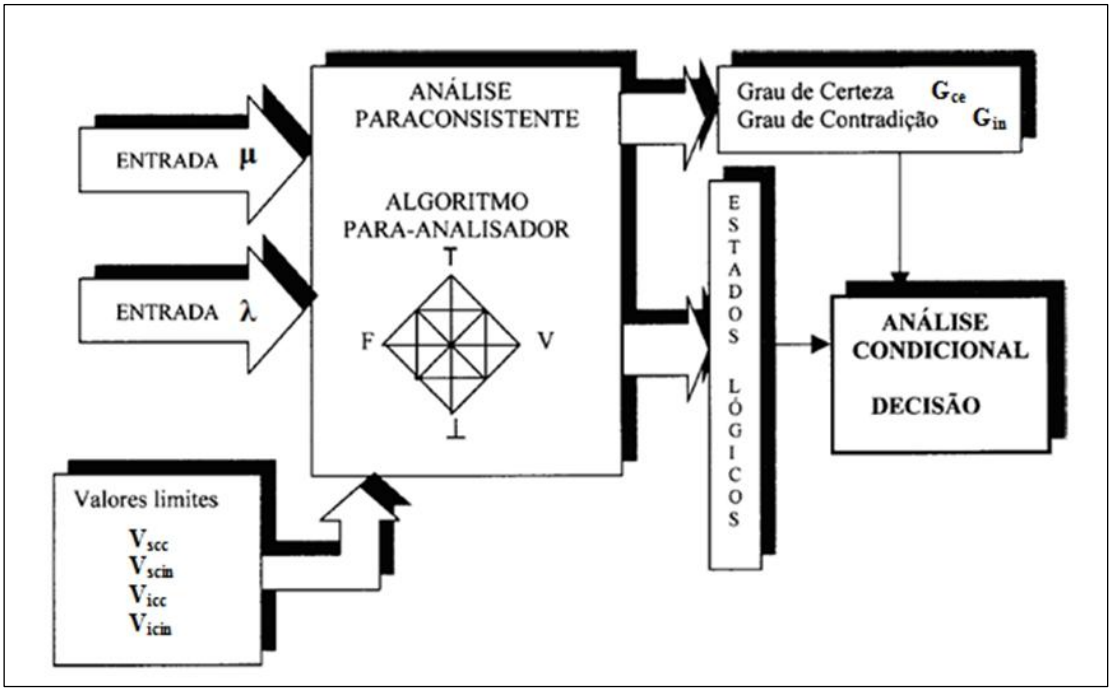


Figura 7 - Representação de um sistema de análise paraconsistente, onde:
entrada μ - evidência favorável, entrada λ - evidência desfavorável ou contrária, G_{ce} - Grau de certeza , G_{in} - Grau de incerteza, V - verdadeiro, F - falso, T - inconsistente, ⊥ - paracompleto, V_{scc} - Valor superior de controle de certeza, V_{scin}- Valor superior de controle de incerteza, V_{icc} - Valor inferior de controle de certeza , V_{icin} - Valor inferior de controle de incerteza

2.3 Rede Neural Artificial Paraconsistente

A RNA proposta em Da Silva Filho e Abe (2001) denominada Rede Neural Artificial Paraconsistente baseia-se na Lógica Paraconsistente Anotada Evidencial Eτ.

Tal lógica, como já observamos, é capaz de manipular conceitos como de incerteza, de inconsistência e de paracompleteza no seu interior sem o perigo de trivialização conforme (Abe, 1992).

Em consequência, a RNAP é capaz também de manipular os conceitos citados diretamente, o que faz diferenciar das teorias de RNAs existentes na literatura. Tais conceitos (incerteza, inconsistência e paracompleteza) estão entre os conceitos mais desafiadores em reconhecimento de padrões.

Uma RNAP é composta por várias Unidades Neurais Artificiais Paraconsistentes (UNAPs) com funções distintas, tais como conexão, aprendizado, memorização etc., que tratam os sinais de entrada da rede provenientes dos graus de evidência favorável e contrário.

Essas UNAPs podem ser caracterizadas como "aglomerados" de Células Neurais Artificiais Paraconsistentes (CNAPs) interligadas entre si.

A seguir apresenta-se uma breve descrição de algumas células que compõem a família das Células Neurais Artificiais Paraconsistentes e no Apêndice E apresenta-se seus respectivos algoritmos.

A *Célula Analítica Real* - *CNAPar* - realiza uma análise das entradas e retorna o grau de evidência real diminuindo, assim, o efeito da inconsistência presente nas entradas (Figura 8).

A *Célula de Detecção de Igualdade* - *CNAPdi* - compara as entradas e retorna um valor representando a igualdade entre elas (Figura 9).

As *Células de Conexão Lógica Simples de Maximização e Minimização* - *CNAPcls* - retornam o valor máximo e mínimo, respectivamente, a partir das entradas. (Figura 10).

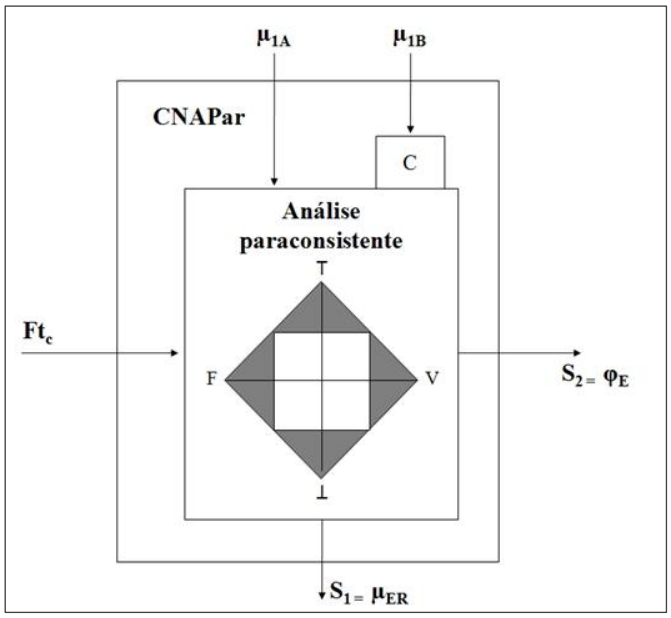


Figura 8 - Célula Neural Artificial Paraconsistente Analítica Real (CNAPar) onde:
μ_{1A} - primeira entrada, μ_{1B} - segunda entrada, C - complemento (1 - μ_{1B}), Ft_c - Fator de Tolerância à Certeza, φ_E - Intervalo de Evidência, μ_{ER} - Grau de Evidência Resultante Real, S_1 - Saída do Grau de Evidência Resultante, S_2 - Saída do sinal de Intervalo de Evidência, V - verdadeiro, F - falso, T - inconsistente, ⊥ - paracompleto

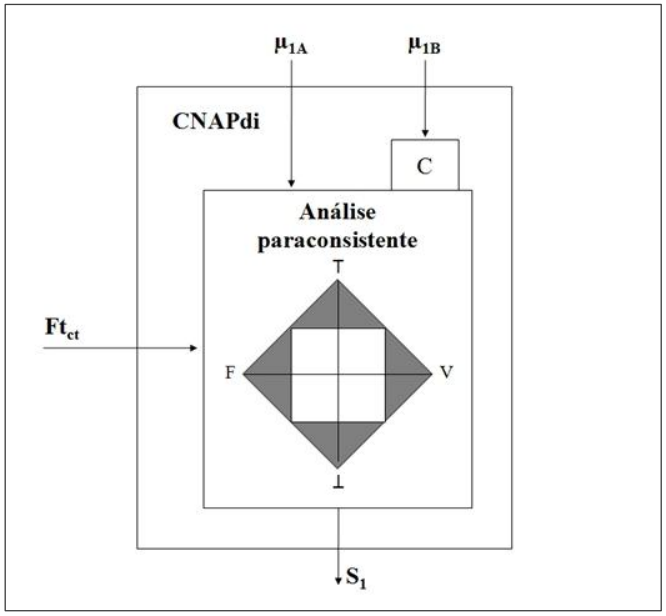


Figura 9 - Célula Neural Artificial Paraconsistente de Detecção de Igualdade (CNAPdi) onde:
μ_{1A} - primeira entrada, μ_{1B} - segunda entrada, C - complemento (1 - μ_{1B}), Ft_{ct} - Fator de Tolerância à Contradição, S_1 - valor 0 ou 1, V - verdadeiro, F - falso, T - inconsistente, ⊥ - paracompleto

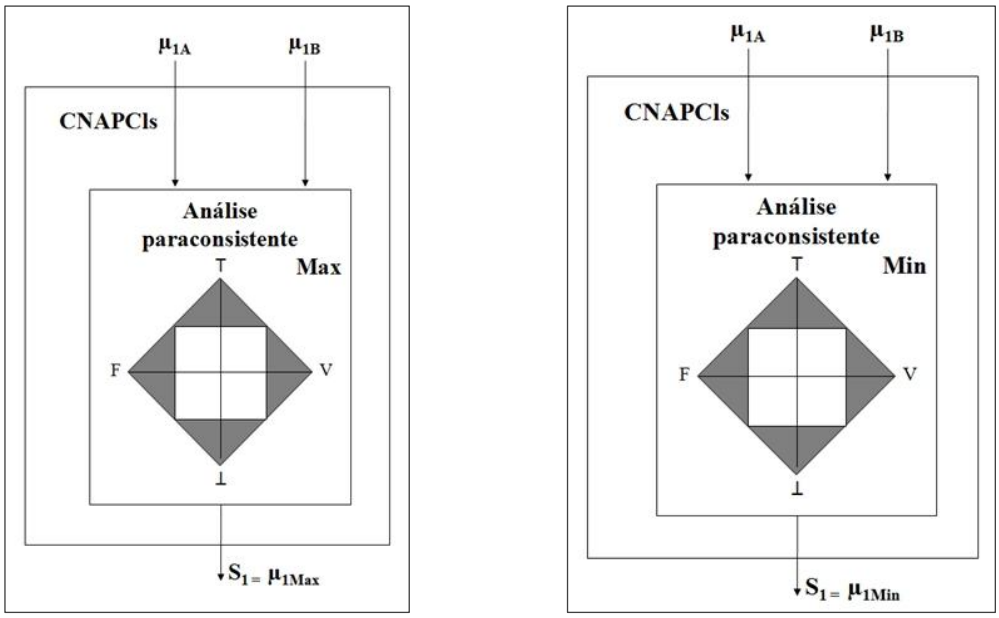


Figura 10 - Células Neurais Artificiais Paraconsistentes de Conexão Lógica de Maximização e de Minimização (CNAPcls Max e CNAPcls Min) onde:
μ_{1A} - primeira entrada, μ_{1B} - segunda entrada, μ_{1Max} - valor máximo da entrada, μ_{1Min} - valor mínimo da entrada, V - verdadeiro, F - falso, T - inconsistente, ⊥ - paracompleto, S_1 - saída da célula

A *Célula de Passagem* - *CNAPpa* - canaliza sinais para determinadas regiões da rede (Figura 11).

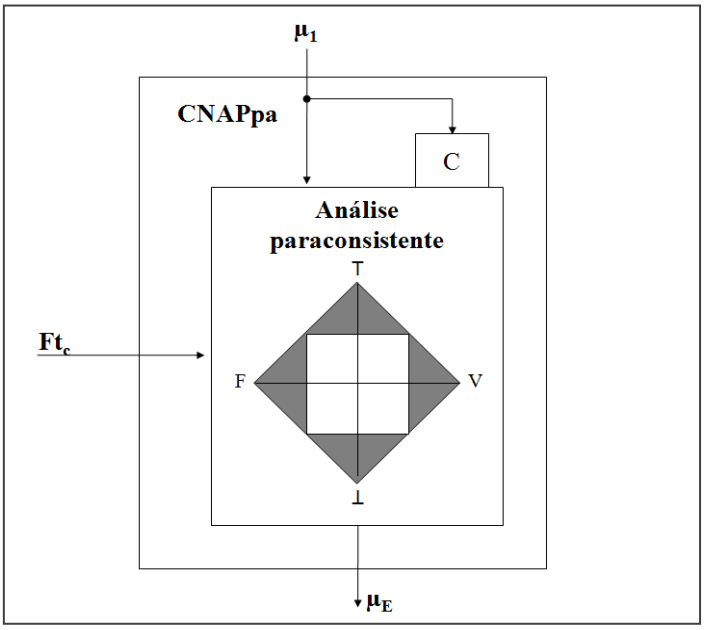


Figura 11 - Célula Neural Artificial Paraconsistente de Passagem (CNAPpa) onde:
μ_1 - entrada, C - complemento (1 - μ_1), Ft_c - Fator de Tolerância à Certeza, V - verdadeiro, F - falso, T - inconsistente, ⊥ - paracompleto, μ_E - grau de evidência resultante

3 MÉTODOS

Atualmente existem vários trabalhos desenvolvidos e em desenvolvimento na área de Inteligência Artificial com base nas RNAPs como Prado (1996), Prado (2007), Da Silva Filho (1999), Lopes (2009), Mário (2006), Torres (2010), Amaral (2013) entre outros. Como se observou na Introdução, o reconhecimento de caracteres destaca-se como uma das mais conhecidas e exploradas modalidades de reconhecimento de padrões, que consiste em características extraídas de um conjunto de caracteres com o objetivo de tentar reproduzir a capacidade humana de ler textos.

Como a técnica de reconhecimento de caracteres classifica as letras a partir das suas características, a maior dificuldade, em sistemas de reconhecimento de padrões, concentra-se na determinação do conjunto de características passíveis de extração devido à interferência de erros e/ou ruídos. Considerando esses detalhes, o desempenho de um sistema automático de reconhecimento de padrões depende fundamentalmente da qualidade dos documentos nas suas formas original e digital e, também de um sistema que consiga manipular os tais dados imprecisos, conflitantes e/ou paracompletos. Assim optamos pelo uso das RNAPs que tem se apresentado como uma importante ferramenta de análise para aplicações que envolvem dados com tais características.

Embora hajam vários estudos no ramo de reconhecimento de caracteres, essa área encontra-se em constante desenvolvimento onde novas técnicas e metodologias surgem ao longo do tempo proporcionando melhor desempenho no reconhecimento.

Esse trabalho propõe uma solução computadorizada para reconhecimento dos caracteres numéricos de cheques bancários brasileiros localizados da linha MICR CMC-7 e no campo do valor numérico. A Figura 12 destaca a localização desses caracteres no cheque.

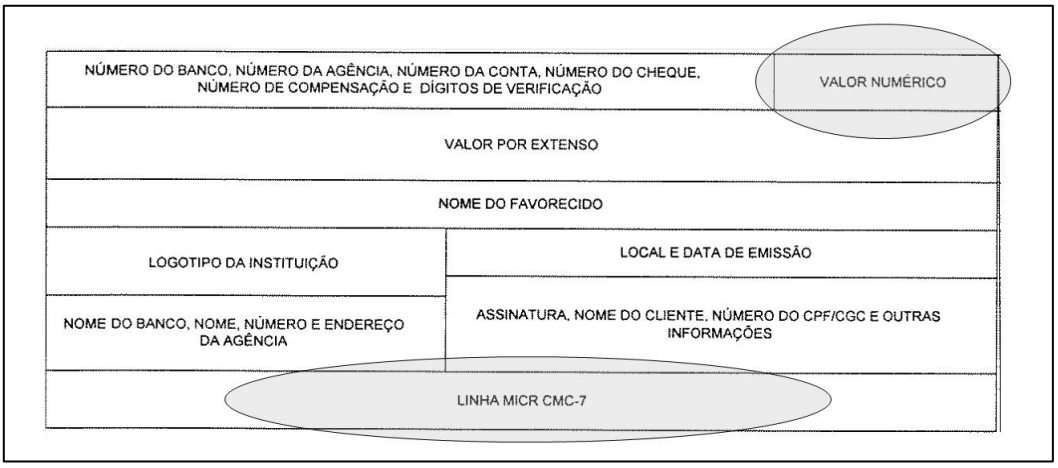


Figura 12 - Estrutura dos campos de um cheque bancário brasileiro, onde:
MICR – Magnetic Ink Character Recognition, CMC-7 - Caracteres Magnéticos Codificado em 7 Barras

A linha MICR CMC-7 pode ser identificada como uma espécie de código de barras composta por um conjunto de caracteres numéricos e alguns caracteres especiais que seguem um determinado padrão (formato), como mostra a Figura 13. Esses caracteres representam um conjunto de dados bancários do cliente como: número do banco, agência, número do cheque e número da conta bancária do cliente.

Figura 13 - Exemplo de linha MICR CMC-7

A seguir apresenta-se um procedimento computacional automático para reconhecimento de caracteres numéricos com um padrão previamente definido e caracteres numéricos manuscritos pelo uso das RNAPs.

A princípio, optou-se por desenvolver o sistema para fazer o reconhecimento de caracteres numéricos com um padrão previamente definido e, posteriormente, com base nos resultados obtidos, as técnicas foram estudas e aperfeiçoadas para fazer o reconhecimento de caracteres numéricos manuscritos.

Para o estudo do caractere numérico com um padrão previamente definido adotou-se os caracteres do tipo CMC-7 (Figura 13), utilizados na linha MICR CMC-7 de cheques bancários brasileiros para codificar os dados da conta bancária do cliente.

O Sistema Computacional Paraconsistente proposto, ao receber como entrada uma imagem digitalizada e previamente tratada de um caractere, é capaz de extrair algumas

características em formato de graus de evidências e, a partir dessas características, faz o reconhecimento do caractere com base nos conceitos das RNAPs.

O processo de reconhecimento do sistema subdivide-se em cinco fases ilustradas na Figura 14: Aquisição da Imagem, Pré-processamento da Imagem, Mapeamento da Imagem em Graus de Evidência, Extração de Características e Reconhecimento.

As duas primeiras fases referem-se à aquisição da imagem (digitalização) e pré-processamento da imagem, as quais não serão abordadas nesse trabalho.

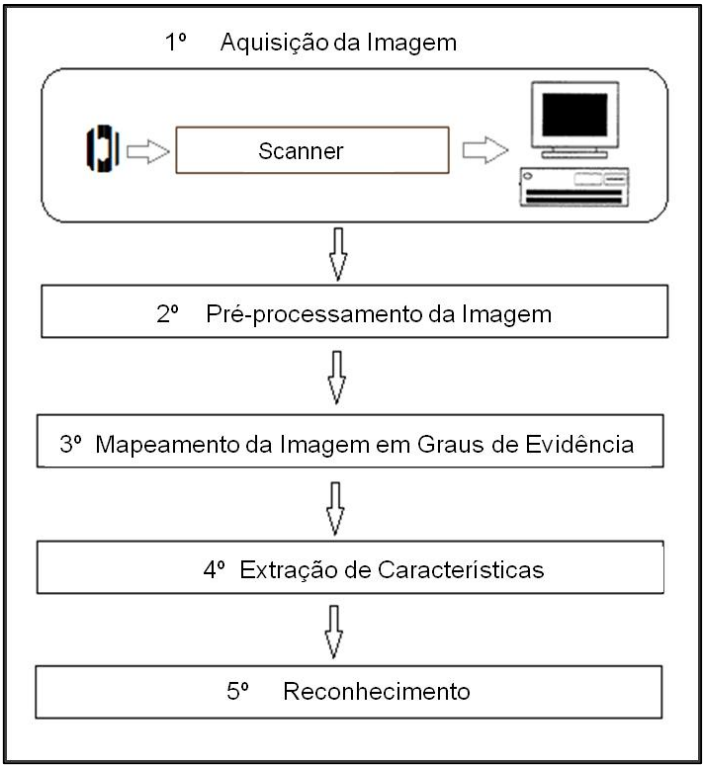


Figura 14 - Fases para o processo de reconhecimento

Apenas como um tratamento simples, adotou-se, na fase de Pré-processamento da Imagem, um filtro que aplica a média dos pixels vizinhos que se repetem, a binarização da imagem, o recorte da imagem nos pontos máximos e mínimos e, o redimensionamento da imagem para 38 (linhas) x 30 (colunas). Ao término dessa fase a imagem estará preparada para a fase de Mapeamento da Imagem em Graus de Evidência. A Figura 15 apresenta um exemplo desses filtros adotados

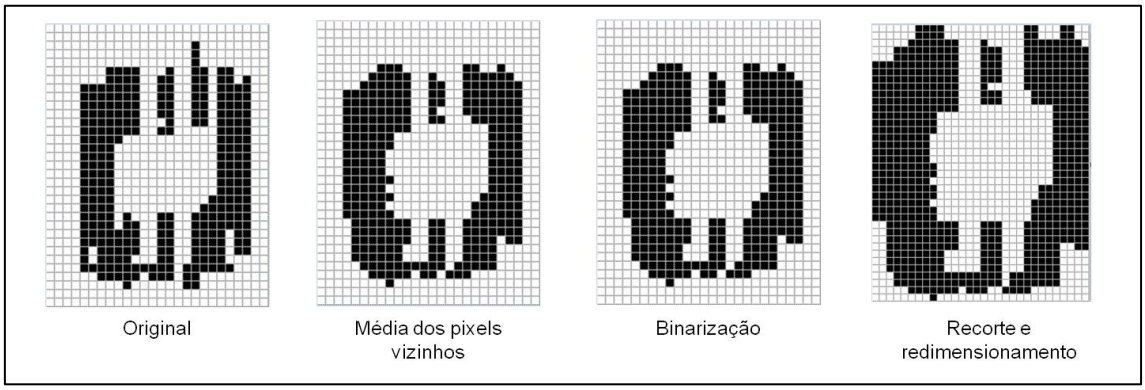


Figura 15 - Exemplo dos tratamentos adotados na fase de pré-processamento da imagem

Considerando como entrada do sistema proposto uma imagem previamente tratada, temos a arquitetura desse sistema apresentada pela Figura 16 composta por quatro camadas, sendo elas, mapeamento da imagem em graus de evidência, extração de características da imagem, reconhecimento das características da imagem e análise paraconsistente das características reconhecidas para definir o caractere reconhecido.

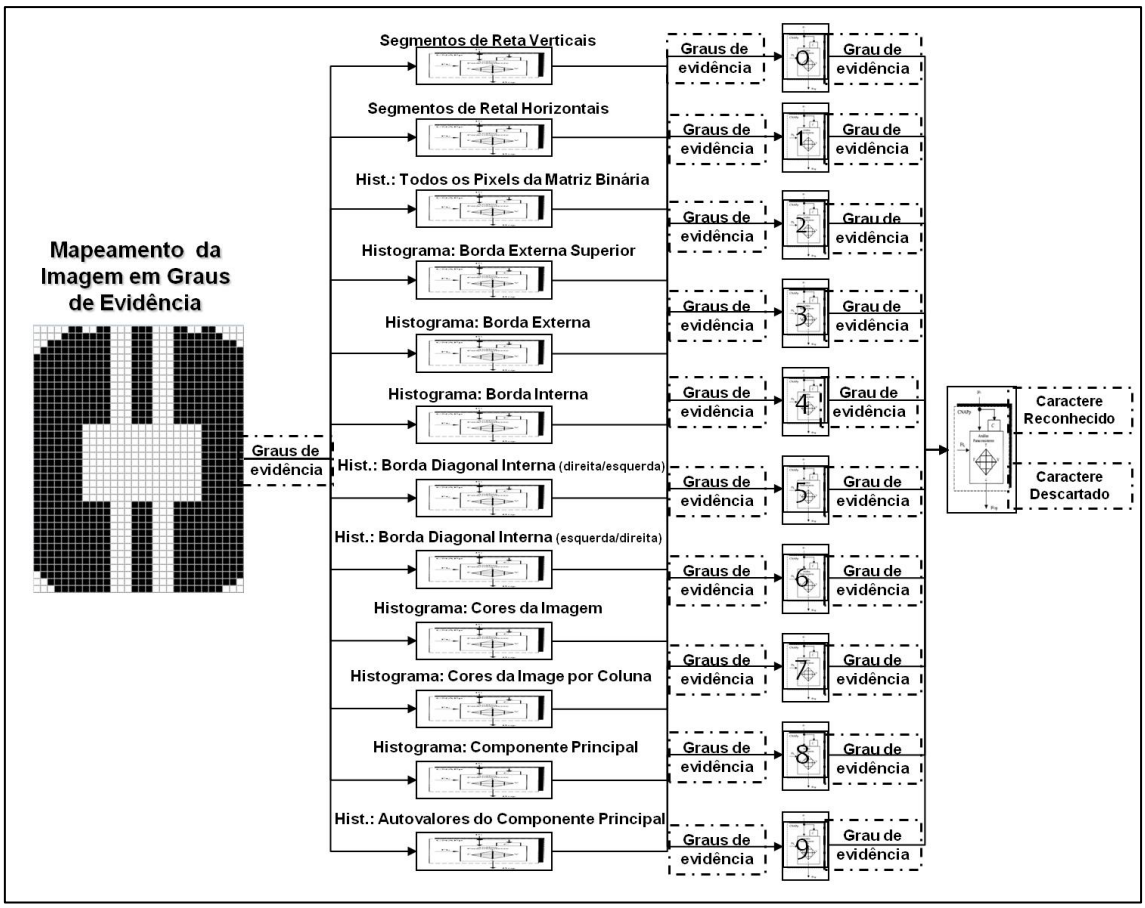


Figura 16 - Arquitetura do Sistema Computacional Paraconsistente proposto

A primeira camada efetua o mapeamento dos pixels da imagem digitalizada em graus de evidência. A partir desse mapeamento algumas características como "Segmentos de Reta Verticais", "Segmentos de Reta Horizontais", "Histograma: Todos os Pixels da Matriz Binária", "Histograma: Borda Externa Superior", "Histograma: Borda Externa", "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (direita/esquerda)", "Histograma: Borda Diagonal Interna (esquerda/direita)", "Histograma: Cores da Imagem", "Histograma: Cores da Imagem por Coluna", "Histograma: Componente Principal" e "Histograma: Autovalores do Componente Principal" são extraídas da imagem. Com base nessas características, o processo de reconhecimento das características da imagem é realizado através do uso das RNAPs e, na última camada o caractere é reconhecido após uma análise paraconsistente das características reconhecidas na terceira camada.

A seguir apresentam-se os detalhes dos procedimentos realizados em cada camada.

3.1 Mapeamento da imagem em graus de evidência

Mapear a imagem em graus de evidência consiste em analisar a matriz de pixels da imagem em várias situações e criar listas de graus de evidência (valores no intervalo [0, 1] que simbolizam a existência de um pixel preto em uma determinada posição da matriz imagem), as quais representam uma projeção da imagem.

Nesse processo se realiza, com base na matriz de pixels da imagem, vários tipos de mapeamento da imagem. A quantidade de diferentes tipos depende do conjunto de características previamente selecionados para o processo de extração de características da imagem.

Neste trabalho, para o reconhecimento dos caracteres numéricos, optou-se pelo seguinte conjunto de tipos de mapeamento da imagem: "Todos os Pixels da Matriz Binária", "Borda Externa", "Borda Interna", "Borda Diagonal Interna (direita/esquerda)", "Borda Diagonal Interna (esquerda/direta)", "Cores da Imagem" (porcentagem de pixels pretos e brancos da imagem como um todo), "Cores da Imagem por coluna" (quantidade de pixels pretos por coluna), "Componente Principal" e

"Autovalores do Componente Principal". A Figura 17 apresenta os cinco primeiros tipos de mapeamento.

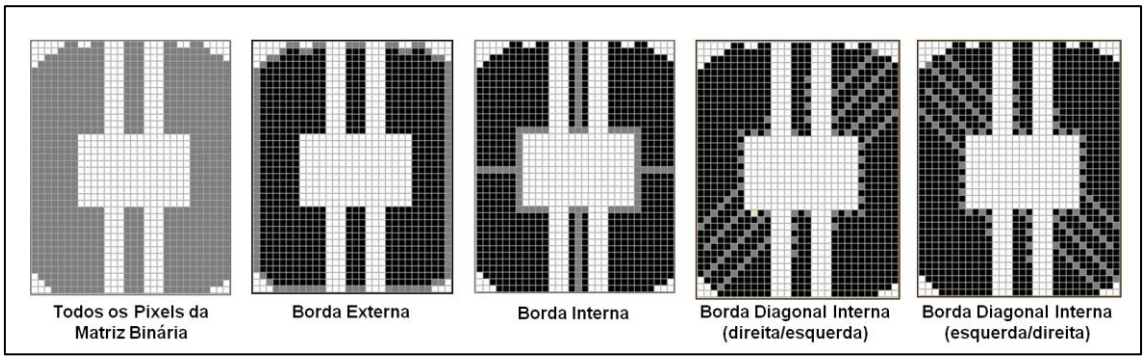


Figura 17 - Tipos de mapeamento da imagem

Considerando que o grau de evidência deve ser um valor no intervalo [0, 1], adotou-se: '0' ao detectar ausência de um pixel preto ao percorrer uma linha ou uma coluna da matriz analisada (para o mapeamento das bordas externa, interna e diagonais internas da imagem); '0' para cada pixel branco encontrado na matriz da imagem (apenas no mapeamento "Todos os Pixels da Matriz Binária") e; um grau de evidência no intervalo [0, 1] para cada pixel preto encontrado na matriz da imagem que possua pelo menos um pixel vizinho da cor preta, conforme descrito a seguir.

Para os tipos de mapeamento "Borda Externa" e "Borda Interna" foram atribuídos graus de evidência para cada pixel preto encontrado nas bordas externa e interna da imagem. Esse processo se subdivide em quatro partes sendo elas: lado esquerdo, lado direito, topo e base da imagem.

No mapeamento da Borda Externa da imagem. Para os lados esquerdo e direito, adotamos como base a normalização do grau de evidencia entre as colunas da imagem, ou seja, utilizamos um regra de três simples para calcular o grau de evidência com base na quantidade de colunas da imagem.

[total de colunas] - · - · - · - · - · - · - · - · ► *1*
[coluna do pixel preto encontrado] - · - · ► *x*

Como exemplo podemos utilizar a Figura 18 com 30 colunas e 38 linhas:

- *lado esquerdo:* a primeira linha da borda externa da imagem possui o primeiro pixel preto na 6ª coluna, portanto, o valor do grau de evidência atribuído equivale à 6/30 = 0,20.

- *lado direito:* a primeira linha da borda externa da imagem possui o último pixel preto na 26ª coluna, portanto o valor do grau de evidência atribuído equivale à 26/30 = 0,87.

Nos casos do topo e da base da imagem, adotamos como base a normalização do grau de evidencia entre as linhas, ou seja, utilizamos um regra de três simples para calcular o grau de evidência com base na quantidade de linhas da imagem.

[total de linhas] ----------► *1*
[linha do pixel preto encontrado] ----► *x*

Assim, na Figura 18 temos:

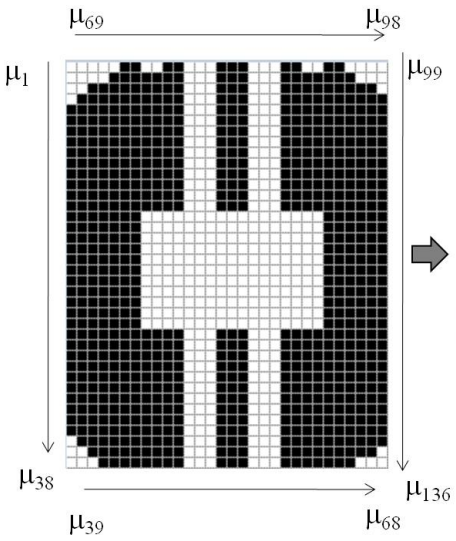
- *topo da imagem:* a primeira coluna da borda externa da imagem possui o primeiro pixel preto na 5ª linha, portanto, o valor do grau de evidência atribuído equivale à 5/38 = 0,13.
- *base da imagem:* a primeira coluna da borda externa da imagem possui o último pixel preto na 35ª coluna, portanto, o valor do grau de evidência atribuído equivale à 35/38 = 0,92.

Para o mapeamento da Borda Interna da imagem adotou-se a mesma técnica do mapeamento da borda externa da imagem, porém, o algoritmo faz a projeção dos pixels a partir do centro da imagem, ou seja, o algoritmo faz um corte vertical no centro na matriz para extrair os graus de evidência dos lados internos esquerdo e direito e, um corte horizontal na matriz para extrair os graus de evidência dos lados internos superior e inferior da imagem.

Como exemplo dos graus de evidência da borda interna, temos na Figura 18:

- *lado interno esquerdo:* a primeira linha da borda interna da imagem possui o primeiro pixel preto na 16ª coluna, portanto, o valor do grau de evidência atribuído equivale à 16/30 = 0,53 .
- *lado interno direito:* a primeira linha da borda interna da imagem possui o último pixel preto na 15ª coluna, portanto, o valor do grau de evidência atribuído equivale à 15/30 = 0,50.
- *lado interno superior:* a primeira coluna da borda interna da imagem possui o primeiro pixel preto na 20ª linha, portanto, o valor do grau de evidência atribuído equivale à 20/38 = 0,53.

- *lado interno inferior:* a primeira coluna da borda interna da imagem possui o última pixel preto na 19ª coluna, portanto, o valor do grau de evidência atribuído equivale à 19/38 = 0,50.



μ_1	0.20	μ_{24}	0.03	μ_{47}	1	μ_{70}	0.11	μ_{93}	0.03	μ_{116}	1
μ_2	0.17	μ_{25}	0.03	μ_{48}	1	μ_{71}	0.08	μ_{94}	0.03	μ_{117}	1
μ_3	0.10	μ_{26}	0.03	μ_{49}	1	μ_{72}	0.08	μ_{95}	0.05	μ_{118}	1
μ_4	0.07	μ_{27}	0.03	μ_{50}	0	μ^{73}	0.05	μ_{96}	0.08	μ_{119}	1
μ_5	0.03	μ_{28}	0.03	μ_{51}	0	μ_{74}	0.03	μ_{97}	0.08	μ_{120}	1
μ_6	0.03	μ_{29}	0.03	μ_{52}	0	μ^{75}	0.03	μ_{98}	0.11	μ_{121}	1
μ_7	0.03	μ_{30}	0.03	μ_{53}	1	μ_{76}	0.05	μ_{99}	0.87	μ_{122}	1
μ_8	0.03	μ_{31}	0.03	μ_{54}	1	μ_{77}	0.05	μ_{100}	0.90	μ_{123}	1
μ_9	0.03	μ_{32}	0.03	μ_{55}	1	μ_{78}	0.03	μ_{101}	0.97	μ_{124}	1
μ_{10}	0.03	μ_{33}	0.03	μ_{56}	0	μ_{79}	0.03	μ_{102}	1	μ_{125}	1
μ_{11}	0.03	μ_{34}	0.03	μ_{57}	0	μ_{80}	0	μ_{103}	1	μ_{126}	1
μ_{12}	0.03	μ_{35}	0.03	μ_{58}	0	μ_{81}	0	μ_{104}	1	μ_{127}	1
μ_{13}	0.03	μ_{36}	0.07	μ_{59}	1	μ_{82}	0	μ_{105}	1	μ_{128}	1
μ_{14}	0.03	μ_{37}	0.10	μ_{60}	1	μ_{83}	0.03	μ_{106}	1	μ_{129}	1
μ_{15}	0.03	μ_{38}	0.13	μ_{61}	1	μ_{84}	0.03	μ_{107}	1	μ_{130}	1
μ_{16}	0.03	μ_{39}	0.92	μ_{62}	1	μ_{85}	0.03	μ_{108}	1	μ_{131}	1
μ_{17}	0.03	μ_{40}	0.95	μ_{63}	1	μ_{86}	0	μ_{109}	1	μ_{132}	1
μ_{18}	0.03	μ_{41}	0.97	μ_{64}	1	μ_{87}	0	μ_{110}	1	μ_{133}	1
μ_{19}	0.03	μ_{42}	1	μ_{65}	1	μ_{88}	0	μ_{111}	1	μ_{134}	1
μ_{20}	0.03	μ_{43}	1	μ_{66}	0.97	μ_{89}	0.03	μ_{112}	1	μ_{135}	0.97
μ_{21}	0.03	μ_{44}	1	μ_{67}	0.97	μ_{90}	0.03	μ_{113}	1	μ_{136}	0.90
μ_{22}	0.03	μ_{45}	1	μ_{68}	0.95	μ_{91}	0.05	μ_{114}	1		
μ_{23}	0.03	μ_{46}	1	μ_{69}	0.13	μ_{92}	0.05	μ_{115}	1		

Figura 18 - Mapeamento da borda externa da imagem em graus de evidência

Ao final do mapeamento das bordas externa e interna da imagem temos oito grupos de graus de evidência, lado esquerdo, lado direito, topo e base (um para cada tipo de borda), representando as bordas externa e interna da imagem, os quais são a base para o processo de extração de característica dos Segmentos de Reta Verticais e Horizontais.

Para o mapeamento "Todos os Pixels da Matriz Binária" adotou-se o valor '0' para cada pixel branco e, para os pixels pretos, utiliza-se a mesma técnica do mapeamento da "Borda Externa", ou seja, para cada pixel preto encontrado na matriz utilizamos um regra de três simples para calcular o grau de evidência com base na quantidade de colunas da imagem.

[total de colunas] -·-·-·-·-·-·-·-·-► *1*
[coluna do pixel preto encontrado] -·-·-► *x*

Para o mapeamento "Borda Diagonal Interna (direita/esquerda)" foram atribuídos graus de evidência para os pixels pretos encontrados nas bordas diagonais internas a partir de dois cortes diagonais na matriz nomeados "superior" e "inferior". Conforme demonstrado na Figura 19(a), o corte diagonal superior (direita/esquerda) inicia-se no último pixel da primeira linha da matriz e se estende até o primeiro pixel da trigésima (30ª) linha da matriz enquanto o corte diagonal inferior (direita/esquerda) inicia-se no último pixel da nona (9ª) linha e estende-se até o primeiro pixel da última coluna da matriz da imagem.

Para o mapeamento "Borda Diagonal Interna (esquerda/ direita)" foram atribuídos graus de evidência para os pixels pretos encontrados nas bordas diagonais internas a partir de dois cortes diagonais na matriz também nomeado "superior" e "inferior". Conforme demonstrado na Figura 19(b), o corte diagonal superior (esquerda/direita) inicia-se no primeiro pixel da primeira linha da matriz e se estende até o último pixel da trigésima (30ª) linha da matriz enquanto o corte diagonal inferior (esquerda/direita) inicia-se no primeiro pixel da nona (9ª) linha e estende-se até o último pixel da última coluna da matriz da imagem.

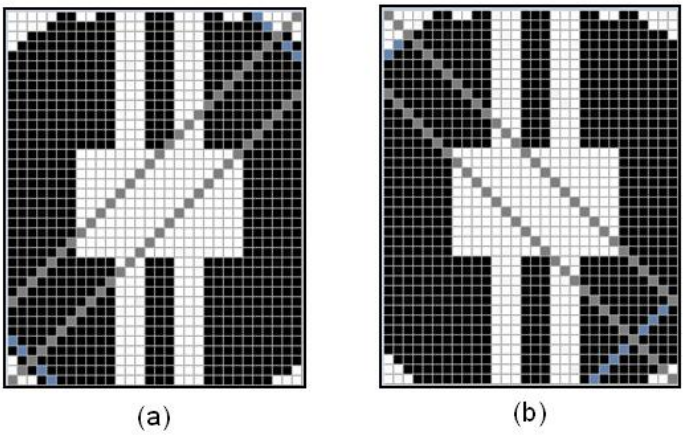


Figura 19 - Cortes diagonais internos superior e inferior onde:
pixels na cor cinza referem-se aos cortes diagonais internos superior e inferior e, pixels na cor azul representam o caminho percorrido pelo algoritmo em busca de um pixel preto

Para mapear cada diagonal o processo se subdivide em duas partes sendo elas: a visão da borda interna acima (topo) e abaixo (base) dos cortes diagonais superior e inferior.

Em ambos os casos, nas diagonais superior e inferior, o algoritmo, ao encontrar um pixel preto na diagonal, atribui um grau de evidência para o mesmo a partir das

coordenadas (x, y) de sua localização na matriz e, de um valor nomeado “peso” (valor atribuído a cada pixel da matriz independente de sua localização) calculado com base no total de pixels da matriz, onde:

peso = 1 / total de pixels da matriz ∴ 1 / (30 * 38) = 0,00088
grau de evidência = coordenada x * coordenada y * peso

Como exemplo dos graus de evidência da Borda Diagonal Interna (direita/esquerda) com base nos cortes diagonais superior e inferior temos na Figura 19(a):

- *Borda Diagonal Interna (direita/esquerda) superior - topo:* a primeira linha diagonal destacada em azul possui um pixel preto na coordenada (2, 27), portanto, o valor do grau de evidência atribuído equivale a 2 * 27 * 0,00088 = 0,047.
- *Borda Diagonal Interna (direita/esquerda) superior - base:* a primeira linha diagonal destacada em azul possui um pixel preto na coordenada (4, 29), portanto, o valor do grau de evidência atribuído equivale a 4 * 29 * 0,00088 = 0,102.
- *Borda Diagonal Interna (direita/esquerda) inferior - topo:* a segunda linha diagonal destacada em azul possui um pixel preto na coordenada (35, 2), portanto, o valor do grau de evidência atribuído equivale a 35 * 2 * 0,00088 = 0,056.
- *Borda Diagonal Interna (direita/esquerda) inferior - base:* a segunda linha diagonal destacada em azul possui um pixel preto na coordenada (37, 4), portanto, o valor do grau de evidência atribuído equivale a 37 * 4 * 0,00088 = 0,130.

Ao final do mapeamento "Borda Diagonal Interna (direita/esquerda)" temos uma lista de graus de evidência, formada pela seguinte sequência dos graus de evidência: borda diagonal interna (direita/esquerda) superior - topo, borda diagonal interna (direita/esquerda) superior - base, borda diagonal interna (direita/esquerda) inferior - topo e borda diagonal interna (direita/esquerda) inferior - base.

Como exemplo dos graus de evidência da Borda Diagonal Interna (esquerda/direita) com base nos cortes diagonais superior e inferior temos na Figura 19(a):

- *Borda Diagonal Interna (esquerda /direita) superior - topo:* a primeira linha diagonal destacada em azul não possui um pixel preto na parte superior, portanto, o valor do grau de evidência atribuído equivale a 0.
- *Borda Diagonal Interna (esquerda /direita) superior - base:* a primeira linha diagonal destacada em azul possui um pixel preto na coordenada (4, 2), portanto, o valor do grau de evidência atribuído equivale a 4 * 2 * 0,00088 = 0,007.
- *Borda Diagonal Interna (esquerda /direita) inferior - topo:* a segunda linha diagonal destacada em azul possui um pixel preto na coordenada (33, 27), portanto, o valor do grau de evidência atribuído equivale a 33 * 27 * 0,00088 = 0,784.
- *Borda Diagonal Interna (esquerda /direita) inferior - base:* a segunda linha diagonal destacada em azul possui um pixel preto na coordenada (35, 25), portanto, o valor do grau de evidência atribuído equivale a 35 * 25 * 0,00088 = 0,77.

Ao final do mapeamento "Borda Diagonal Interna (esquerda/direita)" temos uma lista de graus de evidência, formada pela seguinte sequência dos graus de evidência: borda diagonal interna (esquerda/direita) superior - topo, borda diagonal interna (esquerda/direita) superior - base, borda diagonal interna (esquerda/direita) inferior - topo e borda diagonal interna (esquerda/direita) inferior - base.

Para o mapeamento "Todos os Pixels da Matriz Binária" foram atribuídos graus de evidência para cada pixel da imagem. Atribuiu-se os valores como base a normalização do grau de evidencia entre as colunas da imagem, ou seja, utilizamos uma regra de três simples para calcular o grau de evidência com base na quantidade de colunas da imagem.

[total de colunas] -·-·-·-·-·-·-·-·-► *1*
[coluna do pixel preto encontrado] -·-·-► *x*

Como exemplo, podemos utilizar a Figura 19(a):

- Para os pixels brancos da matriz foram atribuídos grau de evidência = 0.
- Para os pixels pretos da matriz os graus de evidência foram calculados por ordem de coluna conforme descrito acima. O primeiro pixel preto encontrado (linha 5 e coluna 1) o valor do grau de evidência atribuído equivale à 1/30 = 0,033.

Ao final do mapeamento de "Todos os Pixels da Matriz Binária" temos uma lista de graus de evidência, formada pela sequência dos graus de evidência ordenados por coluna.

Para o mapeamento "Cores da Imagem" foram atribuídos graus de evidência equivalentes ao total de pixels brancos e pretos na matriz binária da imagem, por exemplo, se a matriz binária de pixels da imagem possuir 66% dos pixels na cor preta e 34% dos pixels na cor branca conforme Figura19(a), a lista de graus de evidência criada para representar esse mapeamento é equivalente à [0,66, 0,34].

Para o mapeamento "Cores da Imagem por Coluna" foram atribuídos graus de evidência equivalentes a porcentagem de pixels pretos em cada coluna da matriz binária da imagem, por exemplo, a lista de graus de evidência criada para representar as cores das colunas da Figura 19(a) é equivalente à [0,82, 0,87, 0,92, 0,95, 0,97, 1, 1, 0,68, 0,68, 0,71, 0,71, 0, 0, 0, 0,71, 0,71, 0,71, 0, 0, 0, 0,71, 0,71, 0,68, 0,68, 1, 1, 0,97, 0,92, 0,92, 0,87].

Nos mapeamentos "Componente Principal" e "Autovalores do Componente Principal" utilizamos a análise dos componentes principais que consiste em transformar um conjunto de variáveis originais em um conjunto de variáveis de mesma dimensão, nomeadas componentes principais, através de uma técnica estatística multivariada. Essa análise visa reduzir uma massa de dados com menor perda possível de informação. Para esses mapeamentos utilizamos um algoritmo desenvolvido na linguagem de programação Python de (MATH ∩ PROGRAMMING, 2012) para obter o componente principal e os autovalores do componente principal, que nomeamos como "solução Python".

Para os mapeamentos "Componente Principal" e "Autovalores do Componente Principal" os valores obtidos através da "solução Python" são normalizados no intervalo [0, 1] com base nos valores máximo e mínimo.

Considerando a Figura 19(a) temos, ao final do processo de mapeamento da imagem, as seguintes listas de graus de evidência:

- Uma lista referente à Todos os Pixels da Matriz Binária
- Quatro listas referentes à Borda Externa de cada lado da imagem (lado esquerdo, direito, topo e base);
- Quatro listas referentes à Borda Interna de cada lado da imagem (lado esquerdo, direito, topo e base);
- Uma lista referente à Borda Diagonal Interna (direita/esquerda);
- Uma lista referente à Borda Diagonal Interna (esquerda/direita);
- Uma lista referente à Cores da Imagem;
- Uma lista referente à Cores da Imagem por Coluna;
- Uma lista referente ao Componente Principal e;
- Uma lista referente aos Autovalores do Componente Principal.

O conjunto dessas listas forma o conjunto de entradas do processo de extração de características da imagem.

3.2. Extração de características

O processo de extração de característica da imagem consiste em agrupar algumas qualidades do caractere e transformá-las em conjuntos de dados expressos em graus de evidência de modo que possam ser transmitidos e recebidos pelo processo de reconhecimento. Trata-se de uma abstração que caracteriza o caractere e, em geral, permite distinguir um caractere do outro.

Como o processo de reconhecimento é realizado a partir de algumas qualidades do caractere, escolhemos algumas características com base no estudo de algumas técnicas do Grafismo tais como a técnica "valores curvilíneos" da Grafoscopia e as técnicas "altura", "largura" e "direção" da Grafologia (Figura 1), para compor o processo de extração de características do sistema proposto.

As seguintes características foram definidas para o processo de extração de características da imagem: "Segmentos de Reta Verticais" e "Segmentos de Reta

Horizontais" com base nas técnicas de "altura" e "largura" da Grafoscopia; "Histograma: Todos os Pixels da Matriz Binária" com base na técnica de "direção" da Grafologia; "Histograma: Borda Externa Superior", "Histograma:: Borda Externa", "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (direita/esquerda)" e "Histograma: Borda Diagonal Interna (esquerda/direita)" com base na técnica de "valores curvilíneos" da Grafoscopia; "Histograma: Cores da Imagem", "Histograma: Cores da Imagem por Coluna". "Histograma: Componente Principal" e "Histograma: Autovalores do Componente Principal" base na técnica de "regularidade" da Grafologia.

Conforme apresentado na Figura 16, o processo de extração das características é realizado a partir de conjuntos de graus de evidência obtidos no processo de mapeamento da imagem em graus de evidência.

As duas primeiras características, "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais", consistem em analisar o caractere com base apenas nos segmentos de reta verticais e horizontais do contorno externo do caractere e, criar um conjunto de graus de evidência para cada característica (Segmentos de Reta Verticais e Segmentos de Reta Horizontais) de modo que represente essas qualidades do caractere. Para esse procedimento utilizamos uma UNAP conforme apresentada na Figura 20, para detectar os segmentos de reta verticais e horizontais do caractere na imagem.

Essa UNAP nomeada Unidade das Linhas Verticais e Horizontais (ULVH) recebe como entrada os graus de evidências da borda externa do caractere obtidos no processo de mapeamento da imagem e, como saída, gera uma lista com os segmentos de reta identificados no contorno externo do caractere, onde cada segmento de reta compõe-se por um conjunto de graus de evidência com suas peculiaridades descritas a seguir.

A primeira camada da ULVH compõe-se de um conjunto de UNAPs nomeadas Unidade das Células da Primeira Camada (UCPC) e as demais camadas são formadas por outro tipo de UNAPs nomeadas Unidade das Células das Camadas Finais (UCCF).

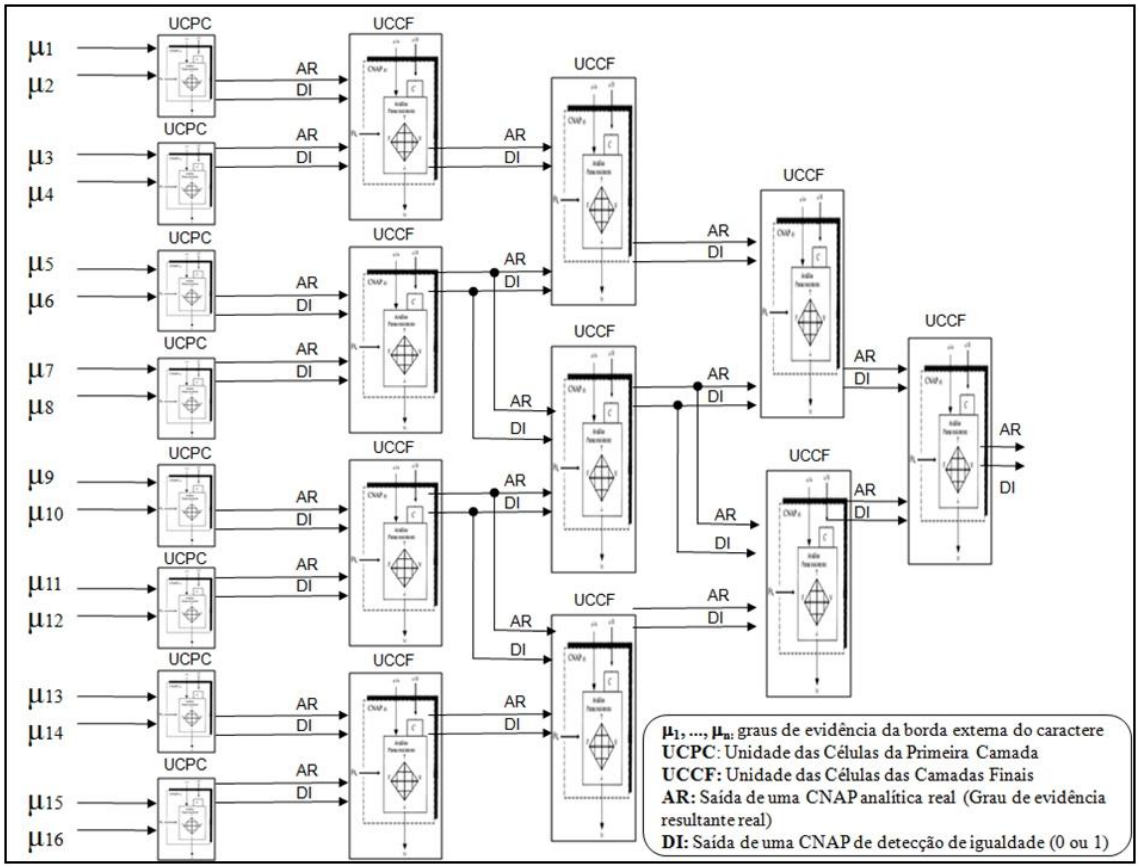


Figura 20 - Arquitetura da Unidade das Linhas Verticais e Horizontais (ULVH)

Na primeira camada dessa arquitetura, cada UCPC recebe duas entradas da sequência dos graus de evidência ordenadamente.

Na camada subsequente (segunda), cada UCCF recebe duas entradas da sequência das saídas da primeira camada ordenadamente.

A partir da terceira camada as UCCF são interconectadas através da segunda entrada da UCCF anterior.

Na primeira camada cada UCPC compõe-se por duas células, CNAPar e CNAPdi, conforme Figura 21.

A CNAPar calcula e retorna o grau de evidência real diminuindo, assim, a inconsistência das entradas, conforme Da Silva Filho et al. (2008). Nas arquiteturas das UCPC e UCCF representa-se o valor retornado por uma célula desse tipo pela sigla ‘AR’.

A CNAPdi verifica se as entradas são iguais retornando os valores 0 ou 1. Nas arquiteturas das UCPC e UCCF representa-se o valor retornado por uma célula desse tipo pela sigla ‘DI’.

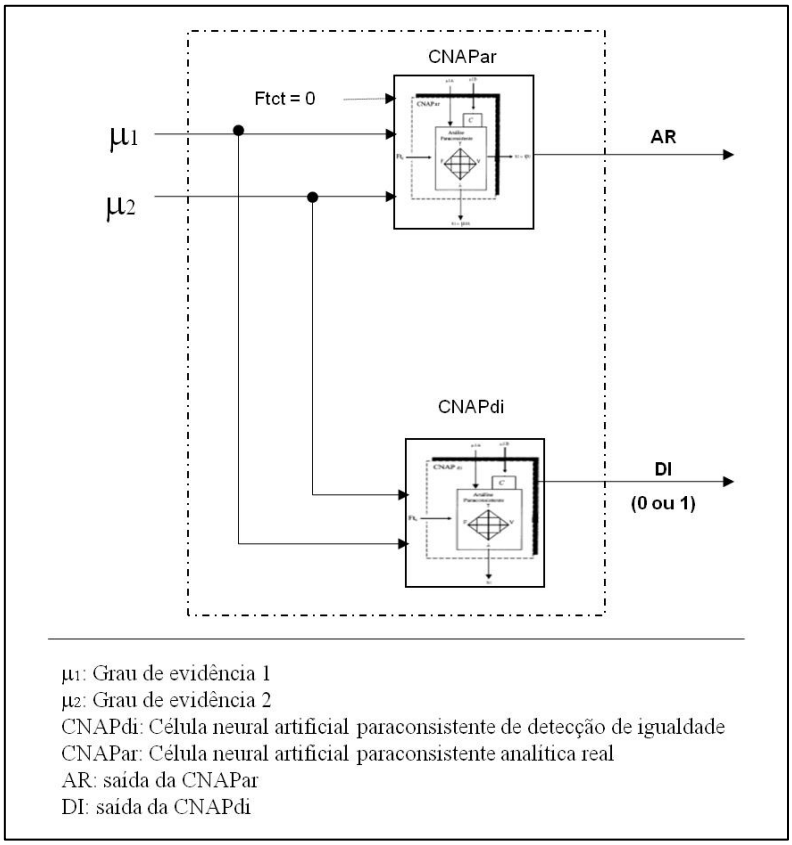


Figura 21 - Arquitetura da Unidade das Células da Primeira Camada (UCPC)

A UCPC retorna uma lista com o resultado de ambas as células, CNAPar e CNAPdi, representadas pelas siglas AR e DI.

Nas demais camadas cada UCCF compõe-se por oito CNAPs, sendo três CNAPcls de Minimização, uma CNAPcls de Maximização, três CNAPdi e uma CNAPar, conforme Figura 22.

Esta unidade recebe como entrada duas unidades sequenciais da camada anterior (UCPC ou UCCF).

A célula 6 é uma CNAPar que recebe como entrada as saídas AR da camada anterior e calcula o grau de evidência real desses valores. O grau de evidência real calculado é a saída AR desta UCCF.

A célula 4 é a CNAPdi principal que recebe como entrada as saídas DI da camada anterior e verifica se existe igualdade entre elas, informando, assim, se existe ou não, um segmento de reta. Esta célula é influenciada pela saída da célula 3 que representa o fator de contradição. Independente dos valores de entrada, se o fator de contradição for

igual a 0, então a saída dessa célula será 0, indicando que não existe um segmento de reta. A saída dessa célula será a saída DI desta UCCF.

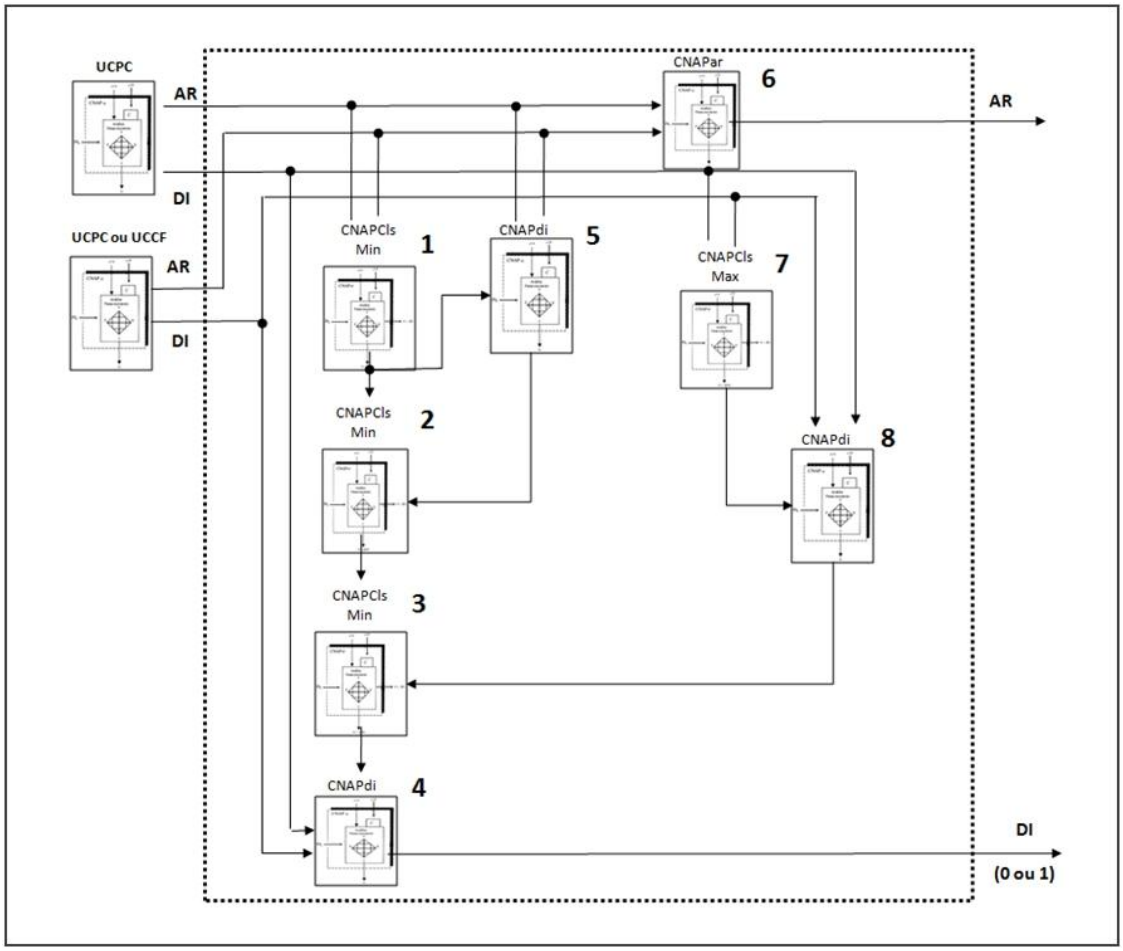


Figura 22 - Arquitetura da Unidade das Células das Camadas Finais (UCCF) onde:
UCPC - Unidade das Células da Primeira Camada, AR - Grau de evidência resultante real, DI - valor 0 ou 1, CNAPcls Min - Célula Neural Artificial Paraconsistente de Conexão Lógica de Minimização, CNAPcls Max - Célula Neural Artificial Paraconsistente de Conexão Lógica de Maximização, CNAPdi - Célula Neural Artificial Paraconsistente de detecção de Igualdade, CNAPar - Célula Neural Artificial Paraconsistente Analítica Real

As demais células foram adicionadas a esta UCCF para tratar três casos especiais apresentados na Figura 23 que detalhamos a seguir.

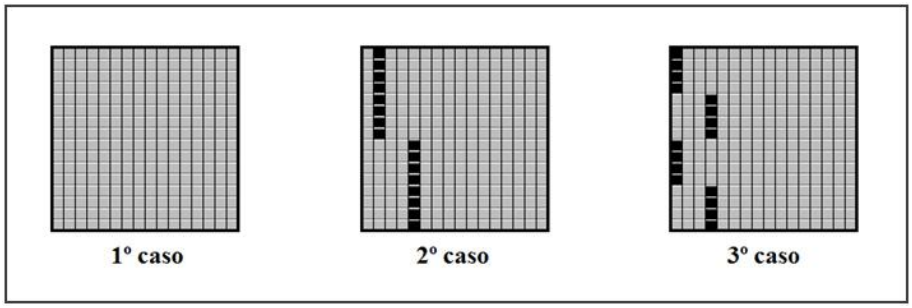


Figura 23 - Exemplos dos casos especiais tratados pela UCCF

1) O primeiro caso ocorre quando as saídas AR e DI da camada anterior são iguais a 0. Isso acontece quando existe ausência de informação.

2) O segundo caso ocorre quando as saídas da camada anterior apresentam os seguintes resultados: saídas AR maiores do que 0 e diferentes entre si e, saídas DI iguais a 1. Isso acontece quando existem dois segmentos de reta distintos em colunas diferentes.

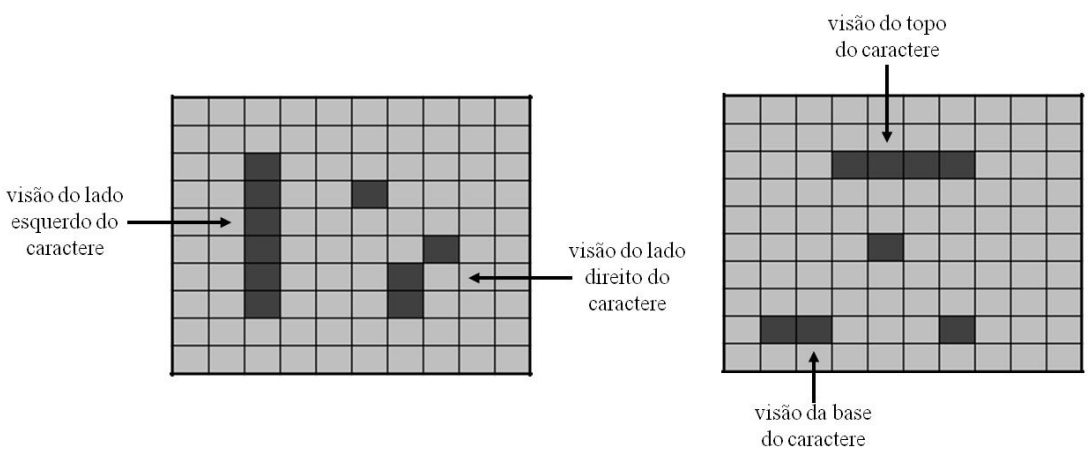
3) O terceiro e último caso ocorre quando as saídas da camada anterior apresentam os seguintes resultados: saídas AR maiores do que 0 e iguais e, saídas DI iguais a 0. Isso acontece quando existem dois segmentos de reta distintos na camada anterior para cada entrada da UCCF. Nesse caso são analisados dois blocos de oito células, onde cada bloco possui dois segmentos de reta distintos.

Assim, os segmentos de reta são identificados a partir da análise das saídas DI retornadas pelas camadas da rede.

Entende-se que por uma sequencia de pixels ‘alinhada’ usaremos o termo segmento de reta por abuso de linguagem. Outros abusos de linguagem serão utilizados conscientemente para maior fluidez na exposição.

Quando a última camada da rede apresenta a saída DI igual a 1, significa que existe um único segmento de reta com tamanho máximo. Caso contrário, o sistema percorre as camadas anteriores em busca das saídas DI igual a 1, que caracterizam um segmento de reta.

Na extração das características "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais", utilizamos quatro UNAPs desse tipo (ULVH), duas para Segmentos de Reta Verticais e duas para Segmentos de Reta Horizontais. Através dessas UNAPs as características "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" da imagem são identificadas e agrupadas individualmente em forma de graus de evidência representando as seguintes informações conforme exemplo na Figura 24:



Característica "Segmentos de Reta Verticais"
visão do lado esquerdo da imagem
Direção: 1
Tamanho mínimo do segmento de reta: 0,3
Localização na coluna: 0,6 (próximo do centro)
Localização na linha: 0,3 (próximo à esquerda)

visão do lado direito da imagem
Direção: 0
Tamanho mínimo do segmento de reta: 0,16
Localização na coluna: 0,8 (próximo à base)
Localização na linha: 0,7 (próximo à direita)

Característica "Segmentos de Reta Horizontais"
visão do topo da imagem
Direção: 1
Tamanho mínimo do segmento de reta: 0,3
Localização na linha: 0,6 (próximo do centro)
Localização na coluna: 0,3 (próximo ao topo)

visão da base da imagem
Direção: 0
Tamanho mínimo do segmento de reta: 0,16
Localização na linha: 0,3 (próximo à esquerda)
Localização na coluna: 0,9 (próximo à base)

Figura 24 - Exemplo de segmentos de reta verticais e horizontais obtidos de uma imagem

- Direção (lado analisado do caractere - esquerdo, direito, topo ou base). Atribuiu-se o valor '1' para o lado esquerdo e o topo e, o valor '0' para o lado direito e a base do caractere.

- Tamanho mínimo estimado do segmento de reta. Valor no intervalo [0, 1] calculado com base no comprimento do segmento de reta (quantidade mínima de pixels do segmento de reta). Primeiramente calcula-se o valor através da multiplicação do índice da camada onde o segmento de reta foi encontrado por 4 (se índice da camada maior do que 0) ou por 2 (se índice da camada igual a 0). Após obter o tamanho do segmento de reta esse valor é normalizado no intervalo [0, 1] através da divisão do valor obtido pela quantidade de graus de evidência da entrada da UCPC. Utilizando como exemplo o lado esquerdo do caractere apresentado na Figura 25 encontra-se o segmento de reta na segunda camada

(índice da camada = 1), portanto, o tamanho mínimo do caractere será igual a (1 * 4) / 12, ou seja, 0,3.

- Localização do segmento de reta na coluna para a característica "Segmentos de Reta Verticais" (topo, centro, base) ou na coluna para a característica "Segmentos de Reta Horizontais" (lado esquerdo, centro, lado direito). Valor no intervalo [0, 1] equivalente à posição do segmento de reta na coluna/linha da matriz do caractere considerando que o valor do grau de evidência é crescente da esquerda para a direita e do topo para a base. Calcula-se o valor através da divisão da posição do elemento que representa o segmento de reta identificado na lista de elementos da camada da rede pela quantidade de elementos da camada. Utilizando como exemplo o lado esquerdo do caractere apresentado na Figura 25 temos o segmento de reta encontrado na segunda posição da segunda camada da rede, portanto, divide-se 2 pela quantidade de elementos da camada 3, ou seja, 2 / 3 = 0,66.

- Localização do segmento de reta na linha para a característica "Segmentos de Reta Verticais" e na coluna para a característica "Segmentos de Reta Horizontais". Valor no intervalo [0, 1] equivalente à posição do segmento de reta na linha/coluna da matriz do caractere considerando que o valor do grau de evidência é crescente da esquerda para a direita e do topo para a base. Esse valor é obtido através da saída AR do elemento que representa o segmento de reta identificado. Como exemplo, temos na Figura 25 o valor 0,3 no segundo elemento da segunda camada da rede para o lado esquerdo.

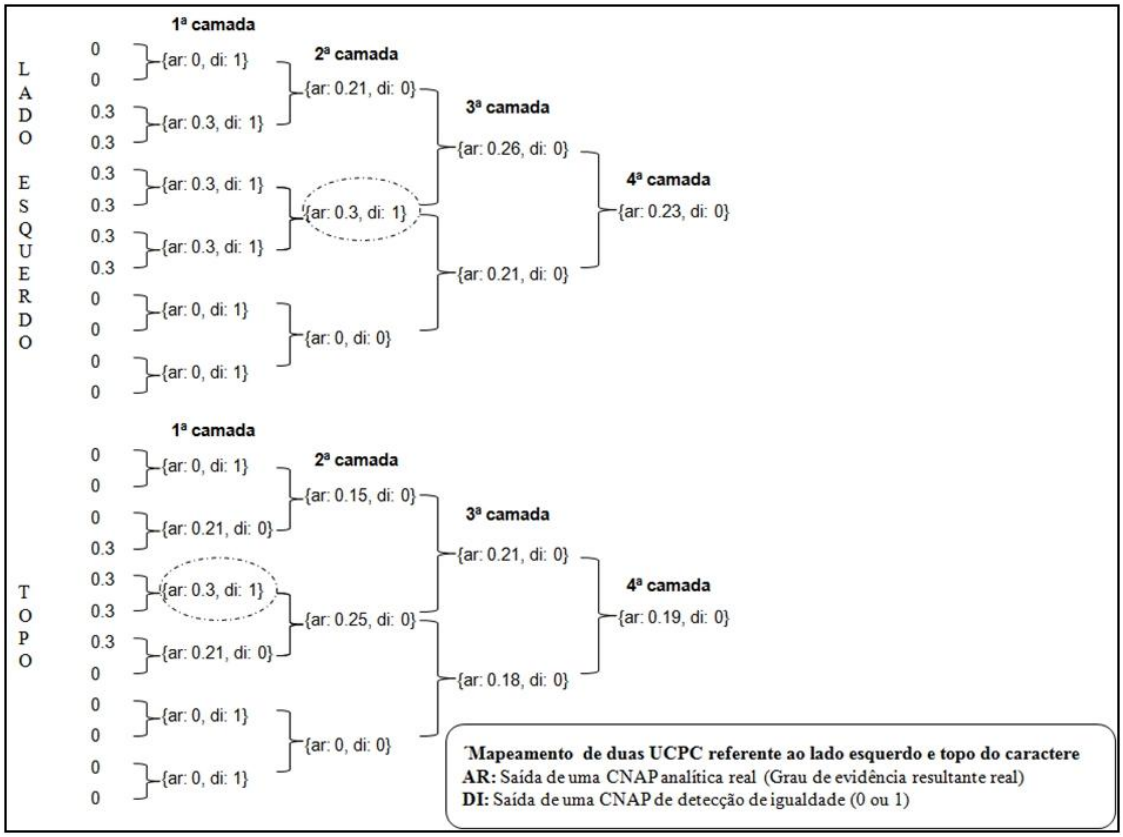


Figura 25 - Exemplo de duas UCPC referente ao lado esquerdo e topo da imagem com os valores obtidos nas camadas de uma ULVH

Para as demais características nomeadas como histogramas, os grupos de graus de evidência obtidos no processo de mapeamento da imagem são agrupados ordenadamente conforme descrito a seguir:

- Cria-se uma lista de graus de evidência para "Histograma: Todos os Pixels da Matriz Binária" através da sequência dos graus de evidência obtidos no processo de mapeamento da imagem, ordenados pelas colunas da matriz do caractere. (Figura 26);

- Cria-se uma lista de graus de evidência, uma para "Histograma: Borda Externa Superior" através da sequencia dos graus de evidência obtidos no processo de mapeamento referente à borda externa do topo da imagem.(Figura 26);

- Criam-se duas listas de graus de evidência, uma para "Histograma: Borda Externa" e outra para "Histograma: Borda Interna" através da sequencia dos graus de evidência obtidos no processo de mapeamento da imagem ordenados por: lado esquerdo, base, topo e lado direito da imagem (Figura 26);

- Criam-se duas listas de graus de evidência, uma para "Histograma: Borda Diagonal Interna (direita/esquerda)" e outra para "Histograma: Borda Diagonal Interna (esquerda/direita)" da sequência dos graus de evidência obtidos no processo de mapeamento da imagem ordenados por: topo e base da borda diagonal interna superior e, na sequência, topo e base da borda diagonal interior inferior da imagem (Figura 26);

- Cria-se uma lista de graus de evidência para "Histograma: Cores da Imagem" com os níveis de cores da imagem como um todo através da sequência dos graus de evidência obtidos no processo de mapeamento da imagem ordenados por: porcentagem de pixels pretos e porcentagem de pixels brancos (Figura 26);

- Cria-se uma lista de graus de evidência para "Histograma: Cores da Imagem por Coluna" com a quantidade de pixels pretos em cada coluna da imagem através da sequencia dos graus de evidência obtidos no processo de mapeamento da imagem, ordenados por coluna (Figura 26);

- Cria-se uma lista de graus de evidência para "Histograma: Componente Principal" através da sequência dos graus de evidência obtidos no processo de mapeamento da imagem, ordenados por coluna (Figura 26);

- Cria-se uma lista de graus de evidência para "Histograma: Autovalores do Componente Principal" através da sequência dos graus de evidência obtidos no processo de mapeamento da imagem, ordenados por coluna (Figura 26).

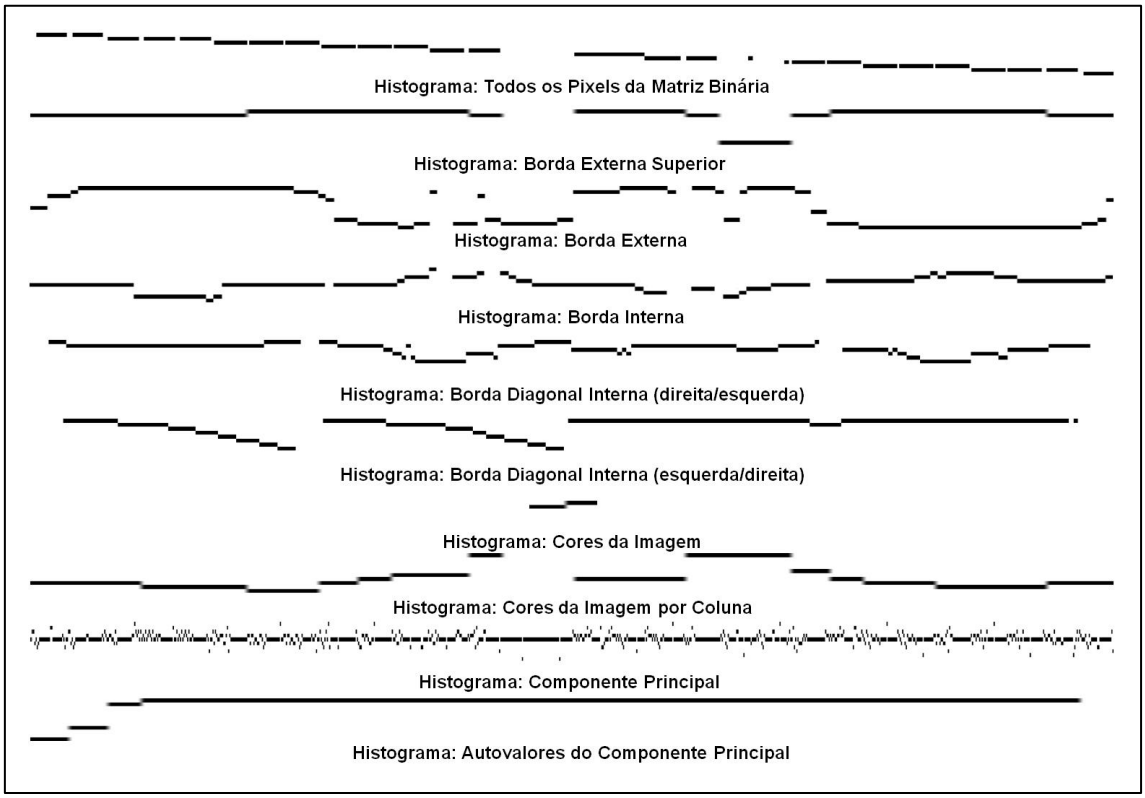


Figura 26 - Exemplo dos histogramas obtidos no processo de extração de características

Ao final desse processo, temos as seguintes listas de graus de evidência referentes às características extraídas do caractere, as quais compõe a entrada do processo de reconhecimento: segmentos de reta verticais identificados de ambos os lados (esquerdo e direito), segmentos de reta horizontais identificados no topo e na base da imagem, histograma de todos os pixels da matriz binária, histograma da borda externa superior, histograma da borda externa, histograma da borda interna, histograma da borda diagonal interna (direita/esquerda), histograma da borda diagonal interna (esquerda/direita), histograma da cores da imagem como um todo, histograma das cores da imagem por coluna, histograma do componente principal e histograma dos autovalores do componente principal.

3.3. Reconhecimento da imagem

No processo de reconhecimento realiza-se uma análise das características do caractere apresentado para cada caractere padrão. O grupo de caracteres padrão do tipo CMC-7 e manuscritos são apresentados na Figura 27.

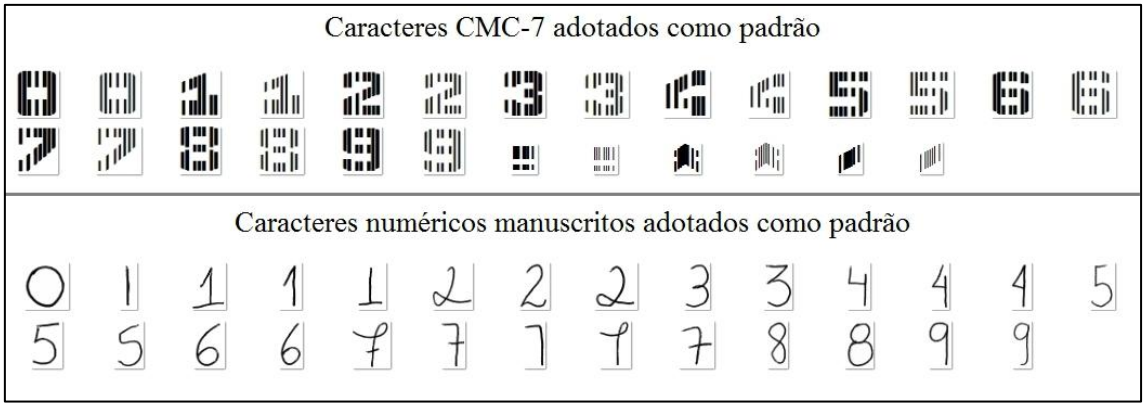


Figura 27 - Conjunto de caracteres dos tipos CMC-7 e numérico manuscrito adotados com caracteres padrão

Para cada análise são realizados os seguintes procedimentos:

1. Para cada caractere padrão, comparam-se as características do caractere apresentado com as características do caractere padrão selecionado;
2. Obtém-se um valor único que represente o grau de evidência de reconhecimento de cada característica comparada do caractere padrão selecionado;
3. Calcula-se um único grau de evidência de reconhecimento do caractere padrão selecionado;
4. Identifica-se o caractere padrão com maior grau de evidência, o qual representa o caractere reconhecido pelo sistema;
5. Identifica-se o caractere padrão com menor grau de evidência, o qual representa o caractere descartado pelo sistema.

O primeiro procedimento é composto por duas UARLs para fazer o reconhecimento das características "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" e, uma unidade de análise paraconsistente nomeada Unidade de Análise Paraconsistente $ParaExtr_{ctr}$ (UAPParaExtr) para cada histograma. Uma UAPParaExtr, que será explicada com mais detalhes a frente, possui como principal componente o Algoritmo Extrator de Efeitos da Contradição (Da Silva Filho, 2009) que, a partir de um conjunto de Graus de Evidência extrai gradativamente a contradição e oferece como resultado um único Grau de Evidencia final representante do grupo.

Assim, conforme mostra a Figura 28, em cada análise comparamos os graus de evidência obtidos no processo de extração de características do caractere apresentado, com os graus de evidência obtidos de cada caractere padrão, ou seja, realiza-se um processamento de reconhecimento de características para cada caractere padrão através do uso de RNAPs, onde a saída representa o grau de evidência de reconhecimento das características do mesmo.

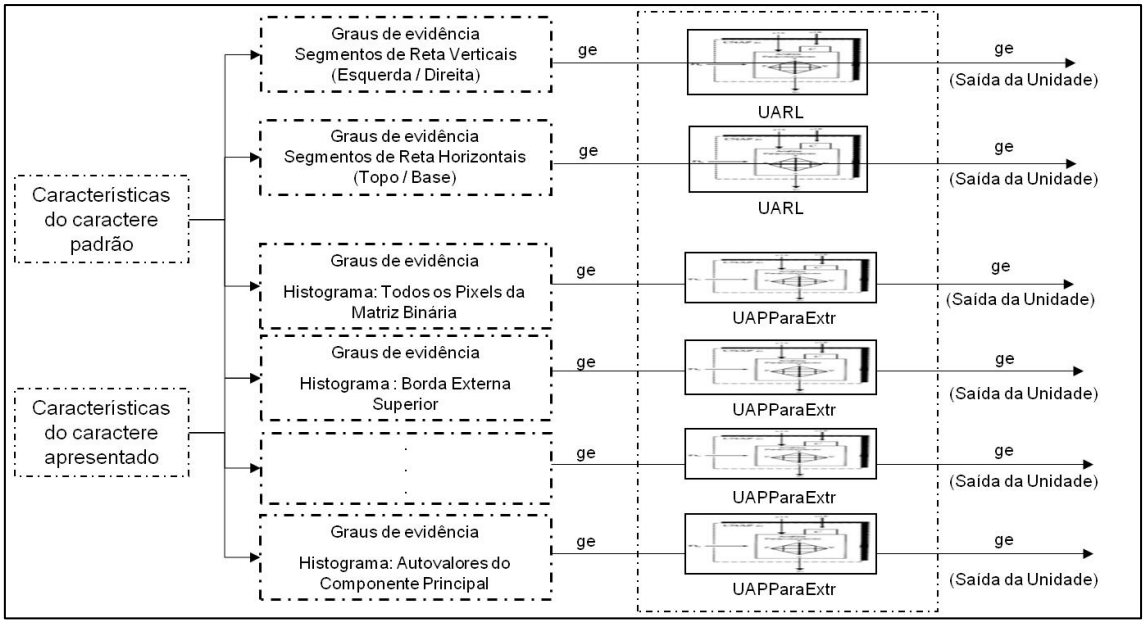


Figura 28 - Processo de reconhecimento das características do caractere apresentado

A UARL, Figura 29, responsabiliza-se por realizar o reconhecimento das características "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" através de duas unidades URP e, transformar a saídas dessas unidades em um único grau de evidência através de uma CNAPa.

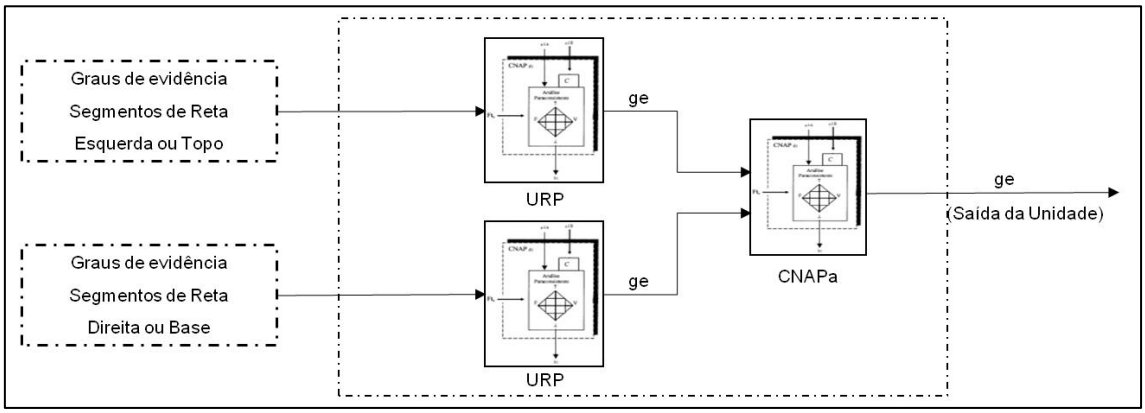


Figura 29 - Arquitetura da Unidade de Agrupamento do Reconhecimento de Linhas (UARL) onde:
URP - Unidade de Reconhecimento de Padrões, ge - grau de evidência resultante, CNAPa - Célula Neural Artificial Paraconsistente Analítica

A URP, Figura 30, realiza, apenas para as características "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais", a comparação entre os graus de evidência obtidos no processo de extração de característica do caractere apresentado e os graus de evidência de um determinado caractere padrão.

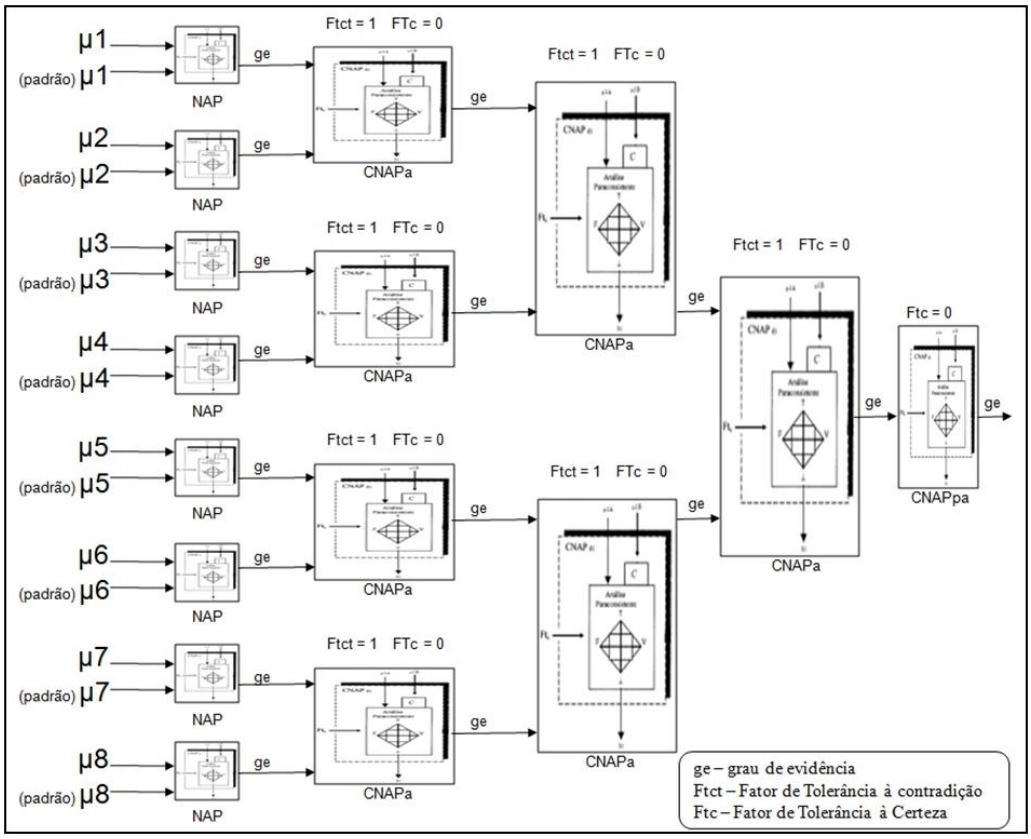


Figura 30 - Arquitetura da Unidade de Reconhecimento de Padrões (URP) onde:
μ_1, ..., μ_n - graus de evidência das características do caractere apresentado, (padrão) μ_1, ..., (padrão) μ_n - graus de evidência das características do caractere padrão, NAP - Nó de Análise Paraconsistente, CNAPdi - Célula Neural Artificial Paraconsistente de Detecção de Igualdade, CNAPa - Célula Neural Artificial Paraconsistente Analítica, CNAPpa - Célula Neural Artificial Paraconsistente de Passagem

Essa URP recebe dois grupos de graus de evidência. Um refere-se ao caractere apresentado ao sistema e o outro a um caractere padrão. A entrada desses dados são organizados em pares e apresentados, na primeira camada, a um algoritmo paraconsistente denominado de Nó de Análise Paraconsistente (NAP) (Da Silva Filho et al, 2010) responsável por calcular o intervalo entre as duas entradas (Figura 31). Posteriormente, as saídas de cada NAP são encaminhadas para as demais camadas, compostas por CNAPa, para transformar essas saídas em um único grau de evidência resultante (segundo procedimento). O algoritmo do NAP é apresentado no Apêndice F.

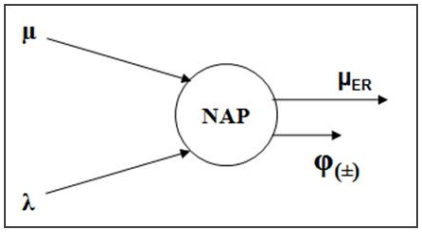


Figura 31 – Representação simbólica de um Nó de Análise Paraconsistente (NAP) onde:
μ - evidência favorável, λ - evidência desfavorável ou contrária, μ_{ER} - Grau de Evidência Resultante Real, φ - Intervalo de Evidência Resultante

Após fazer o reconhecimento das características "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais", o sistema inicia o processamento de reconhecimento das características representadas por histogramas utilizando uma UAPParaExtr para cada característica.

A UAPParaExtr recebe duas listas de graus de evidência sendo uma da característica do caractere apresentado e outra da característica do caractere padrão. Para transformar as saídas dessas unidades em um único grau de evidência que representará a saída final, ou seja, o grau de evidência de reconhecimento da característica, utilizamos um NAP para criar uma única lista que representa o intervalo entre os pares de graus de evidência do caractere apresentado e do caractere padrão e, na sequência, o Algoritmo Paraconsistente Extrator de Efeitos da Contradição ($ParaExtr_{ctr}$) desenvolvido por Da Silva Filho (2009), Figura 32, para calcular um único grau de evidência para a característica analisada (segundo procedimento), o qual tem a característica de extrair, de forma gradativa, os efeitos da contradição em sinais que representam conhecimento incerto. O Algoritmo $ParaExtr_{ctr}$ é apresentado no Apêndice F.

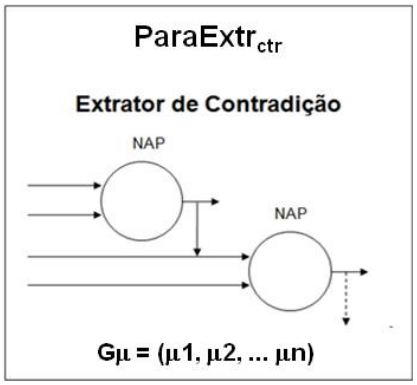


Figura 32 – Algoritmo $ParaExtr_{ctr}$ (Da Silva Filho, 2009)

Para cada histograma compara-se item a item os graus de evidência obtidos do caractere padrão e do caractere apresentado através de uma UAPParaExtr. Para cada par de graus de evidência calculamos o valor indicativo de quanto na análise paraconsistente a contradição está influenciando sobre o valor obtido do grau de evidência resultante real, ou seja, o Intervalo de Evidência (primeiro procedimento).

Considerando que GePadrão representa a lista de graus de evidência do caractere padrão, GeCaractere representa a lista de graus de evidência do caractere apresentado, GeIntervalo representa a lista de graus de evidência resultante do cálculo do intervalo entre as listas GePadrão e GeCaractere, GeP representa um grau de evidência do caractere padrão e GeC representa um grau de evidência do caractere apresentado, temos "μ = GeP" e "λ = (1 – GeC)" para cada par de entradas do NAP, onde cada valor do intervalo de evidência calculado no NAP (Apêndice F - Algoritmo do Nó de Análise Paraconsistente - Passo 10 - Saída S_2) é adicionado à nova lista de graus de evidência nomeada GeIntervalo.

Ex.

GePadrão	=	[0.8, 0.3, 0.7, 0.2]
GeCaractere	=	[0.6, 0.3, 0.2, 0.5]
---	---	---
GeIntervalo	=	[0.8, 1.0, 0.5, 0.7]

Após calcular os graus de evidência referentes ao Intervalo de Evidência entre as características dos caracteres, a lista GeIntervalo é submetida ao algoritmo $ParaExtr_{ctr}$ (Apêndice F - Algoritmo $ParaExtr_{ctr}$) para obter um único grau de evidência que representa o grau de evidência de reconhecimento da característica selecionada (segundo processo).

Ex. [0.8, 1.0, 0.5, 0.7] $\therefore$ 0,71

Na primeira etapa do algoritmo selecionam-se os graus de evidência de maior e menor valor (μ_{max} e μ_{min}) de uma lista de graus de evidência, os quais compõem a entrada do NAP, para a extração dos efeitos da contradição: "$\mu = \mu_{max}$" e "$\lambda = 1 - \mu_{min}$". Ele responsabiliza-se por calcular o intervalo entre as entradas μ e λ.

Após a primeira análise, os graus de evidência selecionados para a primeira etapa são substituídos pelo resultado da análise realizada pelo NAP e, inicia-se a próxima etapa seguindo os mesmos critérios da primeira. Como resultado desse algoritmo, se obtém um único valor que representa o grau de evidência resultante.

Depois de obter um único grau de evidência para cada característica, realiza-se o terceiro procedimento através do algoritmo $ParaExtr_{ctr}$, responsável por calcular um único grau de evidência para o caractere padrão selecionado. Como entrada desse procedimento considera-se uma lista composta pelos graus de evidência calculados para cada característica.

Nesse procedimento, primeiramente, cria-se uma lista apenas com os graus de evidência das características "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" e aplica-se o algoritmo $ParaExtr_{ctr}$, na sequência, cria-se uma nova lista com o valor resultante da primeira lista e os graus de evidência das demais características conforme mostra a Figura 33.

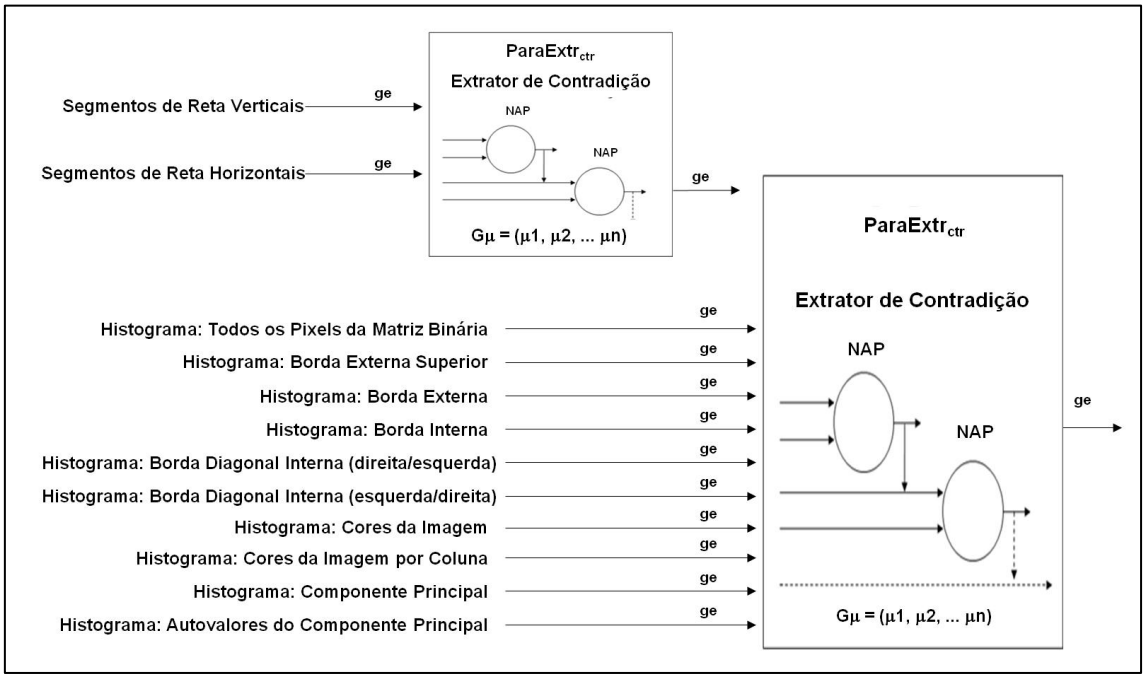


Figura 33 – Esquematização do algoritmo $ParaExtr_{ctr}$ onde:
ge - grau de evidência, NAP - Nó de Análise Paraconsistente, $ParaExtr_{ctr}$ - Algoritmo Paraconsistente Extrator de Efeitos da Contradição, Gμ - Lista de graus de evidência, μ - grau de evidência

Ao final do processo de reconhecimento, cada caractere padrão exibe um grau de evidência, onde o maior valor representa o caractere reconhecido e, o menor valor representa o caractere descartado pelo sistema computacional paraconsistente proposto conforme arquitetura do sistema apresentada na Figura 16.

Assim, o conjunto desses processos são realizados para cada caractere apresentado ao sistema.

A seguir apresenta-se alguns testes realizados com amostras de caracteres dos tipos CMC-7 e numérico manuscrito com os melhores resultados obtidos para cada categoria.

4 RESULTADOS

Para avaliar o desempenho do sistema, a princípio, foram realizados testes com caracteres do tipo CMC-7 e, posteriormente, com caracteres numéricos manuscritos.

Para os testes com os caracteres do tipo CMC-7 foi selecionada uma amostra com 2.092 caracteres distintos a partir de cheques reais digitalizados.

E, para os testes com os caracteres numéricos manuscritos, foi selecionada uma amostra de 1.050 caracteres digitalizados distintos.

Inicialmente o sistema foi implementado e configurado para fazer o reconhecimento dos caracteres considerando apenas as características "Segmentos de Reta Verticais", "Segmentos de Reta Horizontais" e "Borda Externa".

Através dessa primeira configuração do sistema os resultados apresentados para o reconhecimento da amostra de caracteres do tipo CMC-7 foram: 78,92% de acertos, 78,12% de sensibilidade, 98,16% de especificidade, 88,18% de eficiência e 96,77% de acurácia, conforme Tabela 1.

Tabela 1 - Testes realizados pelo sistema configurado com as características "Borda Externa", "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Borda Externa + Segmentos de Reta Verticais + Segmentos de Reta Horizontais							
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	344	92	0,789	0,924	0,857	0,896
1	264	248	16	0,939	0,994	0,967	0,987
2	122	101	21	0,823	0,98	0,904	0,971
3	153	115	38	0,752	0,975	0,863	0,959
4	152	151	1	0,993	0,999	0,996	0,999
5	190	153	37	0,805	0,994	0,9	0,977
6	111	78	33	0,703	0,928	0,815	0,916
7	98	94	4	0,959	0,998	0,979	0,997
8	146	58	88	0,397	0,996	0,697	0,954
9	102	46	56	0,451	0,992	0,722	0,966
⑈	138	98	40	0,71	0,995	0,853	0,977
⑆	90	79	11	0,878	0,988	0,933	0,984
⑇	90	86	4	0,956	0,998	0,977	0,997
Total	2 092	78,92%	21,08%	78,12%	98,16%	88,18%	96,77%

E, os resultados apresentados para o reconhecimento da amostra de caracteres do tipo numérico manuscrito foram: 79,24% de acertos, 79,19% de sensibilidade, 97,67% de especificidade, 88,44% de eficiência e 95,86% de acurácia conforme Tabela 2.

Tabela 2 - Testes realizados pelo sistema configurado com as características "Borda Externa", "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" para o reconhecimentos de caracteres do tipo Manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características: Borda Externa + Segmentos de Reta Verticais + Segmentos de Reta Horizontais							
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	13	96	0,119	0,966	0,543	0,878
1	92	56	36	0,609	0,986	0,798	0,953
2	121	107	14	0,884	0,987	0,936	0,975
3	134	129	5	0,963	0,993	0,978	0,99
4	85	73	12	0,859	0,993	0,926	0,982
5	103	93	10	0,903	0,997	0,95	0,988
6	104	104	0	1	0,985	0,993	0,987
7	88	86	2	0,9777	0,987	0,982	0,987
8	116	89	27	0,767	0,875	0,821	0,863
9	98	82	16	0,837	0,998	0,917	0,983
Total	1 050	79,24%	20,76%	79,19%	97,67%	88,44%	95,86%

Posteriormente, o sistema foi aperfeiçoado através da implementação de novos tipos de mapeamento da imagem como "Todos os Pixels da Matriz Binária", "Borda Interna", "Borda Diagonal Interna (direita/esquerda)", "Borda Diagonal Interna (esquerda/direita)", "Core da Imagem", "Cores da Imagem por Coluna", "Componente Principal" e "Autovalores do Componente Principal" e novos tipos de extração de características como "Histograma: Todos os Pixels da Matriz Binária", "Histograma: Borda Externa Superior", "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (direita/esquerda)", "Borda Diagonal Interna (esquerda/direita", "Histograma: Cores da Imagem", "Histograma: Cores da Imagem por Coluna", "Histograma: Componente Principal" e "Histograma: Autovalores do Componente Principal" e novos testes foram realizados para ambos os tipos de caracteres, CMC-7 e numéricos manuscritos, com diferentes tipos de combinação de características conforme apresentado nos Apêndices A e B.

Após analisarmos os testes realizados com os caracteres do tipo CMC-7 apresentados no Apêndice A observamos que os melhores resultados foram obtidos a partir da combinação das seguintes características: "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (direita/esquerda), conforme mostra Tabela 3 e, "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (esquerda/direita), conforme mostra Tabela 4. Ambas as configurações apresentaram 97,85% de acertos, 97,38% de sensibilidade, 99,81% de especificidade, 98,62% de eficiência e 99,67% de acurácia.

Tabela 3 - Testes realizados pelo sistema configurado com as características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (direita/esquerda)							
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	434	2	0,995	0,999	0,997	0,999
1	264	261	3	0,989	0,999	0,994	0,998
2	122	122	0	1	0,999	1	0,999
3	153	152	1	0,993	0,998	0,996	0,998
4	152	151	1	0,993	1	0,997	0,999
5	190	179	11	0,942	1	0,71	0,995
6	111	109	2	0,982	0,997	0,99	0,997
7	98	94	4	0,959	1	0,98	0,998
8	146	143	3	0,979	0,998	0,989	0,997
9	102	96	6	0,941	0,997	0,969	0,994
⑈	138	133	5	0,964	0,999	0,982	0,997
⑆	90	87	3	0,967	0,99	0,979	0,989
⑇	90	86	4	0,956	0,999	0,977	0,997
Total	2 092	97,85%	2,15%	97,38%	99,81%	96,62%	99,67%

Após analisarmos os testes realizados com os caracteres do tipo numérico manuscrito apresentados no Apêndice B observamos que os melhores resultados foram obtidos a partir da combinação das seguintes características: "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (direita/esquerda), conforme mostra

Tabela 5. Essa configuração apresentou 91,62% de acertos, 91,27% de sensibilidade, 99,07% de especificidade, 95,16% de eficiência e 98,32% de acurácia.

Tabela 4 - Testes realizados pelo sistema configurado com as características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (esquerda/direita)							
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	434	2	0,995	0,999	0,997	0,999
1	264	261	3	0,989	0,999	0,994	0,998
2	122	122	0	1	0,999	1	0,999
3	153	152	1	0,993	0,998	0,996	0,998
4	152	151	1	0,993	1	0,997	0,999
5	190	179	11	0,942	1	0,971	0,995
6	111	109	2	0,982	0,997	0,99	0,997
7	98	94	4	0,959	1	0,98	0,998
8	146	143	3	0,979	0,998	0,989	0,997
9	102	96	6	0,941	0,997	0,969	0,994
⑈	138	133	5	0,964	0,999	0,982	0,997
⑆	90	87	3	0,967	0,99	0,979	0,989
⑇	90	86	4	0,956	0,999	0,977	0,997
Total	2 092	97,85%	2,15%	97,38%	99,81%	98,62%	99,67%

Tabela 5 - Testes realizados pelo sistema configurado com as características "Histograma: Borda Externa", "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda)							
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	109	0	1	0,994	0,997	0,994
1	92	62	30	0,674	0,996	0,835	0,968
2	121	118	3	0,975	0,986	0,981	0,985
3	134	123	11	0,918	0,993	0,956	0,984
4	85	77	8	0,906	0,997	0,951	0,989
5	103	81	22	0,786	0,996	0,891	0,975
6	104	103	1	0,99	0,975	0,982	0,976
7	88	87	1	0,989	0,987	0,988	0,988
8	116	109	7	0,94	0,985	0,962	0,98
9	98	93	5	0,949	0,998	0,973	0,993
Total	1 050	91,62%	8,38%	91,27%	99,07%	95,16%	98,32%

A curva ROC que representa os resultados para caracteres dos tipos CMC-7 e numérico manuscrito descritos nas tabelas 3 e 5 é apresentada no Apêndice D.

Também foi implementado no sistema a opção de comparar as características expressas em histogramas considerando apenas os picos superiores e inferiores do histograma, porém, os resultados obtidos não foram satisfatórios. Para os caracteres do tipo CMC-7 obtivemos apenas 8,03% de acertos considerando os picos das características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (direita/esquerda)" conforme mostra a Tabela 6 e 15,87% de acertos considerando os picos das características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (esquerda/direita)" conforme mostra a Tabela 7. E, para os caracteres do numérico manuscrito obtivemos apenas 18,76% de acertos considerando os picos das características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (direita/esquerda)" conforme mostra a Tabela 8.

Tabela 6 - Testes realizados pelo sistema configurado com os picos superiores e inferiores das características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Picos superiores e inferiores das características Histograma: Borda Interna + Histograma: Borda Diagonal Interna (direita/ esquerda)							
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	30	406	0,069	0,961	0,515	0,775
1	264	10	244	0,038	0,971	0,504	0,853
2	122	1	121	0,009	0,992	0,5	0,935
3	153	4	149	0,026	0,98	0,503	0,911
4	152	2	150	0,013	0,987	0,5	0,916
5	190	3	187	0,015	0,996	0,506	0,907
6	111	2	109	0,018	0,99	0,504	0,938
7	98	15	83	0,153	0,89	0,521	0,855
8	146	4	142	0,027	0,981	0,504	0,915
9	102	2	100	0,02	0,989	0,504	0,942
⑈	138	10	128	0,072	0,966	0,519	0,907
⑆	90	60	30	0,667	0,521	0,594	0,528
⑇	90	25	65	0,278	0,799	0,538	0,777
Total	2 092	8,03%	91,49%	10,81%	92,48%	51,63%	85,84%

Tabela 7 - Testes realizados pelo sistema configurado com os picos superiores e inferiores das características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Picos superiores e inferiores das características Histograma: Borda Interna + Histograma: Borda Diagonal Interna (esquerda/direita)							
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	37	399	0,085	0,987	0,536	0,799
1	264	45	219	0,17	0,932	0,552	0,836
2	122	6	116	0,049	0,976	0,512	0,921
3	153	10	143	0,065	0,973	0,519	0,906
4	152	30	122	0,197	0,982	0,59	0,925
5	190	6	184	0,031	0,993	0,512	0,905
6	111	10	101	0,09	0,97	0,53	0,923
7	98	34	64	0,347	0,802	0,575	0,781
8	146	20	126	0,137	0,954	0,546	0,897
9	102	3	99	0,029	0,972	0,501	0,926
⑈	138	27	111	0,196	0,904	0,55	0,857
⑆	90	58	32	0,644	0,759	0,701	0,754
⑇	90	46	44	0,511	0,901	0,706	0,884
Total	2 092	15,87%	84,13%	19,62%	93,12%	56,38%	87,03%

Tabela 8 - Testes realizados pelo sistema configurado com os picos superiores e inferiores das características "Histograma: Borda Externa", "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Picos superiores e inferiores das características Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda)							
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	5	104	0,046	0,989	0,518	0,891
1	92	76	16	0,826	0,476	0,651	0,507
2	121	25	96	0,201	0,922	0,564	0,84
3	134	1	133	0,001	1	0,504	0,873
4	85	13	72	0,153	0,957	0,555	0,892
5	103	0	103	0	0,498	0,498	0,898
6	104	27	77	0,26	0,935	0,598	0,869
7	88	23	65	0,261	0,933	0,597	0,877
8	116	26	90	0,224	0,898	0,512	0,824
9	98	1	97	0,01	0,996	0,503	0,904
Total	1 050	18,76%	81,24%	19,82%	86,04%	55,00%	83,75%

Além dos testes com caracteres do tipo CMC-7 e numérico manuscrito, optamos por realizar testes com caracteres alfabéticos maiúsculos e minúsculos com as fontes "Arial" e "Times New Roman" e, manuscritos para avaliar como o sistema se comporta com esses tipos de caracteres. Para esses testes não foram selecionados caracteres acentuados e/ou especiais.

As tabelas do Apêndice C mostram:

- os resultados obtidos para o reconhecimento de uma amostra de 1.783 caracteres do tipo alfabético com fonte "Arial", tamanho 12, adotando as características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (direita/esquerda)" e, sem aplicar nenhum pré-tratamento nas imagens. Essa configuração apresentou 82,45% de acertos, 90,97% de sensibilidade, 99,66%% de especificidade, 95,32% de eficiência e 99,34% de acurácia;
- os resultados obtidos para o reconhecimento de uma amostra de 264 caracteres do tipo alfabético com fonte "Times New Roman", tamanho 12, adotando as características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (esquerda/direita)" e, sem aplicar nenhum pré-tratamento nas imagens. Essa configuração apresentou 95,08% de acertos, 95,19% de sensibilidade, 99,90% de especificidade, 97,55% de eficiência e 99,81% de acurácia;
- os resultados obtidos para o reconhecimento de uma amostra de 284 caracteres do tipo alfabético manuscrito, adotando as características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (direita/esquerda)" e, sem aplicar nenhum pré-tratamento nas imagens. Essa configuração apresentou 46,48% de acertos, 47,54% de sensibilidade, 98,95% de especificidade, 73,25% de eficiência e 97,94% de acurácia;

Os testes com caracteres do tipo alfabético foram apenas experimentais, porém, de acordo os resultados obtidos o sistema se mostrou apto para reconhecê-los. Através de algumas adaptações e aperfeiçoamentos, o sistema também estará apto fazer o reconhecimento de caracteres acentuados e especiais com melhor precisão.

5 DISCUSSÃO

O Sistema Computacional Paraconsistente foi construído com o objetivo de desenvolver uma ferramenta capaz de realizar o reconhecimento de caracteres numéricos com um padrão previamente determinado e caracteres numéricos manuscritos através dos métodos descritos no capítulo 3.

O Sistema Computacional Paraconsistente é um sistema web que foi desenvolvido através do uso da linguagem de programação Python e do framework Pyramid. Apresenta-se no Apêndice G o código-fonte dos principais módulos do sistema. Com relação a base de dados dos caracteres padrões e das amostras dos caracteres submetidos aos testes, as imagens foram colocadas em um diretório interno do sistema, ou seja, a base de dados encontra-se em um diretório interno do sistema organizados em pastas. Com relação a base de dados do conjunto de características extraídas dos caracteres padrões e caracteres apresentados das amostras, as características são agrupadas durante o processamento e gravadas em um arquivo texto do tipo "json". Esses arquivos também são gravados em um diretório interno do sistema e organizados em pastas.

Como produto final desse trabalho apresentamos na Figura 34 uma visão geral do Sistema Computacional Paraconsistente.

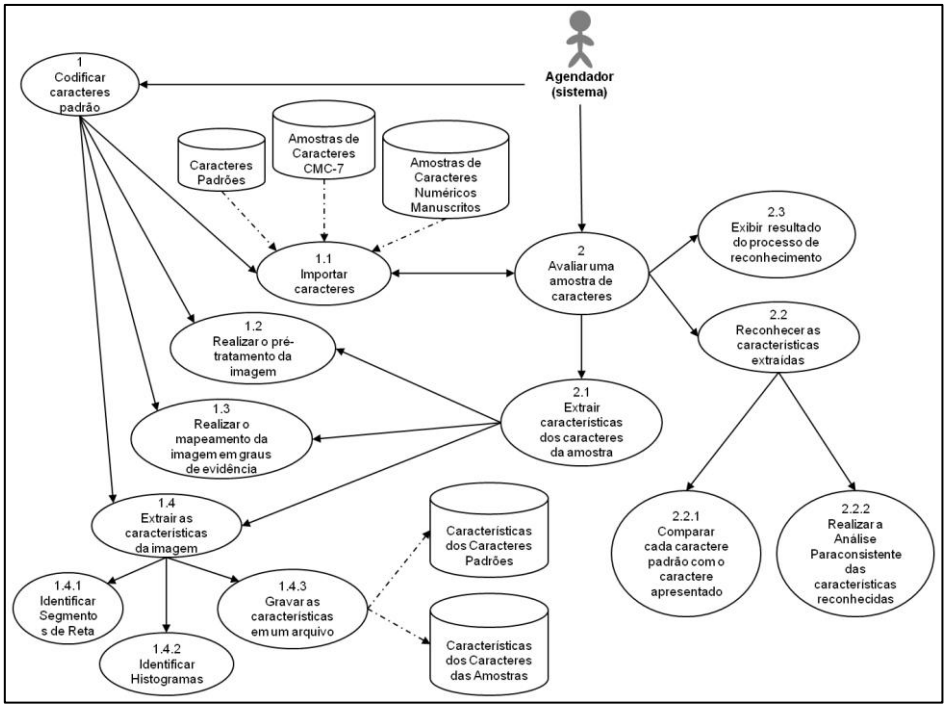


Figura 34 – Visão geral do Sistema Computacional Paraconsistente

Conforme mostra a Figura 34, inicialmente o sistema importa os caracteres padrões (Figura 27) disponibilizados em uma base de dados, extrai as características de cada caractere da amostra em formato de em graus de evidência e grava essas características em uma base de dados em formato de arquivo. Após extrair as características de todos os caracteres padrões, o sistema está apto a fazer o reconhecimento de caracteres do tipo CMC-7 e numérico manuscrito.

No processo de extração das características dos caracteres padrões, todas as características apresentadas na segunda camada da arquitetura geral do sistema (Figura 16) são extraídas dos caracteres, a saber, "Segmentos de Reta Verticais", "Segmentos de Reta Horizontais", "Histograma: Todos os Pixels da Matriz Binária", "Histograma: Borda Externa Superior", "Histograma: Borda Externa", "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (direita/esquerda)", "Borda Diagonal Interna (esquerda/direita)", "Histograma: Cores da Imagem", "Histograma: Cores da Imagem por Coluna", "Histograma: Componente Principal" e "Histograma: Autovalores do Componente Principal"

Para fazer o reconhecimento de uma amostra de caracteres, CMC-7 ou numérico manuscrito, o sistema importa a amostra da base de dados, extrai as características de todos os caracteres da mesma em formato de graus de evidência e, também, grava essas características em uma base de dados em formato de arquivo. Em seguida, o sistema faz o reconhecimento das características de cada caractere da amostra através da comparação das características do caractere da amostra com as características dos caracteres padrões e, posteriormente, faz uma análise dos resultados obtidos no reconhecimento de cada característica através de um algoritmo para definir o caractere reconhecido para cada caractere da amostra. Como finalização do processo de reconhecimento o sistema exibe uma lista dos caracteres da amostra com seus respectivos caracteres reconhecidos.

Inicialmente foi implementado como tipo de mapeamento na primeira camada da arquitetura do sistema (Figura 16) apenas a "Borda Externa" e definimos na segunda camada da arquitetura somente as características "Segmentos de Reta Verticais", "Segmentos de Reta Horizontais" e "Histograma: Borda Externa" para compor o processo de extração de características para ambos os tipos de caracteres, onde a porcentagem de acertos obtida para uma amostra de 2.092 caracteres do tipo CMC-7 foi

de 78,92 % e, para uma amostra de 1.050 caracteres do tipo numérico manuscrito de 79,24%.

Com o objetivo de aumentar a porcentagem de reconhecimento de ambos os tipos de caracteres, adicionamos a primeira camada da arquitetura do sistema novos tipos de mapeamento como "Todos os Pixels da Matriz Binária", "Borda Interna", "Borda Diagonal Interna (direita/esquerda)", "Borda Diagonal Interna (esquerda/direita)", "Core da Imagem", "Cores da Imagem por Coluna", "Componente Principal" e "Autovalores do Componente Principal" e, assim, novos tipos de características foram também acrescentadas a segunda camada da arquitetura como "Histograma: Todos os Pixels da Matriz Binária", "Histograma: Borda Externa Superior", "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (direita/esquerda)", "Borda Diagonal Interna (esquerda/direita)", "Histograma: Cores da Imagem", "Histograma: Cores da Imagem por Coluna", "Histograma: Componente Principal" e "Histograma: Autovalores do Componente Principal". (Figura 16)

A partir desses novos tipos de mapeamento e características, o sistema foi adaptado para oferecer a flexibilidade de combinar vários tipos de características como objetivo de encontrar a melhor combinação de características capaz de oferecer um bom resultado de reconhecimento para cada tipo de caractere, CMC-7 e numérico manuscrito.

Para identificarmos a melhor combinação de características para cada tipo de caractere, foram realizados os testes descritos nas tabelas do Apêndice A para os caracteres do tipo CMC-7 e os testes descritos no Apêndice B para os caracteres do tipo numérico manuscrito.

Após analisarmos os testes do Apêndice A identificamos que a melhor arquitetura do sistema para os caracteres do tipo CMC-7 é composta pelas características "Histograma: Borda Interna" e "Histograma: Borda Interna Diagonal (esquerda/direta)" ou, pelas características "Histograma: Borda Interna" e "Histograma: Borda Interna Diagonal (direita/esquerda). Ambas as configurações apresentaram a mesma porcentagem de acerto (97,85%), sensibilidade (97,38%), especificidade (99,81%), eficiência (98,62%) e acurácia (99,67%).

Com relação aos caracteres numéricos manuscritos, após analisarmos os testes do Apêndice B, identificamos que a melhor arquitetura do sistema para os caracteres do

tipo CMC-7 é composta pelas características "Histograma: Borda Externa" e "Histograma: Borda Interna Diagonal (direita/esquerda)" com porcentagem de acerto (91,62%), sensibilidade (91,27%), especificidade (99,07%), eficiência (95,16%) e acurácia (98,32%).

Tendo como base a Tabela 9, a sensibilidade é calculada por " A / (A + C)", a especificidade por " D / (B + D)", a eficiência por "(sensibilidade + especificidade) / 2" e, a acurácia por "(A+D) / A + B + C + D)". (Cios; Moore, 2002)

Tabela 9 - Possíveis resultados de um teste

Prognóstico (resultado testado)	**Diagnóstico (resultado confirmado)**		**Total**
	Positivo	**Negativo**	
Positivo	A (verdadeiro positivo)	B (falso positivo)	A + B
Negativo	C (falso negativo)	D (verdadeiro negativo)	C + D
Total	A + C	B + D	A + B + C + D

Levando-se em consideração alguns estudos prévios em reconhecimento de caracteres encontramos que o sistema "MICR reader" da VeriFone - baseado nas RNAs - alcançou 99,6% de acurácia conforme Abdleazeem e El-Sherif em Hammerstrom (1993) e Öksüz (2003) apresentou um sistema de reconhecimento de caracteres manuscritos com 90,21% de acertos que combina as vantagens de ambos os sistemas on-line e off-line. Esse último utiliza uma câmera de vídeo regular para detectar as posições da ponta da caneta enquanto os caracteres são escritos e essas posições são utilizadas para calcular a informação direcional. Assim, os caracteres são classificados pela sequência de direções escritas.

Do ponto de vista industrial em Jayadevan et al. (2012) o A2iA "CheckReader" reconhece textos impressos por máquina com 99% de acurácia e textos manuscritos com um formato previamente determinado (*constrained handwritten text*) com 96% de acurácia, ultrapassando alguns dos fornecedores de destaque na área de processamento automático de cheques bancários Mitek, Parascript e SoftPro utilizando RNAs.

Encontramos também a patente de Seiko Epson Corporation (2002) que faz o reconhecimento dos caracteres do tipo CMC-7 com base nas formas de onda detectadas a partir das 7 barras que compõe cada caractere, ou seja, a combinação das larguras dos

espaços entre as barras. A partir dessas formas de onda detectadas para cada caractere, são geradas ondas integradas e determina-se o intervalo efetivo (*effective range*) da onda integrada para cada caractere. Cada caractere consiste em 7 barras, portanto, sua respectiva onda integrada é composta por 7 picos positivos e 7 picos negativos formando "vales" entre os picos positivos e negativos podendo variar entre "profundos" e "rasos" dependendo da largura dos espaços entre as barras. Assim, os caracteres são reconhecidos com base no intervalo válido da onda integrada e na posição relativa dos "vales profundos" na onda integrada.

Assim, o Sistema Computacional Paraconsistente proposto se mostrou capaz de produzir resultados tão bons quanto os outros sistemas.

Como investigações e aplicações práticas futuras se apresenta as seguintes sugestões:

- O estudo da utilização de Redes Neurais Artificiais de aprendizagem na fase de reconhecimento com intuito de melhorar a performance e desempenho do sistema para reconhecer caracteres com ou sem um padrão previamente determinado;
- O estudo e desenvolvimento de um módulo de aquisição da imagem em tempo real via *scanner* para ser acoplado ao sistema e;
- O desenvolvimento de um módulo de pré-tratamento da imagem com um estudo mais aprofundado em filtros de remoção de ruídos da imagem para ser acoplado ao sistema com intuito de proporcionar melhor performance no reconhecimento dos caracteres.

6 CONCLUSÃO

Nesta dissertação propusemos um sistema computacional paraconsistente automatizado - Sistema Computacional Paraconsistente - para o reconhecimento de caracteres numéricos manuscritos, onde através do uso de RNAPs obtivemos resultados com porcentagem de 97,85% de acertos para os caracteres do tipo CMC-7 e 91,62% para os caracteres numéricos manuscritos.

A forma como se considerou a coleta das evidências, bem como a construção do Sistema Computacional Paraconsistente proposto, foi capaz de tratar eficazmente aspectos ambíguos e incertos inerentes a análise de dados ruidosos, de forma direta sem trivialização. Como se pôde ver na exposição, esse sistema é capaz de produzir resultados tão bons quanto os outros sistemas. Assim, o sistema proposto mostrou-se totalmente operacional e, também, aplicável para análise de MICR CMC-7 automatizado.

Os resultados experimentais obtidos para o reconhecimento de caracteres alfabéticos também demonstra que o mesmo sistema pode ser adaptado para fazer o reconhecimentos de outros tipos de caracteres. Dessa forma, a metodologia proposta pode ser adaptada para uso em sistemas administrativos hospitalares com o objetivo de reconhecer dados digitalizados do prontuário do paciente, receitas médicas ou apenas identificar pacientes por código de barras ou protocolo.

Mostra-se, assim, que os resultados obtidos ao longo desse trabalho demonstra que a metodologia adotada para manipular dados imprecisos, inconsistentes e paracompletos durante o processo de reconhecimento de caracteres são eficientes e nos encoraja a aplicar essas idéias nas diversas áreas de aplicação médica, tecnológica e áreas correlatas em reconhecimento de padrões. Em particular, estaremos interessados em aplicar futuramente essa técnica em reconhecimento de imagem e sinais em geral, temas da Biomedicina que possui grande importância no segmento de diagnósticos clínicos, interpretação e análise de imagens, interpretação e análise de sinais, desenvolvimento de drogas e outros temas de importância na área da Medicina.

7 REFERÊNCIAS

Abe JM. *Fundamentos da lógica anotada* [Tese]. São Paulo: Faculdade de Filosofia, Letras e Ciências Humanas, Universidade de São Paulo; 1992.

Abe JM. Some aspects of paraconsistent systems and applications. *Logique Analyse*. 1997;157:83-96.

Abe JM. Annotated logics Qτ and model theory, in logic, artificial intelligence, and robotics, In: Abe JM, Da Silva Filho JI, editors. Proceeding 2nd Congress of Logic Applied to Technology – LAPTEC'2001. Frontiers in artificial intelligence and its applications. Amsterdan: IOS Press; 2001. p.71:1-12.

Abe JM. Introdução à lógica paraconsistente anotada. *Seleção Documental*, Santos. 2006;1(1):11-17.

Abe JM, Akama S. A logical system for reasoning with fuzziness and inconsistencies. In: Proceedings of the IASTED (ASC'99). Honolulu; 1999. P.221-225.

Abe JM, Da Silva Filho JI, Celestino U, De Araújo HC. *Lógica paraconsistente anotada evidencial $E\tau$.* Santos: Comunnicar; 2011.

Abe JM, Lopes HFS, Anghinah R. Paraconsistent Artificial neural network and alzheimer disease: a preliminary study. *Dementia Neuropsychol*. 2007;3:241-7.

Abe JM, Papavero N. *Teoria intuitiva dos conjuntos.* São Paulo: McGraw-Hill, Makron Books; 1991.

Akama S, Abe JM. Annotated rules with uncertainty in expert systems. iN: Proceedings of the IASTED. Innsbruck; 2000. P.817-20.

Amaral FV. *Classificador paraconsistente de atributos de imagens de mamografia no diagnóstico do câncer de mama: apoiado na rede neural artificial paraconsistente* [Tese]. São Paulo: Universidade Paulista; 2013.

Arbib MA. *Brains, machines, and mathematics.* 2nd ed. New York: Springer-Verlag; 1987.

Ávila BC. *Uma abordagem paraconsistente baseada em lógica evidencial para tratar exceções em sistemas de frames com múltipla herança* [Tese]. São Paulo: Universidade de São Paulo; 1996.

Azevedo MA, Brasil LM, De Oliveira RCL. *Redes neurais com aplicações em controle e em sistemas especialistas.* Florianópolis: Visual Books; 2000.

Baxt WG. Application of artificial neural networks to clinical medicine. *Lancet*. 1995;346(8983):1135-8.

Bishop CM. *Neural networks for pattern recognition*. Oxford: Oxford University Press; 1995.

Cios KJ, Moore W. Uniqueness of medical data mining**.** *Intelligence Med*. 2002;26(1-2):1-24.

Da Costa NCA. *Ensaio sobre os fundamentos da lógica***.** São Paulo: Hucitec-Edusp; 1980.

Da Costa NCA. *Logiques classiques et non classiques: essai sur les fondements de la logique*. Paris: Masson; 1997.

Da Costa NCA, Abe JM, Da Silva Filho JI, Murolo AC, Leite CFS. *Lógica paraconsistente aplicada*. São Paulo: Atlas; 1999.

Da Silva Filho JI. *Métodos de aplicações da lógica paraconsistente anotada de anotação com dois valores - LPA2v com construção de algoritmo e implementação de circuitos eletrônicos* [Tese]. São Paulo: Escola Politécnica, Universidade de São Paulo; 1999.

Da Silva Filho JI. Algoritmo paraconsistente extrator de efeitos da contradição - $ParaExtr_{ctr}$. *Seleção Documental*. 2009;15:21/ATC3-28.

Da Silva Filho JI, Abe JM. *Introdução à lógica paraconsistente anotada com ilustrações*. Santos: Emmy; 2000.

Da Silva Filho JI, Abe JM. Paraconsistent analyzer module. *Int J Comput Anticipatory Systems.* 2001;9:346-52.

Da Silva Filho JI, Abe JM, Lambert-Torres G. *Inteligência artificial com as redes de análise paraconsistente: teorias e aplicações*. Rio de Janeiro: LTC; 2008.

Da Silva Filho JI, Lambert-Torres G, Abe JM. *Uncertainty treatment using paraconsistent logic: introducing paraconsistent artificial neural networks*. 21st ed. Amsterdam: IOS Press; 2010.

De Mendonça FLL. *Proposta de arquitetura de um sistema com base em OCR neuronal para resgate e indexação de escritas paleográficas do sec. XVI ao XIX* [Dissertação]. Brasília: Departamento de Engenharia Elétrica, Universidade de Brasília; 2008.

Dedgaonkar SG, Chandavale AA, Sapkal AM. Survey of methods for character recognition. *IJEIT*. 2012;1(5).

Duda RO, Hart PE. *Pattern classification and scene analysis*. New York: John Wiley; 1973.

Duneau L. *Étude et réalisation d'un système adaptatif pour la reconnaissance en ligne de mots manuscripts* [Tese]. França: Université Technologique de Compiègne; 1994.

Engelbrecht AP. *Computational intelligence*: an introduction. 2nd ed. New York: Wiley; 2007.

Fausett L. *Fundamentals of neural network*: architectures, algorithms, and applications. USA: Prentice Hall International;, 1994.

Ferrara LFP. *Redes neurais artificiais aplicada em um reconhecedor de caracteres* [Dissertação]. Uberlândia: Universidade Federal de Uberlândia; 2003.

Fischler MA, Firschein O. Intelligence: the eye, the brain and the computer. USA: Addison-Wesley Publishing; 1987.

Gomide TLF. *Manual de grafoscopia*. 2nd ed. São Paulo: Leud; 2005.

Hammerstrom D. Adaptative solutions Inc. *Spectrum. IEEE*. 1993;30(6):26-32.

Hassounn MH. *Fundamentals of artificial neural network*. Cambridge, MA: MIT Press; 1995.

Haykin S. *Neural networks*. Toronto: McMaster University; 1994.

Hebb D. *The organization of behavior*. New York: Wiley; 1949.

Hecht-Nielsen R. Neural analog processing**.** *Proc SPIE*, Bellingham *WA.* 1982:360:180-9.

Hecht-Nielsen R. *Neurocomputing.* New York: Addison Wesley; 1990.

Hertz J, Krogh A, Palmer R. *An introduction to the theory of neural computation.* New York: Addison-Wesley Publishing; 1991.

Hinton GE, Sejnowski TJ. Neural network architectures for AI. *AAAI-87*, 1987.

Hopfield J. Neural networks and physical systems with emergent collective's computational abilities. *Proc Nat Acad Sci*. 1982;79:2554-8.

Hopfield JJ, Tank TW. Computing with neural circuits: a model. *Science*. 1986;233(4764):625-33.

Jatene DA, Consoni FL, Bernardes R. Avaliação da implementação do prontuário eletrônico do paciente e impactos na gestão dos serviços hospitalares: a experiência do InCor Instituto do Coração. In: *ENANPAD*, Rio de Janeiro; 2012.

Jayadevan R, Kolhe SR, Patil PM, Pal U. Automatic processing of handwritten bank cheque images: a survey. *IJDAR*. 2012;15(4):267-96.

Justino E. *O Grafismo e os modelos escondidos de Markov na verificação automática de assinaturas* [Tese]. Paraná: Pontifícia Universidade Católica do Paraná; 2001.

Justino E. *A análise de documentos questionados*. Relatório técnico para obtenção do grau de Professor Titular. Paraná: Pontifícia Universidade Católica do Paraná; 2002.

Kato N, Suzuki M, Omachi SI, Aso H, Nemoto Y. A handwritten character recognition system using directional element feature and asymmetric mahalanobis distance. *TPAMI*. 1999;21(3):258-62.

Kohonen T. *An introduction to neural computing*. Finland: Helsinki University of Technology; 1987.

Kohonen T. *Self organization and associative memory*. 3rd ed. Berlin: Springer-Verlag; 1989.

Korf RE. *Artificial intelligence search algorithms.* In: *CRC Handbook of Argorithms and Theory of Computation*. FL: CRC Press; 1998.

Kosko B. *Neural networks for signal processing*. New Jersey: Prentice-Hall; 1992.

Lapuerta P, Azen SP, LaBree L. Use of neural networks in predicting the risk of coronary artery disease. *Comput Biomed Res.* 1995;28(1):38-52.

Lau CGY. *Neural network: theoretical foundations and analysis*. USA: IEEE Press; 1991.

Lopes HFS. *Aplicação de Redes Neurais Artificiais Paraconsistentes como método de auxílio no diagnóstico da doença de Alzheimer* [Dissertação]. São Paulo: Faculdade de Medicina, Universidade de São Paulo; 2009.

Lopes HFS; Abe JM.; Anghinah R. Application of paraconsistent artificial neural networks as a method of aid in the diagnosis of Alzheimer disease. *J Med Systems*. 2009:1-9.

Luger GF. *Inteligência artificial: estruturas e estratégias para a resolução de problemas complexos*. 4th ed. Porto Alegre: Bookmann; 2004.

Mantas J. An overview of character recognition methologies, *Pattern Recognition*. 1986;19(6):425-30.

Mário MC. *Modelo de análise de variáveis craniométricas através das redes neurais artificiais paraconsistentes* [Tese]. São Paulo: Faculdade de Medicina, Universidade de São Paulo; 2006.

Mario MC, Abe JM, Ortega N, Del Santo Jr M. Paraconsistent artificial neural network as auxiliary in cephalometric diagnosis. *Artif Organs*. 2010;34(7):215-21.

Martha AS, Salomão PL, Sigulem D, Romani R, Campos CJR. Clinic web: PEP e interação com dispositivos móveis. In: *X Congresso Brasileiro de Informática em Saúde*. Florianópolis; 2006. p.42-7.

Massad E, Menezes RX, Silveira PSP, Ortega NRS. *Métodos quantitativos em medicina*. Barueri: Manole; 2004.

Massad E, Marin HF, Azevedo Neto RS. *O prontuário eletrônico do paciente na assistência, informação e conhecimento médico*. São Paulo: H. de F. Marin; 2003.

Math ∩ Programming. 28 June 2012. Produced by J. Kun [cited 2013 Aug 10]. Available from: http://jeremykun.com/2012/06/28/principal-component-analysis.

McCulloch W, Pitts W. A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys*. 1943;5:115-33.

Miller A, Blott B, Hames T. Review of neural network applications in medical imaging and signal processing. *Med Biol Eng Comput*. 1992;30(5):449-64.

Minsky M, Papert SA. *Perceptrons: an introduction to computational geometry*. 2nd ed. Cambridge, MA: MIT Press; 1988.

Murakami A, Kobayashi LO, Tachinardi U, Gutierrez MA, Furuie SS, Pires FA. Acesso a informações médicas através do uso de sistemas de computação móvel. In: *Congresso Brasileiro de Informática em Saúde*. Ribeirão Preto; 2004. p.39-44.

Nakamatsu K, Abe JM, Suzuki A. Annotated semantics for defeasible deontic reasoning. RSCTC'2000. *LNAI 2005*. 2000;470-8.

Öksüz O. *Vision based handwritten character recognition* [Thesis] Turkey: Bilkent University; 2003 [cited 2013 Aug 17]. Available from: http://www.cs.bilkent.edu.tr/tech-reports/2003/BU-CE-0310.pdf.

Prado JPA. *Uma arquitetura em IA baseada em lógica paraconsistente* [Tese]. São Paulo: Escola Politécnica, Universidade de São Paulo; 1996.

Prado JCA. *Constelação fônica e redes neurais artificiais: aplicabilidade na análise computacional da produção da fala* [Tese]. São Paulo: Faculdade de Filosofia, Letras e Ciências Humanas, Universidade de São Paulo; 2007.

Rich E, Knight K. *Inteligência artificial*. São Paulo: Makron Books; 1993.

Rosemblatt F. *"Principles of neurodynamics"*. New York: Spartan Books; 1962.

Santos CR. *Análise de assinaturas manuscritas baseada nos princípios da grafoscopia* [Dissertação]. Curitiba: Pontifícia Universidade Católica do Paraná; 2004.

Seiko Epson Corporation (Japan). Yuji Takigushi. *Method and apparatus for magnetic pattern recognition and data storage medium carrying computer program means for implementing the method*. EP 1199674A2, 16 Out. 2001, 24 Abr. 2002 [cited 2013 Aug 17]. Available from: https://www.google.com/patents/EP1199674A2?cl=en&dq=cmc7+micr+character+recognition&hl=en&sa=X&ei=7EkIUuq-HYvu9AS334GADg&ved=0CDsQ6AEwAQ.

Siebert W. *Stimulus transformation in peripheral auditory system in recognizing patterns*. Cambridge, MA: MIT Press; 1968.

Silva E. Reconhecimento de caracteres. *Inf Técnico Núcleo de Computação Eletrônica UFRJ*, Rio de Janeiro; 2006 [citado 8 Jun 2013]. Disponível em: http://www.nce.ufrj.br/conceito/artigos/2006/016p1-3.htm.

Silva E, Rodrigues RJ, Thomé ACG. character feature extraction using polygonal projection sweep (contour detection). In: *6th International Work-conference on Artificial and Natural Neural*. Granada; 2001. p.413-8.

Sklansky J. *Pattern recognition: introduction and foundations*. Dowdon: Hutchinson & Ross; 1973.

Suen CY, Berthold M, Mori S. Automatic recognition of handprinted characters: the state of the art. *Proc IEEE*. 1980;68(4).

Torres CR. *Sistema inteligente baseado na lógica paraconsistente anotada evidencial* $e\tau$ *para controle e navegação de robôs móveis autônomos em um ambiente não-estruturado* [Tese]. Itajubá, MG: Universidade Federal de Itajubá; 2010.

Tou JT, Gonzalez RC. *Pattern recognition principles*. USA: Addison-Wesley Publishing; 1974.

Widrow B, Lehr MA. 30 years of adaptive neural networks: perceptron, madaline and backpropagation. *Proc IEEE*. 1990;78(9):1415-42.

APÊNDICES

APÊNDICE A - Testes realizados com caracteres do tipo CMC-7

Tabela 10 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Todos os Pixels da Matriz Binária" para o reconhecimento de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Todos os Pixels da Matriz Binária											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	283	153	283	1646	10	153	0,649	0,994	0,821	0,922
1	264	254	10	254	1815	13	10	0,962	0,993	0,977	0,989
2	122	83	39	83	1948	22	39	0,68	0,989	0,835	0,971
3	153	141	12	141	1787	152	12	0,922	0,922	0,922	0,922
4	152	142	10	142	1907	33	10	0,934	0,983	0,959	0,979
5	190	120	70	120	1898	4	70	0,632	0,998	0,815	0,965
6	111	69	42	69	1978	3	42	0,622	0,998	0,81	0,978
7	98	91	7	91	1847	147	7	0,929	0,926	0,927	0,926
8	146	102	44	102	1921	25	44	0,699	0,987	0,843	0,967
9	102	73	29	73	1972	18	29	0,716	0,991	0,853	0,977
⑈	138	56	82	56	1887	67	82	0,406	0,966	0,686	0,929
⑆	90	79	11	79	1976	26	11	0,878	0,987	0,932	0,982
⑇	90	71	19	71	1994	8	19	0,789	0,996	0,892	0,987
Total	2 092	74,76%	25,24%	1564	24576	528	528	75,52%	97,92%	86,71%	96,11%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 11 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Externa Superior" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Borda Externa Superior											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	72	364	72	1567	89	364	0,165	0,946	0,556	0,783
1	264	259	5	259	1818	10	5	0,981	0,994	0,988	0,993
2	122	21	101	21	1897	73	101	0,172	0,963	0,567	0,917
3	153	23	130	23	1869	70	130	0,15	0,964	0,557	0,904
4	152	50	102	50	1876	64	102	0,329	0,967	0,648	0,921
5	190	24	166	24	1894	8	166	0,126	0,996	0,561	0,917
6	111	67	44	67	1322	659	44	0,604	0,667	0,635	0,664
7	98	13	85	13	1979	15	85	0,133	0,992	0,562	0,952
8	146	1	145	1	1917	29	145	0,007	0,985	0,496	0,917
9	102	14	88	14	1901	89	88	0,137	0,955	0,546	0,915
⑈	138	37	101	37	1881	73	101	0,268	0,962	0,615	0,917
⑆	90	80	10	80	1923	79	10	0,889	0,96	0,925	0,957
⑇	90	72	18	72	1901	101	18	0,8	0,949	0,875	0,943
Total	2 092	35,04%	64,96%	733	23745	1359	1 359	36,62%	94,62%	65,62%	90,00%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 12 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Externa" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Borda Externa											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	435	1	435	1626	30	1	0,998	0,982	0,99	0,985
1	264	261	3	261	1828	0	3	0,989	1	0,994	0,999
2	122	118	4	118	1970	0	4	0,967	1	0,984	0,998
3	153	151	2	151	1935	4	2	0.987	0,998	0,992	0,997
4	152	151	1	151	1940	0	1	0,993	1	0,997	0,999
5	190	171	19	171	1902	0	19	0,9	1	0,95	0,991
6	111	109	2	109	1971	10	2	0,982	0,995	0,995	0,994
7	98	97	1	97	1993	1	1	0,99	0,999	0,999	0,999
8	146	140	6	140	1942	4	6	0,959	0,998	0,998	0,995
9	102	94	8	94	1981	9	8	0,922	0,995	0,995	0,992
⑈	138	117	21	117	1954	0	21	0,848	1	1	0,99
⑆	90	88	2	88	1990	12	2	0,978	0,994	0,994	0,993
⑇	90	88	2	88	2000	2	2	0,978	0,999	0,999	0,998
Total	2 092	96,56%	3,44%	2020	25032	72	72	88,49%	99,69%	99,13%	99,46%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 13 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Interna" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Borda Interna											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	434	2	434	1654	2	2	0,995	0,999	0,997	0,998
1	264	261	3	261	1825	3	3	0,989	0,998	0,993	0,997
2	122	122	0	122	1969	1	0	1	0,999	1	0,999
3	153	152	1	152	1935	4	1	0,993	0,998	0,996	0,998
4	152	151	1	151	1940	0	1	0,993	1	0,997	0,999
5	190	177	13	177	1902	0	13	0,931	1	0,966	0,994
6	111	109	2	109	1973	8	2	0,931	0,996	0,989	0,995
7	98	94	4	94	1994	0	4	0,982	1	0,98	0,998
8	146	143	3	143	1942	4	3	0,959	0,998	0,989	0,997
9	102	96	6	96	1984	6	6	0,979	0,997	0,969	0,994
⑈	138	133	5	133	1954	0	5	0,941	1	0,982	0,998
⑆	90	87	3	87	1985	17	3	0,967	0,991	0,979	0,99
⑇	90	86	4	86	2000	2	4	0,956	0,999	0,977	0,997
Total	2 092	97,75%	2,25%	2045	25057	47	47	97,05%	99,81%	98,57%	99,65%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 14 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Diagonal (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	431	5	431	1649	7	5	0,988	0,996	0,992	0,994
1	264	262	2	262	1815	13	2	0,992	0,993	0,993	0,993
2	122	107	15	107	1960	10	15	0,877	0,995	0,936	0,988
3	153	145	8	145	1924	15	8	0,948	0,992	0,97	0,989
4	152	150	2	150	1940	0	2	0,987	1	0,993	0,999
5	190	158	32	158	1900	2	32	0,832	0,999	0,915	0,984
6	111	109	2	109	1949	32	2	0,982	0,984	0,983	0,984
7	98	96	2	96	1986	8	2	0,98	0,996	0,988	0,995
8	146	129	17	129	1946	0	17	0,884	1	0,942	0,992
9	102	96	6	96	1981	9	6	0,941	0,995	0,968	0,993
⑈	138	124	14	124	1951	3	14	0,898	0,998	0,948	0,992
⑆	90	81	9	81	1995	7	9	0,9	0,996	0,948	0,992
⑇	90	79	11	79	1983	19	11	0,878	0,99	0,934	0,986
Total	2 092	94,02%	5,98%	1967	24979	125	125	92,98%	99,49%	96,23%	99,08%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 15 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Diagonal (esquerda/direita)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	426	10	426	1654	2	10	0,977	0,999	0,988	0,994
1	264	238	26	238	1824	4	26	0,901	0,998	0,95	0,986
2	122	100	22	100	1969	1	22	0,82	0,999	0,91	0,989
3	153	152	1	152	1900	39	1	0,993	0,98	0,987	0,981
4	152	152	0	152	1933	7	0	1	0,996	0,998	0,997
5	190	159	31	159	1887	15	31	0,834	0,992	0,914	0,978
6	111	106	5	106	1973	8	5	0,955	0,996	0,975	0,994
7	98	96	2	96	1970	24	2	0,98	0,988	0,984	0,988
8	146	123	23	123	1943	3	23	0,842	0,998	0,92	0,988
9	102	93	9	93	1947	43	9	0,912	0,978	0,945	0,975
⑈	138	95	43	95	1941	13	43	0,688	0,993	0,841	0,973
⑆	90	62	28	62	1962	40	28	0,689	0,98	0,834	0,967
⑇	90	70	20	70	1981	21	20	0,778	0,989	0,884	0,98
Total	2 092	89,48%	10,52%	1872	24884	220	220	87,45%	99,12%	93,31%	98,38%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 16 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Cores da Imagem" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Cores da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	44	392	44	1440	216	392	0,101	0,867	0,485	0,71
1	264	25	239	25	1588	240	239	0,095	0,869	0,482	0,771
2	122	2	120	2	1919	51	120	0,016	0,974	0,495	0,918
3	153	4	149	4	1851	88	149	0,03	0,955	0,49	0,887
4	152	7	145	7	1814	126	145	0,05	0,935	0,49	0,87
5	190	6	184	6	1829	73	184	0,03	0,962	0,497	0,877
6	111	14	97	14	1719	262	97	0,13	0,868	0,497	0,828
7	98	21	77	21	1704	290	77	0,214	0,855	0,534	0,825
8	146	8	138	8	1851	95	138	0,055	0,952	0,503	0,889
9	102	1	101	1	1937	53	101	0,01	0,973	0,492	0,927
⑈	138	2	136	2	1905	49	136	0,014	0,975	0,495	0,912
⑆	90	1	89	1	1856	146	89	0,011	0,927	0,469	0,888
⑇	90	12	78	12	1746	256	78	0,133	0,872	0,503	0,84
Total	2 092	7,03%	92,97%	147	23159	1945	1945	6,84%	92,18%	49,48%	85,71%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 17 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Cores por Coluna da Imagem" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Cores por Coluna da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	379	57	379	1655	1	57	0,869	0,999	0,934	0,972
1	264	243	21	243	1822	6	21	0,92	0,997	0,956	0,987
2	122	111	11	111	1960	10	11	0,91	0,995	0,952	0,99
3	153	141	12	141	1934	5	12	0,922	0,997	0,96	0,992
4	152	143	9	143	1880	60	9	0,941	0,969	0,955	0,967
5	190	175	15	175	1889	13	15	0,921	0,993	0,957	0,987
6	111	97	14	97	1974	7	14	0,874	0,996	0,935	0,99
7	98	94	4	94	1957	37	4	0,96	0,981	0,97	0,98
8	146	131	15	131	1946	0	15	0,897	1	0,949	0,993
9	102	77	25	77	1988	2	25	0,755	0,999	0,877	0,987
⑈	138	100	38	100	1927	27	38	0,725	0,986	0,855	0,969
⑆	90	76	14	76	1941	61	14	0,844	0,969	0,907	0,964
⑇	90	66	24	66	1972	30	24	0,733	0,985	0,859	0,974
Total	2 092	87,62%	12,38%	1833	24845	259	259	86,70%	98,97%	92,82%	98,09%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 18 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Componente Principal" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Componente Principal											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	75	361	75	1476	180	361	0,172	0,891	0,532	0,741
1	264	199	65	199	1103	725	65	0,754	0,603	0,679	0,622
2	122	4	118	4	1969	1	118	0,033	0,999	0,516	0,943
3	153	11	142	11	1885	54	142	0,072	0,972	0,522	0,906
4	152	23	129	23	1885	55	129	0,151	0,972	0,561	0,912
5	190	11	179	11	1894	8	179	0,058	0,996	0,527	0,911
6	111	6	105	6	1951	30	105	0,054	0,985	0,519	0,935
7	98	13	85	13	1985	9	85	0,133	0,995	0,564	0,955
8	146	18	128	18	1890	56	128	0,123	0,971	0,547	0,912
9	102	26	76	26	1798	192	76	0,255	0,904	0,579	0,872
⑈	138	27	111	27	1648	306	111	0,196	0,843	0,520	0,801
⑆	90	3	87	3	1998	4	87	0,033	0,998	0,516	0,957
⑇	90	7	83	7	1953	49	83	0,078	0,976	0,527	0,937
Total	2 092	20,22%	79,78%	423	23435	1669	1669	16,24%	93,12%	54,68%	87,73%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 19 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Autovalores do Componente Principal" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Autovalores do Componente Principal											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	0	436	0	1656	0	436	0,000	1,000	0,500	0,792
1	264	5	259	5	1789	39	259	0,019	0,979	0,499	0,858
2	122	70	52	70	1816	154	52	0,574	0,922	0,748	0,902
3	153	9	144	9	1736	203	144	0,059	0,895	0,477	0,834
4	152	6	146	6	1756	184	146	0,039	0,905	0,472	0,842
5	190	0	190	0	1902	0	190	0,000	1,000	0,500	0,909
6	111	0	111	0	1945	36	111	0,000	0,982	0,491	0,930
7	98	74	24	74	1118	876	24	0,755	0,561	0,658	0,570
8	146	0	146	0	1937	9	146	0,000	0,995	0,498	0,926
9	102	1	101	1	1983	7	101	0,010	0,996	0,503	0,948
⑈	138	0	138	0	1954	0	138	0,000	1,000	0,500	0,934
⑆	90	14	76	14	1714	288	76	0,156	0,856	0,506	0,826
⑇	90	13	77	13	1898	104	77	0,144	0,948	0,546	0,913
Total	2 092	9,18%	90,82%	192	23204	1900	1900	13,51%	92,61%	53,06%	86,03%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 20 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Segmentos de Reta Verticais + Segmentos de Reta Horizontais											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	95	341	95	1622	34	341	0,218	0,979	0,599	0,821
1	264	138	126	138	1513	315	126	0,523	0,828	0,675	0,789
2	122	18	104	18	1888	82	104	0,148	0,958	0,553	0,911
3	153	33	120	33	1831	108	120	0,216	0,944	0,580	0,891
4	152	50	102	50	1890	50	102	0,329	0,974	0,652	0,927
5	190	57	133	57	1760	142	133	0,300	0,925	0,613	0,869
6	111	38	73	38	1789	192	73	0,342	0,903	0,623	0,873
7	98	24	74	24	1888	106	74	0,245	0,947	0,596	0,914
8	146	22	124	22	1841	105	124	0,151	0,946	0,548	0,891
9	102	7	95	7	1929	61	95	0,069	0,969	0,519	0,925
⑈	138	57	81	57	1758	196	81	0,413	0,900	0,656	0,868
⑆	90	14	76	14	1961	41	76	0,156	0,980	0,568	0,944
⑇	90	21	69	21	1916	86	69	0,233	0,957	0,595	0,926
Total	2 092	27,44%	72,56%	574	23586	1518	1518	25,70%	93,93%	59,82%	88,84%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 21 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Cores por Coluna da Imagem" e "Histograma: Autovalores do Componente Principal" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Cores por Coluna da Imagem + Histograma: Autovalores do Componente Principal											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	376	60	376	1655	1	60	0,862	0,999	0,931	0,971
1	264	243	21	243	1822	6	21	0,920	0,997	0,959	0,987
2	122	112	10	112	1960	10	10	0,918	0,995	0,956	0,990
3	153	141	12	141	1934	5	12	0,922	0,997	0,959	0,992
4	152	143	9	143	1877	63	9	0,941	0,968	0,954	0,966
5	190	174	16	174	1890	12	16	0,916	0,994	0,955	0,987
6	111	97	14	97	1972	9	14	0,874	0,995	0,935	0,989
7	98	94	4	94	1957	37	4	0,959	0,981	0,970	0,980
8	146	132	14	132	1946	0	14	0,904	1,000	0,952	0,993
9	102	76	26	76	1988	2	26	0,745	0,999	0,872	0,987
⑈	138	100	38	100	1929	25	38	0,725	0,987	0,856	0,970
⑆	90	76	14	76	1941	61	14	0,844	0,970	0,907	0,964
⑇	90	66	24	66	1971	31	24	0,733	0,985	0,859	0,974
Total	2 092	87,48%	12,52%	1830	24842	262	262	86,64%	98,98%	92,81%	98,07%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 22 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (esquerda/direita)" e "Histograma: Cores por Coluna da Imagem" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (esquerda/direita) + Histograma: Cores por Coluna da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	424	12	424	1655	1	12	0,972	0,999	0,986	0,994
1	264	261	3	261	1825	3	3	0,989	0,998	0,993	0,997
2	122	119	3	119	1967	3	3	0,975	0,998	0,987	0,997
3	153	150	3	150	1936	3	3	0,980	0,998	0,989	0,997
4	152	150	2	150	1938	2	2	0,987	0,999	0,993	0,998
5	190	184	6	184	1898	4	6	0,968	0,998	0,983	0,995
6	111	107	4	107	1980	1	4	0,964	0,999	0,982	0,998
7	98	96	2	96	1988	6	2	0,980	0,997	0,988	0,996
8	146	140	6	140	1942	4	6	0,959	0,998	0,978	0,995
9	102	95	7	95	1986	4	7	0,931	0,998	0,965	0,995
⑈	138	129	9	129	1952	2	9	0,935	0,999	0,967	0,995
⑆	90	86	4	86	1971	31	4	0,956	0,985	0,970	0,983
⑇	90	84	6	84	1999	3	6	0,933	0,999	0,966	0,996
Total	2 092	96,80%	3,20%	2025	25037	67	67	96,38%	99,74%	98,06%	99,51%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 23 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	435	1	435	1640	16	1	0,998	0,990	0,994	0,992
1	264	261	3	261	1828	0	3	0,989	1,000	0,994	0,999
2	122	119	3	119	1970	0	3	0,975	1,000	0,988	0,999
3	153	151	2	151	1935	4	2	0,987	0,998	0,992	0,997
4	152	151	1	151	1940	0	1	0,993	1,000	0,997	1,000
5	190	171	19	171	1902	0	19	0,900	1,000	0,950	0,991
6	111	109	2	109	1969	12	2	0,982	0,994	0,988	0,993
7	98	97	1	97	1993	1	1	0,990	0,999	0,995	0,999
8	146	143	3	143	1945	1	3	0,979	0,999	0,989	0,998
9	102	98	4	98	1984	6	4	0,961	0,997	0,979	0,995
⑈	138	129	9	129	1954	0	9	0,935	1,000	0,967	0,996
⑆	90	89	1	89	1992	10	1	0,989	0,995	0,992	0,995
⑇	90	88	2	88	2001	1	2	0,978	1,000	0,989	0,999
Total	2 092	97,56%	2,44%	2041	25053	51	51	97,35%	99,79%	98,57%	99,62%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 24 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Externa + Histograma: Borda Diagonal Interna (esquerda/direita)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	435	1	435	1630	26	1	0,998	0,984	0,991	0,987
1	264	261	3	261	1828	0	3	0,989	1,000	0,994	0,999
2	122	119	3	119	1970	0	3	0,975	1,000	0,988	0,999
3	153	151	2	151	1936	3	2	0,987	0,998	0,993	0,998
4	152	151	1	151	1940	0	1	0,993	1,000	0,997	1,000
5	190	172	18	172	1902	0	18	0,905	1,000	0,953	0,991
6	111	109	2	109	1972	9	2	0,982	0,995	0,989	0,995
7	98	97	1	97	1994	0	1	0,990	1,000	0,995	1,000
8	146	140	6	140	1942	4	6	0,959	0,998	0,978	0,995
9	102	94	8	94	1981	9	8	0,922	0,995	0,959	0,992
⑈	138	122	16	122	1954	0	16	0,884	1,000	0,942	0,992
⑆	90	88	2	88	1990	12	2	0,978	0,994	0,986	0,993
⑇	90	88	2	88	2000	2	2	0,978	0,999	0,988	0,998
Total	2 092	96,89%	3,11%	2027	25039	65	65	96,46%	99,73%	98,09%	99,52%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 25 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (esquerda/direita)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	434	2	434	1655	1	2	0,995	0,999	0,997	0,999
1	264	261	3	261	1826	2	3	0,989	0,999	0,994	0,998
2	122	122	0	122	1969	1	0	1	0,999	1	0,999
3	153	152	1	152	1935	4	1	0,993	0,998	0,996	0,998
4	152	151	1	151	1940	0	1	0,993	1	0,997	0,999
5	190	179	11	179	1902	0	11	0,942	1	0,971	0,995
6	111	109	2	109	1976	5	2	0,982	0,997	0,99	0,997
7	98	94	4	94	1994	0	4	0,959	1	0,98	0,998
8	146	143	3	143	1942	4	3	0,979	0,998	0,989	0,997
9	102	96	6	96	1984	6	6	0,941	0,997	0,969	0,994
⑈	138	133	5	133	1953	1	5	0,964	0,999	0,982	0,997
⑆	90	87	3	87	1983	19	3	0,967	0,99	0,979	0,989
⑇	90	86	4	86	2000	2	4	0,956	0,999	0,977	0,997
Total	2 092	97,85%	2,15%	2047	25059	45	45	97,38%	99,81%	98,62%	99,67%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 26 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	434	2	434	1655	1	2	0,995	0,999	0,997	0,999
1	264	261	3	261	1826	2	3	0,989	0,999	0,994	0,998
2	122	122	0	122	1969	1	0	1	0,999	1	0,999
3	153	152	1	152	1935	4	1	0,993	0,998	0,996	0,998
4	152	151	1	151	1940	0	1	0,993	1	0,997	0,999
5	190	179	11	179	1902	0	11	0,942	1	0,71	0,995
6	111	109	2	109	1976	5	2	0,982	0,997	0,99	0,997
7	98	94	4	94	1994	0	4	0,959	1	0,98	0,998
8	146	143	3	143	1942	4	3	0,979	0,998	0,989	0,997
9	102	96	6	96	1984	6	6	0,941	0,997	0,969	0,994
⑈	138	133	5	133	1953	1	5	0,964	0,999	0,982	0,997
⑆	90	87	3	87	1983	19	3	0,967	0,99	0,979	0,989
⑇	90	86	4	86	2000	2	4	0,956	0,999	0,977	0,997
Total	2 092	97,85%	2,15%	2047	25059	45	45	97,38%	99,81%	96,62%	99,67%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 27 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Interna" e "Histograma: Cores por Coluna da Imagem" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Interna + Histograma: Cores por Coluna da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	428	8	428	1654	2	8	0,982	0,999	0,990	0,995
1	264	263	1	263	1818	10	1	0,996	0,995	0,995	0,995
2	122	119	3	119	1969	1	3	0,975	0,999	0,987	0,998
3	153	150	3	150	1933	6	3	0,980	0,997	0,989	0,996
4	152	151	1	151	1939	1	1	0,993	0,999	0,996	0,999
5	190	177	13	177	1898	4	13	0,932	0,998	0,965	0,992
6	111	107	4	107	1979	2	4	0,964	0,999	0,981	0,997
7	98	96	2	96	1981	13	2	0,980	0,993	0,987	0,993
8	146	141	5	141	1937	9	5	0,966	0,995	0,981	0,993
9	102	91	11	91	1978	12	11	0,892	0,994	0,943	0,989
⑈	138	132	6	132	1954	0	6	0,957	1,000	0,978	0,997
⑆	90	81	9	81	1992	10	9	0,900	0,995	0,948	0,991
⑇	90	83	7	83	1999	3	7	0,922	0,999	0,960	0,995
Total	2 092	96,51%	3,49%	2019	25031	73	73	95,68%	99,71%	97,70%	99,46%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 28 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa" e "Histograma: Cores por Coluna da Imagem" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Externa + Histograma: Cores por Coluna da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	276	160	276	1629	27	160	0,633	0,984	0,808	0,911
1	264	260	4	260	1823	5	4	0,985	0,997	0,991	0,996
2	122	119	3	119	1969	1	3	0,975	0,999	0,987	0,998
3	153	150	3	150	1931	8	3	0,980	0,996	0,988	0,995
4	152	152	0	152	1940	0	0	1,000	1,000	1,000	1,000
5	190	175	15	175	1900	2	15	0,921	0,999	0,960	0,992
6	111	107	4	107	1901	80	4	0,964	0,960	0,962	0,960
7	98	97	1	97	1990	4	1	0,990	0,998	0,994	0,998
8	146	139	7	139	1921	25	7	0,952	0,987	0,970	0,985
9	102	90	12	90	1921	69	12	0,882	0,965	0,924	0,961
⑈	138	122	16	122	1953	1	16	0,884	0,999	0,942	0,992
⑆	90	83	7	83	1990	12	7	0,922	0,994	0,958	0,991
⑇	90	85	5	85	1999	3	5	0,944	0,999	0,971	0,996
Total	2 092	88,67%	11,33%	1855	24867	237	237	92,57%	99,06%	95,81%	98,26%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 29 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa", "Histograma: Borda Diagonal Interna (direita/esquerda)" e "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda) + Histograma: Borda Diagonal Interna (esquerda/direita)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	435	1	435	1646	10	1	0,998	0,994	0,996	0,995
1	264	261	3	261	1828	0	3	0,989	1	0,994	0,999
2	122	119	3	119	1970	0	3	0,975	1	0,988	0,999
3	153	152	1	152	1935	4	1	0,993	0,998	0,996	0,998
4	152	151	1	151	1940	0	1	0,993	1	0,997	0,999
5	190	173	17	173	1902	0	17	0,91	1	0,955	0,992
6	111	110	1	110	1969	12	1	0,99	0,994	0,992	0,994
7	98	97	1	97	1994	0	1	0,99	1	0,995	0,999
8	146	145	1	145	1945	1	1	0,993	0,999	0,996	0,999
9	102	98	4	98	1983	7	4	0,961	0,996	0,979	0,995
⑈	138	128	10	128	1954	0	10	0,927	1	0,964	0,995
⑆	90	89	1	89	1991	11	1	0,989	0,994	0,992	0,994
⑇	90	88	2	88	2001	1	2	0,978	0,999	0,989	0,999
Total	2 092	97,80%	2,20%	2046	25058	46	46	97,58%	99,80%	98,72%	99,67%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 30 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa", "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Características: Histograma: Borda Externa + Segmentos de Reta Verticais + Segmentos de Reta Horizontais											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	344	92	344	1531	125	92	0,789	0,924	0,857	0,896
1	264	248	16	248	1817	11	16	0,939	0,994	0,967	0,987
2	122	101	21	101	1930	40	21	0,823	0,98	0,904	0,971
3	153	115	38	115	1891	48	38	0,752	0,975	0,863	0,959
4	152	151	1	151	1938	2	1	0,993	0,999	0,996	0,999
5	190	153	37	153	1891	11	37	0,805	0,994	0,9	0,977
6	111	78	33	78	1838	143	33	0,703	0,928	0,815	0,916
7	98	94	4	94	1991	3	4	0,959	0,998	0,979	0,997
8	146	58	88	58	1938	8	88	0,397	0,996	0,697	0,954
9	102	46	56	46	1975	15	56	0,451	0,992	0,722	0,966
⑈	138	98	40	98	1945	9	40	0,71	0,995	0,853	0,977
⑆	90	79	11	79	1979	23	11	0,878	0,988	0,933	0,984
⑇	90	86	4	86	1999	3	4	0,956	0,998	0,977	0,997
Total	2 092	78,92%	21,08%	1651	24663	441	441	78,12%	98,16%	88,18%	96,77%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 31 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e os picos superiores e inferiores das características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Picos superiores e inferiores das características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	30	406	30	1592	64	406	0,069	0,961	0,515	0,775
1	264	10	244	10	1775	53	244	0,038	0,971	0,504	0,853
2	122	1	121	1	1955	15	121	0,009	0,992	0,5	0,935
3	153	4	149	4	1901	38	149	0,026	0,98	0,503	0,911
4	152	2	150	2	1915	25	150	0,013	0,987	0,5	0,916
5	190	3	187	3	1895	7	187	0,015	0,996	0,506	0,907
6	111	2	109	2	1961	20	109	0,018	0,99	0,504	0,938
7	98	15	83	15	1774	220	83	0,153	0,89	0,521	0,855
8	146	4	142	4	1910	36	142	0,027	0,981	0,504	0,915
9	102	2	100	2	1969	21	100	0,02	0,989	0,504	0,942
⑈	138	10	128	10	1888	66	128	0,072	0,966	0,519	0,907
⑆	90	60	30	60	1045	957	30	0,667	0,521	0,594	0,528
⑇	90	25	65	25	1600	402	65	0,278	0,799	0,538	0,777
Total	2 092	8,03%	91,49%	168	23180	1924	1914	10,81%	92,48%	51,63%	85,84%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 32 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e os picos superiores e inferiores das características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo CMC-7

Testes realizados com caracteres do tipo CMC-7 Picos superiores e inferiores das características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	436	30	406	30	1592	64	406	0,069	0,961	0,515	0,775
1	264	10	244	10	1775	53	244	0,038	0,971	0,504	0,853
2	122	1	121	1	1955	15	121	0,009	0,992	0,5	0,935
3	153	4	149	4	1901	38	149	0,026	0,98	0,503	0,911
4	152	2	150	2	1915	25	150	0,013	0,987	0,5	0,916
5	190	3	187	3	1895	7	187	0,015	0,996	0,506	0,907
6	111	2	109	2	1961	20	109	0,018	0,99	0,504	0,938
7	98	15	83	15	1774	220	83	0,153	0,89	0,521	0,855
8	146	4	142	4	1910	36	142	0,027	0,981	0,504	0,915
9	102	2	100	2	1969	21	100	0,02	0,989	0,504	0,942
⑈	138	10	128	10	1888	66	128	0,072	0,966	0,519	0,907
⑆	90	60	30	60	1045	957	30	0,667	0,521	0,594	0,528
⑇	90	25	65	25	1600	402	65	0,278	0,799	0,538	0,777
Total	2 092	8,03%	91,49%	168	23180	1924	1914	10,81%	92,48%	51,63%	85,84%

APÊNDICE B - Testes realizados com caracteres do tipo numérico manuscrito

Tabela 33 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Todos os Pixels da Matriz Binária" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características: Histograma: Todos os Pixels da Matriz Binária											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	60	49	60	883	58	49	0,55	0,938	0,744	0,898
1	92	31	61	31	919	39	61	0,337	0,959	0,648	0,905
2	121	50	71	50	896	33	71	0,413	0,964	0,689	0,901
3	134	53	81	53	881	35	81	0,395	0,962	0,679	0,889
4	85	58	27	58	921	44	27	0,682	0,954	0,818	0,932
5	103	43	60	43	880	67	60	0,417	0,929	0,673	0,879
6	104	59	45	59	877	69	45	0,567	0,927	0,747	0m891
7	88	81	7	81	890	72	7	0,92	0,925	0,923	0,925
8	116	45	71	45	841	93	71	0,388	0,9	0,644	0,984
9	98	43	55	43	935	17	55	0,439	0,982	0,71	0,931
Total	1 050	49,81%	50,19%	523	8923	527	527	51,08%	94,40%	72,75%	82,44%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 34 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Externa Superior" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Borda Externa Superior											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	59	50	59	883	58	50	0,541	0,938	0,74	0,897
1	92	42	50	42	929	29	50	0,456	0,97	0,713	0,925
2	121	41	80	41	913	16	80	0,339	0,983	0,661	0,909
3	134	66	68	66	883	33	68	0,492	0,964	0,728	0,904
4	85	59	26	59	922	43	26	0,694	0,955	0,825	0,934
5	103	38	65	38	902	45	65	0,369	0,952	0,661	0,895
6	104	73	31	73	847	99	31	0,702	0,895	0,799	0,876
7	88	84	4	84	899	63	4	0,954	0,934	0,944	0,936
8	116	44	72	44	847	87	72	0,379	0,907	0,643	0,849
9	98	46	52	46	927	25	52	0,469	0,973	0,722	0,927
Total	1 050	52,57%	47,43%	552	8952	498	498	53,95%	94,71%	74,36%	90,52%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 35 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Externa" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Borda Externa											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	70	39	70	914	27	39	0,642	0,971	0,807	0,937
1	92	55	37	55	955	3	37	0,598	0,997	0,797	0,962
2	121	118	3	118	919	10	3	0,975	0,989	0,982	0,988
3	134	127	7	127	908	8	7	0,948	0,991	0,969	0,986
4	85	75	10	75	960	5	10	0,882	0,995	0,939	0,986
5	103	83	20	83	942	5	20	0,806	0,995	0,9	0,976
6	104	103	1	103	922	24	1	0,99	0,975	0,982	0,976
7	88	87	1	87	953	9	1	0,989	0,991	0,99	0,99
8	116	95	21	95	881	53	21	0,819	0,943	0,881	0,929
9	98	92	6	92	951	1	6	0,939	0,999	0,969	0,993
Total	1 050	86,19%	13,81%	905	9305	145	145	85,88%	98,46%	92,16%	97,23%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 36 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Interna" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Borda Interna											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	108	1	108	939	2	1	0,991	0,998	0,994	0,997
1	92	61	31	61	938	20	31	0,663	0,979	0,821	0,951
2	121	119	2	119	899	30	2	0,983	0,968	0,976	0,969
3	134	39	95	39	894	22	95	0,291	0,976	0,633	0,889
4	85	76	9	76	958	7	9	0,894	0,993	0,943	0,985
5	103	64	39	64	943	4	39	0,621	0,996	0,809	0,959
6	104	104	0	104	895	51	0	1	0,946	0,973	0,951
7	88	48	40	48	944	18	40	0,545	0,981	0,763	0,945
8	116	97	19	97	931	103	19	0,836	0,89	0,863	0,884
9	98	76	22	76	951	1	22	0,775	0,999	0,887	0,978
Total	1 050	75,43%	24,57%	792	9292	258	258	75,99%	97,26%	86,62%	95,08%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 37 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características: Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	109	0	109	939	2	0	1	0,998	0,999	0,998
1	92	64	28	64	869	89	28	0,696	0,907	0,801	0,889
2	121	88	33	88	894	35	33	0,727	0,962	0,845	0,935
3	134	26	108	26	872	44	108	0,194	0,952	0,573	0,855
4	85	57	28	57	952	13	28	0,671	0,986	0,828	0,961
5	103	47	56	47	931	16	56	0,456	0,983	0,72	0,931
6	104	98	6	98	932	14	6	0,942	0,985	0,964	0,981
7	88	58	30	58	943	19	30	0,659	0,98	0,82	0,953
8	116	76	40	76	811	123	40	0,655	0,868	0,762	0,845
9	98	60	38	60	940	12	38	0,612	0,987	0,8	0,952
Total	1 050	65,05%	34,95%	683	9083	367	367	66,12%	96,08%	81,12%	93,00%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 38 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características: Histograma: Borda Diagonal Interna (esquerda/direita)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	100	9	100	932	9	9	0,917	0,99	0,954	0,983
1	92	58	34	58	855	103	34	0,63	0,892	0,761	0,869
2	121	117	4	117	900	29	4	0,967	0,969	0,968	0,969
3	134	77	57	77	910	6	57	0,575	0,993	0,784	0,94
4	85	54	31	54	947	18	31	0,635	0,981	0,808	0,953
5	103	27	76	27	911	36	76	0,262	0,962	0,612	0,893
6	104	86	18	86	935	11	18	0,827	0,989	0,908	0,972
7	88	81	7	81	904	58	7	0,92	0,94	0,93	0,938
8	116	61	55	61	914	20	55	0,526	0,979	0,752	0,929
9	98	83	15	83	936	16	15	0,847	0,983	0,915	0,97
Total	1 050	70,86%	29,14%	744	9144	306	306	71,06%	96,78%	83,92%	94,16%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 39 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Cores da Imagem" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Cores da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	3	106	3	902	39	106	0,028	0,959	0,493	0,862
1	92	40	52	40	869	89	52	0,435	0,907	0,671	0,866
2	121	0	121	0	922	7	121	0	0,992	0,496	0,878
3	134	15	119	15	868	48	119	0,112	0,948	0,530	0,841
4	85	9	76	9	842	123	76	0,106	0,873	0,489	0,810
5	103	17	86	17	904	43	86	0,165	0,955	0,560	0,877
6	104	23	81	23	786	160	81	0,221	0,831	0,526	0,770
7	88	3	85	3	955	7	85	0,034	0,993	0,513	0,912
8	116	26	90	26	713	221	90	0,224	0,763	0,494	0,704
9	98	11	87	11	786	166	87	0,112	0,826	0,469	0,759
Total	1 050	14,00%	86,00%	147	8547	903	903	14,37%	90,45%	52,41%	82,80%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 40 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Cores por Coluna da Imagem" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Cores por Coluna da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	63	46	63	916	25	46	0,578	0,973	0,776	0,932
1	92	43	49	43	952	6	49	0,467	0,994	0,731	0,948
2	121	64	57	64	811	118	57	0,529	0,873	0,701	0,833
3	134	104	30	104	824	92	30	0,776	0,900	0,838	0,884
4	85	18	67	18	955	10	67	0,212	0,990	0,601	0,927
5	103	9	94	9	923	24	94	0,087	0,975	0,531	0,888
6	104	89	15	89	784	162	15	0,856	0,829	0,842	0,831
7	88	36	52	36	939	23	52	0,409	0,976	0,693	0,929
8	116	21	95	21	871	63	95	0,181	0,933	0,557	0,850
9	98	48	50	48	920	32	50	0,490	0,966	0,728	0,922
Total	1 050	47,14%	52,86%	495	8895	555	555	45,85%	94,08%	69,97%	89,43%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 41 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Componente Principal" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Componente Principal											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	0	109	0	938	3	109	0	0,997	0,498	0,893
1	92	80	12	80	336	622	12	0,870	0,351	0,610	0,396
2	121	2	119	2	924	5	119	0,017	0,995	0,506	0,882
3	134	0	134	0	916	0	134	0	1	0,500	0,872
4	85	1	84	1	921	44	84	0,012	0,954	0,483	0,878
5	103	2	101	2	932	15	101	0,019	0,984	0,502	0,890
6	104	0	104	0	946	0	104	0	1	0,5	0,901
7	88	17	71	17	703	259	71	0,193	0,731	0,462	0,686
8	116	0	116	0	934	0	116	0	1	0,5	0,890
9	98	0	98	0	952	0	98	0	1	0,5	0,907
Total	1 050	9,71%	90,29%	102	8502	948	948	11,10%	90,11%	50,61%	81,94%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 42 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e a característica "Histograma: Autovalores do Componente Principal" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Autovalores do Componente Principal											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	1	108	1	921	20	108	0,009	0,979	0,494	0,878
1	92	26	66	26	899	59	66	0,283	0,938	0,611	0,881
2	121	28	93	28	842	87	93	0,231	0,906	0,569	0,829
3	134	75	59	75	815	101	59	0,560	0,890	0,725	0,848
4	85	19	66	19	816	149	66	0,224	0,846	0,535	0,795
5	103	43	60	43	747	200	60	0,417	0,789	0,603	0,752
6	104	10	94	10	912	34	94	0,096	0,964	0,530	0,878
7	88	11	77	11	921	41	77	0,125	0,957	0,541	0,888
8	116	29	87	29	830	104	87	0,250	0,889	0,569	0,818
9	98	4	94	4	943	9	94	0,041	0,991	0,516	0,902
Total	1 050	23,43%	76,57%	246	8646	804	804	22,36%	91,48%	56,92%	84,69%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 43 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características: Segmentos de Reta Verticais + Segmentos de Reta Horizontais											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	11	98	11	892	49	98	0,101	0,948	0,524	0,860
1	92	18	74	18	873	85	74	0,196	0,911	0,553	0,849
2	121	42	79	42	786	143	79	0,347	0,846	0,597	0,789
3	134	73	61	73	815	101	61	0,545	0,890	0,717	0,846
4	85	39	46	39	940	25	46	0,459	0,974	0,716	0,932
5	103	43	60	43	864	83	60	0,417	0,912	0,665	0,864
6	104	74	30	74	930	16	30	0,712	0,983	0,847	0,956
7	88	20	68	20	855	107	68	0,227	0,889	0,558	0,833
8	116	17	99	17	890	44	99	0,147	0,953	0,550	0,864
9	98	33	65	33	925	27	65	0,337	0,972	0,654	0,912
Total	1 050	35,24%	64,76%	370	8770	680	680	34,87%	92,78%	63,82%	87,05%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 44 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Cores por Coluna da Imagem" e "Histograma: Autovalores do Componente Principal" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Cores por Coluna da Imagem + Histograma: Autovalores do Componente Principal											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	46	63	46	921	20	63	0,422	0,979	0,700	0,921
1	92	38	54	38	950	8	54	0,413	0,992	0,702	0,941
2	121	60	61	60	828	101	61	0,496	0,891	0,694	0,846
3	134	113	21	113	848	68	21	0,843	0,926	0,885	0,915
4	85	19	66	19	952	13	66	0,224	0,987	0,605	0,925
5	103	21	82	21	901	46	82	0,204	0,951	0,578	0,878
6	104	82	22	82	800	146	22	0,788	0,846	0,817	0,840
7	88	44	44	44	920	42	44	0,500	0,956	0,728	0,918
8	116	35	81	35	847	87	81	0,302	0,907	0,604	0,840
9	98	41	57	41	932	20	57	0,418	0,979	0,699	0,927
Total	1 050	47,52%	52,48%	499	8899	551	551	46,10%	94,13%	70,12%	89,50%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 45 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Interna", "Histograma: Borda Diagonal Interna (esquerda/direita)" e "Histograma: Core por Coluna da Imagem" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (esquerda/direita) + Histograma: Cores por Coluna da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	109	0	109	922	19	0	1	0,980	0,990	0,982
1	92	60	32	60	954	4	32	0,652	0,996	0,824	0,966
2	121	116	5	116	881	48	5	0,959	0,948	0,954	0,950
3	134	93	41	93	907	9	41	0,694	0,990	0,842	0,952
4	85	61	24	61	960	5	24	0,718	0,995	0,856	0,972
5	103	51	52	51	938	9	52	0,495	0,990	0,743	0,942
6	104	104	0	104	894	52	0	1	0,945	0,973	0,950
7	88	58	30	58	944	18	30	0,659	0,981	0,820	0,954
8	116	104	12	104	887	47	12	0,897	0,950	0,923	0,944
9	98	79	19	79	948	4	19	0,806	0,996	0,901	0,978
Total	1 050	79,52%	20,48%	835	9235	215	215	78,79%	97,71%	88,25%	95,90%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 46 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo numérico manuscrito

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Borda Interna + Histograma: Borda Diagonal Interna (esquerda/direita)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	108	1	108	939	2	1	0,991	0,998	0,994	0,997
1	92	60	32	60	938	20	32	0,652	0,979	0,816	0,950
2	121	119	2	119	896	33	2	0,983	0,964	0,974	0,967
3	134	48	86	48	895	21	86	0,358	0,977	0,668	0,898
4	85	77	8	77	957	8	8	0,906	0,992	0,949	0,985
5	103	63	40	63	942	5	40	0,612	0,995	0,803	0,957
6	104	104	0	104	901	45	0	1	0,952	0,976	0,957
7	88	52	36	52	940	22	36	0,591	0,977	0,784	0,945
8	116	94	22	94	843	91	22	0,810	0,903	0,856	0,892
9	98	77	21	77	951	1	21	0,786	0,999	0,892	0,979
Total	1 050	76,38%	23,62%	802	9202	248	248	76,89%	97,36%	87,13%	95,28%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 47 - Testes realizados pelo sistema configurado com o pré-tratamento da imagem e as características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (direita/esquerda)", para o reconhecimentos de caracteres do tipo numérico manuscrito.

Testes realizados com caracteres do tipo numérico manuscrito Características: Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	109	0	109	935	6	0	1	0,994	0,997	0,994
1	92	62	30	62	954	4	30	0,674	0,996	0,835	0,968
2	121	118	3	118	916	13	3	0,975	0,986	0,981	0,985
3	134	123	11	123	910	6	11	0,918	0,993	0,956	0,984
4	85	77	8	77	962	3	8	0,906	0,997	0,951	0,989
5	103	81	22	81	943	4	22	0,786	0,996	0,891	0,975
6	104	103	1	103	922	24	1	0,99	0,975	0,982	0,976
7	88	87	1	87	950	12	1	0,989	0,987	0,988	0,988
8	116	109	7	109	920	14	7	0,94	0,985	0,962	0,98
9	98	93	5	93	950	2	5	0,949	0,998	0,973	0,993
Total	1 050	91,62%	8,38%	962	9362	88	88	91,27%	99,07%	95,16%	98,32%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 48 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo numérico manuscrito.

Testes realizados com caracteres do tipo numérico manuscrito Características: Histograma: Borda Externa + Histograma: Borda Diagonal Interna (esquerda/direita)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	75	34	75	923	18	34	0,6881	0,9809	0,8345	0,9505
1	92	58	34	58	955	3	34	0,6304	0,9969	0,8137	0,9648
2	121	119	2	119	918	11	2	0,9835	0,9882	0,9858	0,9876
3	134	128	6	128	910	6	6	0,9552	0,9934	0,9743	0,9886
4	85	77	8	77	958	7	8	0,9059	0,9927	0,9493	0,9857
5	103	82	21	82	943	4	21	0,7961	0,9958	0,8959	0,9762
6	104	103	1	103	925	21	1	0,9904	0,9778	0,9841	0,9790
7	88	87	1	87	950	12	1	0,9886	0,9875	0,9881	0,9876
8	116	101	15	101	889	45	15	0,8707	0,9518	0,9113	0,9429
9	98	92	6	92	951	1	6	0,9388	0,9989	0,9689	0,9933
Total	1 050	87,81%	12,19%	922	9322	128	128	87,48%	98,64%	93,06%	97,56%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 49 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo numérico manuscrito.

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Borda Interna + Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	108	1	108	939	2	1	0,991	0,998	0,994	0,997
1	92	60	32	60	938	20	32	0,652	0,979	0,816	0,950
2	121	119	2	119	896	33	2	0,983	0,964	0,974	0,967
3	134	48	86	48	895	21	86	0,358	0,977	0,668	0,898
4	85	77	8	77	957	8	8	0,906	0,992	0,949	0,985
5	103	63	40	63	942	5	40	0,612	0,995	0,803	0,957
6	104	104	0	104	901	45	0	1	0,952	0,976	0,957
7	88	52	36	52	940	22	36	0,591	0,977	0,784	0,945
8	116	94	22	94	843	91	22	0,810	0,903	0,856	0,892
9	98	77	21	77	951	1	21	0,786	0,999	0,892	0,979
Total	1 050	76,38%	23,62%	802	9202	248	248	76,89%	97,36%	87,13%	95,28%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 50 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Interna" e "Histograma: Cores por Coluna da Imagem" para o reconhecimentos de caracteres do tipo numérico manuscrito.

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Borda Interna + Histograma: Cores por Coluna da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	108	1	108	930	11	1	0,991	0,988	0,990	0,989
1	92	60	32	60	957	1	32	0,652	0,999	0,826	0,969
2	121	119	2	119	892	37	2	0,983	0,960	0,972	0,963
3	134	78	56	78	900	16	56	0,582	0,983	0,782	0,931
4	85	62	23	62	961	4	23	0,729	0,996	0,863	0,974
5	103	60	43	60	942	5	43	0,583	0,995	0,789	0,954
6	104	104	0	104	891	55	0	1	0,942	0,971	0,948
7	88	60	28	60	945	17	28	0,682	0,982	0,832	0,957
8	116	103	13	103	865	69	13	0,888	0,926	0,907	0,922
9	98	77	21	77	948	4	21	0,786	0,996	0,891	0,976
Total	1 050	79,14%	20,86%	831	9231	219	219	78,76%	97,67%	88,21%	95,83%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 51 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa" e "Histograma: Cores por Coluna da Imagem" para o reconhecimentos de caracteres do tipo numérico manuscrito.

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Borda Externa + Histograma: Cores por Coluna da Imagem											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	87	22	87	929	12	22	0,798	0,987	0,893	0,968
1	92	60	32	60	957	1	32	0,652	0,999	0,826	0,969
2	121	118	3	118	911	18	3	0,975	0,981	0,978	0,980
3	134	132	2	132	911	5	2	0,985	0,995	0,990	0,993
4	85	69	16	69	960	5	16	0,812	0,995	0,903	0,980
5	103	81	22	81	942	5	22	0,786	0,995	0,891	0,974
6	104	104	0	104	921	25	0	1	0,974	0,987	0,976
7	88	87	1	87	953	9	1	0,989	0,991	0,990	0,990
8	116	110	6	110	905	29	6	0,948	0,969	0,959	0,967
9	98	87	11	87	946	6	11	0,888	0,994	0,941	0,984
Total	1 050	89,05%	10,95%	935	9335	115	115	88,33%	98,78%	93,56%	97,81%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 52 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa", "Histograma: Borda Diagonal Interna (direita/esquerda)" e "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo numérico manuscrito.

Testes realizados com caracteres do tipo numérico manuscrito Características Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda) + Histograma: Borda Diagonal Interna (esquerda/direita)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	109	0	109	935	6	0	1	0,994	0,997	0,994
1	92	61	31	61	954	4	31	0,663	0,996	0,829	0,967
2	121	117	4	117	915	14	4	0,967	0,985	0,976	0,983
3	134	122	12	122	910	6	12	0,91	0,993	0,952	0,983
4	85	76	9	76	961	4	9	0,894	0,996	0,945	0,988
5	103	79	24	79	944	3	24	0,767	0,997	0,882	0,974
6	104	103	1	103	920	26	1	0,99	0,972	0,981	0,974
7	88	87	1	87	952	10	1	0,989	0,99	0,989	0,989
8	116	110	6	110	917	17	6	0,948	0,982	0,965	0,978
9	98	94	4	94	950	2	4	0,959	0,998	0,978	0,994
Total	1 050	91,24%	8,76%	958	9358	92	92	90,87%	99,03%	94,94%	98,24%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 53 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa", "Segmentos de Reta Verticais" e "Segmentos de Reta Horizontais" para o reconhecimentos de caracteres do tipo numérico manuscrito.

Testes realizados com caracteres do tipo numérico manuscrito Características: Histograma: Borda Externa + Segmentos de Reta Verticais + Segmentos de Reta Horizontais											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	13	96	13	909	32	96	0,119	0,966	0,543	0,878
1	92	56	36	56	945	13	36	0,609	0,986	0,798	0,953
2	121	107	14	107	917	12	14	0,884	0,987	0,936	0,975
3	134	129	5	129	910	6	5	0,963	0,993	0,978	0,99
4	85	73	12	73	958	7	12	0,859	0,993	0,926	0,982
5	103	93	10	93	944	3	10	0,903	0,997	0,95	0,988
6	104	104	0	104	932	14	0	1	0,985	0,993	0,987
7	88	86	2	86	950	12	2	0,9777	0,987	0,982	0,987
8	116	89	27	89	817	117	27	0,767	0,875	0,821	0,863
9	98	82	16	82	950	2	16	0,837	0,998	0,917	0,983
Total	1 050	79,24%	20,76%	832	9232	218	218	79,19%	97,67%	88,44%	95,86%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

Tabela 54 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e os picos superiores e inferiores das características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo numérico manuscrito.

Testes realizados com caracteres do tipo numérico manuscrito Picos superiores e inferiores das características Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
0	109	5	104	5	931	10	104	0,046	0,989	0,518	0,891
1	92	76	16	76	456	502	16	0,826	0,476	0,651	0,507
2	121	25	96	25	857	72	96	0,201	0,922	0,564	0,84
3	134	1	133	1	916	0	133	0,001	1	0,504	0,873
4	85	13	72	13	924	41	72	0,153	0,957	0,555	0,892
5	103	0	103	0	943	4	103	0	0,498	0,498	0,898
6	104	27	77	27	885	61	77	0,26	0,935	0,598	0,869
7	88	23	65	23	898	64	65	0,261	0,933	0,597	0,877
8	116	26	90	26	939	95	90	0,224	0,898	0,512	0,824
9	98	1	97	1	948	4	97	0,01	0,996	0,503	0,904
Total	1 050	18,76%	81,24%	197	8697	853	853	19,82%	86,04%	55,00%	83,75%
Legenda: VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)											

APÊNDICE C - Testes realizados com caracteres do tipo alfabético

Tabela 55 - Testes realizados pelo sistema configurado sem pré-tratamento da imagem e as características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (direita/esquerda)" para o reconhecimentos de caracteres do tipo alfabético fonte "Arial", tamanho 12.

Testes realizados com caracteres do tipo alfabético com fonte Arial - tamanho 12 (sem pré-tratamento da imagem) Características: Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
A	7	7	0	7	1776	0	0	1	1	1	1
a	95	69	26	69	1674	14	26	0,726	0,992	0,859	0,978
B	5	5	0	5	1772	6	0	1	0,997	0,998	0,997
b	21	21	0	21	1762	0	0	1	1	1	1
C	10	10	0	10	1685	88	0	1	0,950	0,975	0,951
c	99	1	98	1	1684	0	98	0,010	1	0,505	0,945
D	5	5	0	5	1767	11	0	1	0,994	0,997	0,994
d	48	48	0	48	1735	0	0	1	1	1	1
E	5	5	0	5	1778	0	0	1	1	1	1
e	211	208	3	208	1572	0	3	0,986	1	0,993	0,998
F	5	5	0	5	1778	0	0	1	1	1	1
f	29	29	0	29	1748	6	0	1	0,997	0,998	0,997
G	5	5	0	5	1778	0	0	1	1	1	1
g	42	41	1	41	1741	0	1	0,976	1	0,988	0,999
H	6	6	0	6	1777	0	0	1	1	1	1
h	75	75	0	75	1708	0	0	1	1	1	1
I	6	0	6	0	1777	0	6	0	1	0,5	0,997
i	120	110	0	110	1664	0	0	1	1	1	1
J	1	1	0	1	1782	0	0	1	1	1	1
j	5	5	0	5	1778	0	0	1	1	1	1
K	5	5	0	5	1778	0	0	1	1	1	1
k	17	17	0	17	1766	0	0	1	1	1	1
L	5	5	0	5	1778	0	0	1	1	1	1
l	49	39	10	39	1701	33	10	0,796	0,981	0,888	0,976
M	5	5	0	5	1778	0	0	1	1	1	1
m	34	34	0	34	1749	0	0	1	1	1	1
N	10	10	0	10	1773	0	0	1	1	1	1
n	136	135	1	135	1647	0	1	0,993	1	0,996	0,999
O	5	5	0	5	1778	0	0	1	1	1	1

Testes realizados com caracteres do tipo alfabético com fonte Arial - tamanho 12 (sem pré-tratamento da imagem) Características: Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda) - conclusão											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
o	119	104	15	104	1654	10	15	0,874	0,994	0,934	0,986
P	4	4	0	4	1777	2	0	1	0,999	0,999	0,999
p	34	32	0	32	1751	0	0	1	1	1	1
Q	5	5	0	5	1773	5	0	1	0,997	0,999	0,997
q	5	5	0	5	1778	0	0	1	1	1	1
R	7	7	0	7	1775	1	0	1	0,999	1,000	0,999
r	130	124	6	124	1652	1	6	0,954	0,999	0,977	0,996
S	5	5	0	5	1665	113	0	1	0,936	0,968	0,937
s	117	4	113	4	1666	0	113	0,034	1	0,517	0,937
T	5	5	0	5	1778	0	0	1	1	1	1
t	154	152	2	152	1629	0	2	0,987	1	0,994	0,999
U	6	6	0	6	1763	14	0	1	0,992	0,996	0,992
u	33	19	14	19	1750	0	14	0,576	1	0,788	0,992
V	5	5	0	5	1777	1	0	1	0,999	1,000	0,999
v	14	13	1	13	1769	0	1	0,929	1	0,964	0,999
Y	5	5	0	5	1778	0	0	1	1	1	1
y	19	19	1	19	1763	0	1	0,95	1	0,975	0,999
X	5	5	0	5	1778	0	0	1	1	1	1
x	5	5	0	5	1776	2	0	1	0,999	0,999	0,999
W	5	5	0	5	1776	2	0	1	0,999	0,999	0,999
w	25	22	2	22	1758	1	2	0,917	0,999	0,958	0,998
Z	5	5	0	5	1776	2	0	1	0,999	0,999	0,999
z	5	3	2	3	1777	1	2	0,6	0,999	0,800	0,998
Total	1783	82,45%	16,88%	1470	90623	313	301	90,97%	99,66%	95,32%	99,34%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 56 - Testes realizados pelo sistema configurado sem pré-tratamento da imagem e as características "Histograma: Borda Interna" e "Histograma: Borda Diagonal Interna (esquerda/direita)" para o reconhecimentos de caracteres do tipo alfabético fonte "Times New Roman", tamanho 12.

Testes realizados com caracteres do tipo alfabético com fonte Times New Roman - tamanho 12 (sem pré-tratamento da imagem) Características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (esquerda/direita)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
A	5	5	0	5	259	0	0	1	1	1	1
a	6	5	1	5	258	0	1	0,833	1	0,917	0,996
B	5	5	0	5	258	1	0	1	0,996	0,998	0,996
b	6	6	0	6	258	0	0	1	1	1	1
C	5	5	0	5	259	0	0	1	1	1	1
c	5	4	1	4	259	0	1	0,8	1	0,9	0,996
D	5	5	0	5	259	0	0	1	1	1	1
d	5	5	0	5	259	0	0	1	1	1	1
E	5	2	3	2	259	0	3	0,4	1	0,7	0,989
e	5	5	0	5	259	0	0	1	1	1	1
F	5	5	0	5	259	0	0	1	1	1	1
f	5	5	0	5	259	0	0	1	1	1	1
G	5	5	0	5	259	0	0	1	1	1	1
g	5	5	0	5	259	0	0	1	1	1	1
H	5	5	0	5	256	3	0	1	0,988	0,994	0,989
h	5	5	0	5	259	0	0	1	1	1	1
I	5	5	0	5	259	0	0	1	1	1	1
i	5	5	0	5	259	0	0	1	1	1	1
J	5	5	0	5	259	0	0	1	1	1	1
j	5	5	0	5	259	0	0	1	1	1	1
K	5	5	0	5	259	0	0	1	1	1	1
k	5	5	0	5	259	0	0	1	1	1	1
L	5	5	0	5	259	0	0	1	1	1	1
l	5	5	0	5	259	0	0	1	1	1	1
M	5	5	0	5	259	0	0	1	1	1	1
m	5	5	0	5	259	0	0	1	1	1	1
N	5	5	0	5	259	0	0	1	1	1	1
n	5	5	0	5	259	0	0	1	1	1	1
O	5	5	0	5	259	0	0	1	1	1	1
o	6	5	1	5	257	1	1	0,833	0,996	0,915	0,992
P	5	5	0	5	258	1	0	1	0,996	0,998	0,996
p	6	5	1	5	258	0	1	0,833	1	0,917	0,996
Q	5	5	0	5	259	0	0	1	1	1	1

Testes realizados com caracteres do tipos alfabético com fonte Times New Roman - tamanho 12 (sem pré-tratamento da imagem) Características: Histograma: Borda Interna + Histograma: Borda Diagonal Interna (esquerda/direita) - conclusão											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
q	5	5	0	5	259	0	0	1	1	1	1
R	5	5	0	5	259	0	0	1	1	1	1
r	5	5	0	5	259	0	0	1	1	1	1
S	5	5	0	5	254	5	0	1	0,981	0,990	0,981
s	5	0	5	0	259	0	5	0	1	0,500	0,981
T	5	5	0	5	259	0	0	1	1	1	1
t	5	5	0	5	259	0	0	1	1	1	1
U	5	5	0	5	259	0	0	1	1	1	1
u	5	5	0	5	258	1	0	1	0,996	0,998	0,996
V	5	5	0	5	259	0	0	1	1	1	1
v	5	4	1	4	259	0	1	0,8	1	0,9	0,996
Y	5	5	0	5	258	1	0	1	0,996	0,998	0,996
y	5	5	0	5	259	0	0	1	1	1	1
X	5	5	0	5	259	0	0	1	1	1	1
x	5	5	0	5	259	0	0	1	1	1	1
W	5	5	0	5	259	0	0	1	1	1	1
w	5	5	0	5	259	0	0	1	1	1	1
Z	5	5	0	5	259	0	0	1	1	1	1
z	5	5	0	5	259	0	0	1	1	1	1
Total	264	95,08%	4,92%	251	13451	13	13	95,19%	99,90%	97,55%	99,81%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

Tabela 57 - Testes realizados pelo sistema configurado com os filtros de pré-tratamento da imagem e as características "Histograma: Borda Externa" e "Histograma: Borda Diagonal Interna (direita/ esquerda)" para o reconhecimentos de caracteres do tipo alfabético manuscrito.

Testes realizados com caracteres do tipo alfabético manuscrito Características: Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda)											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
A	8	3	5	3	264	12	5	0,375	0,957	0,666	0,940
a	7	4	3	4	271	6	3	0,571	0,978	0,775	0,968
B	7	4	3	4	275	2	3	0,571	0,993	0,782	0,982
b	7	4	3	4	272	5	3	0,571	0,982	0,777	0,972
C	8	2	6	2	276	0	6	0,25	1	0,625	0,979
c	6	1	5	1	277	1	5	0,167	0,996	0,582	0,979
D	6	2	4	2	277	1	4	0,333	0,996	0,665	0,982
d	7	4	3	4	274	3	3	0,571	0,989	0,780	0,979
E	5	3	2	3	277	2	2	0,6	0,993	0,796	0,986
e	5	2	3	2	275	4	3	0,4	0,986	0,693	0,975
F	5	3	2	3	279	0	2	0,6	1	0,8	0,993
f	5	3	2	3	278	1	2	0,6	0,996	0,798	0,989
G	4	2	2	2	279	1	2	0,5	0,996	0,748	0,989
g	7	1	6	1	277	0	6	0,143	1	0,571	0,979
H	5	4	1	4	278	1	1	0,8	0,996	0,898	0,993
h	5	2	3	2	277	2	3	0,4	0,993	0,696	0,982
I	4	3	1	3	280	0	1	0,75	1	0,875	0,996
i	5	2	3	2	279	0	3	0,4	1	0,7	0,989
J	4	4	0	4	276	4	0	1	0,986	0,993	0,986
j	6	0	6	0	277	1	6	0	0,996	0,498	0,975
K	5	0	5	0	279	0	5	0	1	0,5	0,982
k	4	0	4	0	277	3	4	0	0,989	0,495	0,975
L	5	3	2	3	277	2	2	0,6	0,993	0,796	0,986

Testes realizados com caracteres do tipo alfabético manuscrito Características: Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda) - continuação											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
l	5	2	3	2	274	5	3	0,4	0,982	0,691	0,972
M	6	3	3	3	277	1	3	0,5	0,996	0,748	0,986
m	5	1	4	1	275	4	4	0,2	0,986	0,593	0,972
N	5	4	1	4	279	0	1	0,8	1	0,9	0,996
n	6	0	6	0	277	1	6	0	0,996	0,498	0,975
O	6	4	2	4	272	6	2	0,667	0,978	0,823	0,972
o	5	4	1	4	272	7	1	0,8	0,975	0,887	0,972
P	6	5	1	5	276	2	1	0,833	0,993	0,913	0,989
p	6	3	3	3	278	0	3	0,5	1	0,75	0,989
Q	5	2	3	2	274	5	3	0,4	0,982	0,691	0,972
q	5	5	0	5	271	8	0	1	0,971	0,986	0,972
R	6	2	4	2	274	4	4	0,333	0,986	0,659	0,972
r	5	3	2	3	266	13	2	0,6	0,953	0,777	0,947
S	7	2	5	2	276	1	5	0,286	0,996	0,641	0,979
s	4	2	2	2	276	4	2	0,5	0,986	0,743	0,979
T	5	5	0	5	279	0	0	1	1	1	1
t	5	1	4	1	278	1	4	0,2	0,996	0,598	0,982
U	5	4	1	4	274	5	1	0,8	0,982	0,891	0,979
u	6	5	1	5	269	9	1	0,833	0,968	0,900	0,965
V	5	3	2	3	272	7	2	0,6	0,975	0,787	0,968
v	5	1	4	1	276	3	4	0,2	0,989	0,595	0,975
Y	5	3	2	3	276	3	2	0,6	0,989	0,795	0,982
y	5	2	3	2	277	2	3	0,4	0,993	0,696	0,982
X	5	4	1	4	277	2	1	0,8	0,993	0,896	0,989
x	5	3	2	3	273	6	2	0,6	0,978	0,789	0,972

Testes realizados com caracteres do tipo alfabético manuscrito Características: Histograma: Borda Externa + Histograma: Borda Diagonal Interna (direita/esquerda) - conclusão											
Tipo da Amostra	**Tamanho da Amostra**	**Acertos**	**Erros**	**VP**	**VN**	**FP**	**FN**	**Sensibilidade**	**Especificidade**	**Eficiência**	**Acurácia**
W	6	1	5	1	278	0	5	0,167	1	0,583	0,982
W	4	2	2	2	279	1	2	0,5	0,996	0,748	0,989
Z	6	0	6	0	277	1	6	0	0,996	0,498	0,975
z	5	0	5	0	279	0	5	0	1	0,5	0,982
Total	284	46,48%	53,52%	132	14332	152	152	47,54%	98,95%	73,25%	97,94%

Legenda:
VP - Verdadeiro Positivo, VN - Verdadeiro Negativo, FP - Falso Positivo, FN - Falso Negativo, Sensibilidade - VP / (VP+FN), Especificidade - VN / (VN+FP), Eficiência - (Sensibilidade + Especificidade) / 2, Acurácia - (VP + VN) / (VP + FP + FN + VN)

APÊNDICE D - Curva ROC

A curva ROC referente aos testes realizados com caracteres do tipo CMC7 e numérico manuscrito nas Tabelas 3 e 5 é apresentada na figura abaixo.

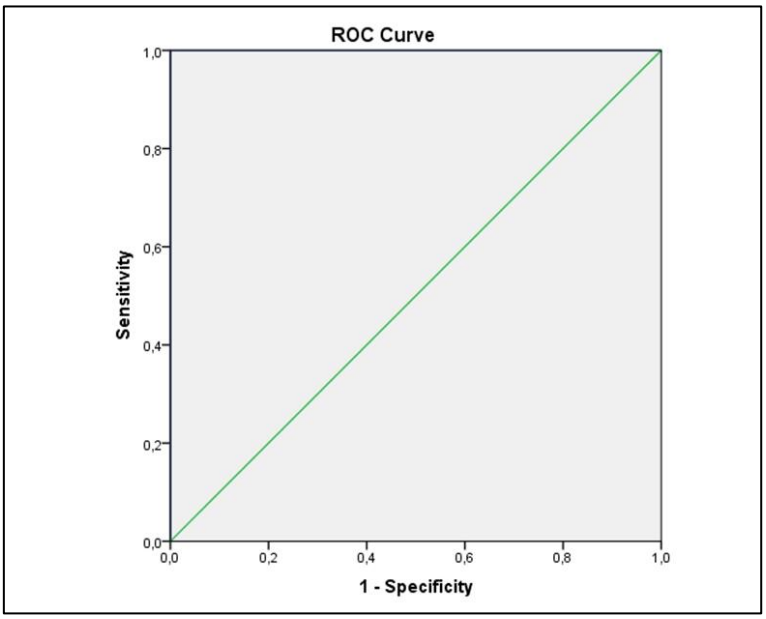


Figura - Curva ROC que representa os testes realizados com caracteres do tipo CMC-7 (Tabela 3) e numérico manuscrito (Tabelas 5) com exceção dos caracteres '1' e '6' da Tabela 5

Para gerar a curva ROC de cada elemento para ambos os tipos de caracteres, CMC-7 e numérico manuscrito, adotou-se a estratégia de vincular as probabilidades de acerto e de erro às tentativas através de uma função exponencial e utilizou-se o software SPSS.

Com exceção dos elementos '1' e '6' do tipo numérico manuscrito, quando considera-se os dados das probabilidades de acerto e erro, os gráficos apresentam a mesma aparência, com área igual a '1', ou seja, só existe um gráfico que representa todas as curvas ROC, onde os pontos encontram-se sobre o eixo 'y' e sobre o eixo 'x-superior'.

Para os elementos '1' e '6' do tipo numérico manuscrito ambos apresentaram 100% de acerto. Devido ao fato da curva ROC somente poder ser construída para dois e somente dois valores, não foi possível construir a curva ROC para esses elementos.

APÊNDICE E - Algoritmos das células neurais artificiais paraconsistentes utilizados no desenvolvimento do sistema

Algoritmo da CNAPar (Da Silva Filho et al., 2008)

1. Entre com o valor do grau de evidência de entrada 1
μ_1 */Grau de evidência 1 $0 \leq \mu_1 \leq 1$ */

2. Entre com o valor do grau de evidência de entrada 2
μ_2 */Grau de evidência 2 $0 \leq \mu_2 \leq 1$ */

3. Transforme o grau de evidência 2 em grau de evidência desfavorável
$\lambda_2 = 1 - \mu_2$ */Grau de evidência desfavorável $0 \leq \lambda_2 \leq 1$ */

4. Calcule o grau de contradição normalizado

$$\mu_{ctr} = \frac{\mu_1 + \lambda_2}{2}$$

5. Calcule o intervalo de evidência resultante

$$\varphi_E = 1 - |2\mu_{ctr} - 1|$$

6. Calcule o grau de evidência resultante

$$\mu_E = \frac{(\mu_1 - \lambda_2) + 1}{2}$$

7. Calcule a distância D

$$D = \sqrt{(1 - |\mu_1 - \lambda_2|)^2 + ((\mu_1 + \lambda_2) - 1)^2}$$

8. Calcule o grau de evidência resultante real
Se $\mu_E > 0{,}5$, $\mu_{ER} = (1 - D/2)$
Se $\mu_E < 0{,}5$, $\mu_{ER} = D/2$
Se $\mu_E = 0{,}5$, $\mu_{ER} = \mu_E$

9. Faça $Ft_{ct} = 1$ e apresente na saída
$S_2 = \varphi_E$
Se $\varphi_E \leq 0{,}25$ então, $S_1 = 0{,}5$ */Indefinição*/
Senão $S_1 = \mu_{ER}$

10. Fim

Algoritmo da CNAPdi (Da Silva Filho et al., 2008)

1. Entre com o valor do grau de evidência de entrada 1
μ_1 */Grau de evidência 1 $0 \leq \mu_1 \leq 1$ */

2. Entre com o valor do grau de evidência de entrada 2
μ_2 */Grau de evidência 2 $0 \leq \mu_2 \leq 1$ */

3. Entre com o valor do Fator de tolerância à contradição
$Ft_{ct} = C_1$ */Fator de tolerância à contradição $0 \leq Ft_{ct} \leq 1$ */

4. Calcule os valores superior e inferior de controle de incerteza

$V_{scin} = \frac{1+C_1}{2}$ */Valor superior de controle de incerteza*/

$V_{icin} = \frac{1-C_1}{2}$ */Valor inferior de controle de incerteza*/

5. Transforme o grau de evidência 2 em grau de evidência desfavorável
$\lambda_2 = 1 - \mu_2$ */Grau de evidência desfavorável $0 \leq \lambda_2 \leq 1$ */

6. Calcule o grau de contradição normalizado

$$\mu_{ctr} = \frac{\mu_1 + \lambda_2}{2}$$

7. Apresente os sinais resultantes das saídas a partir das seguintes condicionais
Se $V_{scin} > \mu_{ctr} > V_{icin}$ então, $S_1 = 1$
Senão $S_1 = 0$

8. Fim

Algoritmo da CNAPcls Max e Min (Da Silva Filho et al., 2008)

1. Entre com o valor do grau de evidência de entrada 1
μ_1 */Grau de evidência 1 $0 \leq \mu_1 \leq 1$ */

2. Entre com o valor do grau de evidência de entrada 2
μ_2 */Grau de evidência 2 $0 \leq \mu_2 \leq 1$ */

3. Transforme o grau de evidência 2 em grau de evidência desfavorável
$\lambda_2 = 1 - \mu_2$ */Grau de evidência desfavorável $0 \leq \lambda_2 \leq 1$ */

4. Calcule o grau de evidência resultante

$$\mu_E = \frac{(\mu_1 - \lambda_2) + 1}{2}$$

5. Apresente os sinais resultantes das saídas a partir das seguintes condicionais
   Para maximização
      Se $\mu_E >= 0,5$ faça, $S_1 = \mu_1$ */Valor maior de evidência de entrada*/
      Se $\mu_E < 0,5$ faça, $S_1 = \mu_2$ */Valor maior de evidência de entrada*/
   Para minimização
      Se $\mu_E >= 0,5$ faça, $S_1 = \mu_2$ */Valor menor de evidência de entrada*/
      Se $\mu_E < 0,5$ faça, $S_1 = \mu_1$ */Valor menor de evidência de entrada*/

6. Fim

Algoritmo da CNAPpa (Da Silva Filho et al., 2008)

1. Entre com o valor do grau de evidência de entrada 1
μ_1 */Grau de evidência 1 $0 \leq \mu_1 \leq 1$ */

2. Entre com o valor do Fator de tolerância à certeza
$Ft_c = C_2$ */Fator de tolerância à certeza $0 \leq Ft_{ct} \leq 1$ */

3. Calcule os valores superior e inferior de controle de certeza

$V_{scc} = \frac{1+C_2}{2}$ */Valor superior de controle de certeza*/

$V_{icc} = \frac{1-C_2}{2}$ */Valor inferior de controle de certeza*/

4. Transforme o grau de evidência 1 em grau de evidência desfavorável
$\lambda = 1 - \mu_1$ */Grau de evidência desfavorável $0 \leq \lambda \leq 1$ */

5. Calcule o grau de evidência resultante

$$\mu_E = \frac{(\mu_1 - \lambda) + 1}{2}$$

6. Apresente na saída
Se $V_{scc} \leq \mu_E$ ou $\mu_E \leq V_{icc}$ então, $S_1 = \mu_1$
Senão, $S_1 = 0{,}5$ */Indefinição*/

7. Fim

APÊNDICE F - Algoritmo do Nó de Análise Paraconsistente e Algoritmo ParaExtr$_{ctr}$

Algoritmo do Nó de Análise Paraconsistente (Da Silva Filho, 2009)

1. Entre com os valores de entrada

μ	*/Grau de evidência favorável	$0 \leq \mu \leq 1$ */
λ	*/Grau de evidência desfavorável	$0 \leq \lambda \leq 1$ */

2. Calcule o grau de contradição normalizado

$$\mu_{ctr} = \frac{\mu + \lambda}{2}$$

3. Calcule o intervalo de evidência resultante

$$\varphi_E = 1 - |2\mu_{ctr} - 1|$$

4. Calcule o grau de certeza

$G_c = \mu - \lambda$

5. Calcule o grau de incerteza

$G_{in} = (\mu + \lambda) - 1$

6. Calcule a distância D

$$D = \sqrt{(1 - |G_c|)^2 + {G_{ct}}^2}$$

7. Determine o sinal da saída
   Se $\varphi_E \leq 0{,}25$ ou $D > 1$ então, $S_1 = 0{,}5$ e $S_2 = \varphi_{E(\pm)}$: Indefinição e vá para o item 11
   Senão vá para o próximo item

8. Determine o grau de certeza real

$$Se\ G_C > 0\ então, G_{CR} = (1 - D)$$

$$Se\ G_C < 0\ então, G_{CR} = (D - 1)$$

$$Se\ G_C = 0\ então, G_{CR} = 0$$

9. Calcule o grau de evidência resultante real

$$\mu_{ER} = \frac{G_{CR} + 1}{2}$$

10. Apresente os resultados na saída

Faça $S_1 = \mu_{ER}$ e $S_2 = \varphi_{E(\pm)}$

11. Fim

Algoritmo ParaExtr$_{ctr}$ (Da Silva Filho, 2009)

1. Apresente os n valores de graus de evidência que compõe o grupo em estudo

$G\mu = (\mu_A, \mu_B, \mu_C, ..., \mu_n)$ */Graus de evidência $0 \leq \mu \leq 1$ */

2. Selecione o maior valor entre os graus de evidência do grupo em estudo

$\mu_{maxA} = \max(\mu_A, \mu_B, \mu_C, ..., \mu_n)$

3. Considere o maior valor entre os graus de evidência do grupo em estudo em graus de evidência favorável

$\mu_{maxA} = \mu$

4. Selecione o menor valor entre os graus de evidência do grupo em estudo

$\mu_{mimA} = \text{mim}(\mu_A, \mu_B, \mu_C, ..., \mu_n)$

5. Transforme menor valor em grau de evidência desfavorável

$1 - \mu_{mimA} = \lambda$

6. Faça a análise paraconsistente entre os valores selecionados

$\mu_{R1} = \mu \lozenge \lambda$ */Utilização de um NAP/

7. Acrescente o valor obtido μ_{R1} no grupo em estudo, excluindo deste os dois valores μ_{max} e μ_{mim} selecionados anteriormente

$G\mu = (\mu_A, \mu_B, \mu_C, ..., \mu_n, \mu_{R1}) - (\mu_{maxA}, \mu_{mimA})$

8. Retorne ao item 2 até que o grupo em estudo tenha um único elemento resultante das análises

$G\mu = (\mu_{ER})$

9. Fim

APÊNDICE G - Código-fonte do Sistema Computacional Paraconsistente

O Sistema Computacional Paraconsistente foi desenvolvido através da linguagem de programação Python e do framework Pyramid. A seguir apresenta-se o código-fonte dos principais módulos do sistema que estão organizados em três diretórios distintos (models, utils, views) dentro do sistema.

Diretório models:

class UAPParaExtr():

```
''' Unidade de Análise Paraconsistente com o algoritmo ParaExtr_ctr - UAPParaExtr
'''
def __init__(self, list_ge):
    ''' Construtor da classe
        Atribui os parâmetros passados ao objeto
        :list_ge: Grau de evidência 1 (0 <= ge <= 1)
    '''
    self.list_ge = list_ge
    self.layers = []

def create_network(self):
    ''' Adiciona um NAP para cada camada da rede
    '''
    # Enquanto a quantidade de elementos de list_ge for maior do que 1
    while len(self.list_ge) > 1:
        # Adiciona a nova camada à lista de camadas
        self.layers.append(Layer())
        # Obtém o maior e menor valor da lista
        max_value = max(self.list_ge)
        min_value = min(self.list_ge)
        # Remove o maior e menor valor da lista
        self.list_ge.remove(max_value)
        self.list_ge.remove(min_value)
        # Calcula mi e lambda (l)
        mi = max_value
        l = 1 - min_value
        # Adiciona um NAP à lista de elementos da camada
        self.layers[-1].elements.append(NAP(mi, l))
        # Adiciona um dicionário vazio à lista de conexões da camada
        self.layers[-1].connections.append({})
        # Adiciona a saída do NAP à lista de valores da camada
        nap_results = self.layers[-1].elements[0].results()
        self.layers[-1].values.append(nap_results)
```

```
        # Adiciona a saída 's1' do NAP na lista list_ge
        nap_s1 = self.layers[-1].values[0]['s1']
        self.list_ge.append(nap_s1)
    def results(self):
        ''' Retorna a saída da rede '''
        self.create_network()
        self.s1 = self.list_ge[0]
        return self.s1
```

class UARCcmc7():

```
    ''' Unidade Neural Artificial Paraconsistente
        Unidade de Agrupamento do Reconhecimento do Caractere CMC7 - UARCcmc7
    '''
    def __init__(        self, ge_pattern, ge_caractere, conf_ftct={'vertical': 0.1, 'horizontal':0.1, 'histogram':0.1},
                         caracteristics=None):
        ''' Construtor da classe
            Atribui os parâmetros passados ao objeto
            :ge_pattern: Dicionário com as seguintes listas de graus de evidência do caractere padrão
                    lado esquerdo, lado direito, topo, base e histogramas (0 <= ge_padrão <= 1)
            :ge_caractere: Dicionário com as seguintes listas de graus de evidência do caractere
                    lado esquerdo, lado direito, topo, base, histograma externo e histograma interno (0 <= ge <= 1)
            :conf_ftct: Define o valor default do fator de tolerância à contradição das células CNAPdi das unidades
                            UARL das linhas verticais, horizontais e do histograma
        '''
        self.ge_pattern = ge_pattern
        self.ge_caractere = ge_caractere
        self.conf_ftct = conf_ftct
        self.ge_dif_histogram = self._get_dif_from_evidences(caracteristics)
        self.caracteristics = caracteristics
        self.layers = []

    def create_network(self):
        ''' Adiciona duas UARL para análise dos segmentos de reta identificados e
            uma URPcmc7 para análise de cada lista de graus de evidência dos histogramas (externo e interno)
            das imagens do caractere e caractere padrão
            :self.ge_pattern['left'], self.ge_caractere['left']: listas de graus de evidência dos segmentos de reta
                         identificados no lado esquerdo das imagens do caractere e caractere padrão
            :self.ge_pattern['right'], self.ge_caractere['right']: listas de graus de evidência dos segmentos de reta
                         identificados no lado direito das imagens do caractere e caractere padrão
            :self.ge_pattern['top'], self.ge_caractere['up']: listas de graus de evidência dos segmentos de reta
                         identificados no topo das imagens do caractere e caractere padrão
            :self.ge_pattern['bottom'], self.ge_caractere['down']: listas de graus de evidência dos segmentos de reta
                         identificados na base das imagens do caractere e caractere padrão
        '''
        # Adiciona a nova camada à lista de camadas
```

```
        self.layers.append(Layer())
        # Adiciona uma UARL na lista de elementos
        if 'vertical_line_segments' in self.caracteristics:
            self.layers[0].elements.append(UARL(self.ge_pattern['external_line_segments']['left'],
                                                self.ge_caractere['external_line_segments']['left'],
                                                self.ge_pattern['external_line_segments']['right'],
                                                self.ge_caractere['external_line_segments']['right'],
                                                self.conf_ftct.get('vertical')))
        # Adiciona uma UARL na lista de elementos
        if 'horizontal_line_segments' in self.caracteristics:
            self.layers[0].elements.append(UARL(self.ge_pattern['external_line_segments']['top'],
                                                self.ge_caractere['external_line_segments']['top'],
                                                self.ge_pattern['external_line_segments']['bottom'],
                                                self.ge_caractere['external_line_segments']['bottom'],
                                                self.conf_ftct.get('horizontal')))
        # Adiciona uma UAPParaExtr para cada caracteristica na lista de elementos do objeto "Layer"
        # exceto para as caracteristicas 'vertical_line_segments' e 'horizontal_line_segments'
        for caracteristic in self.caracteristics:
            if not caracteristic in ('vertical_line_segments', 'horizontal_line_segments', 'top_bottom_peaks'):
                self.layers[0].elements.append(UAPParaExtr(self.ge_dif_histogram[caracteristic]))
        # Defini um indice correspondente à lista de elementos do objeto "Layer"
        # Cada índice corresponde a um item da lista "caracteristics"
        caracteristic_index = 0
        # Adiciona o valor/resultado de cada elemento à lista "values" do objeto "Layer"
        for caracteristic in self.caracteristics:
            if caracteristic != 'top_bottom_peaks':
                self.layers[0].values.append(self.layers[0].elements[caracteristic_index].results())
                # Incrementa o índice para referenciar o próximo elemento da lista de elementos do objeto "Layer"
                caracteristic_index = caracteristic_index + 1

    def results(self):
        ''' Retorna as saídas da RNAP '''
        # Cria e processa a RNAP
        self.create_network()
        # Defini um indice correspondente à lista de elementos do objeto "Layer"
        # Cada índice corresponde a um item da lista "caracteristics"
        caracteristic_index = 0
        # Cria um atributo para cada item da lista "self.caracteristics" no "self" e atribui o valor obtido
        # para cada um deles na rede
        for item in self.caracteristics:
            if item != 'top_bottom_peaks':
                # Ex. self.vertical_line_segments = self.layers[0].values[0]
                setattr(self, '%s' % item, self.layers[0].values[caracteristic_index])
                caracteristic_index = caracteristic_index + 1
        caracteristic_index = 0
        results = {}
        # Adiciona um item ao dicionário "results" para cada item da lista "self.caracteristics" e seu respectivo valor calculado pela RNAP
        for caracteristic in self.caracteristics:
```

```
        if caracteristic != 'top_bottom_peaks':
            # Ex. results[vertical_line_segments] = self.layers[0].values[0]
            results[caracteristic] = self.layers[0].values[caracteristic_index]
            caracteristic_index = caracteristic_index + 1
    return results
def _get_dif_from_evidences(self, caracteristics):
    ''' Retorna uma lista com a diferença entre os graus de evidência
        Ex. ge_diff[0] = fabs(ge_caractere[0] - ge_pattern[0])
    '''
    # Cria um dicionário vazio
    ge_dif = {}
    # Para cada tipo de característica gera uma lista com a diferença entre os graus de evidência
    for tipo in caracteristics:
        if not tipo in ('vertical_line_segments', 'horizontal_line_segments', 'top_bottom_peaks'):
            ge_dif[tipo] = []
            # Se a caracteristica 'top_bottom_peaks' for solicitada completa as listas ge_pattern e
              # ge_caractere com valor "0" até que ambas estejam com o mesmo tamanho
            if 'top_bottom_peaks' in caracteristics:
                while len(self.ge_pattern['histogram'][tipo]) > len(self.ge_caractere['histogram'][tipo]):
                    self.ge_caractere['histogram'][tipo].append(0)
                while len(self.ge_caractere['histogram'][tipo]) > len(self.ge_pattern['histogram'][tipo]):
                    self.ge_pattern['histogram'][tipo].append(0)
            # Para cada par de item de ambas as listas (ge_caractere e ge_pattern) calcula a
              # diferença entre as listas através de uma NAP
            for i in range(len(self.ge_caractere['histogram'][tipo])):
                nap = NAP(self.ge_pattern['histogram'][tipo][i], 1 - self.ge_caractere['histogram'][tipo][i])
                dif = nap.results()['s2']
                ge_dif[tipo].append(dif)
    return ge_dif
```

class UARL():

```
    ''' Unidade Neural Artificial Paraconsistente
        Unidade de Agrupamento do Reconhecimento das Linhas - UARL
    '''
    def __init__(self, ge_default1, ge_caractere1, ge_default2, ge_caractere2, ftct=0.1):
        ''' Construtor da classe
            Atribui os parâmetros passados ao objeto
            :ge_default1: Lista de graus de evidência referente aos segmentos de reta de um lado do
                          caractere padrão (0 <= elemento <= 1)
            :ge_caractere1: Lista de graus de evidência referente aos segmentos de reta de um lado do
                          caractere (0 <= elemento <= 1)
            :ge_default2: Lista de graus de evidência referente aos segmentos de reta de um lado do
                          caractere padrão (0 <= elemento <= 1)
            :ge_caractere2: Lista de graus de evidência referente aos segmentos de reta de um lado do
                          caractere (0 <= elemento <= 1)
            :ftct: Define o valor default do fator de tolerância à contradição das células CNAPdi das
                          unidades URPcmc7
        '''
        self.ge_default1 = ge_default1
        self.ge_caractere1 = ge_caractere1
```

```
        self.ge_default2 = ge_default2
        self.ge_caractere2 = ge_caractere2
        self.ftct = ftct
        self.layers = []

    def create_network(self):
        ''' Adiciona unidades URPcmc7 à primeira camada da rede
        '''
        ''' Adiciona a PRIMEIRA CAMADA da rede '''
        # Adiciona a PRIMEIRA camada à lista de camadas
        self.layers.append(Layer())
        # Adiciona as unidades URPcmc7 à lista de elementos da camada
        self.layers[0].elements.append(URPcmc7(self.ge_default1, self.ge_caractere1, self.ftct))
        self.layers[0].elements.append(URPcmc7(self.ge_default2, self.ge_caractere2, self.ftct))
        # Adiciona um dicionário vazio para cada célula na lista de conexões
        # da camada (Recebe um dicionário vazio porque é a primeira camada)
        self.layers[0].connections.append({})
        self.layers[0].connections.append({})
        # Obtém os valores das URPcmc7
        urpcmc7_1 = self.layers[0].elements[0].results()
        urpcmc7_2 = self.layers[0].elements[1].results()

        # Adiciona os valores das saídas das URPcmc7 na lista de valores da primeira camada
        self.layers[0].values.append(urpcmc7_1)
        self.layers[0].values.append(urpcmc7_2)
        ''' Adiciona a ÚLTIMA CAMADA da rede '''
        # Adiciona a ÚLTIMA camada à lista de camadas
        self.layers.append(Layer())
        # Adiciona uma CNAPa na última camada da rede
        mi1 = self.layers[-2].values[0]
        mi2 = self.layers[-2].values[1]
        ftct = 1
        ftc = 0
        self.layers[-1].elements.append(CNAPa(mi1, mi2, ftct, ftc))
        # Adiciona as conexões na lista de conexões da última camada
        self.layers[-1].connections.append(
                                    {
                                        'mi1': {'layer': 0, 'element': 0},
                                        'mi1': {'layer': 0, 'element': 1}
                                    }
                                  )
        # Adiciona os valores das saídas da CNAPa na lista de valores da camada
        cnapa = self.layers[-1].elements[-1].results()['s1']
        self.layers[-1].values.append(cnapa)

    def results(self):
        ''' Retorna a saída da rede '''
        self.create_network()
        self.s1 = self.layers[-1].values[0]
        return self.s1
```

class UCCF():

```
    ''' Unidade Neural Artificial Paraconsistente
        Unidade das Células das Camadas Finais - UCCF
    '''
    def __init__(self, entrada1, entrada2):
        ''' Construtor da classe
            Atribui os parâmetros passados ao objeto
            :entrada1: Dicionário com as saídas AR e DI do elemento mi1 (0 <= ar1 <= 1 e 0 <= di1 <= 1)
            :entrada2: Dicionário com as saídas AR e DI do elemento mi2 (0 <= ar2 <= 1 e 0 <= di2 <= 1)
```

```
    '''
    self.ar1 = entrada1['ar']
    self.ar2 = entrada2['ar']
    self.di1 = entrada1['di']
    self.di2 = entrada2['di']
    self.layers = []

def create_network(self):
    ''' Adiciona as células CNAPmin(1), CNAPdi(5) e CNAPmax(7) à primeira camada da rede
        Adiciona as células CNAPmin(2) e CNAPdi(8) à segunda camada da rede
        Adiciona a célula CNAPmin(3) à terceira camada da rede
        Adiciona as células CNAPdi(4) e CNAPar(6) à quarta e última camada da rede
    '''
    ''' Adiciona a PRIMEIRA CAMADA da rede '''
    self.layers.append(Layer())

    ### Adiciona as células CNAPmin(1), CNAPdi(5) e CNAPmax(7) à lista de elementos da primeira camada
    # Adiciona a célula CNAPmin(1)
    self.layers[0].elements.append(CNAPmin(self.ar1, self.ar2))
    # Calcula a saída da CNAPmin(1) e atribui o resultado ao fator de tolerância à contradição da CNAPdi(5)
    cnapmin1_results = self.layers[0].elements[0].results()
    # Adiciona a célula CNAPdi(5)
    self.layers[0].elements.append(CNAPdi(self.ar1, self.ar2, cnapmin1_results))
    # Adiciona a célula CNAPmax(7)
    self.layers[0].elements.append(CNAPmax(self.di1, self.di2))
    ### Adiciona as conexões das células da primeira camada
    # Adiciona as conexões da célula CNAPmin(1)
    self.layers[0].connections.append({})
    # Adiciona as conexões da célula CNAPdi(5)
    self.layers[0].connections.append(
                                {
                                    'ftct': {'layer': 0, 'element': 0}
                                }
                            )
    # Adiciona as conexões da célula CNAPmax(7)
    self.layers[0].connections.append({})
    ### Adiciona os valores das saídas das células da primeira camada
    # Adiciona o valor da célula CNAPmin(1)
    self.layers[0].values.append(self.layers[0].elements[0].results())
    # Adiciona o valor da célula CNAPdi(5)
    self.layers[0].values.append(self.layers[0].elements[1].results())
    # Adiciona o valor da célula CNAPmax(7)
    self.layers[0].values.append(self.layers[0].elements[2].results())

    ''' Adiciona a SEGUNDA CAMADA da rede '''
    self.layers.append(Layer())
    ### Adiciona as células CNAPmin(2) e CNAPdi(8) à lista de elementos da segunda camada
    # Obtém as saídas das células CNAPmin(1) e CNAPdi(5) da primeira camada
    cnapmin1 = self.layers[0].values[0]
    cnapdi5 = self.layers[0].values[1]
    # Adiciona a célula CNAPmin(2)
    self.layers[1].elements.append(CNAPmin(cnapmin1, cnapdi5))
    # Obtém a saída da célula CNAPmax(7)da primeira camada
    cnapmax7 = self.layers[0].values[2]
    # Adiciona a célula CNAPdi(8)
    self.layers[1].elements.append(CNAPdi(self.di1, self.di2, cnapmax7))
    ### Adiciona as conexões das células da segunda camada
    # Adiciona as conexões da célula CNAPmin(2)
    self.layers[1].connections.append(
                                {
                                    'mi1': {'layer': 0, 'element': 0},
                                    'mi2': {'layer': 0, 'element': 1}
                                }
```

```
                        )
    # Adiciona as conexões da célula CNAPdi(8)
    self.layers[1].connections.append(
                            {
                                'ftct': {'layer': 0, 'element': 2}
                            }
                        )
    ### Adiciona os valores das saídas das células da segunda camada
    # Adiciona o valor da célula CNAPmin(2)
    self.layers[1].values.append(self.layers[1].elements[0].results())
    # Adiciona o valor da célula CNAPdi(8)
    self.layers[1].values.append(self.layers[1].elements[1].results())

    ''' Adiciona a TERCEIRA CAMADA da rede '''
    self.layers.append(Layer())
    ### Adiciona a célula CNAPmin(3)à lista de elementos da terceira camada
    # Obtém as saídas das células CNAPmin(2) e CNAPdi(8) da segunda camada
    cnapmin2    = self.layers[1].values[0]
    cnapdi8     = self.layers[1].values[1]
    # Adiciona a célula CNAPmin(3)
    self.layers[2].elements.append(CNAPmin(cnapmin2, cnapdi8))
    ### Adiciona as conexões das células da terceira camada
    # Adiciona as conexões da célula CNAPmin(3)
    self.layers[2].connections.append(
                            {
                                'mi1': {'layer': 1, 'element': 0},
                                'mi2': {'layer': 1, 'element': 1}
                            }
                        )
    ### Adiciona os valores das saídas das células da terceira camada
    # Adiciona o valor da célula CNAPmin(3)
    self.layers[2].values.append(self.layers[2].elements[0].results())

    ''' Adiciona a QUARTA CAMADA da rede (ÚLTIMA CAMADA DA REDE)'''
    self.layers.append(Layer())
    ### Adiciona as células CNAPdi(4) e CNAPar(6) à lista de elementos da quarta camada
    # Obtém as saídas da célula CNAPmin(3) da terceira camada
    cnapmin3 = self.layers[2].values[0]
    # Adiciona a célula CNAPmin(4)
    self.layers[3].elements.append(CNAPdi(self.di1, self.di2, cnapmin3))
    # Adiciona a célula CNAPar(6)
    self.layers[3].elements.append(CNAPar(self.ar1, self.ar2))
    ### Adiciona as conexões das células da quarta camada
    # Adiciona as conexões da célula CNAPdi(4)
    self.layers[3].connections.append(
                            {
                                'ftct': {'layer': 2, 'element': 0}
                            }
                        )
    # Adiciona as conexões da célula CNAPar(6)
    self.layers[3].connections.append({})
    ### Adiciona os valores das saídas das células da quarta camada
    # Adiciona o valor da célula CNAPdi(4)
    self.layers[3].values.append(self.layers[3].elements[0].results())
    # Obtém as saídas da célula CNAPar(6) da quarta camada
    cnapar6 = self.layers[3].elements[1].results()
    # Adiciona o valor da célula CNAPar(6)
    self.layers[3].values.append(cnapar6['s1'])


def results(self):
    ''' Retorna as saídas AR e DI da rede '''
    self.create_network()
```

```
    self.ar = self.layers[3].values[1]
    self.di = self.layers[3].values[0]
    results = {
        'ar': self.ar,
        'di': self.di
    }
    return results
```

class UCPC():

```
    ''' Unidade Neural Artificial Paraconsistente
        Unidade das Células da Primeira Camada - UCPC
    '''
    def __init__(self, mi1, mi2, ftct=0.1):
        ''' Construtor da classe
            Atribui os parâmetros passados ao objeto
            :mi1: Grau de evidência 1 (0 <= mi1 <= 1)
            :mi2: Grau de evidência 2 (0 <= mi2 <= 1)
            :ftct: Define o valor default do fator de tolerância à contradição das células CNAPdi
        '''
        self.mi1 = mi1
        self.mi2 = mi2
        self.ftct = ftct
        self.layers = []

    def create_network(self):
        ''' Adiciona uma CNAPar e uma CNAPdi à primeira e única camada da rede
        '''
        # Adiciona a nova camada à lista de camadas
        self.layers.append(Layer())
        # Adiciona as células CNAPar e CNAPdi à lista de elementos da camada
        self.layers[0].elements.append(CNAPar(self.mi1, self.mi2))
        self.layers[0].elements.append(CNAPdi(self.mi1, self.mi2, self.ftct))
        # Adiciona um dicionário vazio para cada célula na lista de conexões
        # da camada (Recebe um dicionário vazio porque é a primeira camada)
        self.layers[0].connections.append({})
        self.layers[0].connections.append({})
        # Obtém os valores das células (CNAPar e CNAPdi)
        ar = self.layers[0].elements[0].results()['s1']
        di = self.layers[0].elements[1].results()
        self.layers[0].values.append(ar)
        self.layers[0].values.append(di)

    def results(self):
        ''' Retorna as saídas AR e DI da rede '''
        self.create_network()
        self.ar = self.layers[0].values[0]
        self.di = self.layers[0].values[1]
        results = {
            'ar': self.ar,
            'di': self.di
        }
        return results
```

class ULVH():

```
    ''' Unidade Neural Artificial Paraconsistente
        Unidade das Linhas Vertical e Horizontal - ULVH
```

```
'''
def __init__(self, list_ge, direcao, ftct=0.1):
    ''' Construtor da classe
        Atribui os parâmetros passados ao objeto
        :list_ge: Lista de graus de evidência referente à um lado da imagem do caracter
        :ftct: Define o valor default do fator de tolerância à contradição das células CNAPdi das unidades UCPC
    '''
    # Enquanto a quantidade de elementos de list_ge for maior do que 2 e não for divisível por 4,
    # adicione graus de evidência igual a 0 na lista até que a quantidade seja divisível por 4
    while len(list_ge) > 2 and len(list_ge) % 4 != 0:
        list_ge.append(0.0)
    self.list_ge = list_ge
    self.direcao = direcao
    self.ftct = ftct
    self.layers = []
    self.line_segments = []

def create_network(self):
    ''' Adiciona unidades UCPC à primeira camada da rede
        Adiciona unidades UCCF à demais camadas da rede
    '''
    ''' Adiciona a PRIMEIRA CAMADA da rede '''
    self.layers.append(Layer())
    # Para cada par de graus de evidência de self.list_ge (ordem sequencial)
    # adiciona uma UCPC à lista de elementos da primeira camada da rede
    for i in range(0, len(self.list_ge), 2):
        mi1 = self.list_ge[i]
        mi2 = self.list_ge[i + 1]
        ftct = self.ftct
        self.layers[0].elements.append(UCPC(mi1, mi2, ftct))
        # Adiciona as conexões na lista de conexões da primeira camada
        self.layers[0].connections.append({})
        # Adiciona os valores das saídas da UCPC na lista de valores da primeira camada
        self.layers[0].values.append(self.layers[0].elements[-1].results())

    ''' Adiciona a SEGUNDA CAMADA da rede '''
    # Se a quantidade de elementos da camada anterior for maior do que 1
    if len(self.layers[-1].elements) > 1:
        # Adiciona uma nova camada
        self.layers.append(Layer())
        # Popula a camada
        self.last_layers(2)
    ''' Adiciona as DEMAIS CAMADAS da rede '''
    # Enquanto a quantidade de elementos da camada anterior for maior do que 1
    while len(self.layers[-1].elements) > 1:
        # Adiciona uma nova camada
        self.layers.append(Layer())
        # Popula a camada
        self.last_layers(1)

def last_layers(self, step):
    ''' Popula da segunda à última camada da rede
        :step:  passo para definir a sequência do looping. O valor 1
                indica que a sequência deverá ser
                [(0,1), (1,2), (2,3), ..., (n-2, n-1), (n-1, n)]
                e o valor 2 indica que a sequência deverá ser
                [(0,1), (2,3), (4,5), ..., (n-3, n-2), (n-1, n)]
    '''
    # Quantidade de elementos da camada anterior
    qtd_elements = len(self.layers[-2].elements)
    # Índice da última camada
    last_layer = len(self.layers) - 2
    # Se a quantidade de elementos da camada anterior for maior do que 1
```

```
        if qtd_elements > 1:
            # Adiciona uma UCCF para cada par de elementos da camada anterior conforme passo definido
            for i in range(0, qtd_elements - 1, step):
                # Seleciona um par de elementos da camada anterior e obtém suas saídas
                entrada1 = self.layers[-2].values[i]
                entrada2 = self.layers[-2].values[i + 1]
                # Adiciona uma UCCF na lista de elementos da camada e atribui os valores de entrada
                self.layers[-1].elements.append(UCCF(entrada1, entrada2))
                # Adiciona as conexões da UCCF na lista de conexões da camada
                self.layers[-1].connections.append(
                                        {
                                            'mi1': {'layer': last_layer, 'element': i},
                                            'mi2': {'layer': last_layer, 'element': i + 1}
                                        }
                                    )
                # Calcula e adiciona os valores de saída da UCCF na lista de valores da camada
                self.layers[-1].values.append(self.layers[-1].elements[-1].results())

    def get_line_segments(self):
        ''' Identifica os segmentos de reta encontrados na lista de graus de evidência informada
            Cada segmento de reta encontrado deve conter um dicionário com:
                'direction': 0 ou 1
                        1) Se lado = esquerda ou topo então
                                direction = 1
                            Senão: (direita ou base)
                                direction = 0
                'size': Tamanho mínimo estimado do segmento de reta
                        1)  Calcular a quantidade mínima de pixels do segmento de reta multiplicando
                            o índice da camada da rede por 4, conforme algoritmo abaixo.
                            Se i > 0 então
                                size = i * 4
                            Senão
                                size = 2
                        2)  Calcular o grau de evidência proporcional ao tamanho do segmento de reta em pixels
                            size = size / quantidade de elementos na lista de graus de evidência (self.list_ge)
                'position': posição do segmento de reta vertical na imagem (topo, centro, base) ou
                            posição do segmento de reta horizontal na imagem (esquerda, centro, direita)
                        1) Calcular o grau de evidência referente a posição da seguinte forma:
                            (índice do elemento na lista de elementos da camada + 1) / quantidade de elementos da camada
                'local': Coluna ou linha onde o segmento de reta se encontra
                         ex. número da coluna onde o segmento de reta vertical se encontra
                                                (grau de evidência equivalente)  número da linha onde o segmento de reta horizontal
                                                 se encontra (grau de evidência equivalente)
                        1) Atribuir o valor AR do elemento
                            local = AR
            Para identificar os segmentos de retas, seguiremos os seguintes critérios:
                1) Se na última camada da rede o valor DI = 1 então:
                    1.1) Existe apenas um segmento de reta com os seguintes dados
                        1.1.1) Adicionar a lista de segmentos de reta da rede apenas um segmento de reta com um dicionário para representá-
lo
                            ex: {
                                    'direction': self.direcao
                                    'size': índice da última camada na lista de camadas * 4
                                    'position': 1
                                    'local': valor AR
                                }
                2) Caso contrário (na última camada da rede valor DI = 0)
                    2.1) Percorrer a rede a partir da penúltima camada até a primeira a procura de:
                        2.1.1) Camadas com pelo menos um elemento com valor DI = 0
                            O elemento só representa um segmento de reta se:
                                valor DI = 1 e possuir ambos os vizinhos com DI = 0 ou
                                valor DI = 1, possuir 1 vizinho com DI = 0 e é o primeiro ou o último elemento da lista
                                    A identificação de segmentos de reta em cada camada é dividida em três partes:
```

```
                1) Verifica se a primeira unidade representa um segmento de reta
                   camada[i][0]    - saída DI igual a 1 e
                   camada[i][1]    - saída DI igual a 0
                2) Verifica se a última unidade representa um segmento de reta
                   camada[i][m]     - saída DI igual a 1 e
                   camada[i][m-1] - saída DI igual a 0
                3) Verifica se as unidades do meio representam um segmento de reta
                   camada[i][m]     - saída DI igual a 1 e
                   camada[i][m-1]   - saída DI igual a 0 e
                   camada[i][m]     - saída DI igual a 0
'''
# Obtém a quantidade de camadas da rede
qtd_layers = len(self.layers)
# Verifica se a última camada da rede possui um elemento que representa um segmento de reta
if self.layers[-1].values[0].get('di') == 1:
    self.line_segments.append(
        {
            'direction': self.direcao,
            'size': ((qtd_layers - 1) * 4.) / len(self.list_ge),
            'position': 1,
            'local': self.layers[-1].values[0]['ar']
        }
    )
else:
    # Percorre a lista de camadas da penúltima até a primeira camada
    for i in range(qtd_layers-2, -1, -1):
        # Define o índice multiplicador e multiplica pelo índice para compor o numerador da divição para calcular
        # a quantidade mínima de pixels de um segmento de reta
        if i > 0:
            numerator = i * 4.
        else:
            numerator = 2.
        qtd_elements = len(self.layers[i].values)
        if self.looking_for_di_0(self.layers[i].values):
            # Verifica se o primeiro elemento da camada representa um segmento de reta
            if  self.layers[i].values[0].get('di') == 1 and \
                self.layers[i].values[1].get('di') == 0:
                self.line_segments.append(
                    {
                        'direction': self.direcao,
                        'size': numerator / len(self.list_ge),
                        'position': 1. / qtd_elements,
                        'local': self.layers[i].values[0]['ar']
                    }
                )
            # Verifica se o último elemento da camada representa um segmento de reta
            if  self.layers[i].values[-1].get('di') == 1 and \
                self.layers[i].values[-2].get('di') == 0:
                self.line_segments.append(
                    {
                        'direction': self.direcao,
                        'size': numerator / len(self.list_ge),
                        'position': 1,
                        'local': self.layers[i].values[-1]['ar']
                    }
                )
            # Verifica se existe algum elemento da segunda e penúltima posição da camada que representa um segmento de reta
            for j in range(1, len(self.layers[i].values)-1):
                if  self.layers[i].values[j].get('di') == 1 and \
                    self.layers[i].values[j + 1].get('di') == 0 and \
                    self.layers[i].values[j - 1].get('di') == 0:
                    self.line_segments.append(
                        {
```

```
                'direction': self.direcao,
                'size': numerator / len(self.list_ge),
                'position': (j + 1.) / qtd_elements,
                'local': self.layers[i].values[j]['ar']
            }
        )


def looking_for_di_0(self, list_values):
    ''' Verifica se a camada possui algum elemento com valor DI = 0
        :list_values: lista dos valores de saída dos elementos de uma determinada camada da rede
        Se um dos valores possuir DI = 0 então retorna 1, indicando que existe um
            elemento representando um segmento de reta
        Caso contrário retorna 0
    '''
    for value in list_values:
        if value.get('di') == 0:
            return 1
    return 0
def get_ge_line_segments(self):
    ''' Gera uma lista com os graus de evidência dos segmentos de reta identificados
    '''
    # Cria uma lista dos graus de evidência que representam os segmentos de reta identificados
    ge_line_segments = []
    # Para cada segmento de reta adicione na lista ge_line_segments os graus de evidência do mesmo
    for line_segment in self.line_segments:
        for param in ('direction', 'size', 'position', 'local'):
            ge_line_segments.append(line_segment[param])
    return ge_line_segments

def results(self):
    ''' Retorna as saídas AR e DI da rede '''
    self.create_network()
    self.get_line_segments()
    self.ge_line_segments = self.get_ge_line_segments()
    results = {
        'segments': self.line_segments,
        'ge_segments': self.ge_line_segments
    }
    return results
```

class URPcmc7():

```
''' Unidade Neural Artificial Paraconsistente
    Unidade de Reconhecimento Paraconsistente CMC7 - URPcmc7
'''
def __init__(self, ge_default, ge_caractere, ftct=0.1):
    ''' Construtor da classe
        Atribui os parametros passados ao objeto
        :ge_default: Graus de evidencia (0 <= ftct <= 1) obtidos no histograma da imagem padrao
        :ge_caractere: Graus de evidencia (0 <= ftct <= 1) obtidos no histograma da imagem
                        apresentada ao sistema
        :ftct: Define o valor default do fator de tolerância à contradição das células CNAPdi
                        da primeira camada da rede
    '''
    self.ge_default = ge_default
    self.ge_caractere = ge_caractere
    self.ftct = ftct
    self.layers = []
    # Ambas as listas de graus de evidência devem conter a mesma quantidade de elementos
```

```
        # Se a quantidade de elementos da lista ge_default for maior do que da lista ge_caractere, adiciona ge = 0.5 na lista ge_caractere
        while len(self.ge_default) > len(self.ge_caractere):
            self.ge_caractere.append(0)
        # Se a quantidade de elementos da lista ge_caractere for maior do que da lista ge_default, adiciona ge = 0.5 na lista ge_default
        while len(self.ge_caractere) > len(self.ge_default):
            self.ge_default.append(0)
        # A quantidade de elementos de ambas as listas deve ser igual a 2 ou divisível por 4
        # Se a quantidade de elementos de ambas as listas for maior do que 2 e não divisível por 4, adiciona ge = 0.5 em ambas as listas
        if len(self.ge_default) > 2:
            while (len(self.ge_default) % 4) != 0:
                self.ge_default.append(0)
                self.ge_caractere.append(0)

    def create_network(self):
        ''' Adiciona células CNAPdi à primeira camada da rede
            Adiciona células CNAPa da segunda a penúltima camada da rede
            Adiciona uma célula CNAPpa à última camada da rede
        '''
        ''' Adiciona a PRIMEIRA CAMADA da rede '''
        self.layers.append(Layer())
        # Obtém a quantidade de elementos de lista ge_default (ambas as listas possuem a mesma quantidade de elementos)
        qtd_elements = len(self.ge_default)
        # Para cada par de graus de evidência constituídos a partir das listas ge e ge_padrao
        # (em ordem sequencial) adiciona uma CNAPdi à lista de elementos da primeira camada da rede
        # (em ordem sequencial) adiciona um NAP à lista de elementos da primeira camada da rede
        for i in range(qtd_elements):
            mi = max([self.ge_caractere[i], self.ge_default[i]])
            l = 1 - min([self.ge_caractere[i], self.ge_default[i]])
            self.layers[0].elements.append(NAP(mi, l))
            # Adiciona as conexões na lista de conexões da primeira camada
            self.layers[0].connections.append({})
            # Adiciona os valores das saídas do NAP na lista de valores da primeira camada
            self.layers[0].values.append(self.layers[0].elements[-1].results()['s2'])
        ''' Adiciona da SEGUNDA a PENÚLTIMA CAMADA da rede '''
        # Define o valor default do fator de tolerância à contradição
        # e do fator de tolerância à certeza das células CNAPa das demais camadas da rede
        ftct = 1
        ftc = 0
        # Enquanto a quantidade de elementos da camada anterior for maior do que 1
        while len(self.layers[-1].elements) > 1:
            # Obtém a quantidade de elementos da última camada
            qtd_elements = len(self.layers[-1].elements)
            # Adiciona uma nova camada na lista de camadas da rede
            self.layers.append(Layer())
            # Para cada par de elementos da camada anterior adicione uma CNAPa à nova camada
            for i in range(qtd_elements - 1):
                mi1 = self.layers[-2].values[i]
                mi2 = self.layers[-2].values[i + 1]
                self.layers[-1].elements.append(CNAPa(mi1, mi2, ftct, ftc))
                # Adiciona as conexões na lista de conexões da nova camada
                self.layers[-1].connections.append(
                                        {
                                            'mi1': {'layer': len(self.layers) - 2, 'element': 0},
                                            'mi2': {'layer': len(self.layers) - 2, 'element': 1},
                                        }
                                    )
                # Adiciona os valores das saídas da CNAPa na lista de valores da camada
                self.layers[-1].values.append(self.layers[-1].elements[-1].results()['s1'])

        ''' Adiciona a ÚLTIMA CAMADA da rede '''
        # Adiciona a ÚLTIMA camada na lista de camadas da rede
        self.layers.append(Layer())
```

```
        # Adiciona uma CNAPpa na última camada da rede
        mi1 = self.layers[-2].values[0]
        ftc = 0
        self.layers[-1].elements.append(CNAPpa(mi1, ftc))
        # Adiciona as conexões na lista de conexões da última camada
        self.layers[-1].connections.append(
                                    {
                                        'mi1': {'layer': len(self.layers) - 2, 'element': 0},
                                    }
                                  )
        # Adiciona os valores das saídas da CNAPpa na lista de valores da camada
        self.layers[-1].values.append(self.layers[-1].elements[-1].results())

    def results(self):
        ''' Retorna as saída da CNAPpa da última camada da rede '''
        self.create_network()
        self.s1 = self.layers[-1].values[0]
        return self.s1
```

class CNAPa(CNAP):

```
    ''' Célula Neural Artificial Paraconsistente Analítica - CNAPa '''
    def __init__(self, mi1, mi2, ftct, ftc):
        ''' Construtor da classe
            Atribui os parâmetros passados ao objeto
            :mi1: Grau de evidência 1 (0 <= mi1 <= 1)
            :mi2: Grau de evidência 2 (0 <= mi2 <= 1)
            :ftct: Fator de tolerância à Contradição (0 <= ftct <= 1)
            :ftc: Fator de tolerância à Certeza (0 <= ftc <= 1)
        '''
        self.mi1 = mi1
        self.mi2 = mi2
        self.ftct = ftct
        self.ftc = ftc

    def results(self):
        ''' Retorna os sinais resultantes (S1 e S2) '''
        vscin = self.get_vscin(self.ftct)
        vicin = self.get_vicin(self.ftct)
        vscc = self.get_vscc(self.ftc)
        vicc = self.get_vicc(self.ftc)
        l = self.get_l(self.mi2)
        mctr = self.get_mctr(self.mi1, l)
        fe = self.get_fe(mctr)
        me = self.get_me(self.mi1, l)
        results = {'s1': 0.5, 's2': fe}
        if vscin > mctr > vicin and (vscc <= me or me <= vicc):
            results['s1'] = me
        self.s1 = results['s1']
        self.s2 = results['s2']
        return results
```

class CNAPar(CNAP):

```
    ''' Célula Neural Artificial Paraconsistente Analítica Real - CNAPar '''
    def __init__(self, mi1, mi2):
        ''' Construtor da classe
            Atribui os parâmetros passados ao objeto
```

```
        :mi1: Grau de evidência 1 (0 <= mi1 <= 1)
        :mi2: Grau de evidência 2 (0 <= mi2 <= 1)
    '''
    self.mi1 = mi1
    self.mi2 = mi2

def get_mer(self, me, d):
    ''' Calcula o grau de evidência resultante real '''
    if me > 0.5:    mer = 1 - (d / 2)
    elif me < 0.5:  mer = d / 2
    else:           mer = me
    return mer

def results(self):
    ''' Retorna os sinais resultantes (S1 e S2) '''
    l = self.get_l(self.mi2)
    mctr = self.get_mctr(self.mi1, l)
    fe = self.get_fe(mctr)
    me = self.get_me(self.mi1, l)
    d = self.get_d(self.mi1, l)
    mer = self.get_mer(me, d)
    results = {'s1': mer, 's2': fe}
    if fe <= 0.25:
        results['s1'] = 0.5
    self.s1 = results['s1']
    self.s2 = results['s2']
    return results
```

class CNAPdi(CNAP):

```
''' Célula Neural Artificial Paraconsistente de Detecção de Igualdade - CNAPdi'''
def __init__(self, mi1, mi2, ftct=0.1):
    ''' Construtor da classe
        Atribui os parâmetros passados ao objeto
        :mi1: Grau de evidência 1 (0 <= mi1 <= 1)
        :mi2: Grau de evidência 2 (0 <= mi2 <= 1)
        :ftct: Fator de tolerância à Contradição (0 <= ftct <= 1) (default = 0)
    '''
    self.mi1 = mi1
    self.mi2 = mi2
    self.ftct = ftct

def results(self):
    ''' Retorna o sinal resultante (S1) '''
    vscin = self.get_vscin(self.ftct)
    vicin = self.get_vicin(self.ftct)
    l = self.get_l(self.mi2)
    mctr = self.get_mctr(self.mi1, l)
    s1 = 0
    if vscin > mctr > vicin:
        s1 = 1
    return s1
```

class CNAPmax(CNAP):

```
''' Célula Neural Artificial Paraconsistente de Conexão Lógica Simples de Maximização - CNAPmax '''
def __init__(self, mi1, mi2):
    ''' Construtor da classe
```

```
        Atribui os parâmetros passados ao objeto
        :mi1: Grau de evidência 1 (0 <= mi1 <= 1)
        :mi2: Grau de evidência 2 (0 <= mi2 <= 1)
    '''
    self.mi1 = mi1
    self.mi2 = mi2

def results(self):
    ''' Retorna o sinal resultante (S1) '''
    l = self.mi2
    me = self.get_me(self.mi1, l)
    s1 = self.mi2
    if me >= 0.5:
        s1 = self.mi1
    return s1
```

class CNAPmin(CNAP):

```
''' Célula Neural Artificial Paraconsistente de Conexão Lógica Simples de Minimização - CNAPmin '''
def __init__(self, mi1, mi2):
    ''' Construtor da classe
        Atribui os parâmetros passados ao objeto
        :mi1: Grau de evidência 1 (0 <= mi1 <= 1)
        :mi2: Grau de evidência 2 (0 <= mi2 <= 1)
    '''
    self.mi1 = mi1
    self.mi2 = mi2

def results(self):
    ''' Retorna o sinal resultante (S1) '''
    l = self.mi2
    me = self.get_me(self.mi1, l)
    s1 = self.mi1
    if me >= 0.5:
        s1 = self.mi2
    return s1
```

class CNAPpa(CNAP):

```
''' Célula Neural Artificial Paraconsistente de Passagem '''
def __init__(self, mi1, ftc):
    ''' Construtor da classe
        Atribui os parâmetros passados ao objeto
        :mi1: Grau de evidência 1 (0 <= mi1 <= 1)
        :ftc: Fator de tolerância à Certeza (0 <= ftc <= 1)
    '''
    self.mi1 = mi1
    self.ftc = ftc

def results(self):
    ''' Retorna o sinal resultante (S1) '''
    vscc = self.get_vscin(self.ftc)
    vicc = self.get_vicin(self.ftc)
    l = self.get_l(self.mi1)
    me = self.get_me(self.mi1, l)
    s1 = 0.5
    if vscc <= me or me <= vicc:
        s1 = self.mi1
    return s1
```

```
class NAP():
    ''' Nó de Análise Paraconsistente - NAP '''
    def __init__(self, mi, l):
        ''' Construtor da classe
            Atribui os parâmetros passados ao objeto
            :mi: Grau de evidência favorável mi (0 <= mi <= 1)
            :l: Grau de evidência desfavorável lambda (0 <= l <= 1)
        '''
        self.mi = mi
        self.l = l

    def get_mctr(self, mi, l):
        ''' Calcula o grau de contradição normalizado
            mctr = (mi + lambda) / 2
        '''
        return (mi + l) / 2.

    def get_fe(self, mctr):
        ''' Calcula o intervalo de evidência resultante
            fe = 1 - |2mctr - 1|
        '''
        return (1 - math.fabs((2 * mctr) - 1))

    def get_gc(self, mi, l):
        ''' Calcula o Grau de Certeza
            gc = mi - l
        '''
        return mi - l

    def get_gin(self, mi, l):
        ''' Calcula o Grau de Contradição
            gin = (mi + l) - l
        '''
        return (mi + l) - 1

    def get_d(self, gc, gin):
        ''' Calcula a distância D
            D = raiz_quadrada((1 - |gc|)**2 + (gin**2)
        '''
        return math.sqrt((1 - math.fabs(gc))**2 + (gin)**2)

    def get_gcr(self, gc, d):
        ''' Determina o Grau de Certeza real
            Se gc >= 0 então faça:
                gcr <-- (1 - d)
            Senão:
                gcr <-- (d - 1)
        '''
        if gc >= 0:
            return 1 - d
        else:
            return d - 1

    def get_mer(self, gcr):
        ''' Calcula o Grau de Evidência resultante real
            mer = (gcr + 1) / 2
        '''
        return (gcr + 1) / 2.

    def results(self):
        ''' Retorna as saídas no NAP '''
        # Calcula o grau de contradição normalizado
        mctr = self.get_mctr(self.mi, self.l)
```

```
        # Calcula o intervalo de evidência resultante
        fe = self.get_fe(mctr)
        # Calcula o grau de certeza
        gc = self.get_gc(self.mi, self.l)
        # Calcula o grau de contradição
        gin = self.get_gin(self.mi, self.l)
        # Calcula a distância D
        d = self.get_d(gc, gin)
        # Determina a saída s2
        self.s2 = fe
        if fe <= 0.25 or d > 1:
            #Determina as saídas do NAP como indefinição
            self.s1 = 0.5
        else:
            # Determina o Grau de Certeza Real
            gcr = self.get_gcr(gc, d)
            # Calcula o grau de evidência resultante real
            mer = self.get_mer(gcr)
            # Determina as saídas do NAP
            self.s1 = mer

        results = {
            's1': self.s1,
            's2': self.s2
        }
        return results
```

Diretório utils

manage_image_mapping.py

```
def get_sample_mapping(caracteristics, images):
    ''' Faz o mapeamento de um grupo de images
        :caracteristics: características que deverão ser extraídas
        :images: dicionário com as caracteristicas das imagens

        Retorna o dicionários de características das imagens atualizado
    '''
    try:
        print u'================= Mapeando as images em graus de evidência ==================='
        for image in images:
            # Converte a matrix binária de lista para array
            matrix = numpy.array(images[image]['binary_matrix'])
            # Mapeamento da imagem em graus de evidência
            images[image]['evidence_degrees'] = get_image_mapping(caracteristics, matrix)
        return images
    except:
        return HTTPNotFound(u'Erro ao tentar obter o mapeamento das características do dicionário
                                        "images" - utils/manage_image_mapping/get_sample_mapping')

def get_image_mapping(caracteristics, matrix):
    ''' Faz o mapeamento da imagem conforme características selecionadas
        :caracteristics: características selecionadas
        :matrix: matriz binária da imagem
        Retorna um dicionário com as evidências mapeadas
    '''
    try:
        evidence_degrees = {}
        evidence_degrees['histogram'] = {}
```

```
        if  'vertical_line_segments' in caracteristics or \
            'horizontal_line_segments' in caracteristics or \
            'external_histogram' in caracteristics or \
            'external_top_border_histogram' in caracteristics:
            # Mapeia as bordas externas da imagem
            evidence_degrees['external_border'] = get_external_border_evidences_from_binary_matrix(matrix)
        if 'all_pixels_histogram' in caracteristics:
            # Mapeia todos os pixels da imagem conforme mapeamento das bordas externas
            #(esquerda e direita) ordenado por coluna
            evidence_degrees['histogram']['all_pixels_histogram'] =
                                        get_evidences_from_all_pixels_from_binary_matrix(matrix)
        if 'internal_histogram' in caracteristics:
            # Mapeia as bordas internas da imagem
            evidence_degrees['internal_border'] = get_internal_border_evidences_from_binary_matrix(matrix)
        if 'diagonal_right_letf_histogram' in caracteristics:
            # Mapeia as bordas das diagonais (direita - esquerda) - superior e inferior
            evidence_degrees['diagonal_right_left_border'] =
                                         get_right_left_diagonal_evidence_degrees_from_binary_matrix(matrix)
        if 'diagonal_letf_right_histogram' in caracteristics:
            # Mapeia as bordas das diagonais (esquerda - direita) - superior e inferior
            evidence_degrees['diagonal_left_right_border'] =
                                        get_left_right_diagonal_evidence_degrees_from_binary_matrix(matrix)
        return evidence_degrees
    except:
        return HTTPNotFound(u'Erro ao tentar obter o mapeamento das características da
                                        imagem - utils/manage_image_mapping/get_image_mapping')

# Extrai os graus de evidência do contorno externo da imagem e retorna um dicionário com os valores
def get_external_border_evidences_from_binary_matrix(matrix):
    ''' Extrai os graus de evidência da borda externa da matriz (contorno externo)
        :matrix: matriz binária da imagem
        Retorna um dicionário com os graus de evidência extraídos.
            external_border = {
                'left':     [evidence, ..., evidence],
                'right':    [evidence, ..., evidence],
                'top':      [evidence, ..., evidence],
                'bottom':   [evidence, ..., evidence],
    '''
    try:
        print '=============== Extrai graus de evidência da borda externa ================'
        # Pega a quantidade de linhas e colunas
        qtd_line = len(matrix)
        qtd_column = len(matrix[0])
        # Pega o índice da última linha e da última coluna da matrix
        index_last_line = qtd_line-1
        index_last_column = qtd_column-1
        external_border = {
            'left':     [],
            'right':    [],
            'top':      [],
            'bottom':   []
        }
        # Percorre cada linha da imagem
        for linha in range(qtd_line):
            evidencia = 0
            direcao = 'left'
            # Para o tipo 'externo' considera-se o total de colunas da matriz
            # O range varia em ordem crescente conforme quantidade de colunas da matriz
            list_range_left = range(qtd_column)
            # O range varia em ordem decrescente conforme quantidade de colunas da matriz
            list_range_right = range(qtd_column-1, -1, -1)
            # Percorre cada coluna da linha em ordem crescente (grau de evidência da Esquerda)
            for coluna in list_range_left:
```

```
        # Se o pixel for preto e possuir pelo menos um pixel vizinho preto, calcula o grau de evidência
        if  matrix[linha,coluna] == True and \
            verify_neighbor_from_binary_matrix(matrix, linha, coluna, index_last_line, index_last_column):
            # Calcula o grau de evidência - Número da coluna = coluna + 1
            evidencia = get_evidence_from_image(coluna + 1, index_last_column + 1)
            # Adiciona o grau de evidência a lista
            external_border[direcao].append(evidencia)
            # O looping é finalizado quando encontra-se o primeiro pixel preto
            break
    # Se foi encontrado um pixel preto no "for" acima, verifica-se a borda do lado direito
    if evidencia > 0:
        direcao = 'right'
        # Percorre cada coluna da linha em ordem decrescente (grau de evidência da Direita)
        for coluna in list_range_right:
            # Se o pixel for preto e possuir pelo menos um pixel preto vizinho, calcula o grau de evidência
            if  matrix[linha,coluna] == True and \
                verify_neighbor_from_binary_matrix(matrix, linha, coluna, index_last_line, index_last_column):
                # Calcula o grau de evidência - Número da coluna = coluna + 1
                evidencia = get_evidence_from_image(coluna + 1, index_last_column + 1)
                break
        # Adiciona o grau de evidência à lista
        external_border[direcao].append(evidencia)
    else:
        # Definição do grau de evidência do lado esquerdo
        external_border[direcao].append(evidencia)
        # Definição do grau de evidência do lado direito
        direcao = 'right'
        # Adiciona o grau de evidência à lista
        external_border[direcao].append(evidencia)
# Percorre cada coluna da matriz
for coluna in range(qtd_column):
    evidencia = 0
    direcao = 'top'
    # Define os range para percorrer o topo e a base da imagem
    # O range varia em ordem crescente conforme quantidade de linhas
    list_range_top = range(qtd_line)
    # O range varia em ordem decrescente conforme quantidade de linhas
    list_range_bottom = range(qtd_line-1, -1, -1)
    # Percorre cada linha em ordem crescente (grau de evidência do Topo)
    for linha in list_range_top:
        # Se o pixel for preto e possuir pelo menos um pexil preto vizinho, calcula o grau de evidência
        if  matrix[linha,coluna] == True and \
            verify_neighbor_from_binary_matrix(matrix, linha, coluna, index_last_line, index_last_column):
            # Calcula o grau de evidência - Número da linha = linha + 1
            evidencia = get_evidence_from_image(linha + 1, qtd_line)
            # Adiciona o grau de evidência à lista
            external_border[direcao].append(evidencia)
            break
    # Se for a evidência for maior do que 0
    if evidencia > 0:
        direcao = 'bottom'
        # Percorre cada linha da mesma coluna em ordem decrescente (grau de evidência da Base)
        for linha in list_range_bottom:
            # Se o pixel for preto e possuir pelo menos um píxel preto vizinho, calcula o grau de evidência
            if  matrix[linha,coluna] == True and \
                verify_neighbor_from_binary_matrix(matrix, linha, coluna, index_last_line, index_last_column):
                # Calcula o grau de evidência - Número da coluna = coluna + 1
                evidencia = get_evidence_from_image(linha + 1, qtd_line)
                break
        # Adiciona o grau de evidência à lista
        external_border[direcao].append(evidencia)
    # Se evidência = 0
    else:
```

```
            # Definição do grau de evidência do Topo
            external_border[direcao].append(evidencia)
            # Definição do grau de evidência da Base
            direcao = 'bottom'
            external_border[direcao].append(evidencia)
        return external_border
    except:
        print u'Erro ao tentar mapear o contorno externo da
                                imagem - utils/manage_image_mapping/get_external_border_evidences_from_binary_matrix'
        return HTTPNotFound(u'Erro ao tentar mapear o contorno externo da
                                imagem - utils/manage_image_mapping/get_external_border_evidences_from_binary_matrix')

# Extrai os graus de evidência do contorno interno da imagem
def get_internal_border_evidences_from_binary_matrix(matrix):
    ''' Extrai os graus de evidência da borda interna da imagem na matriz (contorno interno)
        :matrix: matriz binária da imagem
        Retorna um dicionário com os graus de evidência extraídos.
            internal_border = {
                'left':     [evidence, ..., evidence],
                'right':    [evidence, ..., evidence],
                'top':      [evidence, ..., evidence],
                'bottom':   [evidence, ..., evidence],
    '''
    try:
        print '=============== Extrai graus de evidência da borda interna ================'
        # Pega a quantidade de linhas e colunas
        qtd_line = len(matrix)
        qtd_column = len(matrix[0])
        # Pega o índice da última linha e da última coluna da matrix
        index_last_line = qtd_line-1
        index_last_column = qtd_column-1
        internal_border = {
            'left':     [],
            'right':    [],
            'top':      [],
            'bottom':   []
        }
        # Percorre cada linha da imagem
        for linha in range(qtd_line):
            evidencia = 0
            direcao = 'left'
            # Para a extração de graus de evidência da borda interna o range varia a partir do centro da matriz
            # O range varia em ordem crescente a partir do meio da matriz (considera-se a quantidade de colunas da matriz)
            list_range_left = range((qtd_column / 2), qtd_column)
            # O range varia em ordem decrescente a partir do meio da matriz (considera-se a quantidade de colunas da matriz)
            list_range_right = range((qtd_column / 2)-1, -1, -1)

            # Percorre cada coluna na mesma linha
            for coluna in list_range_left:
                # Se o pixel for preto, calcula o grau de evidência
                if  matrix[linha,coluna] == True and \
                    verify_neighbor_from_binary_matrix(matrix, linha, coluna, index_last_line, index_last_column):
                    # Calcula o grau de evidência - Número da coluna = coluna + 1
                    evidencia = get_evidence_from_image(coluna + 1, index_last_column + 1)
                    # O looping é finalizado quando encontra-se o primeiro pixel preto
                    break
            # Adiciona o grau de evidência à lista (direção: left)
            internal_border[direcao].append(evidencia)
            # Defini direção
            direcao = 'right'
            evidencia = 0
            for coluna in list_range_right:
                # Se o pixel for preto e não possuir nenhum pixels vizinho preto, calcula o grau de evidência
```

```
            if  matrix[linha,coluna] == True and \
                verify_neighbor_from_binary_matrix(matrix, linha, coluna, index_last_line, index_last_column):
                # Calcula o grau de evidência - Número da coluna = coluna + 1
                evidencia = get_evidence_from_image(coluna + 1, index_last_column + 1)
                break
        # Adiciona o grau de evidência à lista (direção: right)
        internal_border[direcao].append(evidencia)
    # Percorre cada coluna
    for coluna in range(qtd_column):
        evidencia = 0
        direcao = 'top'
        # Define o range para percorrer a matriz
        list_range_top = range((qtd_line / 2), qtd_line)
        list_range_bottom = range((qtd_line / 2)-1, -1, -1)
        #Para cada linha
        for linha in list_range_top:
            # Se o pixel for preto e não possui um pixels preto vizinho, calcula o grau de evidência
            if  matrix[linha,coluna] == True and \
                verify_neighbor_from_binary_matrix(matrix, linha, coluna, index_last_line, index_last_column):
                # Calcula o grau de evidência - Número da linha = linha + 1
                evidencia = get_evidence_from_image(linha + 1, qtd_line)
                break
        # Adiciona o grau de evidência à lista
        internal_border[direcao].append(evidencia)
        # Define a direção
        direcao = 'bottom'
        evidencia = 0
        for linha in list_range_bottom:
            # Se o pixel for preto e não possuir um pixel preto vizinho, calcula o grau de evidência
            if  matrix[linha,coluna] == True and \
                verify_neighbor_from_binary_matrix(matrix, linha, coluna, index_last_line, index_last_column):
                # Calcula o grau de evidência - Número da coluna = coluna + 1
                evidencia = get_evidence_from_image(linha + 1, qtd_line)
                break
        # Adiciona o grau de evidência à lista
        internal_border[direcao].append(evidencia)

    return internal_border
except:
    print u'Erro ao tentar mapear o contorno interno da
                        imagem - utils/manage_image_mapping/get_internal_border_evidences_from_binary_matrix'
    return HTTPNotFound(u'Erro ao tentar mapear o contorno interno da
                        imagem - utils/manage_image_mapping/get_internal_border_evidences_from_binary_matrix')

def verify_neighbor_from_binary_matrix(matrix, index_line, index_column, index_last_line, index_last_column):
    ''' Verifica se o pixel possui algum vizinho preto
        :matrix: matriz transposta da matriz binária da imagem (cada coluna da matriz corresponde a uma linha da imagem)
        :index_column: índice da coluna do pixel
        :index_line: índice da linha do pixel
        :index_last_line: índice da última linha na matriz
        :index_last_column: índice da última coluna na matriz
    '''
    try:
        # Verifica se existe vizinhos na mesma linha da matriz (vizinhos da direita e esquerda)
        if  (index_column < index_last_column and matrix[index_line, index_column+1] == True) or \
            (index_column > 0 and matrix[index_line, index_column-1] == True):
            return True
        # Verifica se existe vizinhos na próxima linha (vizinhos de baixo)
        if index_line < index_last_line:
            if  matrix[index_line+1, index_column] == True or \
                (index_column < index_last_column and matrix[index_line+1, index_column+1] == True) or \
                (index_column > 0 and matrix[index_line+1, index_column-1] == True):
                return True
```

```
        # Verifica se existe vizinhos na linha anterior (vizinhos de cima)
        if index_line > 0:
            if  matrix[index_line-1, index_column] == True or \
                (index_column < index_last_column and matrix[index_line-1, index_column+1] == True) or \
                (index_column > 0 and matrix[index_line-1, index_column-1] == True):
                return True
        # Retorna False se não houver vizinhos
        return False
    except:
        print u'Erro ao verificar os vizinhos dos pixels -
                                utils/manage_image_mapping/verify_neighbor_from_binary_matrix'
        return HTTPNotFound(u'Erro ao verificar os vizinhos dos pixels -
                                utils/manage_image_mapping/verify_neighbor_from_binary_matrix')

# Calcula o grau de evidência do pixel
def get_evidence_from_image(index, total):
    ''' Calcula o grau de evidência
        total = quantidade de linhas ou colunas
        indice = número da linha ou coluna onde o pixel preto foi encontrado
        Para calcular o grau de evidência realiza-se uma regra de três simples
            total   ------  1
            indice  ------  evidencia
            evidencia = indice / total
        PS. * Multiplica-se o 'indice' por '1.0' para realizar operação com números reais
            * Utiliza-se a função 'round(x, 2)' para arredondar o valor obtido para números reais de 2 casas decimais.
    '''
    try:
        index = 1. * index
        evidence = round((index / total), 2)
        return evidence
    except:
        print u'Erro ao calcular o grau de evidência - utils/manage_image_mapping/get_evidence_from_image'
        return HTTPNotFound(u'Erro ao calcular o grau de evidência -
                                utils/manage_image_mapping/get_evidence_from_image')

def get_evidences_from_all_pixels_from_binary_matrix(matrix):
    ''' Retorna uma lista de graus de evidência ordenados por colunas
        :matrix: matriz binária da imagem
    '''
    try:
        print u'=================== Mapeando todos os pixels da matriz binária ==============='
        # Pega a matrix transposta para pegar os pixels por ordem de columa
        matrix = matrix.T
        # Pega os índices da última linha e da última columa da matriz
        index_last_line = len(matrix)-1
        index_last_column = len(matrix[0])-1
        # Cria uma lista para guardar os graus de evidência
        list_evidences = []
        # Percorre cada linha da matriz (corresponde a cada coluna da matriz original) e calcula o grau de evidência de cada pixel
(linha-a-linha da matriz original)
        for index_line, line in enumerate(matrix):
            evidencia = 0
            # Percorre cada coluna da matriz (corresponde a cada linha da matriz original)
            for index_column, pixel in enumerate(line):
                # Se o pixel for preto e houver vizinhos calcula e atribuiu um grau de evidência ao pixel
                if  pixel == True and \
                    verify_neighbor_from_binary_matrix(matrix, index_line, index_column, index_last_line, index_last_column):
                    # Calcula grau de evidência - Número da coluna = index_column + 1
                    #evidencia = get_evidence_from_image(index_column+1, index_last_column+1)
                    evidencia = get_evidence_from_image(index_line+1, index_last_line+1)
                # Adiciona o grau de evidência na lista
                list_evidences.append(evidencia)
        return list_evidences
```

```
    except:
        print u'Erro ao mapear todos os pixels -
                                utils/manage_image_mapping/get_evidences_from_all_pixels_from_binary_matrix'
        return HTTPNotFound(u'Erro ao mapear todos os pixels -
                                 utils/manage_image_mapping/get_evidences_from_all_pixels_from_binary_matrix')

def get_right_left_diagonal_evidence_degrees_from_binary_matrix(matrix):
    ''' Retorna os graus de evidência da diagonal (direita/esquerda) considerando os cortes superior e inferior
        :matrix: matriz binária da imagem
        Retorna um dicionário com os graus de evidência mapeados
            diagonal_right_left_border = {
                'upper_top': [],
                'upper_botton': [],
                'lower_top': [],
                'lower_bottom': []
            }
    '''
    try:
        print u'============= Mapeando a diagonal (direita / esquerda) da matriz binária ============='
        # Pega o índice da última linha e da última coluna da matrix
        qtd_line = len(matrix)
        qtd_column = len(matrix[0])
        # Cria um dicionário vazio para os graus de evidência
        diagonal_right_left_border = {
            'upper_top': [],
            'upper_bottom': [],
            'lower_top': [],
            'lower_bottom': []
        }
        # Define o grau de evidência de um pixel
        peso = 1. / (qtd_column * qtd_line)
        # Define alguns parâmetros iniciais
        type = 'upper'    # define o tipo de grau de evidência extraído (diagonal_superior ou diagonal_inferior)
        # Percorre as linhas (quantidade máxima de linhas = quantidade de colunas)
        for linha_base in range(qtd_column):
            # Define a coluna onde inicia-se a verificação dos pixels
            coluna_base = qtd_column - (linha_base + 1)
            #### upper_top ###
            # Define o passo (topo = -1)
            step_column = -1
            step_line = -1
            # Percorre os pixels na diagonal em busca de pixels preto da segunda coluna base - up
            if coluna_base > 0:
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, (coluna_base - 1), linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "upper_top"
                diagonal_right_left_border['upper_top'].append(evidence_degree)
            else:
                # Percorre os pixels na diagonal em busca de pixels preto da primeira coluna base - up
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, coluna_base, linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "upper_top"
                diagonal_right_left_border['upper_top'].append(evidence_degree)
            #### upper_bottom ###
            # Define o passo (base = 1)
            step_column = 1
            step_line = 1
            # Percorre os pixels na diagonal em busca de pixels preto da segunda coluna base - down
            if coluna_base > 0:
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, (coluna_base - 1), linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "upper_bottom"
                diagonal_right_left_border['upper_bottom'].append(evidence_degree)
```

```
            else:
                # Percorre os pixels na diagonal em busca de pixels preto da primeira coluna base - down
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, coluna_base, linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "upper_bottom"
                diagonal_right_left_border['upper_bottom'].append(evidence_degree)
        # Define alguns parâmetros iniciais
        type = 'lower'     # define o tipo de grau de evidência extraído (diagonal_superior ou diagonal_inferior)
        # Percorre as linhas (quantidade máxima linhas = quantidade de colunas) - corte na diagonal a partir da última linha
        for linha_base in  range((qtd_line-qtd_column), qtd_line):
            # Define a coluna onde inicia-se a verificação dos pixels
            coluna_base = qtd_line - (linha_base + 1)
            #### lower_top ###
            # Define o passo (topo = -1)
            step_column = -1
            step_line = -1
            # Percorre os pixels na diagonal em busca de pixels preto da segunda coluna base - up
            if coluna_base > 0:
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, (coluna_base - 1), linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "lower_top"
                diagonal_right_left_border['lower_top'].append(evidence_degree)
            else:
                # Percorre os pixels na diagonal em busca de pixels preto da primeira coluna base - up
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, coluna_base, linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "lower_top"
                diagonal_right_left_border['lower_top'].append(evidence_degree)
            #### lower_bottom ###
            # Define o passo (base = 1)
            step_column = 1
            step_line = 1
            # Percorre os pixels na diagonal em busca de pixels preto da segunda coluna base - down
            if coluna_base > 0:
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, (coluna_base - 1), linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "lower_bottom"
                diagonal_right_left_border['lower_bottom'].append(evidence_degree)
            else:
                # Percorre os pixels na diagonal em busca de pixels preto da primeira coluna base - down
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, coluna_base, linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "lower_bottom"
                diagonal_right_left_border['lower_bottom'].append(evidence_degree)
        return diagonal_right_left_border
    except:
        print u'Erro ao mapear a diagonal (direita/esquerda) -
                 utils/manage_image_mapping/get_right_left_diagonal_evidence_degrees_from_binary_matrix'
        return HTTPNotFound(u'Erro ao mapear a diagonal (direita/esquerda) -
                 utils/manage_image_mapping/get_right_left_diagonal_evidence_degrees_from_binary_matrix')

def get_left_right_diagonal_evidence_degrees_from_binary_matrix(matrix):
    ''' Retorna os graus de evidência da diagonal (esquerda/direita) considerando os cortes superior e inferior
        :matrix: matriz binária da imagem

        Retorna um dicionário com os graus de evidência mapeados
            diagonal_left_right_border = {
                'upper_top': [],
                'upper_bottom': [],
                'lower_top': [],
                'lower_bottom': []
            }
    '''
```

```
try:
    print u'============= Mapeando a diagonal (esquerda / direita) da matriz binária ==============='
    # Pega o índice da última linha e da última coluna da matrix
    qtd_line = len(matrix)
    qtd_column = len(matrix[0])
    # Cria um dicionário vazio para os graus de evidência
    diagonal_left_right_border = {
        'upper_top': [],
        'upper_bottom': [],
        'lower_top': [],
        'lower_bottom': []
    }
    # Define o grau de evidência de um pixel
    peso = 1. / (qtd_column * qtd_line)
    # Define alguns parâmetros iniciais
    type = 'upper'            # define o tipo de grau de evidência extraído (diagonal_superior ou diagonal_inferior)
    # Percorre as linhas (quantidade máxima de linhas = quantidade de colunas)
    for linha_base in range(qtd_column):
        # Define a coluna onde inicia-se a verificação dos pixels
        coluna_base = linha_base
        #### upper_top ###
        # Define o passo (topo = -1)
        step_column = 1
        step_line = -1
        # Percorre os pixels na diagonal em busca de pixels preto da segunda coluna base - up
        if coluna_base > 0:
            evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                            qtd_line, qtd_column, peso, (coluna_base + 1), linha_base, type, step_column, step_line)
            # Grava o grau de evidência na lista "upper_top"
            diagonal_left_right_border['upper_top'].append(evidence_degree)
        else:
            # Percorre os pixels na diagonal em busca de pixels preto da primeira coluna base - up
            evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                            qtd_line, qtd_column, peso, coluna_base, linha_base, type, step_column, step_line)
            # Grava o grau de evidência na lista "upper_top"
            diagonal_left_right_border['upper_top'].append(evidence_degree)
        #### upper_bottom ###
        # Define o passo (base = 1)
        step_column = -1
        step_line = 1
        # Percorre os pixels na diagonal em busca de pixels preto da segunda coluna base - down
        if coluna_base > 0:
            evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                            qtd_line, qtd_column, peso, (coluna_base + 1), linha_base, type, step_column, step_line)
            # Grava o grau de evidência na lista "upper_bottom"
            diagonal_left_right_border['upper_bottom'].append(evidence_degree)
        else:
            # Percorre os pixels na diagonal em busca de pixels preto da primeira coluna base - down
            evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                            qtd_line, qtd_column, peso, coluna_base, linha_base, type, step_column, step_line)
            # Grava o grau de evidência na lista "upper_bottom"
            diagonal_left_right_border['upper_bottom'].append(evidence_degree)
    # Define alguns parâmetros iniciais
    type = 'lower'         # define o tipo de grau de evidência extraído (diagonal_superior ou diagonal_inferior)
    # Percorre as linhas (quantidade máxima linhas = quantidade de colunas) - corte na diagonal a partir da última linha
    index_column_base = -1
    for linha_base in  range((qtd_line-qtd_column), qtd_line):
        # Define a coluna onde inicia-se a verificação dos pixels
        coluna_base = index_column_base = 1
        #### lower_top ###
        # Define o passo (topo = -1)
        step_column = 1
        step_line = -1
```

```
            # Percorre os pixels na diagonal em busca de pixels preto da segunda coluna base - up
            if coluna_base > 0:
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, (coluna_base + 1), linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "lower_top"
                diagonal_left_right_border['lower_top'].append(evidence_degree)
            else:
                # Percorre os pixels na diagonal em busca de pixels preto da primeira coluna base - up
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, coluna_base, linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "lower_top"
                diagonal_left_right_border['lower_top'].append(evidence_degree)
            #### lower_bottom ###
            # Define o passo (base = 1)
            step_column = -1
            step_line = 1
            # Percorre os pixels na diagonal em busca de pixels preto da segunda coluna base - down
            if coluna_base > 0:
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, (coluna_base + 1), linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "lower_bottom"
                diagonal_left_right_border['lower_bottom'].append(evidence_degree)
            else:
                # Percorre os pixels na diagonal em busca de pixels preto da primeira coluna base - down
                evidence_degree = looking_for_diagonal_black_pixel_from_binary_matrix(matrix,
                                qtd_line, qtd_column, peso, coluna_base, linha_base, type, step_column, step_line)
                # Grava o grau de evidência na lista "lower_bottom"
                diagonal_left_right_border['lower_bottom'].append(evidence_degree)
        return diagonal_left_right_border
    except:
        print u'Erro ao mapear a diagonal (esquerda/direita) -
                 utils/manage_image_mapping/get_left_right_diagonal_evidence_degrees_from_binary_matrix'
        return HTTPNotFound(u'Erro ao mapear a diagonal (esquerda/direita) -
                 utils/manage_image_mapping/get_left_right_diagonal_evidence_degrees_from_binary_matrix')


def looking_for_diagonal_black_pixel_from_binary_matrix(matrix, qtd_line, qtd_column, peso, coluna, linha, type, step_column,
step_line):
    ''' Busca o primeiro pixel preto de uma linha vertical e,
        Retorna um dicionário com as características do grau de evidência
        :matrix: matriz binária
        :qtd_line: quantidade de linhas
        :qtd_column: quantidade de colunas
        :peso: peso do pixel
        :coluna: coluna base para procura do primeiro pixel preto na diagonal
        :linha: linha base para procura do primeiro pixel preto na diagonal
        :type: "upper" ou "lower" - diagonal superior ou diagonal inferior
        :step: passo para percorrer as linhas/colunas da matrix (top = -1, bottom = 1)
        Retorna o grau de evidência do pixel
    '''
    try:
        # Se a diagonal superior
        if type == 'upper':
            last_line = qtd_column
        # Se for a diagonal inferior (type == 'lower')
        else:
            last_line = qtd_line
        # Percorre os pixels na diagonal em busca de pixels preto da primeira coluna base
        while 0 <= coluna < qtd_column and 0 <= linha < last_line:
            # Verifica se o pixel é preto e se possui vizinhos
            if  matrix[linha,coluna] == True and \
                verify_neighbor_from_binary_matrix(matrix, linha, coluna, qtd_line-1, qtd_column-1):
                # Calcula o grau de evidência
```

```
            evidence = (coluna + 1) * (linha + 1) * peso
            return evidence
        else:
            coluna = coluna + step_column
            linha  = linha + step_line
    evidence = 0
    return evidence
except:
    print u'Erro ao calcular grau de evidência da diagonal (direita/esquerda) -
            utils/manage_image_mapping/looking_for_diagonal_black_pixel_from_binary_matrix'
    return HTTPNotFound(u'Erro ao calcular grau de evidência da diagonal (direita/esquerda) -
             utils/manage_image_mapping/looking_for_diagonal_black_pixel_from_binary_matrix')
```

manage_image_details.py

```
def get_image_details_by_histogram(image, caracteristics, image_name):
    ''' Cria uma imagem no disco para cada histograma conforme características selecionadas
        :image: imagem codificada no dicionário "images"
        :caracteristics: características selecionadas
        :image_name: nome da imagem
    '''
    # Define o diretório para salvar as imagens dos histogramas
    datadir_histogram =
     '\\dev\\fmusp\\fmusp_reconhecer\\fmusp_reconhecer\\static\\data\\ImagensPreTratadas\\HISTOGRAM\\'

    # Define as dimensões da imagem inicial
    altura = 11
    for type_histogram in caracteristics:
        if not type_histogram in ['vertical_line_segments', 'horizontal_line_segments', 'top_bottom_peaks']:
            # Pega o histograma no dicionário "images"
            histogram = image['evidence_degrees']['histogram'][type_histogram]
            # Define a largura da imagem do histograma
            largura = len(histogram)
            # Cria uma nova imagem
            picture = Image.new('1', (largura, altura), 255)
            # Constroi a imagem do histograma
            pixels = picture.load()
            for mi in range(largura):
                if histogram[mi] > 0:
                    pixels[mi, (10 * float(histogram[mi]))] = 0
            # Redimensiona a imagem
            picture = picture.resize((largura*6.5, altura*6.5), Image.ANTIALIAS)
            # Define o complemento do path e salva a imagem no disco
            path = '%s\\%s\\%s' % (datadir_histogram, type_histogram, image_name)
            picture.save(path, quality=100)
    return True
```

manage_characteristics_extraction.py

```
def get_sample_caracteristics_extraction(caracteristics, images):
    '''
    Pega as características solicitadas
    caracteristics = ['vertical_line_segments', 'horizontal_line_segments',
                      'all_pixels_histogram', 'top_border_histogram', \
                      'external_histogram', 'internal_histogram',
                      'diagonal_right_letf_histogram', 'diagonal_letf_right_histogram', \
```

```
            'color_image_histogram', 'color_column_histogram',
                        'pca_histogram', 'autovalues_pca_histogram']
    Para cada imagem no dicionário 'images' faça:
        Para cada caracteristica solicitada:
            Extrair a característica solicitada
            Gravar no dicionário 'images' a característica extraída
    Retorna o dicionário 'images'
    '''
    try:
        print u'================ Extraindo características das images ====================='
        #Para cada imagem no dicionário 'images' faça:
        for image in images:
            ##### Para cada caracteristica solicitada, extrai as características em graus de evidência #####
            # Pega a matriz binária da imagem
            matrix = numpy.array(images[image]['binary_matrix'])
            #import pdb;pdb.set_trace()
            images[image]['binary_matrix'] = []
            if 'vertical_line_segments' in caracteristics or 'horizontal_line_segments' in caracteristics:
                # Cria um dicionário vazio para guardar a lista dos graus de evidência dos segmentos de reta identificados
                images[image]['evidence_degrees']['external_line_segments'] = {}
                # Identifica os segmentos de reta verticais
                if 'vertical_line_segments' in caracteristics:
                    # Cria listas vazias para pegar os graus de evidência da borda externa da imagem
                    list_ge_left = []
                    list_ge_right = []
                    # Identifica os segmentos de reta do lado esquerdo
                    side = 'left'
                    list_ge_left.extend(images[image]['evidence_degrees']['external_border'][side])
                    images[image]['evidence_degrees']['external_line_segments'][side] =
                                get_extracted_line_segment(list_ge_left, side)
                    # Identifica os segmentos de reta do lado direito
                    side = 'right'
                    list_ge_right.extend(images[image]['evidence_degrees']['external_border'][side])
                    images[image]['evidence_degrees']['external_line_segments'][side] =
                                 get_extracted_line_segment(list_ge_right, side)
                # Identifica os segmentos de reta horizontais
                if 'horizontal_line_segments' in caracteristics:
                    # Cria listas vazias para pegar os graus de evidência da borda externa da imagem
                    list_ge_top = []
                    list_ge_bottom = []
                    # Identifica os segmentos de reta do lado esquerdo
                    side = 'top'
                    list_ge_top.extend(images[image]['evidence_degrees']['external_border'][side])
                    images[image]['evidence_degrees']['external_line_segments'][side] =
                                get_extracted_line_segment(list_ge_top, side)
                    # Identifica os segmentos de reta do lado direito
                    side = 'bottom'
                    list_ge_bottom.extend(images[image]['evidence_degrees']['external_border'][side])
                    images[image]['evidence_degrees']['external_line_segments'][side] =
                                 get_extracted_line_segment(list_ge_bottom, side)
            # Cria um histograma com os graus de evidência identificados no topo da imagem
            if 'external_top_border_histogram' in caracteristics:
                # Utilizar o campo images[image]['evidence_degrees']['external_border']['top'] como
                    #  histograma para a característica
                images[image]['evidence_degrees']['histogram']['external_top_border_histogram'] =
                            images[image]['evidence_degrees']['external_border']['top']
            # Cria um histograma com os graus de evidência do contorno externo da imagem na seguinte sequência: lado esquerdo, base,
topo e lado direito
            if 'external_histogram' in caracteristics:
                # Extrai a característica solicitada
                # Agrupa as listas de graus de evidência das bordas internas
                external_histogram = []
                for item in ('left', 'bottom', 'top', 'right'):
```

```
            external_histogram.extend(images[image]['evidence_degrees']['external_border'][item])
        # Grava o histograma das bordas internas no dicionário "images"
        images[image]['evidence_degrees']['histogram']['external_histogram'] = external_histogram
    # Cria um histograma com os graus de evidência do contorno interno da imagem na seguinte sequência: lado esquerdo, base,
topo e lado direito
    if 'internal_histogram' in caracteristics:
        # Extrai a característica solicitada
        # Agrupa as listas de graus de evidência das bordas internas
        internal_histogram = []
        for item in ('left', 'bottom', 'top', 'right'):
            internal_histogram.extend(images[image]['evidence_degrees']['internal_border'][item])

        # Deixa o item images[image]['evidence_degrees']['internal_border'] do dicionário vazio
        images[image]['evidence_degrees']['internal_border'] = {}
        # Grava o histograma das bordas internas no dicionário "images"
        images[image]['evidence_degrees']['histogram']['internal_histogram'] = internal_histogram
    # Cria um histograma com os graus de evidência das diagonais (direita/esquerda) da imagem
    # na seguinte sequência:
    # diagonal superior (topo, base) e diagonal inferior (topo, base)
    if 'diagonal_right_letf_histogram' in caracteristics:
        # Extrai a característica solicitada
        # Agrupa as listas de graus de evidência das diagonais esquerda/direita superiores e inferiores
        diagonal_right_letf_histogram = []
        for item in ('upper_top', 'upper_bottom', 'lower_top', 'lower_bottom'):
diagonal_right_letf_histogram.extend(images[image]['evidence_degrees']['diagonal_right_left_border'][item])
        # Deixa o item images[image]['evidence_degrees']['diagonal_right_left_border'] do dicionário vazio
        images[image]['evidence_degrees']['diagonal_right_left_border'] = []
        # Grava o histograma das diagonais esquerda/direita no dicionário "images"
        images[image]['evidence_degrees']['histogram']['diagonal_right_letf_histogram'] =
                    diagonal_right_letf_histogram
    # Cria um histograma com os graus de evidência das diagonais (esquerda/direita) da imagem na seguinte sequência:
    # diagonal superior (topo, base) e diagonal inferior (topo, base)
    if 'diagonal_letf_right_histogram' in caracteristics:
        # Extrai a característica solicitada
        # Agrupa as listas de graus de evidência das diagonais esquerda/direita superiores e inferiores
        diagonal_letf_right_histogram = []
        for item in ('upper_top', 'upper_bottom', 'lower_top', 'lower_bottom'):
diagonal_letf_right_histogram.extend(images[image]['evidence_degrees']['diagonal_left_right_border'][item])
        # Deixa o item images[image]['evidence_degrees']['diagonal_left_right_border'] do dicionário vazio
        images[image]['evidence_degrees']['diagonal_left_right_border'] = []
        # Grava o histograma das diagonais esquerda/direita no dicionário "images"
        images[image]['evidence_degrees']['histogram']['diagonal_letf_right_histogram'] =
                    diagonal_letf_right_histogram
    # Cria um histograma dos graus de evidência das cores da matriz na seguinte
    # sequência: % pixels preto, % pixels brancos
    if 'color_image_histogram' in caracteristics:
        # Extrai a porcentagem de pixels pretos e brancos da matriz e adiciona no dicionário 'images'
        color_image_histogram = get_color_image_histogram(matrix)
        # Grava no dicionário 'images' a característica extraída
        # Ex. [% pixels pretos, % de pixels brancos]
        images[image]['evidence_degrees']['histogram']['color_image_histogram'] = color_image_histogram
    # Cria um histograma dos graus de evidência das cores de cada coluna da matriz ordenados por coluna
    if 'color_column_histogram' in caracteristics:
        # Extrai a característica solicitada
        list_color_by_column = get_color_by_column_histogram(matrix)
        # Grava no dicionário 'images' a característica extraída
        images[image]['evidence_degrees']['histogram']['color_column_histogram'] = list_color_by_column
    # Se uma das características 'componentes principais' ou 'autovalores dos componentes principais' forem solicitadas
    if 'pca_histogram' in caracteristics or 'autovalues_pca_histogram' in caracteristics:
        # Obtém as características dos componentes principais
        pca_caracteristics = get_pca_caracteristics(matrix)
        # Grava a matriz dos componentes principais no dicionário "images"
        #images[image]['pca_matrix'] = pca_caracteristics['pca_matrix']
```

```
            # Cria um histograma dos graus de evidência dos autovalores dos componentes
              # principais da matriz binária
            images[image]['evidence_degrees']['histogram']['autovalues_pca_histogram'] =
                        pca_caracteristics['vectors']
            # Cria um histograma dos graus de evidência das componentes principais da matriz binária
            if 'pca_histogram' in caracteristics:
              # Grava o histograma dos graus de evidência das conjunto principais no dicionário
              images[image]['evidence_degrees']['histogram']['pca_histogram'] = pca_caracteristics['list_ge_pca']
        return images
    except:
        print u'Erro ao extrair características das imagens -
                        utils/manage_caracteristics_extraction/get_sample_caracteristics_extraction'
        return HTTPNotFound(u'Erro ao extrair características das imagens -
                        utils/manage_caracteristics_extraction/get_sample_caracteristics_extraction')

def principalComponents(matrix):
    ''' Calcula as componentes principais da matriz binária da imagem
        matrix: matriz binária da imagem
        Retorna os autovalores e os componentes principais da matriz
    '''
    try:
        # Columns of matrix correspond to data points, rows to dimensions.
        deviationMatrix = (matrix.T - numpy.mean(matrix, axis=1)).T
        covarianceMatrix = numpy.cov(deviationMatrix)
        eigenvalues, principalComponents = numpy.linalg.eig(covarianceMatrix)
        # sort the principal components in decreasing order of corresponding eigenvalue
        indexList = numpy.argsort(-eigenvalues)
        eigenvalues = eigenvalues[indexList]
        principalComponents = principalComponents[:, indexList]
        return eigenvalues, principalComponents
    except:
        print u'Erro ao calcular componentes principais -
                        utils/manage_caracteristics_extraction/principalComponents'
        return HTTPNotFound(u'Erro solicitar pré-tratamento das imagens -
                        utils/manage_caracteristics_extraction/principalComponents')


def get_pca_caracteristics(matrix):
    ''' Retorna a um vetor com a lista de graus de evidência que representam
        os componente principais da matriz binária da imagem
        matrix: matriz binária da imagem
    '''
    try:
        def get_ge_pca(max, min, value):
            ''' Calcula o grau de evidência do componente principal
                :max: valor máximo do intervalo
                :min: valor mínimo do intervalo
                :value: valor a ser normalizado
                Função para calcular o valor = ((1. / (max-min)) * value) + ((1. / (max - min)) * (0 - min))
                Retorna o valor normalizado entre o intervalo [0,1] definido como grau de evidência
            '''
            # Calcula o intervalo entre os pontos máximo e mínimo
            distance = max - min
            # Calcula o grau de evidência
            ge = ((1. / distance) * value) + ((1. / distance) * (0 - min))
            return ge.real
        # Obtém os autovalores e os componentes principais da matriz
        # Utiliza a matrix transposta para considerar as colunas como linhas no cálculo das componentes principais
        ev, pca = principalComponents(matrix.T)
        # Normaliza os autovalores em graus de evidência
        ge_list_autovalues = [get_ge_pca(ev.max().real, ev.min().real, autovalue) for autovalue in ev]
        # Pega os valores máximo e mínimo da matrix
```

```
        max = pca.max()
        min = pca.min()
        # Transforma o array em uma lista
        list_pca = pca.tolist()
        # Cria uma lista vazia para agrupar todas as linhas da matriz dos componentes principais
        list_values = []
        # Cada linha corresponde a coluna da matriz original porque se trata da matriz transposta
        # Agrupa as colunas da matriz
        for line in list_pca:
            list_values += line
        # Transforma todos os valores em números reais
        real_list_values = [get_ge_pca(max, min, value) for value in list_values]
        # Retorna apenas a lista dos pca da matriz
        pca_values = {
            'pca_matrix': pca,
            'list_ge_pca': real_list_values,
            'vectors':  ge_list_autovalues
        }
        return pca_values
    except:
        print u'Erro ao calcular transformar os componentes principais em graus de evidência -
                                utils/manage_caracteristics_extraction/get_pca_caracteristics'
        return HTTPNotFound(u'Erro ao calcular transformar os componentes principais em graus de evidência -
                                utils/manage_caracteristics_extraction/get_pca_caracteristics')

def get_color_by_column_histogram(matrix):
    ''' Cria uma lista com o grau de evidência da porcentagem de pixels pretos de cada coluna
        :matrix: matriz binária
        Retorna uma lista com os graus de evidência equivalentes a porcentagem de pixels pretos de cada coluna da matriz binária da
imagem
    '''
    try:
        # Cria uma lista vazia
        list_color_by_column = []
        # Calcula a matriz transposta da matriz para considerar as colunas da matriz
        matrix = matrix.T
        # Para cada linha da matriz transposta faça, (a linha da matriz transposta corresponde a
        # coluna da matriz binária)
        for line in range(len(matrix)):
            # Conta a quantidade de pixels pretos na coluna
            qtd_black_pixels = numpy.sum(matrix[line])
            # Calcula grau de evidência
            evidence = qtd_black_pixels / (len(matrix[line]) * 1.)

            # Adiciona à lista o grau de evidência correspondente a porcentagem de pixels pretos na coluna
            list_color_by_column.append(evidence)
        return list_color_by_column
    except:
        print u'Erro ao criar histograma das cores das colunas da imagem -
                                utils/manage_caracteristics_extraction/get_color_by_column_histogram'
        return HTTPNotFound(u'Erro ao criar histograma das cores das colunas da imagem -
                                utils/manage_caracteristics_extraction/get_color_by_column_histogram')

def get_color_image_histogram(matrix):
    ''' Cria uma lista de graus de evidência que equivale a porcentagem de pixels pretos e brancos
        na matriz na seguinte sequência: [% pixels pretos, % pixels brancos]
        :matrix: matriz binária
        Retorna uma lista com os graus de evidência que equivale a porcentagem de pixels pretos e
                brancos Ex. [% black, % white]
    '''
    try:
        # Calcula o total de pixels da matriz
        total_pixels = matrix.size
```

```
        # Calcula o total de pixels pretos da matriz
        total_black_pixels = numpy.sum(matrix)
        # Calcula o total de pixels brancos da matriz
        total_white_pixels = total_pixels - total_black_pixels
        # Calcula os graus de evidência referente as cores branco e preto
        black = total_black_pixels / (total_pixels * 1.)
        white = total_white_pixels / (total_pixels * 1.)
        return [black, white]
    except:
        print u'Erro ao criar histograma das cores da matrix -
                                utils/manage_caracteristics_extraction/get_color_image_histogram'
        return HTTPNotFound(u'Erro ao criar histograma das cores da matrix -
                                utils/manage_caracteristics_extraction/get_color_image_histogram')

def get_extracted_line_segment(list_ge, side):
    ''' Extrai os segmentos de reta de cada borda externa da imagem e
        grava os graus de evidência de cada um deles no dicionário "images"
        :image: uma imagem do dicionário de imagens a ser reconhecida
        :sides: bordas externas da imagem que devem ser extraídos os segmentos de reta
        Retorna o dicionário "images" atualizado
    '''
    try:
        # Defini o valor do fator de tolerância à contradição das unidades UCPC
        ucpc_ftct = 0.15  #(bom p/ o 2 - 0.02)
        # Para cada lado da imagem (side) identifica os segmentos de reta e grava os graus de
        # evidência no dicionário "images"
        ''' Lados esquerdo, direito, topo e base da imagem '''
        # Define o valor do parâmetro direction
        if side in ('left', 'top'):
            # lados esquerdo e topo
            direction = 1
        else:
            # lados direito e base
            direction = 0
        # Cria uma ULVH para identificar os segmentos de reta
        ulvh = ULVH(list_ge, direction, ucpc_ftct)
        # Pega o resultado da RNAP "ULVH"
        # dicionário {'segments': segmentos de reta identificados, 'ge_segments': lista de graus de
        # evidência dos segmentos de reta identificados}
        line_segment = ulvh.results()
        return line_segment['ge_segments']
    except:
        print u'Erro identificar segmentos de reta - utils/manage_caracteristics_extraction/extract_line_segment'
        return HTTPNotFound(u'Erro identificar segmentos de reta -
                utils/manage_caracteristics_extraction/extract_line_segment')
```

manage_recognition.py

```
def get_recognition(caracteristics, images, pattern_datadir, pattern_filename_db):
    ''' Esse módulo tem por objetivo fazer os processamentos necessários para comparar as
        características extraídas do caractere apresentado com as características extraídas
        dos caracteres padrões.
    '''
    try:
        print u'=================== Reconhecendo características ============================='
        # Obtém os graus de evidência dos padrões
        pattern_images = manage_file.get_content_from_file(pattern_datadir, pattern_filename_db)
        # Cria um dicionário para as características reconhecidas do caractere
        recognized_caracteristics = {}
```

```
        # Para cada imagem do dicionário "images" compara as características com os caracteres padrão
        for image in images:
            # Cria um dicionário vazio para a imagem no dicionário "recognized_caracteristics"
            recognized_caracteristics[image] = {}
            # Cria um dicionário vazio para guardar as características do caractere apresentado
            ge_caractere = {
                'external_line_segments': {},
                'histogram': {}
            }
            # Pega as caracteristicas do caractere (amostra)
            images_external_line_segments = images[image]['evidence_degrees'].get('external_line_segments')
            # Pega os segmentos de reta identificados do caractere apresentado
            if images_external_line_segments:
                ge_caractere['external_line_segments'] = images_external_line_segments
            # Pega cada histograma do caractere apresentado conforme caracteristicas selecionadas
            for caracteristic in caracteristics:
                # Cria uma lista vazia para cada característica, exceto para
                    # 'top_bottom_peaks', 'vertical_line_segments' e 'horizontal_line_segments'
                if not caracteristic in ('top_bottom_peaks', 'vertical_line_segments', 'horizontal_line_segments'):
                    ge_caractere['histogram'][caracteristic] = []
                    # Se a característica "top_bottom_peaks" estiver na lista "característics" calcula os
                        # picos superiores e inferiores dos histogramas
                    if 'top_bottom_peaks' in caracteristics:
                        caractere_peaks =
                                get_peaks_from_the_wave(images[image]['evidence_degrees']['histogram'][caracteristic])
                        ge_caractere['histogram'][caracteristic].extend(caractere_peaks)
                    # Grava no dicionário "ge_caractere" cada histograma do caractere apresentado
                    else:
                        caractere_histogram = images[image]['evidence_degrees']['histogram'][caracteristic]
                        ge_caractere['histogram'][caracteristic].extend(caractere_histogram)

            # Faz o reconhecimento das características de cada imagem e
            # grava no dicionário recognized_caracteristics
            recognized_caracteristics[image] = get_recognition_from_pattern(image, pattern_images, caracteristics, ge_caractere)
        return recognized_caracteristics
    except:
        print u'Erro ao reconhecer características  - utils/manage_recognition/get_recognition'
        return HTTPNotFound(u'Erro ao reconhecer características - utils/manage_recognition/get_recognition')

def get_recognition_from_pattern(image_name, pattern_images, caracteristics, ge_caractere):
    ''' Retorna o reconhecimento de cada caractere padrão para um caractere apresentado
        :ge_caractere: características do caractere apresentado
        Retorna o reconhecimento de cada caractere padrão
    '''
    try:
        # Cria um dicionário vazio para guardar o reconhecimento de cada padrão
        image_recognized_caracteristics = {}
        image_recognized_caracteristics['patterns'] = {}
        image_recognized_caracteristics['name'] = image_name.split('_')[0]
        # Define o ftct para as características 'vertical_line_segments' e 'horizontal_line_segments' conforme caractere
        conf_ftct = {
            '0': {'vertical': 0.20, 'horizontal': 0.20},
            '1': {'vertical': 0.18, 'horizontal': 0.20},
            '2': {'vertical': 0.10, 'horizontal': 0.10},
            '3': {'vertical': 0.12, 'horizontal': 0.14},
            '4': {'vertical': 0.15, 'horizontal': 0.15},
            '5': {'vertical': 0.20, 'horizontal': 0.17},
            '6': {'vertical': 0.16, 'horizontal': 0.18},
            '7': {'vertical': 0.20, 'horizontal': 0.20},
            '8': {'vertical': 0.10, 'horizontal': 0.10},
            '9': {'vertical': 0.18, 'horizontal': 0.14},
            '10': {'vertical': 0.20, 'horizontal': 0.20},
            '11': {'vertical': 0.20, 'horizontal': 0.20},
```

```
            '12': {'vertical': 0.20, 'horizontal': 0.20}
        }
        # Compara as características com cada caractere padrão
        for pattern in pattern_images:
            # Cria um dicionário vazio para guardar as características do caractere padrão
            ge_pattern = {
                'external_line_segments': {},
                'histogram': {}
            }
            # Pega as caracteristicas do caractere padrão
            pattern_external_line_segments = pattern_images[pattern]['evidence_degrees'].get('external_line_segments')
            # Pega os segmentos de reta identificados do caractere apresentado
            if pattern_external_line_segments:
                ge_pattern['external_line_segments'] = pattern_external_line_segments
            # Pega cada histograma do caractere padrão conforme caracteristicas selecionadas
            for caracteristic in caracteristics:
                # Cria uma lista vazia para cada característica, exceto para
                    # 'top_bottom_peaks', 'vertical_line_segments' e 'horizontal_line_segments'
                if not caracteristic in ('top_bottom_peaks', 'vertical_line_segments', 'horizontal_line_segments'):
                    ge_pattern['histogram'][caracteristic] = []
                    # Se a característica "top_bottom_peaks" estiver na lista "característics" calcula
                        # os picos superiores e inferiores dos histogramas
                    if 'top_bottom_peaks' in caracteristics:
                        pattern_peaks =
                   get_peaks_from_the_wave(pattern_images[pattern]['evidence_degrees']['histogram'][caracteristic])
                        ge_pattern['histogram'][caracteristic].extend(pattern_peaks)
                    # Grava no dicionário "ge_caractere" cada histograma do caractere padrão
                    else:
                        pattern_histogram = pattern_images[pattern]['evidence_degrees']['histogram'][caracteristic]
                        ge_pattern['histogram'][caracteristic].extend(pattern_histogram)
            # Cria uma rede UARCcmc7
            selected_conf_ftct = conf_ftct.get(pattern_images[pattern]['name'])
            uarccmc7 = UARCcmc7(ge_pattern, ge_caractere, selected_conf_ftct, caracteristics)
            # Obtém as características reconhecidas
            results = uarccmc7.results()
            # Adiciona o resultado do reconhecimento ao dicionário "recognized_caracteristics"
            image_recognized_caracteristics['patterns'][pattern] = {}
            image_recognized_caracteristics['patterns'][pattern]['caracteristics'] = results
        return image_recognized_caracteristics
    except:
        print u'Erro ao reconhecer características de um caractere  -
                utils/manage_recognition/get_recognition_from_pattern'
        return HTTPNotFound(u'Erro ao reconhecer características de um caractere -
                utils/manage_recognition/get_recognition_from_pattern')

def get_peaks_from_the_wave(list_ge):
    ''' Retorna um dicionário com os picos superiores e inferiores da lista normalizados em graus de evidência
        :list_ge: lista de valores
    '''
    try:
        # Cria as listas vazias dos picos superiores e inferiores da lista apresentada
        upper_peaks = []
        lower_peaks = []
        # Cria variáveis de contagem do tamanho dos picos (superiores e inferiores) com valor 0
        first_upper_peaks = 0
        last_upper_peaks = 0
        # Percorre a lista em busca de picos superiores
        for item in range(len(list_ge)-1):
            # Se o próximo item for maior do que o item atual e não existir valor em "first_upper_peaks"
            # adiciona o item atual a "first_upper_peaks"
            if list_ge[item+1] > list_ge[item]:
                if not first_upper_peaks:
                    first_upper_peaks = list_ge[item]
```

```
        # Se não, calcula a distância entre o ponto inicial e final do pico e grava o valor na
        # lista "upper_peaks"
        else:
            if first_upper_peaks:
                last_upper_peaks = list_ge[item]
                ge_value = last_upper_peaks - first_upper_peaks
                upper_peaks.append(ge_value)
                first_upper_peaks = 0
                last_upper_peaks = 0
    # Se existir valor em "first_upper_peaks", calcular o grau de evidência considerando
    # o último valor da lista como "last_upper_peaks"
    if first_upper_peaks:
        last_upper_peaks = list_ge[-1]
        ge_value = last_upper_peaks - first_upper_peaks
        upper_peaks.append(ge_value)
    # Cria variáveis de contagem do tamanho dos picos (superiores e inferiores) com valor 0
    first_lower_peaks = 0
    last_lower_peaks = 0
    # Percorre a lista em busca de picos superiores
    for item in range(len(list_ge)-1):
        # Se o próximo item for menor do que o item atual e não existir valor em
        # "first_lower_peaks" adiciona o item atual a "first_lower_peaks"
        if list_ge[item+1] < list_ge[item]:
            if not first_lower_peaks:
                first_lower_peaks = list_ge[item]
        # Se não, calcula a distância entre o ponto inicial e final do pico e grava o valor na lista "lower_peaks"
        else:
            if first_lower_peaks:
                last_lower_peaks = list_ge[item]
                ge_value = first_lower_peaks - last_lower_peaks
                lower_peaks.append(ge_value)
                first_lower_peaks = 0
                last_lower_peaks = 0
    # Se existir valor em "first_upper_peaks", calcular o grau de evidência considerando
    # o último valor da lista como "last_upper_peaks"
    if first_lower_peaks:
        last_lower_peaks = list_ge[-1]
        ge_value = first_lower_peaks - last_lower_peaks
        lower_peaks.append(ge_value)
    top_bottom_peaks = upper_peaks + lower_peaks
    return top_bottom_peaks
except:
    print u'Erro identificar picos superiores e inferiores do histograma  -
            utils/manage_recognition/get_peaks_from_the_wave'
    return HTTPNotFound(u'Erro identificar picos superiores e inferiores do histograma -
            utils/manage_recognition/get_peaks_from_the_wave')
```

manage_paraconsistent_analysis.py

```
def get_paraconsistent_analysis(recognized_caracteristics):
    ''' Faz a análise paraconsistente para cada imagem do dicionário "recognized_caracteristics"
        Cada imagem terá os seguintes dados
        recognized_caracteristics[image] = {
            'name': nome da imagem apresentada
            'patterns': {
                pattern: {
                    'caracteristics': {},
                },
            },
```

```
        'paraconsistent_analysis': {
            pattern: valor da análise paraconsistente
        }
        'recognized_caractere': {
            'recognized_pattern': nome do caractere padrão reconhecido,
            'evidence_degree': valor do grau de evidencia de reconhecimento do caractere padrão reconhecido,
            'name': nome do caractere reconhecido
        },
        'discarded_caractere': {
            'discarded_pattern': nome do caractere padrão discartado,
            'evidence_degree': valor do grau de evidencia de reconhecimento do caractere padrão descartado,
            'name': nome do caractere discartado
        },
        'recognized_flag': Falso ou verdadeiro
    }
    Retorna um dicionário com:
    :analysis: um dicionário com o grau de reconhecimento de cada caractere padrão
    :caractere_reconhecido: caractere padrão com maior grau de evidência
    :caractere_excluído: caractere padrão com menor grau de evidência
'''
try:
    # Percorre cada imagem do dicionário "recognized_caracteristics"
    for image in recognized_caracteristics:
        # Cria um dicionário vazio para guardar o resultado da análise paraconsistente de cada caractere padrão
        recognized_caracteristics[image]['paraconsistent_analysis'] = {}
        # Percorre cada caractere padrão para fazer a análise paraconsistente das características
        for pattern in recognized_caracteristics[image]['patterns']:
            # Cria uma lista vazia para os graus de evidência das características reconhecidas
            list_ge_recognized_caracteristics = []
            # Faz a análise paraconsistente de cada image
            if 'vertical_line_segments' and 'horizontal_line_segments' in
                        recognized_caracteristics[image]['patterns'][pattern]['caracteristics']:
                list_ge_recognized_caracteristics =
               [recognized_caracteristics[image]['patterns'][pattern]['caracteristics']['vertical_line_segments'],
               recognized_caracteristics[image]['patterns'][pattern]['caracteristics']['horizontal_line_segments']]
                # Cria uma UAPParaExtr para os segmentos de reta verticais e horizontais
                uapparaextr_line_segments = UAPParaExtr(list_ge_recognized_caracteristics)
                # Obtém o resultado do algoritmo (análises paraconsistentes)
                result_linhas = uapparaextr_line_segments.results()
                # Grava apenas o resultado do algoritmo na 'list_ge_recognized_caracteristics'
                    # (apaga os valores anteriores da lista)
                list_ge_recognized_caracteristics = [result_linhas]
            # Adiciona os grau de evidência das características reconhecidas listadas na lista 'caracteristics' com exceção das
            # características 'vertical_line_segments' e 'horizontal_line_segments'
            for caracteristic in recognized_caracteristics[image]['patterns'][pattern]['caracteristics']:
                if not caracteristic in ['vertical_line_segments', 'horizontal_line_segments']:
list_ge_recognized_caracteristics.append(recognized_caracteristics[image]['patterns'][pattern]['caracteristics'][caracteristic])
            # Cria outra UAPParaExtr para fazer a análise paraconsistente de todas as características
            analysis_uapparaext = UAPParaExtr(list_ge_recognized_caracteristics)
            # Obtém o resultado do algoritmo (análises paraconsistentes)
            results = analysis_uapparaext.results()

            # Grava no dicionário "recognized_caracteristics" o resultado da análise paraconsistente
            recognized_caracteristics[image]['paraconsistent_analysis'][pattern]= results
        # Pega o caracter padrão reconhecido e discartado
        recognized_caracteristics[image]['recognized_caractere'] = {
            'recognized_pattern': max(recognized_caracteristics[image]['paraconsistent_analysis'].iteritems(),
                                      key=operator.itemgetter(1))[0]
        }
        recognized_caracteristics[image]['recognized_caractere']['evidence_degree'] =
             recognized_caracteristics[image]['paraconsistent_analysis'][recognized_caracteristics[image]['recognized_caractere']['rec
ognized_pattern']]
        recognized_caracteristics[image]['recognized_caractere']['name'] =
```

```
            recognized_caracteristics[image]['recognized_caractere']['recognized_pattern'].split('_')[0]
        # Pega o caracter padrão reconhecido e discartado
        recognized_caracteristics[image]['discarded_caractere'] = {
            'discarded_pattern': min(recognized_caracteristics[image]['paraconsistent_analysis'].iteritems(),
                                     key=operator.itemgetter(1))[0]
        }
        recognized_caracteristics[image]['discarded_caractere']['evidence_degree'] =
            recognized_caracteristics[image]['paraconsistent_analysis'][recognized_caracteristics[image]['discarded_caractere']['disc
arded_pattern']]
        recognized_caracteristics[image]['discarded_caractere']['name'] =
            recognized_caracteristics[image]['discarded_caractere']['discarded_pattern'].split('_')[0]

        # Verifica se o caractere apresentado foi reconhecido corretamente
        if recognized_caracteristics[image]['name'] ==
                        recognized_caracteristics[image]['recognized_caractere']['name'] and \
            recognized_caracteristics[image]['discarded_caractere']['discarded_pattern'] !=
                        recognized_caracteristics[image]['recognized_caractere']['recognized_pattern']:
            recognized_caracteristics[image]['recognized_flag'] = True
        else:
            recognized_caracteristics[image]['recognized_flag'] = False
    # Retorna o dicionário "recognized_caracteristics" atualizado
    return recognized_caracteristics
  except:
    print u'Erro ao fazer análise paraconsistente  -
                        utils/manage_paraconsistente_analysis/get_paraconsistent_analysis'
    return HTTPNotFound(u'Erro ao fazer análise paraconsistente -
                         utils/manage_paraconsistente_analysis/get_paraconsistene_analysis')
```

Diretório view

image.py

```
def view_images(request):
  caractere_type = request.matchdict.get('type')
  if 'pattern' in caractere_type:
    folder = 'PATTERN'
    pattern_type = caractere_type.split('pattern_')[1]
    # Ex. 'CMC7\\'
    complement_dir = '%s\\%s\\' % (folder, pattern_type)
    page_name = u'%s - %s' % (folder, pattern_type)
    page_view = 'create_pattern_file'
    link_name = u'Codificar com os filtros '
  else:
    folder = 'SAMPLE'
    selected_item = request.matchdict.get('item')
    # Ex. 'CMC7\\0\\'
    complement_dir = '%s\\%s\\%s\\' % (folder, caractere_type, selected_item)
    page_name = u'%s - Caractere %s' % (caractere_type, selected_item)
    page_view = 'recognize_sample'
    link_name = u'Reconhecer Caracteres com os filtros '
  navigation = {
    'current': {
      'caractere_type': caractere_type,
      'name': u'Lista de Caracteres - %s' % page_name,
      'view': page_view,
      'link': link_name
    }
  }
```

```
    datadir =  request.registry.settings['fmusp_reconhecer.datadir'] + complement_dir
    list_characters = os.listdir(os.path.expanduser(datadir))
    return {
        'folder': folder,
        'complement_dir': complement_dir,
        'list_characters': list_characters,
        'navigation': navigation
    }

def recognize_sample(request):
    ''' Apresenta os caracteres reconhecidos
    '''
    navigation = {
        'current': {
            'name': u'Caracteres Reconhecidos',
            'view': ''
        }
    }
    caractere_type = request.matchdict.get('type')
    selected_item = request.matchdict.get('item')
    save_sample_in_file = True
    # Define filtros conforme selecionado pelo usuário
    filters = {}
    index_filter = -1
    if request.POST.get('medianfilter'):
        index_filter = index_filter + 1
        filters[index_filter] = request.POST.get('medianfilter')
    if request.POST.get('modefilter'):
        index_filter = index_filter + 1
        filters[index_filter] = request.POST.get('modefilter')
    # Adiciona o filtro de binarização da imagem (obrigatório)
    index_filter = index_filter + 1
    filters[index_filter] = 'binaryfilter'
    if request.POST.get('minfilter'):
        index_filter = index_filter + 1
        filters[index_filter] = request.POST.get('minfilter')
    # Adiciona o filtro para recorte dos pontos máximos e mínimos da imagem (obrigatório)
    index_filter = index_filter + 1
    filters[index_filter] = 'trimfilter'
    caracteristics = []
    for item in request.POST:
        if 'caracteristic_' in item:
            caracteristics.append(item.split('caracteristic_')[1])
    if request.POST.get('sample'):
        # Define o tipo da amostra como 'diretório'
        type = 'folder'
        filename = None
    else:
        for item in request.POST:
            if 'buttom' in item:
                # Define o tipo da amostra como 'arquivo'
                type = 'file'
                filename = item.split('buttom_')[1]
                break
    # Define path do diretório da amostra
    datadir =  request.registry.settings['fmusp_reconhecer.datadir']
    datadir_complement = 'SAMPLE\\%s\\%s\\' % (caractere_type, selected_item)
    ##### Faz o pre-tratamento das imagens #####
    images = manage_image_treatment.get_treatment(filters, datadir, datadir_complement, type, filename=filename)
    ##### Mapeamento das imagens em graus de evidência #####
    images = manage_image_mapping.get_sample_mapping(caracteristics, images)
    ##### Extrai as características das imagens #####
    images = manage_caracteristics_extraction.get_sample_caracteristics_extraction(caracteristics, images)
```

```
##### Pega as características dos caracteres padrão #####
# Define path do arquivo dos caracteres padrão
datadir_db = request.registry.settings['fmusp_reconhecer.datadir_db'] + 'PATTERN\\'
# Define o filename dos caracteres padrao conforme filtros selecionados
pattern_filename_complement = filters[0]
for index_filter in range(1, len(filters)):
    pattern_filename_complement =  '%s_%s' % (pattern_filename_complement, filters[index_filter])
# Define o path do arquivo dos caracteres padrão
if 'HandwrittenNumeric' in caractere_type:
    # Define path do diretório dos caracteres padrão e o nome do arquivo dos caracteres padrão
    pattern_datadir = '%s%s\\' % (datadir_db, 'HandwrittenNumeric')
    pattern_filename_db = '%s_%s' % ('HandwrittenNumeric', pattern_filename_complement)
else:
    # Define path do diretório dos caracteres padrão e o nome do arquivo dos caracteres padrão
    pattern_datadir = '%s%s\\' % (datadir_db, caractere_type)
    pattern_filename_db = '%s_%s' % (caractere_type, pattern_filename_complement)
# Pega no arquivo o dicionário de características dos caracteres padrão
pattern_images = manage_file.get_content_from_file(pattern_datadir, pattern_filename_db)
##### Compara as características com os caracteres padrão - reconhecimento #####
recognized_caracteristics = manage_recognition.get_recognition(caracteristics, images, pattern_datadir, pattern_filename_db)
##### Faz a análise paraconsistente do reconhecimento #####
recognized_caracteristics = manage_paraconsistent_analysis.get_paraconsistent_analysis(recognized_caracteristics)

##### Grava o dicionário de características das imagens no arquivo ####
if save_sample_in_file is True:
    # Define o nome do arquivo do banco de dados
    filename_db = caractere_type
    if type != 'folder':
        filename_db = '%s_%s' % (selected_item, filename.split('.jpg')[0])
    filename_db =  '%s_%s_%s' % (selected_item, filename_db, pattern_filename_complement)
    # Grava o dicionário no arquivo
    datadir_db = request.registry.settings['fmusp_reconhecer.datadir_db'] + 'RESULTS\\'
    save_file = manage_file.save_dict_in_file(datadir_db, filename_db, recognized_caracteristics)
# Define o complemento dos diretórios para mostrar as imagens na tela
complement_dir = {}
complement_dir['original'] = 'SAMPLE/%s/%s' % (caractere_type, selected_item)
complement_dir['cut'] = 'TREATMENT/CUT'
if 'HandwrittenNumeric' in caractere_type:
    complement_dir['pattern'] = 'PATTERN/HandwrittenNumeric'
else:
    complement_dir['pattern'] = 'PATTERN/%s' % caractere_type
msg = 'Caracteres analisados pelo sistema: '
##### Calcula especificidade e sensibilidade ######
'''
    acuracia = (VP + VN) / (VP + FP + FN +VN)
    sensibilidade = VP / (VP+FN)
    especificidade = VN / (VN+FP)
    Eficiência = (SENS + ESPEC) / 2
    onde,
        vp = Verdadeiro Positivo, vn = Verdadeiro Negativo
        fp = Falso Positivo, fn = Falso Negativo
'''
dict_sens_spec = {}
# Verifica se a amostra selecionada é 'ALL'
if selected_item == 'ALL':
    # Para cada tipo de caractere
    if caractere_type == 'CMC7':
        defined_range = range(13)
        #defined_range = range(10)
    elif caractere_type in ('ArialAlfabetico', 'TimesNewsRomanAlfabetico', 'HanwrittenAlfabetico'):
        digit_alpha = ['AA','BB','CC','DD','EE','FF','GG','HH','II','JJ','KK','LL','MM','NN','OO','PP',
                                        'QQ','RR','SS','TT','UU','VV','YY','XX','WW','ZZ','a','b','c','d','e','f','g','h','i',
                                        'j','k','l','m','n','o','p','q','r','s','t','u','v','y','x','w','z']
```

```
        defined_range = range(52)
    else:
        defined_range = range(10)
    for digit in defined_range:
        if caractere_type in ('ArialAlfabetico', 'TimesNewsRomanAlfabetico', 'HanwrittenAlfabetico'):
            digit = digit_alpha[digit]
        dict_sens_spec[digit] = {}
        # Cria variáveis "verdadeiro-positivo", "verdadeiro-negativo", "falso-positivo" e
        # "falso_negativo" com valores igual a zero
        dict_sens_spec[digit]['vp'] = 0
        dict_sens_spec[digit]['vn'] = 0
        dict_sens_spec[digit]['fp'] = 0
        dict_sens_spec[digit]['fn'] = 0

        # Para cada imagem no dicionário "images" calcula o vp, vn, fp e fn
        for image in recognized_caracteristics:
            # Calcula vp
            if recognized_caracteristics[image]['recognized_caractere']['name'] == str(digit) and
                        recognized_caracteristics[image]['name'] == str(digit):
                # caractere apresentado = 1 e caractere reconhecido = 1
                dict_sens_spec[digit]['vp'] = dict_sens_spec[digit]['vp'] + 1
            # Calcula vn
            #elif recognized_caracteristics[image]['recognized_caractere']['name'] == str(digit) and
                        recognized_caracteristics[image]['name'] != str(digit):
            elif recognized_caracteristics[image]['recognized_caractere']['name'] != str(digit) and
                        recognized_caracteristics[image]['name'] != str(digit):
                # caractere apresentado = 1 e caractere reconhecido = 1
                dict_sens_spec[digit]['vn'] = dict_sens_spec[digit]['vn'] + 1
            # Calcula fp
            #elif recognized_caracteristics[image]['recognized_caractere']['name'] != str(digit) and
                        recognized_caracteristics[image]['name'] != str(digit):
            elif recognized_caracteristics[image]['recognized_caractere']['name'] == str(digit) and
                        recognized_caracteristics[image]['name'] != str(digit):
                # caractere apresentado = 1 e caractere reconhecido = 1
                dict_sens_spec[digit]['fp'] = dict_sens_spec[digit]['fp'] + 1
            # Calcula fn
            #elif recognized_caracteristics[image]['recognized_caractere']['name'] != str(digit) and
                        recognized_caracteristics[image]['name'] == str(digit):
            elif recognized_caracteristics[image]['recognized_caractere']['name'] != str(digit) and
                        recognized_caracteristics[image]['name'] == str(digit):
                # caractere apresentado = 1 e caractere reconhecido = 1
                dict_sens_spec[digit]['fn'] = dict_sens_spec[digit]['fn'] + 1
        dict_sens_spec[digit]['sensibility'] =
                        dict_sens_spec[digit]['vp'] * 1. / (dict_sens_spec[digit]['vp'] + dict_sens_spec[digit]['fn'])
        dict_sens_spec[digit]['specificity'] =
                        dict_sens_spec[digit]['vn'] * 1. / (dict_sens_spec[digit]['vn'] + dict_sens_spec[digit]['fp'])
        dict_sens_spec[digit]['efficiency'] =
                        (dict_sens_spec[digit]['sensibility'] + dict_sens_spec[digit]['specificity']) / 2
        # acuracia = (VP + VN) / (VP + FP + FN +VN)
        dict_sens_spec[digit]['accuracy'] =
                        (dict_sens_spec[digit]['vp'] + dict_sens_spec[digit]['vn']) * 1. /
                                (dict_sens_spec[digit]['vp'] + dict_sens_spec[digit]['fp'] +
                                dict_sens_spec[digit]['fn'] + dict_sens_spec[digit]['vn'])
# Calcula a quantidade de caracteres reconhecidos
total_recognized_characteres = 0
for image in recognized_caracteristics:
    #import pdb;pdb.set_trace()
    #print recognized_caracteristics[image]['recognized_flag']
    if  recognized_caracteristics[image]['recognized_flag'] == True:
        total_recognized_characteres += 1
# Se a amostra tiver mais do que 1 elemento
if len(recognized_caracteristics) > 1:
    details = False
```

```
        image_name = ''
    else:
        image_name = images.keys()[0]
        # Cria a imagem dos histogramas no disco conforme características selecionadas
        details = manage_image_details.get_image_details_by_histogram(images[image_name], caracteristics, image_name)
    return {
        'navigation': navigation,
        'list_images':  images,
        'recognized_caracteristics':  recognized_caracteristics,
        'total_recognized_characteres': total_recognized_characteres,
        'complement_dir': complement_dir,
        'dict_sens_spec': dict_sens_spec,
        'details': details,
        'image_name': image_name,
        'caracteristics': caracteristics,
        'msg': msg
    }

def create_pattern_file(request):
    ''' Cria o arquivo com as características dos caracteres padrões
    '''
    navigation = {
        'current': {
            'name': u'Caracteres Padrões',
            'view': ''
        }
    }
    # Pega o tipo do caractere Ex. CMC7
    caractere_type = request.matchdict.get('type')
    # Define as característica a serem extraídas
    caracteristics = ['vertical_line_segments', 'horizontal_line_segments', \
                      'all_pixels_histogram', 'external_top_border_histogram', \
                      'external_histogram', 'internal_histogram', \
                      'diagonal_right_letf_histogram', 'diagonal_letf_right_histogram', \
                      'color_image_histogram', 'color_column_histogram', \
                      'pca_histogram', 'autovalues_pca_histogram']
    ##### Define filtros conforme selecionado pelo usuário #####
    filters = {}
    index_filter = -1
    if request.POST.get('medianfilter'):
        index_filter = index_filter + 1
        filters[index_filter] = request.POST.get('medianfilter')
    if request.POST.get('modefilter'):
        index_filter = index_filter + 1
        filters[index_filter] = request.POST.get('modefilter')
    # Adiciona o filtro de binarização da imagem (obrigatório)
    index_filter = index_filter + 1
    filters[index_filter] = 'binaryfilter'
    if request.POST.get('minfilter'):
        index_filter = index_filter + 1
        filters[index_filter] = request.POST.get('minfilter')
    # Adiciona o filtro para recorte dos pontos máximos e mínimos da imagem (obrigatório)
    index_filter = index_filter + 1
    filters[index_filter] = 'trimfilter'

    ##### Define path do diretório #####
    datadir =  request.registry.settings['fmusp_reconhecer.datadir']
    datadir_complement = 'PATTERN\\%s\\' % caractere_type.split('pattern_')[1]
    type = 'folder'
    ##### Pre-tratamento das imagens #####
    images = manage_image_treatment.get_treatment(filters, datadir, datadir_complement, type, filename=None)
    ##### Mapeamento das imagens em graus de evidência #####
    images = manage_image_mapping.get_sample_mapping(caracteristics, images)
```

```
    ##### Extrai as características das imagens #####
    images = manage_caracteristics_extraction.get_sample_caracteristics_extraction(caracteristics, images)
    ##### Grava o dicionário de características das imagens no arquivo ####
    # Define o nome do arquivo do banco de dados
    filename_db = caractere_type.split('pattern_')[1]
    for index_filter in range(len(filters)):
        filename_db =  '%s_%s' % (filename_db, filters[index_filter])
    # Grava o dicionário no arquivo
    datadir_db = request.registry.settings['fmusp_reconhecer.datadir_db'] + datadir_complement
    save_file = manage_file.save_dict_in_file(datadir_db, filename_db, images)
    msg = u'Caracteres Padrões armazenados em arquivo: '
    return {
        'filters': filters,
        'datadir': datadir,
        'list_images':  images,
        'msg': msg,
        'navigation':   navigation
    }

def process_pattern(request):
    ''' Lista de caracteres padrões
        Exibe a lista de caracteres padrões processados
    '''
    navigation = {
        'current': {
            'name': u'Lista de Caracteres Padrões',
            'view': 'imagem_pattern'
        }
    }
    manage_pattern.get_pattern_evidences(True)
    datadir =  request.registry.settings['fmusp_reconhecer.datadir'] + 'PATTERN'
    list_images = os.listdir(os.path.expanduser(datadir))
    # Define os registros da listagem (banco de dados ou no diretório)
    registros = list_images
    if request.method != 'POST':
        return {
            'image_path':   'C:\\%s\\' % datadir ,
            'registros':    registros,
            'navigation': navigation
        }
```