Universidade Paulista

Ciência da Computação

Atividade Prática Supervisionada

DESENVOLVIMENTO DE UM SISTEMA PARA RECONHECIMENTO DE ESCRITA A MÃO

Aleksander Rocha - R.A.: C630IH-0

Daniel Jançanti - R.A.: C630IG-2

Ingrid Oliveira - R.A.: C51791-7

Rafaela Aranas - R.A.: C29CII-0

Araraquara – 2017

Sumário

1 Objetivo e motivação do trabalho

Este trabalho apresenta um estudo sobre o reconhecimento visual de caracteres através da utilização das redes neurais. São abordados os assuntos referentes ao processamento digital de imagens, aos sistemas de reconhecimento de caracteres, e às redes neurais.

Em relação ao processamento digital de imagens são apresentados temas, abrangendo os assuntos referentes à aquisição de imagens, ao tratamento e reconhecimento de padrões.

2 Introdução

Segundo Osorio (1991), um sistema de reconhecimento de caracteres permite que o computador possa adquirir informações através da leitura de textos, ou seja, ao invés de entrar as informações pelo teclado, é possível implementar um sistema computadorizado, associado a um dispositivo ótico, para a aquisição automática de informações descritas sob a forma textual.

Os sistemas de reconhecimento de caracteres possuem uma grande importância junto ao processamento de dados. Isto é, devido ao fato da linguagem escrita ser a forma mais usual do ser humano armazenar e transmitir informações.

As **redes neurais** são um novo paradigma de desenvolvimento de sistemas, que tem se difundido muito na atualidade. As redes neurais tem se apresentado como uma solução muito adequada para sistemas de reconhecimento de padrões, como é o caso de um sistema de reconhecimento de caracteres, onde os padrões os para serem reconhecidos são os próprios caracteres. Esta característica, a ser demonstrada em maiores detalhes posteriormente, foi o que levou ao estudo e emprego das redes neurais junto a este trabalho.

O **processamento de imagens** é uma área de estudos da computação que tem crescido muito nos últimos anos. Seu grande crescimento se deve principalmente ao desenvolvimento de equipamentos, cada vez mais sofisticados e baratos, utilizados para a captura e tratamento de imagens digitais. O processamento gráfico engloba as área de processamento de imagem e computação gráfica, onde o processamento de imagens está relacionado ao tratamento e reconhecimento de padrões em imagens, e a computação gráfica está ligada à síntese de imagens e à geração de descrições (dados não pictóricos) utilizadas na obtenção dessas imagens.

O objeto de trabalho no processamento de imagens é a imagem digital, onde esta é definida como sendo uma matriz de M x N elementos (vetores com informações referentes aos pontos da imagem). Cada elemento da imagem digital é denominado de pixel, tendo associado a si uma informação referente à luminosidade e à cor. Esta informação pode ser um valor indicativo de intensidade luminosa. Esta apresentação através de uma matriz bidimensional de pontos é resultante da manipulação da imagem como sendo uma área de memória do computador. A cada pixel é associada uma ou mais posições de memória, as quais deve armazenar as diversas informações referentes a estes. Com isto pode-se armazenar e processar as informações referentes a uma imagem para posteriormente exibi-la.

O **processo de digitalização** consiste em realizar a aquisição de uma cena, a qual é passada para o computador em um formato adequado para que este possa manupulá-la. As informações visuais são convertidas em sinais elétricos por sensores óticos, e esses sinais são quantificados em valores binários e armazenados na memória do computador. No processo de digitalização, os sinais são amostrados espacialmente e quantificados em amplitude, de forma a obter a imagem digital.

No processo de digitalização, a imagem sofrerá uma amostragem e uma discretização da intensidade luminosa, que é denominada de quantização. No **processo de quantização**, uma imagem com tons contínuos é convertida em uma de tons discretos. Para o armazenamento e processamento por um computador, cada tonalidade (intensidade da luz refletida por cada ponto da imagem) é representada por um valor armazenado de forma binária. Cada ponto amostrado possuirá portanto um valor binário correspondente à intensidade luminosa da imagem naquele ponto. As imagens podem ser do tipo monocromáticas ou policromáticas. Nas imagens monocromáticas somente uma faixa de comprimentos de onda (uma cor) é analisada pelo sensor, determinando as intensidades de luminosidade para esta faixa. As intensidades de luminosidades descritas acima são denominadas de tonalidades ou níveis de intensidade de cor. Para as imagens policromáticas a digitalização é feita para diferentes faixas de comprimento de onda (diferentes cores).

3 Sistemas de reconhecimento de caracteres

Segundo Silva (2003), os sistemas de reconhecimento ótico de caracteres são sistemas desenvolvidos para, de certa forma, reproduzir a capacidade humana de ler textos.

3.1 Origens e Evolução

Mantas (1986) disse que as origens da tentativa de sumular a leitura humana datam de 1870, quando foi inventado por Carey, o scanner de retina. A partir da evolução dos dispositivos capazes de captar e traduzir imagens em sinais que podiam ser medidos, novos horizontes foram descobertos. Entretanto, a primeira tentativa bem sucedida de reconhecimento de caracteres foi realizada somente em 1990 pelo cientista russo Tyutin, que desenvolveu um sistema que visava o auxílio de deficientes visuais.

Alguns dos motivos do desenvolvimento de um sistema de reconhecimento de caracteres são:

- Aquisição de dados numéricos comerciais;
- Compactação de dados;
- Leitura automática de formulários;
- Identificação de endereçamento postal;
- Reconhecimento de cheques bancários;
- Sistemas de aquisição de textos para a tradução automática;
- Interface homem x máquina mais natural e sem necessidade de recodificação dos dados (sujeitos a um menor número de erros, pois não é necessário redigitar os textos, introduzindo novos erros);
- Auxílio a deficientes visuais, na leitura de textos (tradução e impressão automática em braile)

3.2 Tipos de Sistemas de Reconhecimento de Caracteres

Os sistemas de reconhecimento de caracteres podem ser desenvolvidos utilizando-se diferentes procedimentos tanto de aquisição de dados como de processamento de informações além de serem orientados para diferentes tipos de aplicações.

Podem ser classificados primeiramente quanto ao tipo de mecanismo utilizado na aquisição dos textos a serem reconhecidos. Nesta categoria, encontram-se os sistemas magnéticos, sistemas mecânicos e sistemas óticos.

Dentro dos sistemas OCR, são encontradas duas classes importantes, aqueles que são baseados no processamento sequencial das informações e os baseados no processamento paralelo. Existem três categorias principiapais: sistema de reconhecimento online, sistema de reconhecimento de caracteres isolados e sistema de reconhecimento de escrito cursiva.

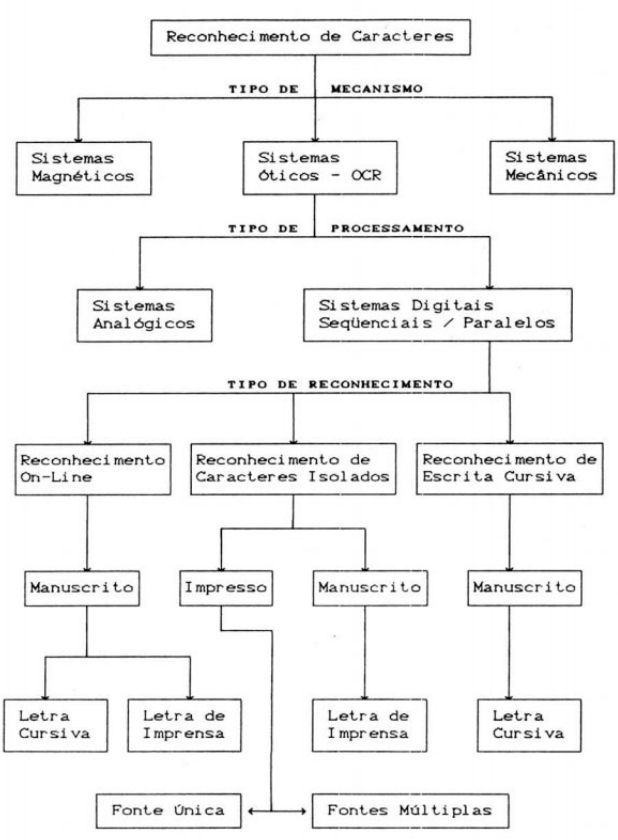

Figura 1: Organograma

3.3 Descrição dos Sistemas OCR

Existem diferentes etapas de processamento das informações visando o reconhecimento de textos por um sistema OCR. Estas etapas se estendem desde a cap-

tura da imagem do texto até a obtenção final de uma identificação deste. Em um sistema de COR não se deve considerar apenas a parte referente às técnicas empregadas na resolução do problema de classificação, mas um item de extrema importância é a medicação e avaliação dos resultados obtidos por estes.

3.3.1 Etapas de Processamento

Em geral, os sistemas de OCR possuem implementadas as seguintes funções: aquisição da imagem do texto, tratamento da imagem, localização e separação dos caracteres, pré-processamento dos padrões, extração de atributos, reconhecimento/-classificação dos padrões e pós-processamento.

Para a separação dos caracteres, o algoritmo tem que levar em consideração diversos fatores, entre eles: o espaçamento entre caracteres e tamanho do caractere (fixo ou variável), a utilização de uma grade auxiliar (no caso de formulários com espaçamento bem definidos), e o tipo de texto a ser reconhecido, onde podem haver caracteres que se tocam ou caracteres com descontinuidade (características muito importante para módulos de segmentação). Esta etapa de processamento é até hoje uma das etapas mais críticas, sendo um problema ainda não completamente solucionado.

Após terem sido isolados os caracteres, e preferencialmente, com a realização de um ajuste de tamanho e posição pré-definidos, pode-se então realizar uma etapa de extração de atributos. Esta etapada é opcional e dependerá exclusivamente do tipo de técnica empregada no reconhecimento e classificação dos padrões.

3.3.2 Técnicas de Reconhecimento

Existem diferentes técnicas empregadas no reconhecimento de padrões, mas estas podem ser classificadas em três grandes categorias, técnicas baseadas em:

- Atributos globais
- Distribuição de pontos e atributos geométricos
- Topológicos

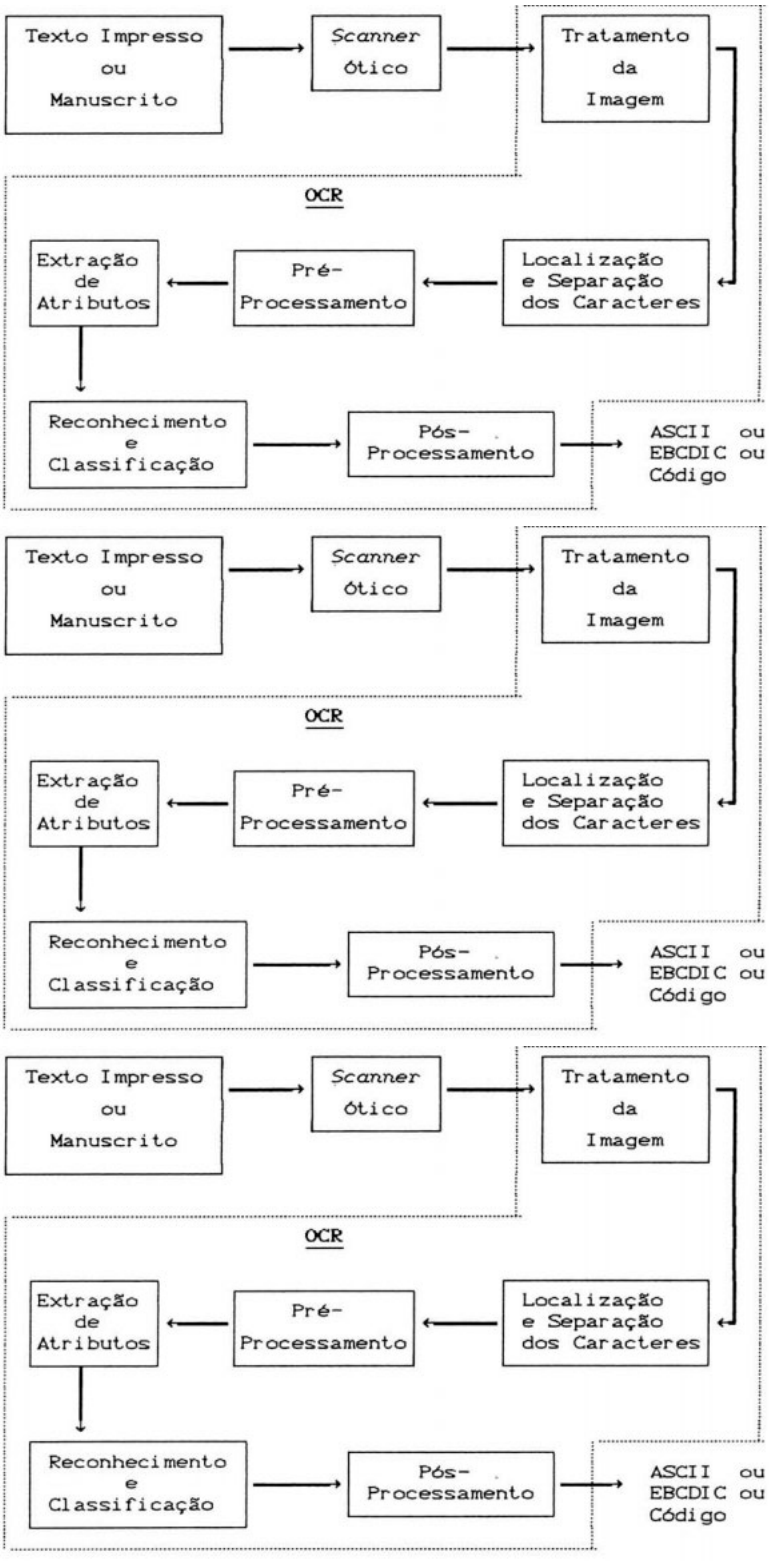

Figura 2: Etapas do processo de reconhecimento

4 Plano de desenvolvimento da aplicação

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

5 Projeto

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

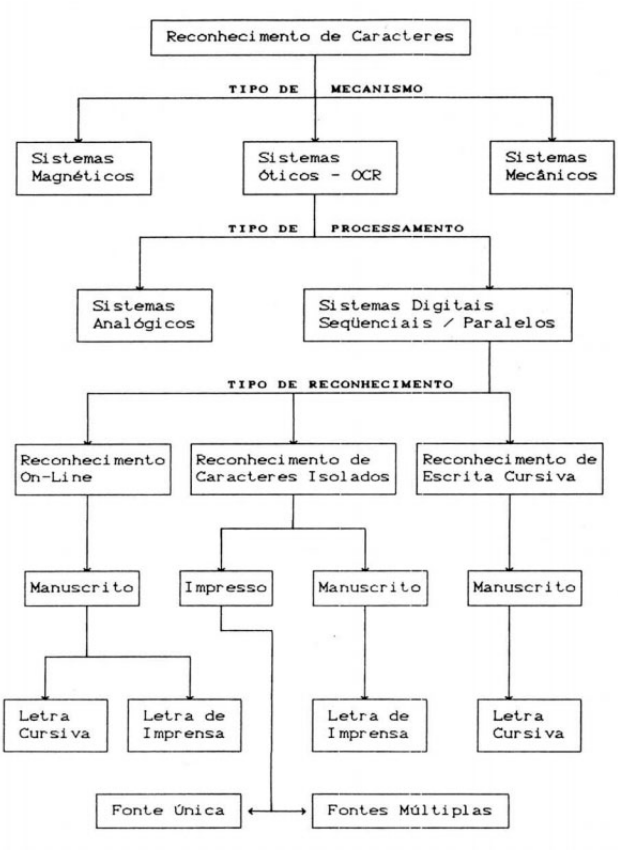

Figura 3: Organograma

6 Código fonte

CharacterImage.java

```
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;


public class CharacterImage implements Serializable {

	private static final long serialVersionUID = 1L;

	private List<Position> positions = new ArrayList<Position>();
	private Position first;
	private BufferedImage image;
	private int width, height;
	private int initWidth, initHeight;

	private double proximidade;

	private char character;

	private boolean isLineBreak;

	public char getCharacter() {
		return character;
	}

	public void setCharacter(char character) {
		this.character = character;
	}

	public CharacterImage(Position fist) {
		this.first = fist;
		this.initWidth = this.first.getX();
		this.initHeight = this.first.getY();
		this.add(fist);
	}

	public boolean contains(Position position) {
		return positions.contains(position);
	}

	public void add(Position position) {
		if (width < position.getX()+1) {
			width = position.getX()+1;
		}
		if (height < position.getY()+1) {
			height = position.getY()+1;
		}
		if (initHeight > position.getY()) {
			initHeight = position.getY();
		}
		if (initWidth > position.getX()) {
			initWidth = position.getX();
		}
		positions.add(position);
	}

	public void add(CharacterImage character) {
		positions.addAll(character.getPositions());
		if (character.initWidth < initWidth) {
			initWidth = character.initWidth;
			first = character.first;
		}
		if (character.initHeight < initHeight) {
```

```
                initHeight = character.initHeight;
        }
        if (character.width > width) {
                width = character.width;
        }
        if (character.height > height) {
                height = character.height;
        }
}

public void addPosition(int x, int y) {
        add(new Position(x, y));
}

public boolean isCompl(CharacterImage character) {
        int space1 = initHeight - character.height;
        int space2 = character.initHeight - height;
        return (positions.size()/2) > character.positions.size()
                        && character.initWidth >= initWidth && character.width <= width
                        && ((space1 >= 0 && space1 < getHeight()/2)
                                        || (space2 >= 0 && space2 < getHeight()/2));
}

public boolean isSelf(CharacterImage character) {
        for (Position position: positions) {
                if (character.haveSibling(position)) {
                        return true;
                }
        }
        return false;
}

public boolean haveSibling(Position olter) {
        for (Position position: positions) {
                if (position.isSibling(olter)) {
                        return true;
                }
        }
        return false;
}

public boolean isValid(int x, int y) {
        int with = image.getWidth(), height = image.getHeight();
        return x >= 0 && y >= 0 && with > x && height > y;
}

public List<Position> getPositionScale() {
        List<Position> positionsScale = new ArrayList<Position>();
        for (int x = 0; x < image.getWidth(); x++) {
                for (int y = 0; y < image.getHeight(); y++) {
                        if (image.getRGB(x, y) != Color.WHITE.getRGB()) {
                                positionsScale.add(new Position(x, y));
                        }
                }
        }
        return positionsScale;
}

public void analisarProximidade(CharacterImage outer) {
        proximidade = 0;
        if (outer.image == null) {
                outer.newImage();
        }
        newImageResize(outer.getWidth(), outer.getHeight());
        List<Position> outerPositions = outer.getPositionScale();
        List<Position> positionsScale = getPositionScale();
        for (Position pScale: positionsScale) {
                if (outerPositions.contains(pScale)) {
                        proximidade++;
                }
```

```
        }

        proximidade = ((proximidade*2)/(positionsScale.size()+outerPositions.size()))*100;

        image = null;
}

public void analisarProximidade2(CharacterImage outer) {
        proximidade = 0;
        newImage();
        outer.newImageResize(getWidth(), getHeight());
        List<Position> outerPositions = outer.getPositionScale();
        List<Position> positionsScale = getPositionScale();
        for (Position pScale: positionsScale) {
                if (outerPositions.contains(pScale)) {
                        proximidade++;
                }
        }

        proximidade = ((proximidade*2)/(positionsScale.size()+outerPositions.size()))*100;

        image = null;
}

public BufferedImage newImageScale(int scale) {

        this.image = newImage();

        BufferedImage bi = new BufferedImage(scale * image.getWidth(null),
                        scale * image.getHeight(null), BufferedImage.TYPE_INT_ARGB);

        Graphics2D grph = (Graphics2D) bi.getGraphics();
        grph.scale(scale, scale);

        grph.drawImage(image, 0, 0, null);
        grph.dispose();

        this.image = bi;
        return bi;
}

public  BufferedImage newImageResize(int width, int height) {
        this.image = newImage();
        int type=0;
        type = image.getType() == 0? BufferedImage.TYPE_INT_ARGB : image.getType();
        BufferedImage resizedImage = new BufferedImage(width, height,type);
        Graphics2D g = resizedImage.createGraphics();
        g.drawImage(image, 0, 0, width, height, null);
        g.dispose();
        this.image = resizedImage;
        return resizedImage;
}

public BufferedImage newImage() {
        this.image = createImage(width-initWidth, height-initHeight);
        for (Position position: positions) {
                image.setRGB(position.getX() - initWidth, position.getY() - initHeight, Color.BLACK.getRGB());
        }
        return this.image;
}

private BufferedImage createImage(int width, int height) {
        BufferedImage newImage = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);

        Graphics2D g2d = (Graphics2D) newImage.getGraphics();
        g2d.setColor(Color.white);
        g2d.fillRect(0, 0, width, height);
        g2d.dispose();

        return newImage;
```

```
        }

        public List<Position> getPositions() {
                return positions;
        }

        public int getWidth() {
                return width-initWidth;
        }

        public int getHeight() {
                return height-initHeight;
        }

        public double getProximidade() {
                return proximidade;
        }

        public int getInitHeight() {
                return initHeight;
        }

        public int getInitWidth() {
                return initWidth;
        }

        public double calculateSpaceWidth(CharacterImage characterImage) {
                return characterImage.getInitWidth()-width;
        }

        public double calculateSpaceHeight(CharacterImage characterImage) {
                return characterImage.getInitHeight()-height;
        }

        public int getCenterY() {
                return height-(getHeight()/2);
        }

        public int getCenterX() {
                return width-(getWidth()/2);
        }

        public boolean isLineBreak() {
                return isLineBreak;
        }

        public void setLineBreak(boolean isLineBreak) {
                this.isLineBreak = isLineBreak;
        }
}
```

CharacterProcessing.java

```
import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import javax.imageio.ImageIO;


public class CharacterProcessing {

        private static BufferedImage grayscale, binarized;

        private double averageSpacingWidth;
        private double averageSpacingHeight;
```

```
public final static int WHITE = Color.WHITE.getRGB(), BLACK = Color.BLACK.getRGB();

public List<CharacterImage> discover(BufferedImage image) throws IOException {
	image = processing(image);

	List<CharacterImage> characters = new ArrayList<CharacterImage>();
	for (int x = 0; x < image.getWidth(); x++) {
		for (int y = 0; y < image.getHeight(); y++) {
			if (WHITE != image.getRGB(x, y)) {
				Position position = new Position(x, y);
				CharacterImage fist = null;
				List<CharacterImage> remove = new ArrayList<CharacterImage>();
				for (CharacterImage character: characters) {
					if (character.haveSibling(position)) {
						if (fist == null) {
							character.add(position);
							fist = character;
						}
						else {
							fist.add(character);
							remove.add(character);
						}
						//break;
					}
				}
				characters.removeAll(remove);
				if (fist == null) {
					characters.add(new CharacterImage(position));
				}
			}
		}
	}

	List<Integer> heightLines = new ArrayList<>();
	boolean init = false;
	for (int y = 0; y < image.getHeight(); y++) {
		boolean isLine = true;
		for (int x = 0; x < image.getWidth(); x++) {
			if (WHITE != image.getRGB(x, y)) {
				init = true;
				isLine = false;
				break;
			}
		}
		if (init && isLine) {
			init = false;
			heightLines.add(y);
		}
	}

	List<CharacterImage> charactersByLine = new ArrayList<>();
	CharacterImage last = null;
	Integer topHeightLine = null;
	for (Integer heightLine: heightLines) {
		for (CharacterImage character: characters) {
			if (character.getInitHeight()+character.getHeight() <= heightLine
					&& (topHeightLine == null || character.getInitHeight() > topHeightLine)) {
				charactersByLine.add(character);
				last = character;
			}
		}
		last.setLineBreak(true);
		topHeightLine = heightLine;
	}
	last.setLineBreak(false);
	characters = charactersByLine;

	List<CharacterImage> remove = new ArrayList<CharacterImage>();
	for (CharacterImage character: characters) {
		for (CharacterImage characterOther: characters) {
```

```
                        if (!character.equals(characterOther) && !remove.contains(characterOther)) {
                                if (character.isCompl(characterOther)) {
                                        character.add(characterOther);
                                        remove.add(characterOther);
                                }
                                else if (characterOther.isCompl(character)) {
                                        characterOther.add(character);
                                        remove.add(character);
                                }
                        }
                }
        }
        characters.removeAll(remove);

        CharacterImage previus = null;
        double sumSpacesWidth = 0, amountSpacesWidth = 0;
        double sumSpacesHeight = 0, amountSpacesHeight = 0;

        for (CharacterImage character: characters) {
                if (previus != null) {
                        double spaceWidh = previus.calculateSpaceWidth(character);
                        if (spaceWidh > 0) {
                                sumSpacesWidth+=spaceWidh;
                                amountSpacesWidth++;
                        }

                        double spaceHeight = previus.calculateSpaceHeight(character);
                        if (spaceHeight > 0) {
                                sumSpacesHeight+=spaceHeight;
                                amountSpacesHeight++;
                        }
                }
                previus = character;
        }
        if (sumSpacesWidth > 0 && amountSpacesWidth > 0) {
                averageSpacingWidth = sumSpacesWidth / amountSpacesWidth;
        }
        if (sumSpacesHeight > 0 && amountSpacesHeight > 0) {
                averageSpacingHeight = sumSpacesHeight / amountSpacesHeight;
        }

        return characters;
}

public static BufferedImage processing(BufferedImage image) throws IOException {
        grayscale = toGray(image);
        binarized = binarize(grayscale);
        //writeImage("output_f");
        return binarized;
}

public static BufferedImage processing2(BufferedImage image) throws IOException {
        return toBinary(toGrayscale(image), 215);
}

private static BufferedImage toGrayscale(BufferedImage image)
                throws IOException {
        BufferedImage output = new BufferedImage(image.getWidth(),
                        image.getHeight(), BufferedImage.TYPE_BYTE_GRAY);
        Graphics2D g2d = output.createGraphics();
        g2d.drawImage(image, 0, 0, null);
        g2d.dispose();
        return output;
}

private static BufferedImage toBinary(BufferedImage image, int t) {
        int BLACK = Color.BLACK.getRGB();
        int WHITE = Color.WHITE.getRGB();

        BufferedImage output = new BufferedImage(image.getWidth(),
```

```
                        image.getHeight(), BufferedImage.TYPE_BYTE_GRAY);

        for (int y = 0; y < image.getHeight(); y++)
                for (int x = 0; x < image.getWidth(); x++) {
                        Color pixel = new Color(image.getRGB(x, y));
                        output.setRGB(x, y, pixel.getRed() < t ? BLACK : WHITE);
                }

        return output;
}

private static void writeImage(String output) throws IOException {
        File file = new File("test-resource/result-processing/"+output+".png");
        ImageIO.write(binarized, "jpg", file);
}

public static int[] imageHistogram(BufferedImage input) {
        int[] histogram = new int[256];

        for(int i=0; i<histogram.length; i++) histogram[i] = 0;

        for(int i=0; i<input.getWidth(); i++) {
                for(int j=0; j<input.getHeight(); j++) {
                        int red = new Color(input.getRGB (i, j)).getRed();
                        histogram[red]++;
                }
        }

        return histogram;
}

private static BufferedImage toGray(BufferedImage original) {
        int alpha, red, green, blue;
        int newPixel;

        BufferedImage lum = new BufferedImage(original.getWidth(), original.getHeight(), original.getType());

        for(int i=0; i<original.getWidth(); i++) {
                for(int j=0; j<original.getHeight(); j++) {

                        alpha = new Color(original.getRGB(i, j)).getAlpha();
                        red = new Color(original.getRGB(i, j)).getRed();
                        green = new Color(original.getRGB(i, j)).getGreen();
                        blue = new Color(original.getRGB(i, j)).getBlue();

                        red = (int) (0.21 * red + 0.71 * green + 0.07 * blue);
                        newPixel = colorToRGB(alpha, red, red, red);

                        lum.setRGB(i, j, newPixel);

                }
        }

        return lum;
}

private static int otsuTreshold(BufferedImage original) {
        int[] histogram = imageHistogram(original);
        int total = original.getHeight() * original.getWidth();

        float sum = 0;
        for(int i=0; i<256; i++) sum += i * histogram[i];

        float sumB = 0;
        int wB = 0;
        int wF = 0;

        float varMax = 0;
        int threshold = 0;
```

```
            for(int i=0 ; i<256 ; i++) {
                    wB += histogram[i];
                    if(wB == 0) continue;
                    wF = total - wB;

                    if(wF == 0) break;

                    sumB += (float) (i * histogram[i]);
                    float mB = sumB / wB;
                    float mF = (sum - sumB) / wF;

                    float varBetween = (float) wB * (float) wF * (mB - mF) * (mB - mF);

                    if(varBetween > varMax) {
                            varMax = varBetween;
                            threshold = i;
                    }
            }

            return threshold;
    }

    private static BufferedImage binarize(BufferedImage original) {
            int red;
            int newPixel;

            int threshold = otsuTreshold(original);

            BufferedImage binarized = new BufferedImage(original.getWidth(), original.getHeight(), original.getType());

            for(int i=0; i<original.getWidth(); i++) {
                    for(int j=0; j<original.getHeight(); j++) {

                            red = new Color(original.getRGB(i, j)).getRed();
                            int alpha = new Color(original.getRGB(i, j)).getAlpha();
                            if(red > threshold) {
                                    newPixel = 255;
                            }
                            else {
                                    newPixel = 0;
                            }
                            newPixel = colorToRGB(alpha, newPixel, newPixel, newPixel);
                            binarized.setRGB(i, j, newPixel);

                    }
            }
            return binarized;
    }

    private static int colorToRGB(int alpha, int red, int green, int blue) {
            int newPixel = 0;
            newPixel += alpha;
            newPixel = newPixel << 8;
            newPixel += red; newPixel = newPixel << 8;
            newPixel += green; newPixel = newPixel << 8;
            newPixel += blue;

            return newPixel;
    }

    public boolean isBlankSpace(CharacterImage previus, CharacterImage next) {
            boolean blankSpace = false;
            if (previus != null && next != null) {
                    double space = previus.calculateSpaceWidth(next);
                    blankSpace = space > averageSpacingWidth*1.8;
            }
            return blankSpace;
    }
}
```

Classification.java

```
import java.util.List;

public interface Classification {
	CharacterImage analyze(CharacterImage character);
	List<CharacterImage> getCharacters();
}
```

OCRProcessing.java

```
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;

import javax.swing.JTextArea;

public class OCRProcessing {

	private static final String BASE_DIR = "test-resource";

	private JTextArea txtConsole;
	private Classification chassification;

	public OCRProcessing() {
		try {
			this.chassification = new PixelsPositionsClassification(loadData());
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
	public OCRProcessing(JTextArea txtConsole) {
		this();
		this.txtConsole = txtConsole;
	}

	public void learm(String letras, BufferedImage image) throws IOException, ClassNotFoundException {
		letras = letras.replaceAll("␣", "");

		CharacterProcessing processing = new CharacterProcessing();
		List<CharacterImage> characters = processing.discover(image);

		for (int index = 0; index < characters.size(); index++) {
			characters.get(index).setCharacter(letras.charAt(index));;
		}

		save(characters);
	}

	public String recogner(BufferedImage image) throws FileNotFoundException, ClassNotFoundException, IOException {
		CharacterProcessing processing = new CharacterProcessing();
		List<CharacterImage> characters = processing.discover(image);
		StringBuilder txt = new StringBuilder();

		CharacterImage previus = null;
		for (CharacterImage character: characters) {
			//ImageIO.write(character.newImage(), "png", new File(BASE_DIR+"/result-processing/"+Instant.now().toEpochMilli()+".png"))
			CharacterImage candidate = chassification.analyze(character);

			if (processing.isBlankSpace(previus, character)) {
				txt.append("␣");
			}

			txt.append(candidate.getCharacter());
```

```
                if (character.isLineBreak()) {
                        txt.append("\n");
                }

                print(String.format("Caractere: %s, aprox: %s\n", candidate.getCharacter(), candidate.getProximidade()));
                previus = character;
        }

        print(String.format("O resultado e: %s\n", txt.toString()));

        return txt.toString();
    }

    private void save(List<CharacterImage> characters) throws IOException, ClassNotFoundException {
        File fileData = new File(BASE_DIR+"/data.ser");

        if (!fileData.getParentFile().exists()) {
            fileData.getParentFile().mkdirs();
        }

        chassification.getCharacters().addAll(characters);
        FileOutputStream fos = new FileOutputStream(fileData);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(chassification.getCharacters());
        oos.close();
        fos.close();
    }

    private List<CharacterImage> loadData() throws FileNotFoundException, IOException, ClassNotFoundException {
        File fileData = new File(BASE_DIR+"/data.ser");

        List<CharacterImage> characters = null;

        if (fileData.exists()) {
            FileInputStream fis = new FileInputStream(fileData);
            ObjectInputStream ois = new ObjectInputStream(fis);
            characters = (List<CharacterImage>) ois.readObject();
            fis.close();
            ois.close();
        }
        else {
            characters = new ArrayList<CharacterImage>();
        }

        return characters;
    }

    private void print(String text) {
        if (txtConsole == null) {
            //System.out.print(text);
        }
        else {
            txtConsole.append(text);
        }
    }
}
```

OCRView.java

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.GraphicsEnvironment;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.font.FontRenderContext;
import java.awt.geom.AffineTransform;
```

```
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import javax.imageio.ImageIO;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.filechooser.FileNameExtensionFilter;

import br.com.deployxtech.ocr.OCRProcessing;

public class OCRView extends JDialog {

	private static final long serialVersionUID = 1L;

	private JButton btnTest = new JButton("Testar");
	private JButton btnLearm = new JButton("Treinar");
	private JPanel pnlControl = new JPanel();

	private JLabel lblImage = new JLabel();
	private JComboBox<String> cmbFonts = new JComboBox<>();
	private JButton btnSearchImage = new JButton("Selecionar...");
	private JPanel pnlImage = new JPanel();
	private JTextArea txtResult = new JTextArea();
	private JTextArea txtConsole = new JTextArea();
	private JPanel pnlProcessing = new JPanel(new BorderLayout(10,10));

	private JFileChooser fc = new JFileChooser(".");

	private OCRProcessing processing = new OCRProcessing(txtConsole);

	private BufferedImage image;

	public OCRView() {
		setTitle("Reconhecimento de Caracteres");
		setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
		init();
	}

	private void init() {
		GraphicsEnvironment e = GraphicsEnvironment.getLocalGraphicsEnvironment();
		Font[] fonts = e.getAllFonts();
		for (Font font: fonts) {
			cmbFonts.addItem(font.getFontName());
		}

		setLayout(new BorderLayout(10,10));
		pnlControl.add(btnSearchImage);
		pnlControl.add(btnTest);
		pnlControl.add(btnLearm);
		pnlControl.add(cmbFonts);
		pnlControl.setPreferredSize(new Dimension(800, 60));
		getContentPane().add(pnlControl, BorderLayout.NORTH);
		pnlProcessing.setLayout(new GridLayout(2, 1));
		JScrollPane imageScroll = new JScrollPane(lblImage);
		imageScroll.setBorder(BorderFactory.createLineBorder(Color.BLACK,5));
		pnlProcessing.add(imageScroll, BorderLayout.NORTH);
		pnlProcessing.add(txtResult, BorderLayout.CENTER);
		getContentPane().add(pnlProcessing, BorderLayout.CENTER);
```

```
txtConsole.setEditable(false);
JScrollPane consoleScroll = new JScrollPane(txtConsole);
consoleScroll.setPreferredSize(new Dimension(800, 300));
getContentPane().add(consoleScroll, BorderLayout.SOUTH);

btnSearchImage.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		try {
			FileNameExtensionFilter imageFilter = new FileNameExtensionFilter(
					"Image␣files", ImageIO.getReaderFileSuffixes());
			fc.setFileFilter(imageFilter);
			fc.setAcceptAllFileFilterUsed(false);
			int returnVal = fc.showOpenDialog(OCRView.this);
			if (returnVal == JFileChooser.APPROVE_OPTION) {
				File fileImage = fc.getSelectedFile();
				image = ImageIO.read(fileImage);
				lblImage.setIcon(new ImageIcon(image));

				Pattern pattern = Pattern.compile("(\\()(.*?)(\\))");
				Matcher matcher = pattern.matcher(fileImage.getName());
				if (matcher.find()) {
					String letras = matcher.group(2);
					txtResult.setText(letras);
				}
				else {
					txtResult.setText("");
				}
			}
		}
		catch (Exception ex) {
			ex.printStackTrace();
		}
	}
});

btnTest.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent arg0) {
		try {
			if (image == null) {
				JOptionPane.showMessageDialog(OCRView.this, "E␣preciso␣selecionar␣uma␣imagem.");
			}
			else if (!txtResult.getText().equals("")) {
				JOptionPane.showMessageDialog(OCRView.this, "E␣preciso␣apagar␣a␣caixa␣de␣texto␣para␣realizar␣o␣tes
			}
			else {
				String result = processing.recogner(image);
				txtResult.setText(result);
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
});

btnLearm.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent arg0) {
		try {
			if (image == null) {
				JOptionPane.showMessageDialog(OCRView.this, "E␣preciso␣selecionar␣uma␣imagem.");
			}
			else if (txtResult.getText().equals("")) {
				JOptionPane.showMessageDialog(OCRView.this, "E␣preciso␣escrever␣as␣letras␣exatamente␣como␣esta␣na␣i
			}
			else {
				processing.learm(txtResult.getText(), image);
				txtResult.setText("");
				lblImage.setIcon(null);
				JOptionPane.showMessageDialog(OCRView.this, "Letras␣aprendidas...");
			}
		} catch (Exception e) {
```

```
                                e.printStackTrace();
                        }
                }
        });

        cmbFonts.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                        String text = "abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ .,!?-_ 0123456789";
                        Font font = fonts[cmbFonts.getSelectedIndex()];
                        font = new Font(font.getFontName(), Font.PLAIN, 40);

                        Rectangle2D rectagleText = font.getStringBounds(text, new FontRenderContext(new AffineTransform(), true, true));

                        int width = new Double(rectagleText.getWidth()).intValue()+20, height = new Double(rectagleText.getHeight()).intVa


                        BufferedImage newImage = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);

                        Graphics2D g2d = (Graphics2D) newImage.getGraphics();
                        g2d.setColor(Color.white);
                        g2d.fillRect(0, 0, width, height);
                        g2d.setColor(Color.black);
                        g2d.setFont(font);
                        g2d.drawString(text, 10, height-15);
                        g2d.dispose();

                        lblImage.setIcon(new ImageIcon(newImage));
                        txtResult.setText(text.replace(" ", ""));
                        image = newImage;
                }
        });;
    }

    /**
     * @param args
     */
    public static void main(String[] args) {
        OCRView view = new OCRView();
        view.setSize(400, 600);
        view.setLocationByPlatform(true);
        view.setVisible(true);
    }

}
```

PixelsPositionsClassification.java

```
import java.util.List;

public class PixelsPositionsClassification implements Classification {

    private List<CharacterImage> characters;
    private double averageProximity = 100d;
    private int averageQt = 0;

    public PixelsPositionsClassification(List<CharacterImage> characters) {
        this.characters = characters;
    }

    @Override
    public CharacterImage analyze(CharacterImage character) {
        int qt = 0;
        int qtCandidate = 0;
        CharacterImage bestCandidate = null;
        for (CharacterImage charAnalyze: characters) {
            qt++;
            charAnalyze.analisarProximidade2(character);
            if (bestCandidate == null || (bestCandidate.getProximidade() < charAnalyze.getProximidade())) {
                bestCandidate = charAnalyze;
                qtCandidate = qt;
```

```
                }

                if (bestCandidate.getProximidade() > averageProximity && qt > averageQt) {
                        break;
                }
        }

        averageProximity = (averageProximity+bestCandidate.getProximidade())/2;
        averageQt = (averageQt+qtCandidate)/2;

        if (this.characters.remove(bestCandidate)) {
                this.characters.add(0,bestCandidate);
        }

        return bestCandidate;
    }

    @Override
    public List<CharacterImage> getCharacters() {
        return characters;
    }
}
```

Position.java

```
import java.io.Serializable;

public class Position implements Serializable {

    private static final long serialVersionUID = 1L;

    private int x;
    private int y;

    public Position(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int getX() {
        return x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return y;
    }

    public void setY(int y) {
        this.y = y;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj instanceof Position) {
            Position outer = (Position)obj;
            return x == outer.x && y == outer.y;
        }
        else {
            return false;
        }
    }

    public boolean isSibling(Position position) {
        return (x+1 == position.x && y+1 == position.y)
                || (x+1 == position.x && y == position.y)
                || (x+1 == position.x && y-1 == position.y)
                || (x == position.x && y+1 == position.y)
```

```
                        || (x == position.x && y-1 == position.y)
                        || (x-1 == position.x && y+1 == position.y)
                        || (x-1 == position.x && y == position.y)
                        || (x-1 == position.x && y-1 == position.y);
        }

        @Override
        public String toString() {
                return String.format("%s - %s", x, y);
        }
}
```

7 Bibliografia

Referências

Mantas, J. (1986). An overview of character recognition methodologies.

Osorio, F. S. (1991). Um estudo sobre reconhecimento visual de caracteres através de redes neurais.

Silva, Eugênio e Thomé, A. (2003). Reconhecimento de caracteres manuscritos utilizando time de redes neurais. In *Congresso da Sociedade Brasileira de Computação, XXIII*, pages 1–8.