UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LAILLA MILAINNY SIQUEIRA BINE

MannAR: Um método de interpretação de imagens de autômatos aplicado às tecnologias assistivas para deficientes visuais

Maringá

2019

LAILLA MILAINNY SIQUEIRA BINE

MannAR: Um método de interpretação de imagens de autômatos aplicado às tecnologias assistivas para deficientes visuais

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.
Área de concentração: Ciência da Computação

Orientadora: Profª. Drª. Linnyer Beatrys Ruiz Aylon

Maringá

2019

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá – PR., Brasil)

Bine, Lailla Milainny Siqueira

B612m MannAR: um método de interpretação de imagens de autômatos aplicado às tecnologias assistivas para deficientes visuais/ Lailla Milainny Siqueira Bine. -- Maringá, 2019.

158 f. : il. Color., figs. , tabs.

Orientadora: Prof.a. Dr.a. Linnyer Beatrys Ruiz Aylon.

Dissertação (mestrado) – Universidade Estadual de Maringá, Centro de Tecnologia, Programa de Pós-graduação em Ciência da Computação, 2019.

1. Computação – Autômatos Finitos. 2. Deficientes Visuais - Acessibilidade. 3. Deficientes Visuais – Tecnologia Assistiva. 4. Teoria da Computação. I. Aylon, Linnyer Beatrys Ruiz, orient. II. Universidade Estadual de Maringá. Centro de Tecnologia. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 22. ED.005.131

Jane Lessa Monção CRB 9/1173

FOLHA DE APROVAÇÃO

LAILLA MILAINNY SIQUEIRA BINE

MannAR: Um método de interpretação de imagens de autômatos aplicado às tecnologias assistivas para deficientes visuais

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Profa. Dra. Linnyer Beatrys Ruiz Aylon
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Luciana Andréia Fondazzi Martimiano
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Daniela Eloise Flôr
Instituto Federal do Paraná – IFPR-Paranavaí

Aprovada em: 30 de maio de 2019.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

AGRADECIMENTOS

Agradeço a Deus por ter me abençoado nesta caminhada e as pessoas que contribuíram na realização deste trabalho. Em especial:

Aos meus pais, Antonio e Marcianita, por estarem sempre presentes e me ensinarem a seguir o caminho certo. Pai, obrigada por ser um exemplo de aluno e me inspirar a também ser. Mãe, muito obrigada por ser um exemplo de mulher e pela paciência nas correções deste trabalho.

À minha orientadora Professora Doutora Linnyer Beatrys Ruiz Aylon por todas as sugestões e conselhos, não só durante o período deste trabalho mas durante toda a pós-graduação, meu muito obrigada.

Ao Alisson, pelas inúmeras sugestões e conselhos. Com certeza o resultado deste trabalho não seria o mesmo sem a sua participação. Obrigada também pelo incentivo, por sempre acreditar em mim e por estar ao meu lado em todos os momentos.

Ao meu irmão Wuigor, pela ajuda em todas as etapas deste trabalho e principalmente pela paciência e preocupação de fazer meus dias melhores.

Aos meus amigos da pós-graduação e do laboratório Manna que amenizavam os momentos difíceis com alegria e descontração.

Aos professores que participaram da banca que contribuíram com comentários e sugestões no desenvolvimento deste trabalho. Também agradeço a todos que contribuíram de forma direta ou indireta.

E a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro concedido a este trabalho.

“Inclusão é sair das escolas dos diferentes e promover a escola das diferenças.”

Maria Teresa Eglér Mantoan

MannAR: Um método de interpretação de imagens de autômatos aplicado às tecnologias assistivas para deficientes visuais

RESUMO

Disciplinas da área de Ciência da Computação usam frequentemente gráficos e diagramas no processo de ensino e aprendizagem. Esse é o caso dos conteúdos em aulas de Teoria da Computação, Circuitos Digitais, Grafos, Estruturas de Dados, Engenharia de Software, entre outras. Visando contribuir com o campo de Tecnologia assistiva, definiu-se como primeiro passo na descrição automática de diagramas relacionados a Ciência da Computação o escopo de imagens de reconhecedores de linguagens formais, devido a Teoria da Computação ser uma das bases para a área. Para lidar com o desafio de passar de forma completa e acessível esse conteúdo a alunos, este trabalho propõe o MannAR (junção do nome do grupo Manna com *Automata Recognition*), um método de tradução de imagens de diagramas de estados de transição de mecanismos reconhecedores de linguagens formais para formatos alternativos que sejam acessíveis a deficientes visuais. Transmitir a informação que está presente em uma imagem digital envolve conceitos de Visão Computacional, Processamento de Imagens e Inteligência Artificial. As etapas deste trabalho são: investigação de ferramentas já existentes; criação de uma base de imagens de reconhecedores de linguagens formais; criação do método e implementação deste em um protótipo como prova de conceito; e realização de estudos de caso com usuários deficientes visuais. Os autômatos finitos da disciplina de Teoria da Computação foram escolhidos como prova de conceito do MannAR dando origem ao MannAR-FA. Como resultados têm-se um modelo de Rede Neural Convolucional que reconhece os tipos de autômatos em imagem (Autômatos Finitos, Autômatos de Pilha e Máquinas de Turing), a Base de imagens digitais de Autômatos, o método MannAR, e o protótipo MannAR-FA. Foram elaborados testes tanto para o método MannAR, como para a ferramenta MannAR-FA. O MannAR obteve um acerto de 72,5% na base de imagens AF80 e quando comparado a trabalhos relacionados obteve um desempenho superior. Relacionado à ferramenta, obteve-se sucesso em testes com usuários reais, nos quais apenas um erro foi relatado. Assim, considera-se o método proposto uma solução com melhor adaptabilidade aos diferentes usuários trazendo benefícios, não só a usuários com deficiência, mas para todos.

Palavras-chave: Acessibilidade. Tecnologia assistiva. Deficientes visuais. Autômatos Finitos. Teoria da Computação.

MannAR: An automata image interpretation method applied to assistive technologies for the visually impaired

ABSTRACT

Computer Science disciplines often use graphics and diagrams in the teaching and learning process. This is the case of contents in classes of Computer Theory, Digital Circuits, Graphs, Data Structures, Software Engineering, among others. Aiming to contribute to the field of Assistive Technology, the first step in the automatic description of diagrams related to Computer Science was the scope of formal languages recognizers images, due to Computer Theory being one of the bases for the area. To deal with the challenge of passing this content completely and easily to students, this work proposes MannAR (junction of the name Manna group with Automata Recognition), a method of translating transitional states diagrams from mechanisms recognizing formal languages to alternative formats that are accessible to the visually impaired . Transmitting information that is present in a digital image involves Computational Vision, Image Processing and Artificial Intelligence concepts. The stages of this work are: existing tools investigation; formal language recognizers image base creation; a method creation and its implementation in a prototype as proof of concept; and conducting case studies with visually impaired users. The finite automata of the Computer Theory discipline were chosen as proof of concept of MannAR giving rise to MannAR-FA. As results, there is a Convolutional Neural Network model that recognizes the automata types in an image (Finite Automata, Pushdown Automata and Turing Machines), the Digital Automata Images Database, the MannAR method, and the MannAR-FA prototype. Tests were performed for both the MannAR method and the MannAR-FA tool. The MannAR achieved a 72.5% hit on the AF80 image base and when compared to related work it achieved superior performance. Related to the tool, we succeeded in testing with real users, which only one error was reported. Thus, the proposed method is considered a solution with better adaptability to the different users bringing benefits not only to users with disabilities, but to all.

Keywords: Accessibility. Assistive Technology. Visually impaired. Finite Automata. Computer Theory.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AD	Automata Database
AF	Autômato Finito
AP	Autômato de Pilha
BFS	Busca em Largura (*Breadth-First Search*)
CNN	Rede Neural Convolucional (*Convolutional Neural Network*)
CSS	*Cascading Style Sheets*
COPEP	Comitê Permanente de Ética em Pesquisa com Seres Humano
HTML	*Hypertext Markup Language*
JSON	*JavaScript Object Notation*
LA	*Learning Automata*
MannaHap	*Manna Haptic*
MannAR	*Manna Automata Recognition*
MODI	*Microsoft Office Document Imaging*
MT	Máquina de Turing
PROPAE	Programa Multidisciplinar de Pesquisa e Apoio à Pessoa com Deficiência e Necessidades Educativas Especiais
OCR	*Optical Character Recognition*
ReLu	*Rectified Linear Unit*
UML	*Unified Modeling Language*
VGG	*Very Deep Convolutional Networks*
WAI	*Web Accessibility Initiative*
WCAG	*Web Content Accessibility Guidelines*
W3C	*World Wide WEb Consortion*
XMI	*XML Metadata Interchange*
XML	*Extensible Markup Language*

SUMÁRIO

1

Introdução

A Ciência da Computação é uma escolha popular entre os alunos com deficiências porque, em muitos casos, o computador permite que eles realizem atividades que de outra forma não estariam disponíveis (PANSANATO et al., 2012). Um dos maiores desafios no ensino dessa área aos deficientes visuais está na aprendizagem de conteúdos que envolvem abstrações didáticas utilizando imagens e diagramas, como é o caso das disciplinas: Circuitos Digitais, Grafos, Estrutura de Dados, Engenharia de Software e Teoria da Computação.

A grande maioria dos diagramas utilizados em disciplinas de Ciência da Computação se concentram no estilo nó-link, ou seja, nós interligados por arestas. Apesar da utilização do mesmo estilo de diagrama, existem variações de representação dependendo do seu uso. Um exemplo é a diferença entre um diagrama de estados de autômatos e um diagrama de caso de uso. No primeiro, existem símbolos adicionais para representar os estados iniciais e finais. No segundo, há a presença do símbolo que representa o ator.

Considerando esse aspecto, foi necessário escolher um diagrama como o foco deste trabalho. Devido à área de Teoria da Computação ser considerada fundamental para o bom desempenho no curso de Ciência da Computação, optou-se por definir o escopo deste trabalho em imagens digitais de diagramas de estados de transição de mecanismos reconhecedores de linguagens formais.

De acordo com Luque et al. (2018), o uso frequente de gráficos e diagramas cria obstáculos adicionais na aprendizagem de deficientes visuais. Pansanato et al. (2012, p. 33) afirmam que a “natureza gráfica inerente dos diagramas faz com que sejam parcial ou totalmente

inacessíveis a estudantes cegos".

Segundo Francioni e Smith (2002) devem existir duas preocupações relacionadas à acessibilidade de diagramas. A primeira está relacionada com a geração de uma versão tátil da imagem ou do diagrama. Para esta etapa está sendo desenvolvida uma pesquisa em paralelo, denominada MannaHap[1]. Já a segunda se preocupa em passar a informação que está contida nessas representações. Realizar apenas a primeira não garante que a segunda esteja sendo transmitida, portanto, essa etapa é o foco deste trabalho.

A passagem correta do conteúdo das imagens é consequência da descrição dos elementos que ela contém. Isso pode ser feito pela área na computação que trabalha com a descrição de imagens que, segundo Bhowmick e Hazarika (2017), está entre as tendências de tópicos para pesquisa de Tecnologias Assistivas para deficientes visuais.

Esta área se relaciona com diversos campos da Ciência da Computação como: a Visão Computacional que, de acordo com Szeliski (2010), busca descrever uma imagem por meio da reconstrução de suas propriedades como forma, iluminação e cor; o Processamento de Imagens Digitais que é utilizado para extrair as propriedades da imagem; e a Inteligência Artificial utilizada para classificar as propriedades extraídas de acordo com o que está se buscando descrever.

Práticas que auxiliam uma pessoa com deficiência, seja ela qual for, e buscam autonomia, inclusão social e qualidade de vida desse indivíduo são denominadas Tecnologias Assistivas (BRASIL, 2009). Esta é uma área de conhecimento de característica interdisciplinar que engloba produtos, recursos, metodologias, estratégias e serviços.

As Tecnologias Assistivas buscam a promoção da acessibilidade. A acessibilidade não está só relacionada com conceitos físicos, ela também se preocupa com a acessibilidade no espaço digital. Esta consiste em tornar disponível ao usuário, de forma autônoma, toda a informação, sem prejuízos quanto ao conteúdo.

Muitas vezes, é necessária a combinação de múltiplas formas de apresentação da informação para maximizar as habilidades dos usuários que possuem deficiências. Uma forma de viabilizar esse processo é por meio de um sistema automático de transcrição de mídias (TORRES et al., 2002).

Na literatura encontram-se esforços da comunidade para auxiliar a criação e

[1]O MannaHap está sendo desenvolvido no laboratório Manna, do Departamento de Informática da Universidade Estadual de Maringá. A pesquisa estuda a construção de um modelo de sistema háptico DiY (*Do It Yourself*) de representação de imagens digitais para deficientes visuais, munido de um *display* de pinos *refreshable*, com um custo financeiro acessível quando comparado aos já existentes.

manipulação de diagramas, relacionados à Ciência da Computação, por deficientes visuais, como as implementadas pelos autores Miller (2009), Zapirain et al. (2010), Pansanato et al. (2012), Crescenzi et al. (2012), Balik et al. (2013), Sauter (2015) e Luque et al. (2016).

Apesar da existência das ferramentas citadas, essas não realizam a descrição de imagens e não se preocupam em dar acessibilidade a imagens de diagramas já existentes na literatura, como as presentes em livros e materiais elaborados por professores. Docentes estão acostumados a preparar seu material de apoio para as aulas utilizando recursos gráficos como meio facilitador de aprendizagem. Porém, um deficiente visual, mesmo utilizando um computador com leitor de tela e tendo versões digitais de livros e apostilas, não consegue ter acesso a todo o conteúdo, demandando um esforço do professor para traduzi-los.

De acordo com o Artigo 5 da Constituição Federal Brasileira (1988), todos os indivíduos são iguais, sem qualquer tipo de distinção, ou seja, indivíduos com deficiência visual têm o direito, conforme previsto em lei, de participar plenamente na sociedade. Neste contexto, a educação inclusiva assume importância fundamental para a supressão dos limites enfrentados pelos indivíduos com deficiência (TEIXEIRA, 2014).

Outro ponto, é que os alunos com deficiência possuem o direito de ter acesso aos mesmos recursos que todos os outros discentes, o que nem sempre acontece no caso das imagens disponibilizadas em materiais didáticos (LUQUE et al., 2018).

O objetivo geral deste trabalho é desenvolver um método de tradução de diagramas de estados de transição de mecanismos reconhecedores de linguagens formais utilizados na disciplina de Teoria da Computação, denominado MannAR[2]. Almejando alcançar o objetivo geral, têm-se os seguintes objetivos específicos:

- estudar o uso das Redes Neurais Convolucionais (CNN - *Convolutional Neural Network*) como auxílio no processo de identificação dos objetos presentes em um diagrama de estados de transição de mecanismos reconhecedores de linguagens formais e que facilite o processo de reconhecimento;
- estudar métodos de localização de círculos em imagem para que os estados possam ser corretamente reconhecidos e verificar qual o melhor para ser utilizado na versão final do método produzido;
- elaborar um protótipo que utilize o método desenvolvido;

[2]junção do nome do grupo *Manna* com *A*utomata *R*ecognition (Reconhecimento de Autômatos)

- investigar a melhor abordagem para que a tradução realizada seja acessível e útil para deficientes visuais quando estes são estudantes de Ciência da Computação;
- integrar o protótipo construído com ferramentas que possam ser utilizadas pelos usuários para manipular e navegar pelos diagramas reconhecidos;
- realizar testes de casos de uso com usuários deficientes visuais.

A contribuição com a área de Tecnologias Assistivas está relacionada à descrição de imagens para deficientes visuais. Segundo Henry et al. (2014), embora o foco da acessibilidade seja a deficiência, a pesquisa e desenvolvimento em acessibilidade trazem benefícios para todos.

Este documento está organizado como segue: o Tópico 2 apresenta o referencial teórico deste trabalho; o Tópico 3 apresenta os trabalhos relacionados que estão classificados em procedimentos associados ao reconhecimento automático de diagramas presentes na área da ciência da computação e em ferramentas assistivas relacionadas a reconhecedores de linguagens; o Tópico 4 apresenta o método proposto; o Tópico 5 detalha o protótipo elaborado o Tópico 6 apresenta os resultados deste trabalho; e ao final apresenta-se o Tópico 7 com as conclusões e na sequência as referências.

2

REFERENCIAIS TEÓRICOS

O reconhecimento de imagens é um processo complexo, pois envolve diferentes subtarefas, como o reconhecimento dos objetos, de suas relações e a construção do significado que a imagem representa. Em um primeiro momento é necessário entender quais são os objetos que caracterizam um reconhecedor de linguagem (Autômato Finito, Autômato de Pilha e Máquina de Turing) em uma imagem digital e suas variações de representação. Tais conceitos são abordados na Seção 2.1.

Para que seja construída uma Tecnologia Assistiva, é necessário entender os conceitos relacionados a ela. Na Seção 2.2, é realizada a conceituação de Tecnologia Assistiva e apresentam-se alguns guias de desenvolvimento, heurísticas e conceitos que podem nortear a construção da ferramenta e a realização de testes com a mesma.

No desenvolvimento do método de tradução de imagens de diagramas de estados de transição, para possibilitar o acesso a deficientes visuais às imagens, têm-se desafios relacionados à identificação e descrição dos elementos que compõem a imagem, este processo é a essência dos Sistemas de Visão Computacional. Szeliski (2010) considera esta área como um problema árduo, já que esses sistemas buscam descrever objetos em imagens digitais da forma como um ser humano as descreveria.

A Visão Computacional é um campo que engloba o processamento e análise de imagens para reconhecimento de objetos e extração de informações (SUN, 2016), por conseguinte são necessários para o desenvolvimento deste trabalho alguns conceitos relacionados à área, apresentados na Seção 2.3, a saber:

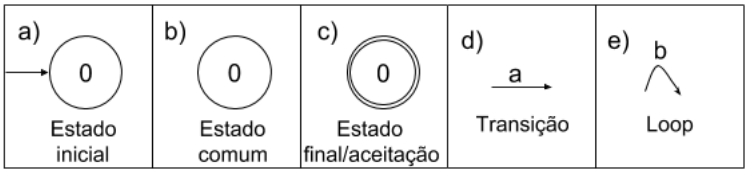
- conceitos fundamentais de processamento digital de imagens;
- métodos de detecção de círculos: neste trabalho considera-se como o cerne do reconhecimento do autômato a descoberta primeiramente dos estados representados por objetos circulares;
- redes neurais convolucionais: para auxiliar o reconhecimento e descrição da imagem são utilizadas Redes Neurais Convolucionais (CNNs) em duas abordagens. A primeira atua no reconhecimento do número de estados que o autômato possui, pressupondo-se que essa informação possa melhorar e facilitar a detecção dos círculos na imagem. A segunda, na detecção do tipo do reconhecedor de linguagem visando facilitar a descrição final do autômato;
- descritores de formas: além de objetos circulares, as imagens de autômatos possuem caracteres e flechas que possuem suas características extraídas por meio de descritores.

2.1 DIAGRAMAS DE ESTADOS DE TRANSIÇÃO

A Teoria da Computação é considerada a base da Ciência da Computação e estuda modelos computacionais genéricos para especificar qualquer função computável e seus limites (DIVERIO; MENEZES, 2009). Entre esses modelos estão os reconhecedores de linguagem conhecidos como Máquinas de Turing (MT), Autômatos Finitos (AF) e Autômatos de Pilha (AP).

Conceitos de Teoria da Computação são definidos matematicamente, porém, para efeito didático, é comum o uso de representações por meio de imagens, que são os diagramas de estado de transição. Tais abstrações ilustram os AFs, APs e MTs e são caracterizados por diagramas do tipo nó-link, sendo que uma imagem de um reconhecedor de linguagem pode possuir variados componentes, como mostra a Figura 2.1.

Figura 2.1: Componentes de um diagrama de estados de transição



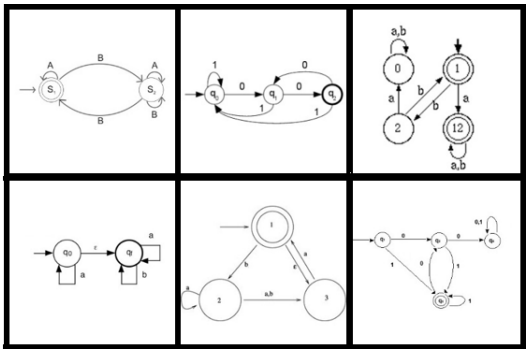
Fonte: (PRÓPRIA, 2019)

Um estado inicial, como mostra a parte (a) da Figura 2.1, é composto por um elemento circular, que representa o estado, e uma flecha. O estado comum é representado apenas por um elemento circular, como mostra a parte (b) da Figura 2.1, e um estado final/aceitação (parte (c)) é representado por 2 círculos concêntricos, sendo um menor que o outro. Todos os estados possuem um rótulo que representa o "nome" daquele estado, facilitando a identificação das transições.

A parte (d) da Figura 2.1 representa uma transição que liga dois estados diferentes, essa transição pode estar em qualquer direção ou ângulo. Um caso particular de uma transição é quando temos um *Loop*, ou seja, uma ligação do estado para ele mesmo, como mostra a parte (e) da Figura 2.1. Todas as transições possuem também uma descrição que representa uma ação a ser tomada pelo reconhecedor.

Na Figura 2.2 são apresentados vários exemplos, retirados da Internet, de reconhecedores de linguagens formais, mostrando algumas das possíveis formas que as transições podem assumir. Observa-se que um diagrama pode utilizar linhas retas, linhas curvas ou ainda os dois tipos em uma mesma imagem.

Figura 2.2: Exemplos de diferentes formatos de transições

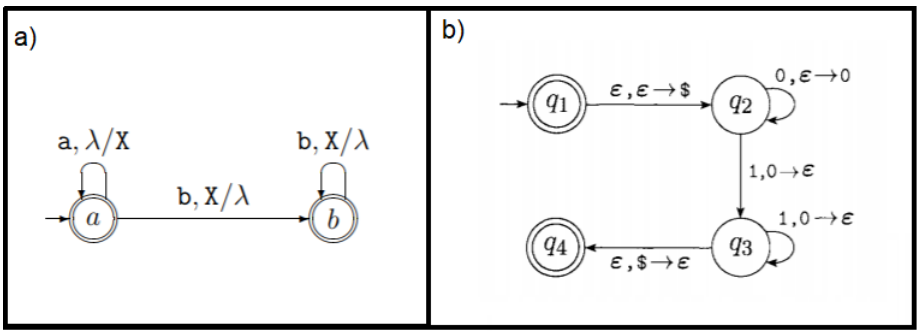


Fonte: (IMAGENS RETIRADAS DA INTERNET)

Outro elemento, que possui variações, é a representação do rótulo da transição. Apesar dessa variação não ser comum para AFs, ela é bastante presente nos APs e nas MTs, pois cada autor ou sistema utiliza uma forma diferente de representar os elementos da descrição de uma transição.

Na Figura 2.3 observa-se a diferença entre dois modelos de transição de Autômatos de Pilha. Na parte (a), o autor Vieira (2006) utiliza o padrão $(a,b/x)$, enquanto o autor Sipser (2006) (parte (b)) utiliza o padrão $(a,b \rightarrow x)$.

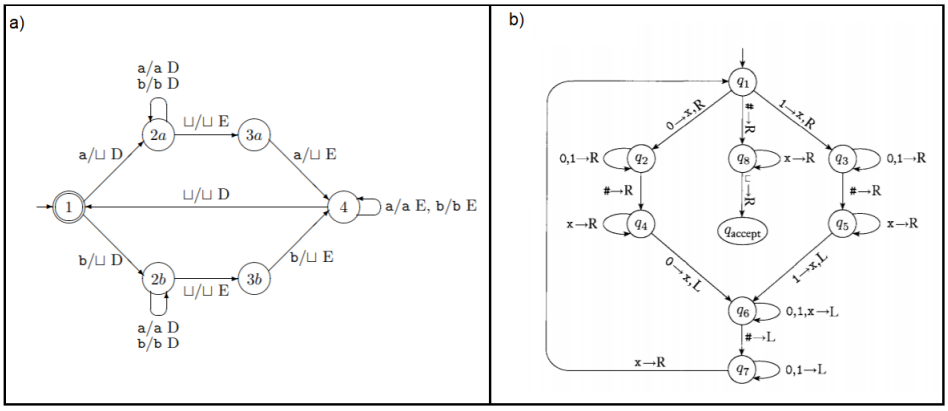
Figura 2.3: Diferença do modelo de transição entre os autores Vieira (2006), na parte (a), e Sipser (2006), na parte (b), para Autômato a pilha



Fonte: (PRÓPRIA, 2019)

Também existem diferenças para a representação de modelos de transição de Máquina de Turing. Na Figura 2.4 é possível observar a diferença entre dois modelos de transição de Máquinas de Turing, na qual na parte (a) o autor Vieira (2006) utiliza o padrão $(a/b\ D)$ e na parte (b) o autor Sipser (2006) utiliza o padrão $(a \to b, D)$.

Figura 2.4: Diferença do modelo de transição entre os autores Vieira (2006), na parte (a), e Sipser (2006), na parte (b), para Máquina de Turing



Fonte: (PRÓPRIA, 2019)

Apesar de existirem alguns padrões na forma como os reconhecedores de linguagens podem ser representados em imagem, têm-se muitas variações, que vão desde o modelo utilizado na transição, até outras variáveis relacionadas ao estilo do desenho, como por exemplo, o formato do *loop*, tipo de flecha e forma da flecha.

2.2 TECNOLOGIAS ASSISTIVAS PARA DEFICIENTES VISUAIS

O conceito de Tecnologia Assistiva no Brasil foi definido pela Secretaria Especial dos Direitos Humanos da Presidência da República por meio do Comitê de Ajudas Técnicas (CAT). Esta definição destaca que a Tecnologia Assistiva é uma área interdisciplinar que engloba qualquer prática relacionada às pessoas com deficiência, visando a autonomia, inclusão social e qualidade de vida desse indivíduo (BRASIL, 2009).

A definição de Tecnologia Assistiva no contexto de deficiência visual é bastante complexa devido aos inúmeros aspectos e pontos de vista que podem ser abordados, sendo que as pesquisas desse tema podem estar relacionadas tanto com "fatores fisiológicos associados com a perda de visão, quanto aos fatores psicológicos e humanos influenciando orientação, mobilidade e acesso à informação" (BHOWMICK; HAZARIKA, 2017, p. 152, tradução própria).

De acordo com Bhowmick e Hazarika (2017), as tecnologias assistivas relacionadas a deficientes visuais geralmente se concentram em mobilidade e reconhecimento de objetos, sendo que a segunda é o foco deste trabalho. Na última década houve um aumento do interesse da comunidade acadêmica por temas relacionados, como por exemplo: bengalas inteligentes (RAMIREZ et al., 2017; VELÁZQUEZ et al., 2018), dispositivos *wearable* (FENG, 2016; TAPU et al., 2018), aplicativos acessíveis para *smartphone* (CUTTER; MANDUCHI, 2017; MOHAMAD et al., 2018), olhos biônicos, também denominados implantes corticais (LUO; CRUZ, 2014; LEWIS et al., 2015) e *displays* e interfaces táteis (GÖTZELMANN, 2016; GUO et al., 2017).

Uma Tecnologia Assistiva, muitas vezes, é um software dedicado apenas a uma parcela de usuários que necessitam de um auxílio na realização de alguma tarefa. Por outro lado, existe o software que se adapta e atende a todas as pessoas, ou a maioria possível. Esse software, além de acessível, segue os princípios do Design Universal, que busca a inclusão e a não segregação do indivíduo com necessidades especiais.

O desenvolvimento de softwares com acessibilidade é norteado por guias e padrões de acessibilidades. Muitos desses guias são destinados principalmente para páginas Web, contudo, essas diretrizes podem ser expandidas para demais softwares.

A World Wide Web Consortion (W3C), principal organização de padrões Web, criou em 1999 o Web Accessibility Initiative (WAI). Esse grupo de pesquisa criou um conjunto de documentos que passaram a nortear o desenvolvimento acessível, sendo que o Web

Content Accessibility Guidelines (WCAG)[1] é o que se relaciona com este trabalho. O W3C recomenda que as políticas de acessibilidade na Web sejam baseadas nas versões mais novas dos documentos, sendo que a versão mais atual do WCAG é a 2.0.

Ressalta-se ainda que as avaliações são uma parte importante do processo de acessibilidade. O W3C sugere que se realize uma validação por meio de ferramentas automáticas. Para a organização é indispensável que ocorra uma validação manual para tentar encontrar problemas a partir do julgamento humano, sendo que o próprio W3C fornece uma lista de pontos a serem verificados manualmente. Também recomendam-se testes com usuários reais que poderão dizer se o site está realmente acessível ou apenas tecnicamente acessível.

Outro ponto a ser destacado é a usabilidade do sistema. Esta é um fator importante no desenvolvimento com acessibilidade, pois por meio dela é possível avaliar quão fácil é a realização de tarefas no sistema. Para avaliar este ponto serão utilizadas as Heurísticas de Usabilidade de Nielsen (NIELSEN, 1994), que possuem um foco maior no usuário do que nos métodos de desenvolvimento empregados.

Os documentos da W3C, as Heurísticas de Nielsen e os princípios do Design Universal são detalhados nas próximas seções.

2.2.1 Guias de acessibilidade

O WCAG 2.0 (WORLD WIDE WEB CONSORTIUM, 2008) é um documento composto por 12 recomendações divididas em quatro princípios. Cada recomendação possui um ou mais *checkpoints* a serem atendidos. Os quatro princípios são:

1. Perceptível: 1.1 - fornecer alternativas textuais para qualquer conteúdo não textual, para que possa ser transformado em outras formas de acordo com as necessidades dos usuários, tais como impressão com tamanho de fontes maiores, braille, fala, símbolos ou linguagem mais simples; 1.2 - fornecer alternativas para mídias baseadas em tempo; 1.3 - criar conteúdo que pode ser apresentado de diferentes maneiras (por exemplo, um layout simplificado) sem perder informação ou estrutura; e 1.4 - facilitar a audição e a visualização de conteúdo aos usuários, incluindo a separação entre o primeiro plano e o plano de fundo.

2. Operável: 2.1 - fazer com que toda funcionalidade fique disponível a partir de um teclado; 2.2 - fornecer aos usuários tempo suficiente para ler e utilizar o conteúdo; 2.3 - não criar

[1] https://www.w3.org/WAI/standards-guidelines/wcag/

conteúdo de uma forma conhecida por causar convulsões; e 2.4 - fornecer maneiras de ajudar os usuários a navegar, localizar conteúdos e determinar onde se encontram.

3. Compreensível: 3.1 - tornar o conteúdo de texto legível e compreensível; 3.2 - fazer com que as páginas Web apareçam e funcionem de modo previsível; e 3.3 - ajudar os usuários a evitar e corrigir erros.

4. Robusto: 4.1 - maximizar a compatibilidade entre os atuais e futuros agentes de usuário, incluindo tecnologias assistivas.

Segundo a World Wide Web Consortium (2008), seguir essas diretrizes tornarão o conteúdo acessível a uma gama mais ampla de pessoas com deficiência, além de também tornar o conteúdo da Web mais útil para usuários em geral. Para estar em conformidade com os critérios da WCAG 2.0, todos os *checkpoints* devem ser cumpridos. Existem três níveis de conformidade A, AA, AAA. Para a página Web ser nível A, todos os *checkpoints* do nível A devem ser atendidos, para ser nível AA, todos os do nível A e do nível AA devem ser atendidos e assim por diante.

Estas diretrizes serão utilizadas em três momentos neste trabalho, a saber: na validação automática das páginas HTML (*Hypertext Markup Language*), do protótipo desenvolvido, na validação manual realizada pelos próprios desenvolvedores e na validação realizada por deficientes visuais.

2.2.2 Heurísticas de usabilidade de Nielsen

Segundo Nielsen (1994), a usabilidade é composta por diferentes componentes e é tradicionalmente associada com cinco fatores:

- aprendizagem: o sistema deve ser de fácil aprendizado;
- eficiência: após o usuário aprender a utilizar o sistema, existe a possibilidade de um alto nível de produtividade;
- memória: as funcionalidades do sistema devem ser fáceis de lembrar. Assim, um usuário, que fica um tempo sem acessar o sistema, deve ser capaz de utilizá-lo sem precisar reapreender;
- erros: a taxa de erro do sistema deve ser baixa;
- satisfação: o sistema deve ser prazeroso de ser utilizado.

Baseado nesses itens, Nielsen definiu 10 heurísticas que são utilizadas como métricas para avaliação de acessibilidade.

- visibilidade do *status* do sistema: o usuário deve sempre ter um *feedback* adequado sobre o que está acontecendo no sistema, em um tempo hábil;
- correspondência entre sistema e mundo real: o sistema deve possuir uma linguagem familiar ao usuário, fazendo com que as informações surjam de maneira lógica e natural;
- controle do usuário e liberdade: o sistema deve prover meios de desfazer e refazer ações errôneas do usuário de maneira rápida;
- consistência e padronização: o sistema deve ser conciso e padronizado no que tange à apresentação das informações, na linguagem utilizada e nas diferentes formas de interação;
- prevenção de erros: o sistema deve realizar o máximo esforço para prevenir a ocorrência de um erro. No entanto, caso não seja possível evitá-lo, deve-se prover mensagens de erro da maneira mais clara e informativa possível;
- reconhecer ao invés de memorizar: o sistema deve prover instruções de uso de maneira fácil e recuperável, de modo que o usuário, ao usar diferentes módulos, realize as ações pela familiaridade com a interface.
- flexibilidade e eficiência de uso: o sistema deve ser customizável para diferentes perfis de usuários, desde os iniciantes até os mais experientes.
- *design* minimalista: os componentes do sistema devem possuir apenas as informações necessárias para a interação. Deve-se primar pela máxima "menos é mais".
- ajuda aos usuários para reconhecimento, diagnóstico e recuperação de erros: as mensagens de erro devem ser sempre expressas na linguagem natural do usuário (nunca por códigos técnicos), indicando precisamente o problema e fornecendo maneiras de solucioná-lo.
- ajuda e documentação: O sistema deve possuir, quando necessário, um manual de ajuda para as interações, que possa ser obtido de maneira rápida e simples.

Segundo Nielsen (1994), a avaliação da usabilidade deve ter foco no usuário, pois dizer que uma tecnologia possui acessibilidade é relativa às preferências e categorias de cada usuário. Assim, pretende-se utilizá-las como parte dos testes com usuários deficientes visuais.

2.2.3 Design Universal

O Centro de Habitação Acessível (*Center for Accessible Housing*), que fica na Universidade da Carolina do Norte, desenvolveu os Princípios do Design Universal (THE CENTER OF UNIVERSAL DESING, 1997) que buscam nortear o desenvolvimento de ambientes e produtos para serem utilizados por todas as pessoas, ou pela maior parte possível, sem a necessidade de adaptação ou de design especializado. No Quadro 2.1 a seguir são apresentados os sete princípios e suas definições.

Quadro 2.1: Princípios do Design Universal

Princípio	**Definição**
Uso Equitativo	O design é útil e comercializável para pessoas com habilidades diversas
Uso Flexível	O design acomoda uma ampla gama de preferências e habilidades individuais
Uso Simples e Intuitivo	O uso do design é fácil de entender, independentemente da experiência, conhecimento, habilidades linguísticas ou nível de concentração atual do usuário
Informação Perceptível	O design comunica efetivamente a informação necessária ao usuário, independentemente das condições ambientais ou das habilidades sensoriais do usuário
Tolerância ao Erro	O design minimiza o risco e as consequências adversas de ações acidentais ou não intencionais
Baixo Esforço Físico	O design pode ser usado de forma eficiente e confortável e com um mínimo de fadiga
Tamanho e Espaço de Aproximação e Uso	O tamanho e os espaços apropriados são fornecidos para aproximação, alcance, manipulação e uso, independentemente do tamanho, postura ou mobilidade do usuário

Fonte: (THE CENTER OF UNIVERSAL DESIGN, 1997, p. 107-108, tradução própria)

Os sete princípios podem ser aplicados para avaliar os projetos existentes, orientar o processo de design e ensinar sobre as características de produtos e ambientes mais utilizáveis (THE CENTER OF UNIVERSAL DESING, 1997).

Inicialmente, o Design Universal foi elaborado para tornar acessíveis ambientes físicos, porém, os princípios vêm sendo utilizados em outras áreas como na elaboração de currículos de educação geral (HITCHCOCK; STAHL, 2003) e no desenvolvimento de software (BALIK et al., 2013). Pretende-se aplicar esses princípios no desenvolvimento deste trabalho, permitindo que o projeto seja útil a uma gama maior de pessoas além dos deficiente visuais.

2.3 SISTEMAS DE VISÃO COMPUTACIONAL

Um sistema de visão computacional executa tarefas de processamento de informações, na qual a entrada consiste em uma imagem, que representa projeções de uma cena, e a saída é uma descrição concisa da mesma retratada. Tal processo geralmente envolve a descrição dos objetos e suas inter-relações, mas também pode incluir informações sobre outras características físicas, como forma, textura, cor e material (BARROW; TENENBAUM, 1981).

De acordo com Szeliski (2010), "como humanos, percebemos a estrutura tridimensional do mundo à nossa volta com aparente facilidade". Na visão computacional tenta-se fazer o inverso, ou seja, realizar a descrição de uma imagem reconstruindo suas propriedades, como forma, iluminação e cor. Enquanto os algoritmos de visão computacional são tão propensos a erros, é notável que humanos e animais façam isso sem esforço (SZELISKI, 2010).

Um dos ramos das aplicações de sistemas de visão computacional são as tecnologias que auxiliam pessoas com deficiência visual. Entre elas, podem-se citar alguns temas, como sistemas que ajudam na mobilidade (SOUSA; MARENGONI, 2012; GREWE et al., 2018), os que ajudam na leitura de textos impressos (SHILKROT et al., 2018; GOEL et al., 2018), na inclusão das pessoas com deficiência visual no ambiente acadêmico (SILVA et al., 2015; REIS et al., 2018) e na descrição de imagens para deficientes visuais (BABALOLA, 2015), sendo o último o foco deste trabalho.

Nas próximas seções são abordados os conceitos básicos de processamento digital de imagens relacionados a sistemas de visão computacional. Além disso, são apresentados alguns métodos de detecção de círculos em imagens, a saber: *Hough* para círculos, *Learning Automata* e *Haar Cascade*.

Também são apresentados métodos de classificação de propriedades extraídas da imagem, a saber: CNN para identificar o tipo do diagrama de estados de transição (AF, AP e MT) e quantos estados estão presentes na imagem; e descritores de formas regionais utilizados neste trabalho na classificação de flechas, *loops*, caracteres e ruídos.

2.3.1 Conceitos básicos de processamento digital de imagens

Uma imagem digital é composta por um conjunto de *pixels*, cada um deles é representado por uma cor, podendo ser uma imagem binária, em escala de cinza ou colorida. De acordo com Gonzalez e Woods (2007), uma imagem pode ser definida como uma função

bidimensional $f(x,y)$, na qual x e y são coordenadas espaciais, e a amplitude de f em qualquer parte das coordenadas (x,y) é chamada de intensidade do nível de cinza.

Os pontos ao redor de um *pixel* p, com coordenadas (x,y), são denominados vizinhos de p. Os 4 *pixels* horizontais e verticais dados pelas coordenadas na Equação 1 são denominados 4-vizinhança e é denotado por $N_4(p)$.

$$(x+1,y),(x-1,y),(x,y+1),(x,y-1) \tag{1}$$

Considerando também os *pixels* localizados na diagonal de p mais os pontos $N_4(p)$, temos 8 *pixels* que formam a 8-vizinhança denotado por $N_8(p)$. As coordenadas dos *pixels* das diagonais são apresentados na Equação 2.

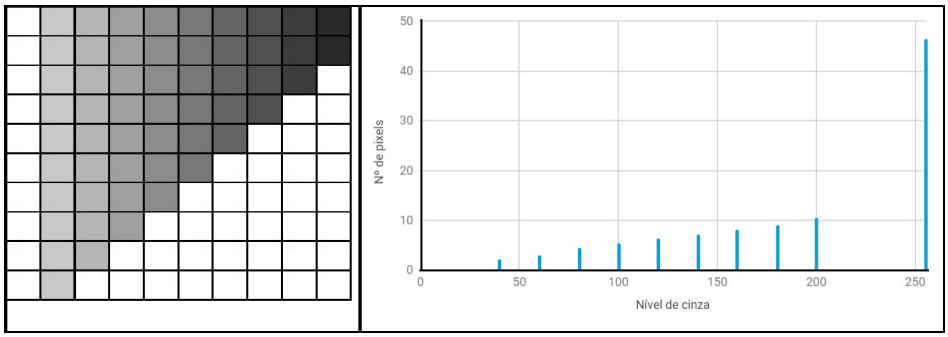
$$(x+1,y+1),(x+1,y-1),(x-1,y+1),(x-1,y-1) \tag{2}$$

Na sequência, são apresentadas algumas técnicas que podem ser realizadas no processo de manipulação de imagens, a saber: histograma, *thresholding*, Método Canny e a base da morfologia matemática.

2.3.1.1 Histograma

Um histograma de uma imagem digital em níveis de cinza representa a frequência com que cada cor aparece na imagem. Considerando uma imagem I, na qual os níveis de cinza vão de $[0,L-1]$, sendo L o maior nível de cinza disponível, teremos uma função discreta $h(r_k) = n_k$, na qual r_k é o k-ésimo nível e n_k representa o número de *pixels* na imagem com a cor r_k (GONZALEZ; WOODS, 2007). Na Figura 2.5 é apresentado um exemplo de Histograma.

Figura 2.5: Imagem em níveis de cinza, na qual cada quadrado representa um *pixel* e seu histograma correspondente



Fonte: (PRÓPRIA, 2019)

Na Figura 2.5 têm-se a representação de uma imagem de tamanho 10×10, na qual cada quadrado representa um *pixel*. Também é apresentado o histograma de níveis de cinza correspondente. É possível perceber que há vários picos no histograma que vão aumentando de tamanho correspondendo ao degradê contido na imagem.

2.3.1.2 *Thresholding*

Thresholding é uma técnica utilizada para particionar a imagem em regiões baseadas nos valores de intensidade de cor de cada *pixel* (GONZALEZ; WOODS, 2007). Para uma imagem em escala de cinza é possível realizar um *thresholding* binário, que a transformará em preta e branca, a partir de um limiar T. Considera-se como ponto de objeto qualquer ponto (x,y) da imagem, tal que $f(x,y) > T$ e como pontos de fundo os que forem menores ou iguais a T. O processo de *thresholding* binário é dado pela Equação 3.

$$g(x,y) = \begin{cases} 1 & \text{se } f(x,y) > T \\ 0 & \text{se } f(x,y) \leq T \end{cases} \tag{3}$$

Descobrir o melhor valor para o limiar T é um problema que pode ser visto como uma decisão estatística. Uma boa solução para encontrar o limiar é o método Otsu (GONZALEZ; WOODS, 2007), proposto por Nobuyuki Otsu, que recebe uma imagem em tons de cinza e encontra o valor ideal para o *threshold* da imagem, buscando o valor que minimiza a soma das variâncias intraclasses (OTSU, 1979). Esse limiar ótimo pode ser obtido por meio da Equação 4.

$$\eta = \frac{\sigma_b^2}{\sigma_t^2} \tag{4}$$

na qual:

- σ_b^2 é a variância entre as classes;
- σ_t^2 é a variância total.

A primeira etapa é a construção do histograma. Como resultado para cada nível de cinza tem-se a quantidade encontrada de *pixels* com aquele tom na figura. Para cada resultado da etapa anterior é realizada a divisão do valor pelo número total de *pixels* obtendo a porcentagem de cada um deles na imagem.

Considerando o histograma dividido em 2 partes, classes A e B, na qual, a primeira classe corresponde ao intervalo de $[0,t]$ e a segunda de $[t+1, L-1]$, sendo L o número máximo

de tons de cinza. Portanto, é possível calcular a probabilidade de cada nível de cinza ser de cada classe, por meio das Equações 5 para a classe A e 6 para a classe B.

$$\omega_0 = \sum_{i=0}^{t} P_i \tag{5}$$

$$\omega_1 = \sum_{t+1}^{L-1} P_i \text{ ou } \omega_1 = 1 - \omega_0 \tag{6}$$

Para calcular as variâncias também são utilizadas as médias de cada classe (Equação 9 para a classe A e 10 para a classe B) que são dadas pelas equações:

$$\mu_t = \sum_{i=0}^{t} iP_i \tag{7}$$

$$\mu T = \sum_{i=t+1}^{L-1} iP_i \tag{8}$$

$$\mu_0 = \frac{\mu_t}{\omega_0} \tag{9}$$

$$\mu_1 = \frac{\mu T - \mu_t}{\omega_1} \tag{10}$$

Com as probabilidades e médias das classes já calculadas, a variância total (Equação 11) e a variância entre as classes (Equação 12) podem ser obtidas pelas equações:

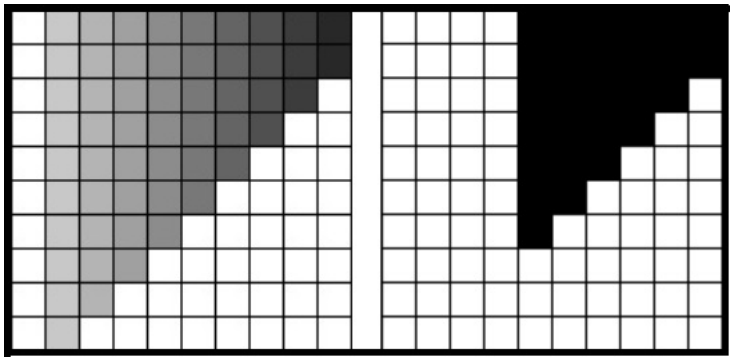
$$\sigma_t^2 = \sum_{i=0}^{L-1} (i - \mu T)^2 P_i \tag{11}$$

$$\sigma_b^2 = \omega_0 \omega_1 (\mu_0 - \mu_1)^2 \tag{12}$$

Na sequência, é possível calcular η, já apresentado anteriormente (Equação 4). Como essas funções são calculadas várias vezes alterando o valor de t, que é o limiar testado, ao final o maior η será o limiar que deverá ser utilizado no *threshold*.

Na Figura 2.6 apresenta-se um exemplo do *threshold* binário utilizando o método de Otsu.

Figura 2.6: Resultado da aplicação do *threshold* Otsu, na qual na primeira parte tem-se a imagem original e na segunda o resultado do método



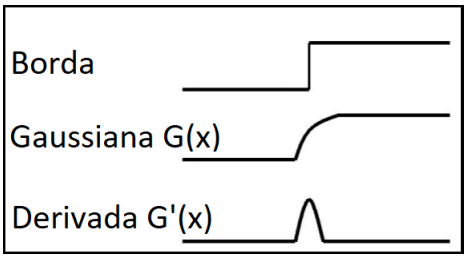
Fonte: (PRÓPRIA, 2019)

2.3.1.3 Método Canny

O método de detecção de bordas Canny foi desenvolvido em 1986 por John F. Canny (CANNY, 1986). Esse consiste em realizar uma convolução da imagem pela primeira derivada de um filtro Gaussiano, que possui o efeito de suavizar a imagem.

Na Figura 2.7 tem-se a diferença dos processos de convolução utilizando uma função gaussiana e utilizando a primeira derivada dessa função sobre uma borda. Quando se utiliza apenas a função, o resultado é uma variação contínua entre o valor inicial e final da borda original. Já utilizando a derivada, o resultado mostra que o máximo da nova função corresponde ao máximo da função original.

Figura 2.7: Exemplo de utilização da função gaussiana $G(x)$ e de sua derivada $G'(x)$ sobre uma borda



Fonte: (PRÓPRIA,2019)

As bordas da imagem são indicadas pelos pontos máximos resultantes da convolução. Esse processo pode ser realizado por meio de uma função gaussiana de duas dimensões ou de uma dimensão só para x e depois o mesmo processo para y. A função Gaussiana de uma dimensão e sua derivada são apresentadas na sequência.

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}} \tag{13}$$

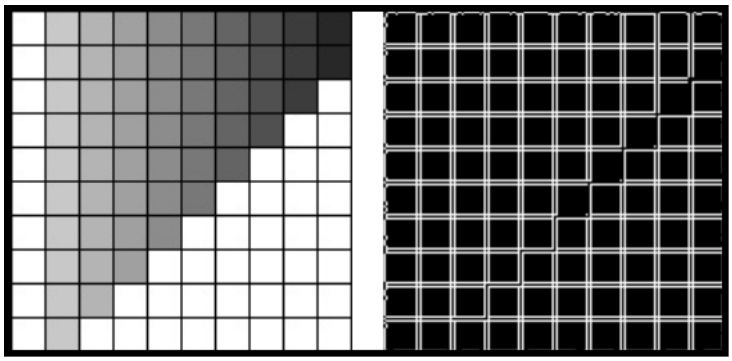
$$G'(x) = \frac{-x}{\sqrt{2\pi\sigma^3}} e^{\frac{-x^2}{2\sigma^2}} \tag{14}$$

O algoritmo de Canny é multi-passos e tem como resultado as bordas da imagem. O primeiro passo é a redução de ruído. A redução é realizada convoluindo a imagem original por um filtro Gaussiano. A imagem de saída possuirá uma suavização em relação a original. Quanto maior o filtro gaussiano utilizado, mais a imagem será suavizada.

O segundo passo é calcular os gradientes de intensidade da imagem. Para isso são utilizados 4 filtros, um para detectar bordas horizontais, outro para verticais e dois para bordas diagonais (45º e 135º). Na sequência, para cada *pixel* verifica-se o resultado das convoluções escolhendo o de maior intensidade e a direção que ele representa.

Assim, os gradientes mais altos possuem uma maior probabilidade de serem bordas. Para elas serem identificadas é utilizado um *threshold* com dois limiares, um superior e um inferior. Na Figura 2.8, exibida a seguir, mostra-se o resultado da aplicação do método para a figura do exemplo.

Figura 2.8: Exemplo de aplicação do método Canny, na qual na primeira parte da figura temos a imagem original e na segunda o resultado do método



Fonte: (PRÓPRIA, 2019)

2.3.1.4 Morfologia Matemática

A morfologia matemática é o estudo da extração de componentes de uma imagem que são úteis em sua representação e descrição, como forma, bordas, esqueletos, entre outros (GONZALEZ; WOODS, 2007). Há duas operações fundamentais para o processamento de morfologia, a saber: a dilatação e a erosão.

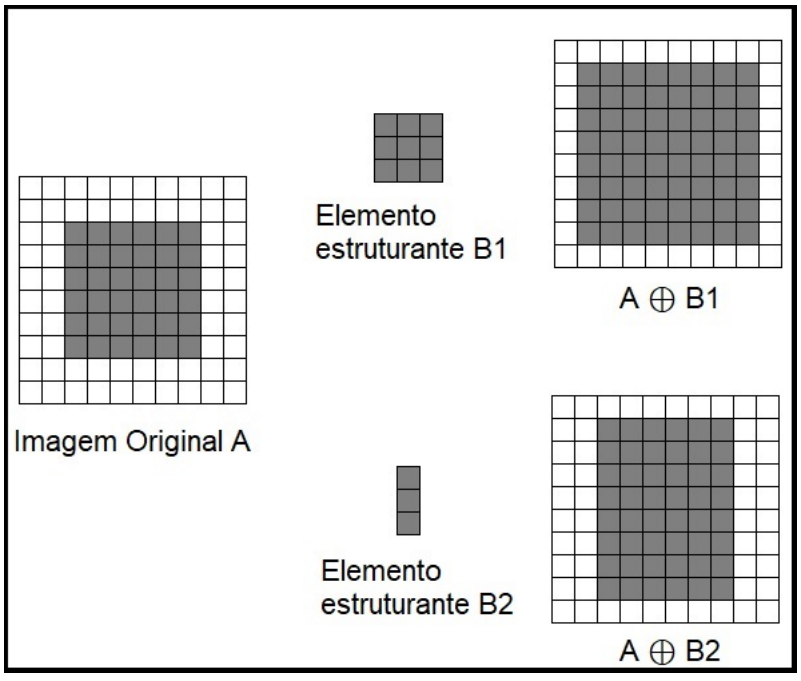
A dilatação da imagem A pelo elemento estruturante B é definida pela Equação 15:

$$A \oplus B = \left\{ z \mid (\widehat{B})_z \cap A \neq \varnothing \right\} \tag{15}$$

A Equação 15 indica que $(\widehat{B})_z$ deve ser posicionado e centrado no *pixel* z para verificar se existe alguma interseção com A. Se houver a interseção, este *pixel* z é considerado relevante e ele será atribuído como objeto na imagem de saída. O mesmo processo deve ser repetido para todos os *pixels* da imagem.

A dilatação é uma operação que "aumenta" ou "engrossa" objetos em uma imagem binária. Esse processo pode preencher buracos e/ou conectar objetos próximos. O formato do elemento estruturante é importante para determinar a nova forma do objeto (GONZALEZ; WOODS, 2007). Na Figura 2.9 é apresentada a diferença de dilatação da mesma imagem utilizando dois elementos estruturantes diferentes.

Figura 2.9: Exemplo de dilatação utilizando dois elementos estruturantes diferentes



Fonte: (PRÓPRIA, 2019)

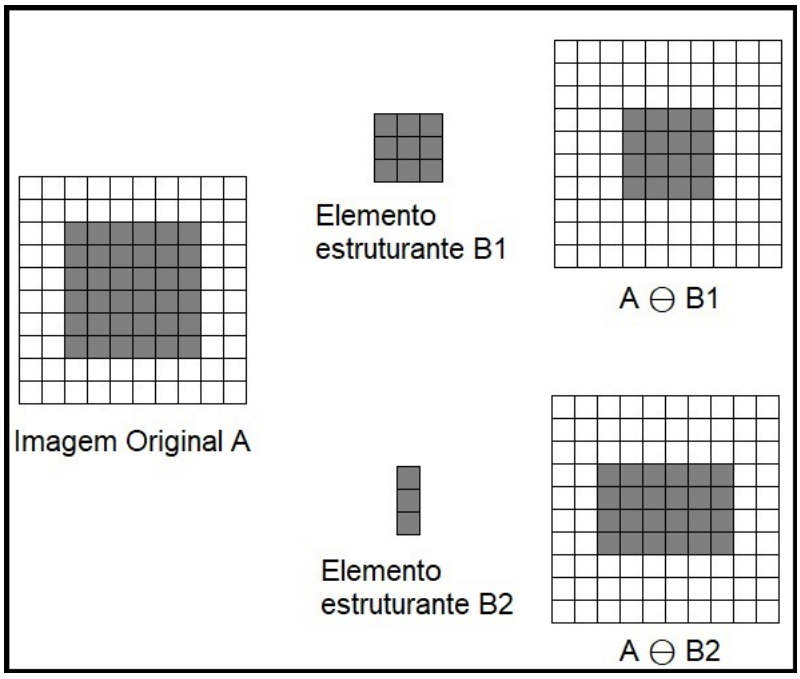
Já a operação de erosão é o contrário da dilatação, pois "diminui" ou "afina" a imagem. A erosão da imagem A pelo elemento estruturante B é definida pela Equação 16:

$$A \ominus B = \left\{ z \mid (\widehat{B})_z \subseteq A \right\} \tag{16}$$

A Equação 16 indica que $(\widehat{B})_z$ deve ser posicionado e centrado no *pixel* z e todo o elemento estruturante B deve estar contido na imagem A. Se estiver contido, então o *pixel* z é considerado relevante e ele será atribuído como objeto na imagem de saída. Da mesma forma, este processo deve ser realizado para todos os *pixels* da imagem.

Na erosão novamente tem-se que o formato do elemento estruturante afeta a imagem resultante. Na Figura 2.10 é mostrada a diferença de erosão da mesma imagem utilizando dois elementos estruturantes diferentes.

Figura 2.10: Exemplo de erosão utilizando dois elementos estruturantes diferentes



Fonte: (PRÓPRIA, 2019)

2.3.2 Detecção de círculos

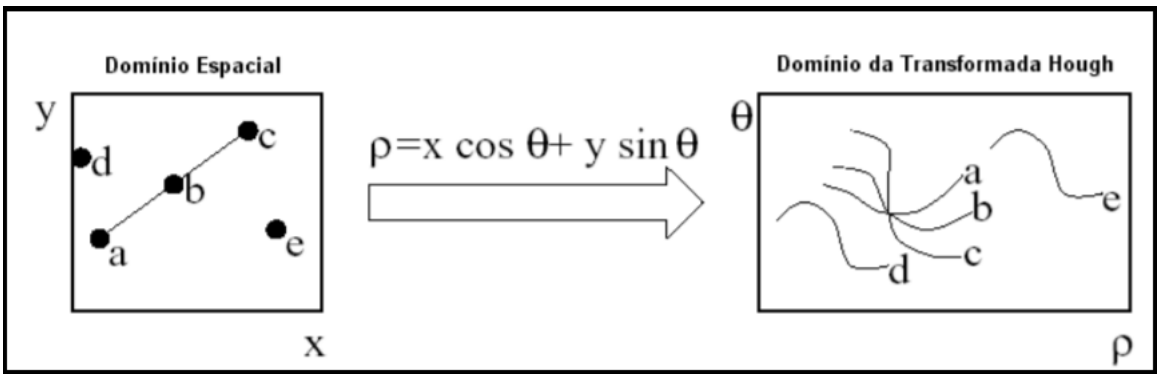
Segundo Costa e Jr (2010) a detecção de forma é necessária em muitas tarefas de Visão Computacional. Neste trabalho considera-se o reconhecimento do círculo o cerne do MannAR e esse processo foi considerado preliminar na construção do método.

Foram realizadas análises de três abordagens. A primeira foi a transformada de *Hough*, por ser um método de análise de formas há muito tempo reconhecido pela comunidade acadêmica. O segundo foi a técnica *Learning Automata*, por propor um método sem se basear na transformada de *Hough*, oferecendo uma perspectiva diferente para o trabalho. O terceiro foi o algoritmo *Haar Cascade*, por ser uma abordagem baseada em aprendizagem de máquina. Cada uma delas é detalhada nas próximas seções.

2.3.2.1 Transformada de Hough para círculos

A transformada de Hough foi introduzida, pela primeira vez, como um método de detecção de padrões entre pontos em dados de imagens binárias (HOUGH, 1962). Ela propõe que, ao representar uma imagem do espaço digital (x,y) na forma de parâmetros descritos pela curva, é possível encontrar a forma desejada da mesma. A ideia é que a transformação apresente o mapeamento em um único ponto de todos os pontos pertencentes a uma mesma curva, como mostra a Figura 2.11.

Figura 2.11: Exemplo de mapeamento de uma linha existente no domínio espacial para o domínio da Transformada de Hough



Fonte: (MACEDO, 2005)

O uso da Transformada de Hough para detectar círculos foi inicialmente apresentado por DudaR e Hart (1972). Um círculo pode ser parametrizado por suas coordenadas centrais (x_0, y_0) e seu raio, e tais valores estão diretamente relacionadas à posição dos pontos de borda (x,y), que formam o círculo através da Equação 17.

$$(x-x_0)^2 + (y-y_0)^2 = r^2 \tag{17}$$

Para determinar os possíveis círculos na imagem utiliza-se uma matriz acumuladora tridimensional contendo os parâmetros: x_0, y_0 *e* r. Inicialmente é determinado o intervalo dos tamanhos de raios a serem calculados, ou seja, qual o maior e o menor círculo que serão procurados. Assim, para cada ponto na imagem será calculada a coordenada central do círculo. O cálculo é realizado para todos os raios no intervalo considerado.

Cada centro encontrado associado a um raio possui sua posição, representada discretamente na matriz acumuladora, incrementada em uma unidade. Ao final, buscam-se os maiores valores na matriz para representar os círculos (YUEN et al., 1990).

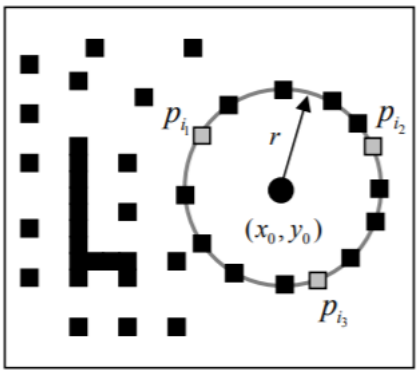
2.3.2.2 Técnica *Learning Automata*

A técnica *Learning Automata* para detecção de círculos foi apresentado por Cuevas et al. (2013) e se propõe a detectar círculos em imagens ruidosas sem utilizar princípios da transformada de Hough. Esse algoritmo fundamenta-se em *Learning Automata* (LA), que "é um método de otimização probabilística que explora um ambiente aleatório desconhecido, melhorando progressivamente o desempenho, por meio de um sinal de reforço (função objetivo)" (CUEVAS et al., 2013, p. 1, tradução própria).

O primeiro passo dessa técnica é utilizar o método Canny para descobrir as bordas da imagem de entrada. Cada *pixel* da borda é adicionado ao vetor de *pixels* $P_t = \left\{p_1, p_2, p_3, ..., p_n\right\}$, na qual n é a quantidade de *pixels* de borda na imagem. Segundo o autor, apenas 5% dos *pixels* de borda do vetor P_t devem ser selecionados e isso pode ser feito aleatoriamente, formando assim o novo vetor $P = \left\{p_1, p_2, p_3, ..., p_n\right\}$, no qual n representa a quantidade de *pixels* selecionados.

Na próxima etapa é criada uma lista de círculos candidatos. Cada um deles será representado por três pontos do vetor P, ou seja, $C_i = (p_{i1}, p_{i2}, p_{i3})$, ao realizar esse agrupamento, pressupõe-se que os pontos fazem parte do mesmo círculo. Na Figura 2.12 é possível observar um círculo candidato com os pontos p_{i1}, p_{i2} *e* p_{i3}.

Figura 2.12: Círculo candidato formado pelo agrupamento dos pontos p_{i1}, p_{i2} *e* p_{i3}



Fonte: (CUEVAS et al., 2013)

Como vimos na Subseção 2.3.2.1, cada ponto (x,y) pertencente a um círculo é caracterizado pela Equação 17, ou seja, podemos definir um círculo pelo seu ponto central e pelo seu raio. Porém, como nesse caso temos 3 pontos de borda os *pixels* centrais, x_0 *e* y_0 podem ser definidos pelas Equações 18 e 19, respectivamente.

$$x_0 = \frac{det(A)}{4((x_{i2}-x_{i1})(y_{i3}-y_{i1})-(x_{i3}-x_{i1})(y_{i2}-y_{i1}))} \tag{18}$$

$$y_0 = \frac{det(B)}{4((x_{i2}-x_{i1})(y_{i3}-y_{i1})-(x_{i3}-x_{i1})(y_{i2}-y_{i1}))}, \tag{19}$$

Sendo que os determinantes de A e B são definidos pelas matrizes:

$$A = \begin{bmatrix} x_{i2}^2+y_{i2}^2-(x_{i1}^2+y_{i1}^2) & 2(y_{i2}-y_{i1}) \\ x_{i3}^2+y_{i3}^2-(x_{i1}^2+y_{i1}^2) & 2(y_{i3}-y_{i1}) \end{bmatrix} \tag{20}$$

$$B = \begin{bmatrix} 2(x_{i2}-x_{i1}) & x_{i2}^2+y_{i2}^2-(x_{i1}^2+y_{i1}^2) \\ 2(x_{i3}-x_{i1}) & x_{i3}^2+y_{i3}^2-(x_{i1}^2+y_{i1}^2) \end{bmatrix} \tag{21}$$

O raio do círculo candidato é calculado pela equação:

$$r = \sqrt{(x_0-x_d)^2+(y_0-y_d)^2} \tag{22}$$

Os valores x_d e y_d podem ser qualquer um dos pontos $p_{i1},\ p_{i2}\ ou\ p_{i3}$. Todas as possíveis combinações de pontos do vetor P serão utilizadas para a criação dos possíveis círculos.

Com a lista de círculos candidatos pronta, é necessária a verificação de quais círculos realmente existem na imagem original. Para isso, é calculado um sinal de reforço $\beta(C_i)$ que representa o erro correspondente entre os *pixels* do círculo candidato C_i e os *pixels* de borda que existem na imagem original S_i. O sinal de reforço é calculado de acordo com a equação:

$$\beta(C_i) = \frac{\sum_{N_s}^{h=1} E(S_h)}{N_s} \tag{23}$$

Na qual, N_s é o número de *pixels* que existem no perímetro de C_i. Já $E(S_h)$ é uma função que verifica se o *pixel* S_h, sendo $S_h \in S_i$, é um *pixel* de borda existente na imagem original. A função $E(S_h)$ é definida como:

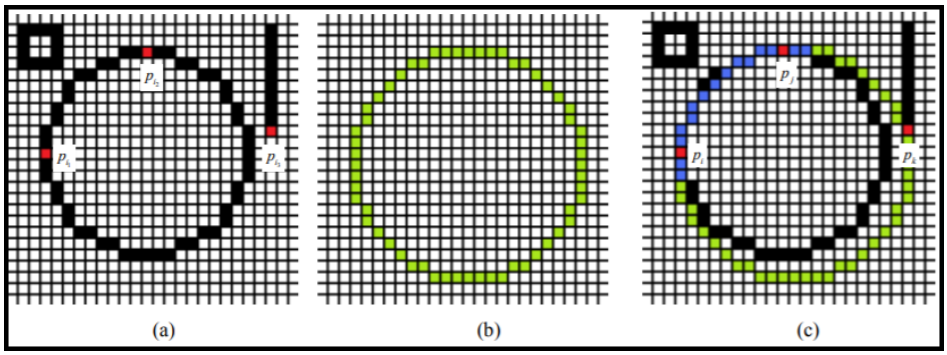
$$E(S_h) = \begin{cases} 1 & \text{se o } pixel\ S_h \text{ é um } pixel \text{ de borda} \\ 0 & \text{caso contrário} \end{cases} \tag{24}$$

Na Figura 2.13 tem-se um exemplo do cálculo do sinal de reforço. Na parte (a) da figura apresenta-se a seleção dos três pontos aleatórios na imagem. Na parte (b) observa-se um

círculo criado, utilizando os pontos selecionados em (a) e na parte (c) mostra-se a sobreposição dos dois círculos, sendo o original na cor preta e o círculo candidatado na cor verde.

Ao realizar o cálculo do sinal de reforço é observado que apenas a parte em azul resultaria em saída 1 na função $E(S_h)$ e a soma de todos os resultados de $E(S_h)$ divididos pela quantidade total de *pixels* testados N_s corresponde ao resultado de $\beta(C_i)$.

Figura 2.13: Exemplo do processo de cálculo de sinal de reforço



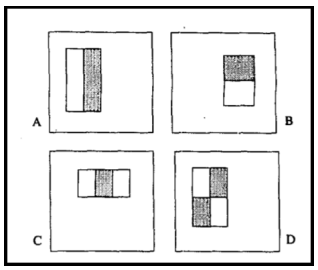
Fonte: (CUEVAS et al., 2013)

De forma geral, o processo consiste em pegar um círculo C_i da lista de círculo candidatos, descobrir o valor de seu sinal de reforço $\beta(C_i)$, armazenar e ir para o próximo círculo. De acordo com Cuevas et al. (2013), existem duas formas de se obter uma solução. A primeira é que todos os círculos que tiverem o sinal de reforço abaixo do limite pré-estabelecido serão dados como resposta. A segunda é que ao final do processo, a resposta será o círculo candidato com a maior probabilidade de ser um círculo.

2.3.2.3 *Haar Cascade*

O *Haar Cascade* é um algoritmo de busca de objetos em imagens proposto por Viola e Jones (2001). Os autores utilizam máscaras (*Haar Features*) que caracterizam o objeto procurado, e estas máscaras representam amplitudes e direções do objeto. Na Figura 2.14 têm-se alguns exemplos de *features*.

Figura 2.14: Exemplo de *features*



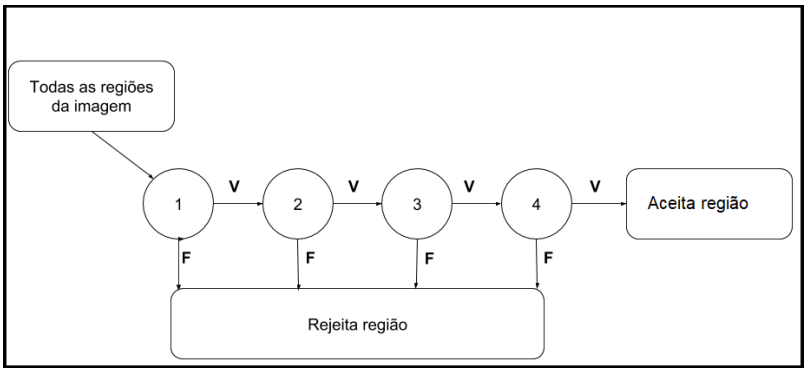
Fonte: (VIOLA; JONES, 2001)

Para encontrar as *features* do objeto procurado (no caso círculos) são necessárias duas classes de imagens, sendo uma de imagens positivas, que possuem o objeto, e uma de imagens negativas, que não possuem o objeto. A partir disso, são geradas as máscaras do objeto e então elas são aprendidas por um algoritmo de aprendizagem de máquina chamado *AdaBoost*. Esse algoritmo gera vários classificadores, sendo um para cada *Haar Feature* gerada anteriormente.

A ideia do algoritmo consiste em dividir a imagem de entrada em regiões com tamanhos predefinidos. Os classificadores gerados anteriormente formam uma lista que recebe as regiões da imagem. Cada região é processada pelo primeiro classificador que fornece como resposta “Falso” ou “Verdadeiro”.

Caso o resultado do classificador seja “Falso”, considera-se que aquela região não possui o objeto procurado e a região é rejeitada. Caso o resultado seja “Verdadeiro”, a região passa para o próximo classificador até passar por todos formando um processo de cascata. Ao passar por todos os classificadores, é considerado que a imagem possui o objeto. Esse processo é representado pelo diagrama da Figura 2.15.

Figura 2.15: Esquema da detecção em cascata



Fonte: (PRÓPRIA, 2019)

Assim, basta que o resultado de um classificador seja falso para uma região da imagem ser descartada. Para ser considerada uma região com o objeto procurado, é necessário passar por todos os classificadores.

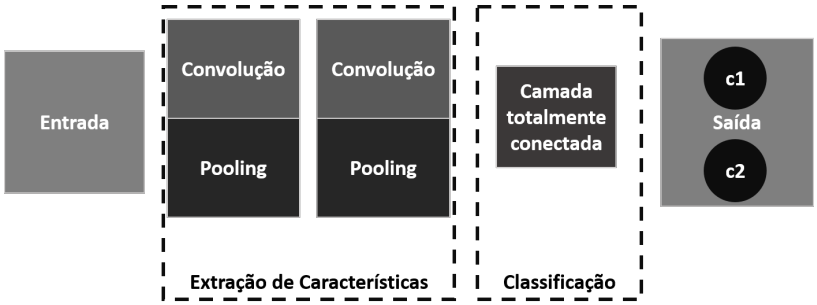
2.3.3 Redes Neurais Convolucionais

As Redes Neurais Convolucionais são uma variação da Rede Neural tradicional, sendo que uma das principais diferenças é que a CNN recebe diretamente imagens. A primeira CNN foi proposta por LeCun et al. (1998) e ficou conhecida como Lenet5. Ao longo dos anos outras redes se tornaram conhecidas como AlexNet (KRIZHEVSKY et al., 2012), VGG (*Very Deep Convolutional Networks*) (SIMONYAN; ZISSERMAN, 2014) e GoogLeNet (SZEGEDY et al., 2015).

Neste trabalho, as CNNs foram utilizadas com uma parte preliminar do método de tradução de imagens de reconhecedores de linguagens formais. Pressupõe-se que saber antecipadamente algumas características da imagem pode ajudar em uma detecção mais precisa da imagem. Duas abordagens foram escolhidas: detectar o tipo do reconhecedor de linguagem (AF, AP ou MT) e detectar quantos estados a imagem possui.

Uma das vantagens de utilizar CNN em comparação com outros métodos de classificação é que além da classificação, a extração de características também é realizada pela rede. Uma CNN é composta por uma sucessão de camadas, sendo que inicialmente têm-se camadas de convolução e na sequência tem-se uma ou mais camadas totalmente conectadas. Na Figura 2.16 é apresentada uma arquitetura típica de uma CNN.

Figura 2.16: Arquitetura típica de uma CNN



Fonte: (PRÓPRIA, 2019)

As camadas de convolução fazem o processo de extração de características locais, posteriormente têm-se as camadas totalmente conectadas que fazem a extração de

características globais (pode ser uma ou mais camadas) e a última camada desse grupo realiza a classificação. Por fim, tem-se a saída da rede. O número de saídas depende da quantidade da classe. Para cada imagem, tem-se como resultado a predição de cada classe, na qual a decisão é tomada de acordo com a melhor previsão. Algumas das camadas são detalhadas na sequência:

- camadas de convolução: nesta camada é realizado o processo de convolução de imagem por um conjunto de filtros. Esse processo consiste em percorrer a imagem com um filtro, que também é conhecido como Kernel, e em cada *pixel* da imagem é calculada a soma de produto ponto a ponto entre a região do tamanho do filtro e o próprio filtro (GONZALEZ; WOODS, 2007). Este processo geralmente necessita de um tratamento de borda na imagem. A versão utilizada neste trabalho é conhecida como *zero-padding* que adiciona uma camada de zeros ao redor das imagens;

- camadas de *pooling*: a proposta desta camada é diminuir a quantidade de parâmetros que devem ser aprendidos pela rede. Neste trabalho, foi utilizado o *max pooling* por ser o mais utilizado. Ele consiste no processo de obter o valor mais alto de uma região $n \times n$, em que n é o número de *pixels*. Os *pixels* selecionados formam uma nova matriz, que é uma representação menor da versão anterior;

- camadas totalmente conectadas: nesta camada, cada saída da etapa anterior é conectada a um neurônio. O erro obtido no final da última camada é propagado para as camadas anteriores, fazendo com que os pesos dos filtros sejam ajustados (STAMFORD; PEACH, 2016). Esse processo, conhecido como *backpropagation*, possibilita o aprendizado da rede.

Ao final de cada camada existe a possibilidade da utilização de uma função de ativação. Uma versão bastante utilizada é a ReLU (*Rectified Linear Unit*). Essa função consiste em mudar os valores negativos para 0 no resultado de cada camada e foi utilizada neste trabalho.

Para a melhoria da performance da rede é comum a utilização de *cross-validation*. Este processo consiste em dividir as imagens do *database* em *folds*. Assim, o treinamento deve ser realizado baseado no número de *folds* existentes. Por exemplo, se houver dez *folds*, são realizados 10 treinamentos da rede, sendo que para cada vez um *fold* diferente será dedicado para teste e os outros nove para o treinamento da rede. Cada execução gera um resultado, e a resposta final é dada pela média dos resultados gerados.

Outro processo realizado para melhoria do desempenho é o *Late Fusion*, que é uma técnica que toma a decisão final com base nas decisões de mais de um classificador (LIU et

al., 2013). Para isso as respostas de cada classificador podem ser combinadas utilizando regras como: soma, produto, máximo ou mínimo. A resposta final será dada pela regra que obtiver um melhor desempenho.

As redes neurais trouxeram um grande avanço para a área de aprendizagem de máquina, pois podem resolver processos não-lineares e complexos encontrando relações entre entradas e saídas que não são facilmente percebidas pelo ser humano.

2.3.4 Descritores de forma regionais

Um objeto em uma imagem pode ser descrito por suas características. Existem duas formas de representar um objeto pela sua região, por meio de suas características externas como a borda ou por meio de características internas que são os *pixels* que compõe a região (GONZALEZ; WOODS, 2007).

Na grande maioria das vezes os descritores regionais internos e externos são utilizados de forma conjunta (GONZALEZ; WOODS, 2007). Assim, um objeto pode ser representado pela sua borda, e por meio da borda pode-se obter algumas características como comprimento, orientação e número de concavidades e também pelos *pixels* que o representam, no qual é possível extrair dados como cor e textura. No Quadro 2.2 apresentam-se algumas características regionais que serão utilizadas neste trabalho.

Quadro 2.2: Descrição das características de objeto utilizadas

Característica	*Descrição*
Área	Quantidade de *pixels* do objeto inscrito em um quadrado
Área do objeto	Quantidade de *pixels* apenas que representam o objeto, no caso apenas os *pixels* pretos
Solidez	Razão entre a área e a área do objeto
Excentricidade	Razão entre o eixo de maior comprimento da área pelo eixo de menor comprimento
Pontos extremos	Localização dos pontos mais a direita, a esquerda, alto, baixo do objeto.

Fonte: Adaptado de (BABALOLA, 2015)

Apesar de existirem inúmeros descritores de características, as características selecionadas são suficientes para distinguir os objetos presentes nos reconhecedores de linguagens.

2.4 CONSIDERAÇÕES FINAIS

Neste Tópico foram apresentados os referencias teóricos deste trabalho. Primeiramente, explanou-se sobre os objetos que compõem as imagens de Autômatos Finitos, Autômatos de Pilha e Máquinas de Turing que são importantes para entendimento da dimensionalidade do problema e quantidade de variáveis envolvidas.

Na sequência, visando a construção de um protótipo acessível foram apresentadas algumas diretrizes de desenvolvimento de Tecnologias Assistivas. Elas também são utilizadas nos testes com usuários deficientes visuais. Outro ponto, que será avaliado, é a usabilidade do sistema por meio das Heurísticas de Nielsen.

Também apresentou-se o conceito do *Desing Universal*, que neste trabalho é utilizado visando a construção de um método mais inclusivo e que possa ter utilidade por uma gama maior de pessoas.

Por fim, apresentaram-se conceitos relacionados a Sistemas de Visão Computacional que abrangem os fundamentos de processamento digital de imagens, a detecção de objetos, como círculos e formas, e as Redes Neurais Convolucionais que auxiliam a descrição das imagens. Todos esses conceitos são importantes para realização do núcleo do trabalho, que é o desenvolvimento do método de reconhecimento de imagens de autômatos.

3

TRABALHOS RELACIONADOS

Os trabalhos relacionados estão organizados em duas categorias. A primeira consiste nos trabalhos que buscam realizar a classificação e/ou conversão e/ou descrição automática de imagens, em que foram considerados apenas diagramas que são relacionados com a área da computação. A segunda categoria é voltada para uma visão das ferramentas assistivas já desenvolvidas, sendo elas para auxílio de criação, construção e manipulação de reconhecedores de linguagens por deficientes visuais.

Considera-se a segunda categoria como de grande relevância, pois, por meio dela, é possível estudar o que já foi investigado na literatura relacionado às representações alternativas de autômatos para deficientes visuais. Além disso, há a possibilidade das próprias ferramentas apresentadas nesses trabalhos serem uma versão alternativa. Exemplificando: uma imagem de autômato poderia ser convertida para um arquivo que seja compatível com tais ferramentas, que comumente dão suporte à navegação, manipulação e compartilhamento para deficientes visuais.

Por último, é apresentado o projeto MannaHap, que é um sistema háptico assistivo de representação de imagens digitais para deficientes visuais. Segundo Luque et al. (2018), é de grande relevância representar uma imagem digital por outro meio além da textual, sendo uma alternativa aceita a representação tátil. Este trabalho foi desenvolvido conjuntamente com o MannaHap, no qual os resultados oriundos do MannAR são integrados.

3.1 RECONHECIMENTO E DESCRIÇÃO AUTOMÁTICA DE DIAGRAMAS

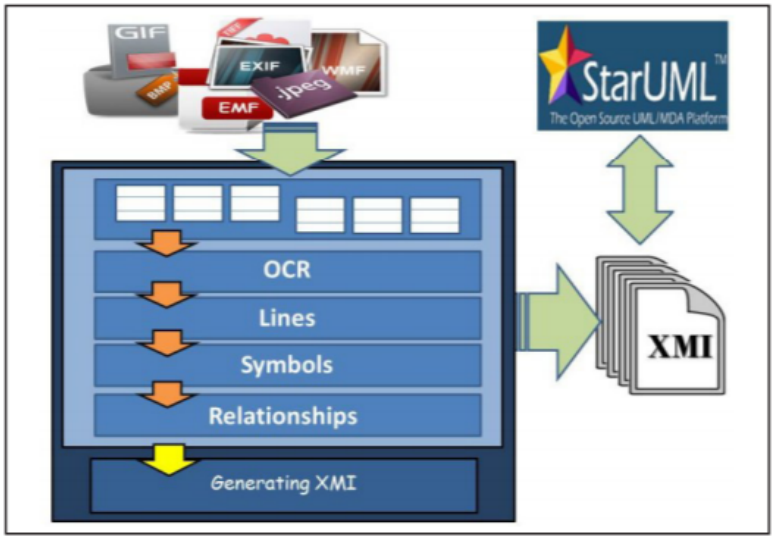
Trabalhos referentes ao reconhecimento automático de diagramas estão relacionados à classificação automática de diagramas a partir de sua imagem e/ou a tradução do diagrama para uma representação alternativa para cegos. Na sequência, são descritos três trabalhos, a saber: Karasneh e Chaudron (2013), Babalola (2015) e Moreno et al. (2016).

3.1.1 Img2UML

O projeto Img2UML foi idealizado pelos pesquisadores Karasneh e Chaudron (2013). A proposta do trabalho consiste em desenvolver uma ferramenta que converte automaticamente imagens de diagramas de classe UML (*Unified Modeling Language*) para o formato XMI (*XML Metadata Interchange*), que é baseado no padrão aberto XML (*Extensible Markup Language*). De acordo com os autores, esses modelos são importantes por serem a ligação entre o *design* do *software* e a implementação.

Para realizar o reconhecimento das imagens de diagramas de classe é necessário segmentar formas, símbolos, linhas e textos. Também é necessário identificar a regra de cada elemento do diagrama no modelo. Na Figura 3.1, é apresentada a visão geral do método de reconhecimento de diagramas de classe utilizado pelos autores Karasneh e Chaudron (2013).

Figura 3.1: Visão geral do método img2UML



Fonte: (KARASNEH; CHAUDRON, 2013)

Para realizar o reconhecimentos dos objetos e suas relações presentes no diagrama de classe os autores dividiram o processo de reconhecimento em 4 partes:

- detecção das classes na imagem: Para realizar a detecção das classes é necessário encontrar os retângulos na imagem e para isso foi utilizado um *framework* chamado Aforge.NET[1]. Também foi realizado um pré-processamento na imagem que incluiu transformar a imagem para níveis de cinza e aplicar a técnica *threshold*;

- reconhecimento de texto presente nas classes: o retângulo que representa a classe possui diversas partes com texto para serem identificadas, como: o nome da classe, os atributos e as funções. Para realizar o reconhecimento do texto foi utilizado a biblioteca para OCR (*Optical Character Recognition*) denominada MODI (*Microsoft Office Document Imaging*);

- detecção das relações: As relações entre os objetos são representadas por linhas que ligam duas classes, ou seja, dois retângulos. Essas relações podem ter várias direções, sendo linhas retas ou curvas, porém, o método Img2UML detecta apenas linhas retas. Nessa etapa do processo os autores identificam apenas a quais classes uma relação pertence, e não o tipo dessa relação;

- detecção do tipo da relação: existem quatro tipos de relações que podem ser identificadas, a saber: associação, generalização, dependência e composição. Todos os tipos são representados por pequenas formas geométricas que são reconhecidas pelo *framework* Aforge.NET.

Ao final do processo de reconhecimento dos objetos os dados foram organizados para serem representados em XMI. Para realização dos testes foram utilizadas 10 imagens de diagramas de classes de tamanho e resolução diferentes. Os resultados apresentaram uma acurácia de 100% para o reconhecimento de classes, 97% para o reconhecimento de relações e 85% para o reconhecimento de símbolos. Apesar dos bons resultados no reconhecimento, os autores apresentaram as seguintes limitações de seu trabalho:

- falha na criação do XMI: os autores afirmam que a resolução da imagem afeta a precisão do reconhecimento. Outro ponto, é que a maioria dos problemas surge quando um diagrama apresenta um grande número de relações. Dependendo do tipo de falha no reconhecimento, como o não reconhecimento de uma classe, o arquivo XMI pode não ser criado.

[1]Mais informações em: http://www.aforgenet.com/

- padrão *StarUML*: a ferramenta Img2UML utiliza o padrão StarUML, assim, caso o diagrama de classe tenha sido feito em outros modelos que utilizem outros formatos, o diagrama não será reconhecido.

- OCR e relações: a ferramenta pode não reconhecer corretamente nomes de atributos e funções. Um dos problemas apresentados é quando se tem os caracteres "()", pois, muitas vezes eles podem ser reconhecidos como a letra "O". De acordo com os autores, a versão apresentada no artigo da ferramenta Img2UML (KARASNEH; CHAUDRON, 2013) não pode detectar com segurança linhas tracejadas que estejam na diagonal. Também não pode detectar texto em relacionamentos. Assim, se a imagem não contiver texto nos relacionamentos, a oportunidade de exportar a imagem para XMI torna-se maior.

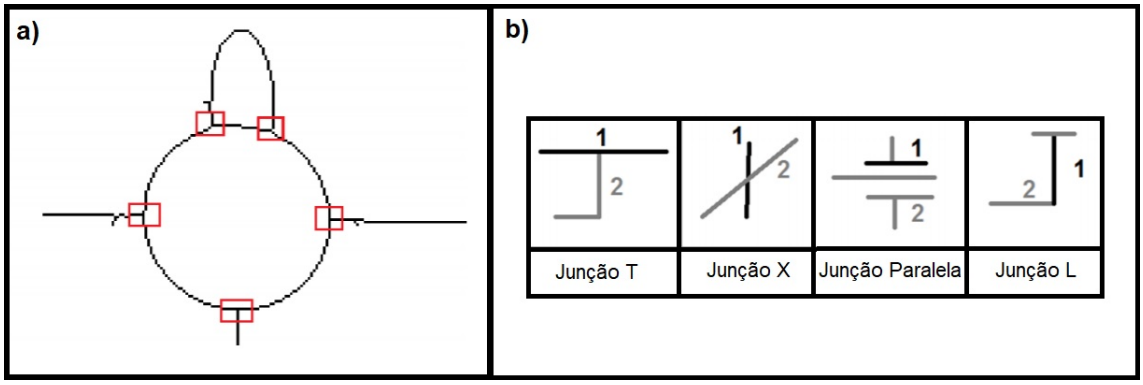
Apesar das limitações, os autores julgam que esta ferramenta pode ser útil para extração de informações do diagrama de classe UML, economizando tempo e esforço. Outro ponto, é que o Aforge.NET realiza a descoberta de símbolos geométricos da imagem somente se eles não estiverem conectados a nenhum outro objeto. Por isso, os autores realizam a descoberta das linhas antes de reconhecer os retângulos. No entanto, os autores não descrevem de forma clara como este processo é realizado, já que dão a entender que já sabem as classes no reconhecimento das linhas e em outro momento apresentam que reconhecem as linhas primeiro.

3.1.2 Reconhecimento de imagens de Autômatos Finitos

O trabalho de Babalola (2015) apresenta um sistema para o reconhecimento de diagramas de autômatos finitos. A proposta do trabalho é realizar a análise de uma aplicação prática desse sistema. Inicialmente, o autor apresenta os métodos de processamento de imagens utilizados na construção do sistema de reconhecimento.

Na primeira fase do processo o autor aplica um pré-processamento da imagem utilizando a técnica *threshold*. Na sequência, realiza a separação e reconhecimento dos objetos da imagem. Esta etapa inclui encontrar todos os pontos de junção da imagem, ou seja, onde dois objetos se cruzam, como mostra a parte (a) da Figura 3.2. Para encontrar os pontos de junção, o autor utilizou um processo de análise dos vizinhos de um pixel, assim ele procura por padrões, ilustrados na parte (b) da Figura 3.2.

Figura 3.2: Exemplos de pontos de junção



Fonte: (BABALOLA, 2015)

Após esse processo é possível separar cada objeto e classificá-los. Nesse caso, o autor utiliza as *features* apresentadas no Quadro 3.1 para decidir se o objeto é um círculo, uma curva, uma linha reta ou um texto.

Quadro 3.1: Condições utilizadas para identificar os objetos na imagem

Elementos do Diagrama	*Features*
Círculos	Área $>$ (tamanho mediano da área x 2) (Área / menor comprimento do eixo) ≈ 2 Excentricidade < 0.5
Linhas retas	Área $>$ (tamanho mediano da área x 2) (Área / menor comprimento do eixo) > 10 Excentricidade > 0.96 Solidez < 0.09
Curvas	Área $>$ (tamanho mediano da área x 2) (Área / menor comprimento do eixo) entre 1.95 e 2.9 Menor comprimento do eixo > 100 Solidez < 0.1
Texto	Área $<$ (tamanho mediano da área x 2) Menor comprimento do eixo < 10 Solidez > 0.04

Fonte: (BABALOLA, 2015, p. 54, tradução própria)

Considere excentricidade como a razão entre os valores do maior e do menor eixo das coordenadas, sendo que o valor é um número entre 0 e 1. A solidez é dada pela área dividida

pela área convexa, sendo esta segunda o número de *pixels* compreendidos na envoltória convexa de um objeto. Se o objeto não se encaixar em nenhuma das classificações ele é considerado um objeto desconhecido.

Após a classificação, os elementos que são textos são enviados para um sistema OCR externo. Já para os objetos classificados como linhas são realizados dois processos, sendo: a descoberta da direção da flecha e a quais estados a linha se relaciona. Os processos são detalhados na sequência.

- descoberta da direção da flecha: o autor divide a flecha em 4 regiões e realiza a contagem dos *pixels* nas regiões das extremidades. A região que obtiver a maior soma é considerada como sendo a direção da flecha.
- descoberta dos estados relacionados a uma linha: o autor realiza a análise de uma região envolta da linha. Se um círculo é descoberto ele é considerado como sendo um estado relacionado àquela linha.

Ao final desta fase do sistema de reconhecimento, uma imagem de diagrama será reduzida de um arranjo de *pixels* para uma coleção de elementos de diagrama. O sistema elaborado apresenta algumas limitações, como o não reconhecimento de linhas pontilhadas, tracejadas ou que se cruzam.

Para realização dos testes foram utilizadas três imagens, submetidas ao sistema. Para cada uma, o autor coloca como resultado o número de círculos na imagem, o número de arcos e o número de caracteres detectados, sendo que as três imagens testadas apresentaram um bom desempenho.

3.1.3 Classificação automática de Diagramas UML

O trabalho de Moreno et al. (2016) propõe a classificação automática de imagens da Web e informa se elas são ou não diagramas UML. A ferramenta elaborada recebe uma imagem que pode ser de diferentes formatos (BMP, GIF, JPEG, PNG e TIFF) e então as classifica como diagramas UML ou não. De acordo com os autores, a aplicação pode ser utilizada em vários cenários, a saber:

- após a utilização de um buscador genérico. Segundo os autores, a ferramenta pode melhorar a precisão dos resultados de uma busca por diagramas UML realizada em um motor de dados genérico, como o Google, excluído os falsos positivos;

- integrada a um motor de busca específico para diagramas UML. A busca direta por diagramas UML pode ter um aumento de precisão utilizando essa busca. Os autores informam que eles próprios estavam construindo essa busca específica por diagramas UML e que a ferramenta de classificação apresentada seria parte de trabalhos futuros;

- repositórios particulares. A ferramenta também pode ser utilizada na busca de diagramas UML em repositórios de organizações, auxiliando na reutilização de software, em que por meio dessa busca poderia se encontrar material útil na produção de novos softwares.

A classificação automática dos diagramas é realizada por meio da extração de características da imagem, na qual a combinação delas permite a classificação por regras de indução. As características extraídas foram: Histograma de tons de cinza, que informa quantos *pixels* de cada tom de cinza a imagem possui; Histograma de cor que informa quantos *pixels* de cada cor a imagem possui; e Formas geométricas elementares, que identificam as formas geométricas básicas, como quadrados, retângulos e círculos.

Para os testes os pesquisadores utilizaram cerca de 18899 imagens retiradas da Internet, sendo 794 imagens de diagramas UML, 6703 imagens de outros diagramas e 11402 de imagens que não são diagramas. O resultado do trabalho apresenta um acerto de 94,4% do método empregado, possibilitando a conclusão dos autores de que a ferramenta é efetiva nos cenários de aplicação já listados. Apesar da ferramenta não estar no contexto de acessibilidade, o reconhecimento de diagramas UML poderia auxiliar também deficientes visuais.

3.2 FERRAMENTAS ASSISTIVAS RELACIONADAS À RECONHECEDORES DE LINGUAGENS

A segunda categoria de trabalhos relacionados aborda ferramentas que já permitem o compartilhamento, criação e manipulação de imagens de autômatos por deficientes visuais. A literatura apresenta trabalhos como os dos autores (COHEN et al., 2006), (STALLMANN et al., 2007), (MILLER, 2009), (CRESCENZI et al., 2012) e (BALIK et al., 2013).

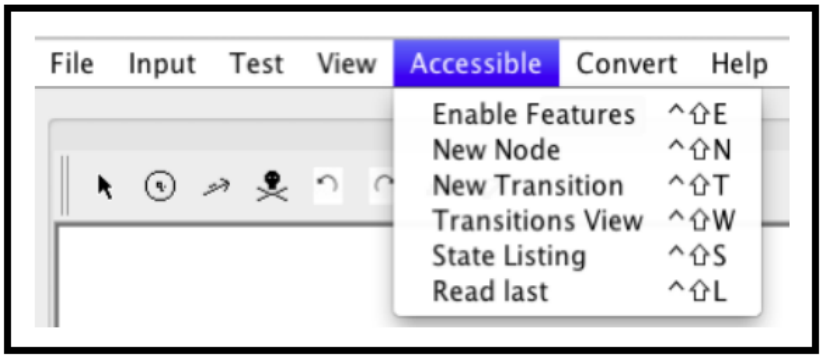
Duas questões são avaliadas para que a ferramenta possa ser uma versão alternativa, ou seja, converter a imagem do autômato para uma versão compatível com uma ferramenta já acessível. São elas: a ferramenta ser gratuita; e a forma utilizada para representar os autômatos. Devido a esses fatores apenas duas ferramentas foram detalhadas nas próximas seções.

3.2.1 JFLAP acessível

A ideia desse trabalho era incluir acessibilidade na ferramenta JFLAP, um simulador de Máquinas de Turing. O JFLAP possui o uso de XMI como formato de seu arquivo, o que facilita a conversão da imagem para o formato .jff. A proposta dos pesquisadores Crescenzi et al. (2012) era a inserção da acessibilidade de forma que a versão desenvolvida não tivesse grandes mudanças de *layout*, permitindo que ela fosse utilizada por deficientes visuais e pessoas sem deficiência visual.

Para inserção de acessibilidade, os autores inseriram atalhos de teclado em operações como: adição de novo nó, adição de nova transição, visualização das transições, lista de estados, entre outros, como é possível visualizar na Figura 3.3.

Figura 3.3: Novo menu na ferramenta JFLAP com os atalhos de teclado



Fonte: (CRESCENZI et al., 2012)

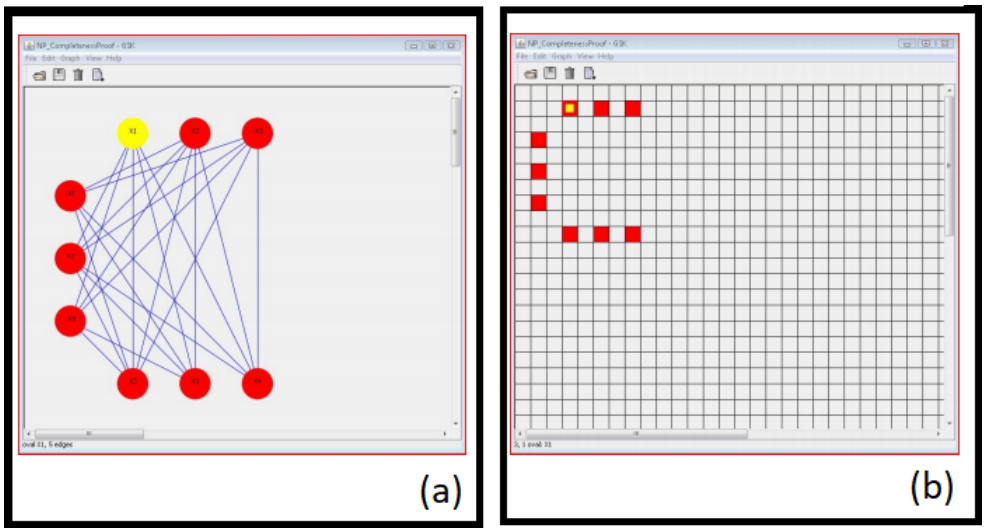
Outras funções adicionadas foram novas telas para criação de nó, criação de transição, visualização dos detalhes das transições e uma nova tela de simulação do autômato. Essas novas funcionalidades permitem ao deficiente visual especificar os dados necessários para a construção do autômato utilizando como auxílio a ferramenta *JAWS*, que é o leitor de tela para o sistema operacional *Windows*.

A avaliação da ferramenta foi realizada com um aluno de Ciência da Computação que é deficiente visual. Foram dadas as especificações de três Máquinas de Turing, que o aluno desenvolveu no JFLAP. Após, os pesquisadores avaliaram a corretude dos exercícios realizados. Segundos os autores do trabalho o resultado foi satisfatório, pois, uma máquina estava certa, outra estava quase correta e a última continha um erro muito simples.

3.2.2 *Graph SKetching tool* (GSK)

A ferramenta *Graph SKetching tool* (GSK), elaborada Balik et al. (2013), é uma aplicação que pode ser utilizada por usuários de computador com deficiência visual, independentemente de dispositivos e hardware especializados. O *software* GSK permite que seus usuários criem, editem e compartilhem diagramas do tipo nó-link em tempo real. Na Figura 3.4 têm-se as duas visualizações possíveis para o mesmo diagrama que o *software* implementa.

Figura 3.4: Visualização do mesmo diagrama no formato tradicional (a) e no formato alternativo para deficientes visuais (b)



Fonte: (BALIK et al., 2013)

O trabalho de Balik et al. (2013) está baseado nos princípios do Design Universal, que se preocupa com a utilização do software por qualquer pessoa, sem segregação de usuários. Para seguir essas recomendações, o GSK permite que o mesmo diagrama possua duas representações, uma para ser utilizada por deficientes visuais e uma por pessoas que enxergam, permitindo que os dois grupos de pessoas trabalhem juntos no mesmo projeto.

Além da realização da avaliação realizada no contexto de disciplinas da área de Ciência da Computação, no artigo do mesmo grupo de pesquisadores (BALIK et al., 2014) foi apresentada uma avaliação da mesma ferramenta com nove alunos cegos de outras áreas. A conclusão geral a respeito da ferramenta foi que ela permite a criação e edição de diagramas por pessoas com deficiência visual em um tempo razoável. O trabalho relata que um avaliador comentou que a ferramenta o ajudou a memorizar mais facilmente a estrutura do diagrama, colaborando para a compreensão do mesmo.

3.3 MANNAHAP

O MannaHap (junção do nome do grupo dos desenvolvedores *Manna* + a palavra *Háptico*), é um sistema que utiliza uma abordagem háptica, ou seja, ele promove a interação entre o usuário e o dispositivo por meio do toque, desencadeando percepções secundárias, como a auditiva.

O sistema é composto por um dispositivo intermediário, que realiza a conversão da imagem digital para o formato utilizado no MannaHap, e um hardware composto por mesa de pinos, que se movem de acordo com os dados recebidos (*pinos refreshable*), a fim de representar a figura escolhida pelo usuário. Assim, qualquer imagem digital pode ser representada no sistema tátil desde que ela seja transformada para o formato necessário. Na Figura 3.5 tem-se o conceito de um usuário usando o sistema háptico assistivo. O detalhe apresenta uma interação do usuário, na qual ele está tocando a tela atualizável e escutando um contexto de áudio.

Figura 3.5: Representação da utilização do sistema háptico assistivo

Fonte: (SVAIGEN et al., 2018)

O MannAR pode funcionar como um dispositivo intermediário do sistema MannaHap. Assim, para representar a imagem de um reconhecedor de linguagens é necessário gerar um arquivo no formato JSON (*JavaScript Object Notation*)[2]. O sistema utiliza características que são independentes do tipo de imagem a ser convertida. As características são:

- linhas: altura da imagem (quantidade de linhas na imagem original de entrada);
- colunas: largura da imagem (quantidade de colunas da imagem original de entrada);

[2]https://www.json.org/

- objetos: representados por um vetor, compreendem os objetos identificados na imagem digital de entrada. Cada objeto possui uma lista com características distintas, a saber:
  - ID: Identificação numérica única do objeto;
  - coordenada de origem: par ordenado numérico inteiro positivo (n_l, n_c), no qual n_l indica a coordenada referente à linha e n_c indica a coordenada referente à coluna. Tem-se como restrição que $n_l \leq linhas$ e $n_c \leq colunas$;
  - matriz de *pixels*: matriz em níveis de cinza que representa o objeto, sendo 0 indica que o pino deve permanecer abaixado e quanto maior o valor, mais alto o pino deverá ser levantado;
  - descrição: representada por um texto, provê uma contextualização do objeto na qual será gerado um áudio para o usuário.

Utilizando as características listadas, é possível mapear a imagem gerada por um sistema intermediário para o MannaHap. As demais particularidades, arquitetura de funcionamento, e a descrição dos estados do sistema não serão abordados por este trabalho.

3.4 CONSIDERAÇÕES FINAIS

No que tange aos trabalhos referentes ao reconhecimento e descrição automática de diagramas, foi constatado que existem poucos trabalhos na literatura na área de diagramas relacionados a Ciência da Computação. No entanto, os trabalhos de Crescenzi et al. (2012) e Babalola (2015) possuem uma proposta similar a deste trabalho.

Apesar do trabalho de Babalola (2015) ser sobre o reconhecimento de imagens de autômatos, o autor apresenta testes com poucas imagens (apenas três), o que sugere limitações nos métodos propostos quando se trata de padrões utilizados nos diagramas (forma como um estado ou uma transição é representada).

Já o trabalho de Moreno et al. (2016) apresenta uma abordagem diferente dos anteriores por ser apenas uma classificação de imagens, informando se a imagem é ou não um diagrama UML. Neste caso, se percebe que os testes foram realizados com uma boa quantidade de imagens, aumentando a qualidade do método. Um fator interessante é que não foi encontrado na literatura um trabalho que apresente a união de reconhecimento com classificação automática das imagens.

Os dois trabalhos elencados como ferramentas assistivas apresentam soluções que permitem que usuários deficientes visuais e não deficientes visuais possam interagir no mesmo

ambiente. Assim, as duas soluções se tornam possibilidades interessantes de serem ferramentas que possam ter as imagens de diagramas convertidas para arquivos que sejam compatíveis com elas e por serem gratuitas.

Por fim, as ferramentas assistivas de manipulação de diagramas podem ser abordagens utilizadas como meios alternativos de representação de reconhecedores de imagens, já que possuem todas as funcionalidades acessíveis. Outra opção, é integrar o MannAR com um sistema de representação tátil como o MannaHap. Assim, os usuários podem expandir as percepções da imagem por meio do toque.

4

MANNAR: RECONHECIMENTO AUTOMÁTICO DE IMAGENS DE AUTÔMATOS

Este Tópico apresenta o método desenvolvido por este trabalho para tradução de diagramas de estados de transição de mecanismos reconhecedores de linguagens formais utilizados na disciplina de Teoria da Computação, denominado MannAR. A solução apresentada contempla estratégias de processamento de imagens para segmentar os objetos e prover o devido significado à cada parte obtida.

Nas próximas seções são apresentados os detalhes sobre a solução proposta. Inicialmente, são apresentados dois processos preliminares ao método de desenvolvimento, a saber:

- modelos de reconhecimento do tipo e da quantidade de estados de um reconhecedor de linguagem formal: construção de duas CNNs, sendo uma que identifica o tipo do reconhecedor de linguagem formal e outra a quantidade de estados na imagem. Esse processo não faz parte do método em si, porém, seus resultados, que são dois modelos de identificação, possuem potencial de serem utilizados no método;
- técnicas de identificação de círculos em imagens: estudo de três técnicas de reconhecimento de círculos. No caso deste processo, a técnica com o melhor desempenho é utilizada como parte do método.

Na sequência apresenta-se uma visão geral do método proposto e suas demais etapas são explanadas, como o pré-processamento, o reconhecimento dos estados, o reconhecimento das transições e o pós-processamento.

4.1 ETAPAS PRELIMINARES AO MÉTODO PROPOSTO

Foram realizados dois processos preliminares à construção do método proposto com o objetivo de estudar técnicas que poderiam ser utilizadas no desenvolvimento e de facilitar a construção do mesmo. O primeiro processo realizado fornece algumas informações auxiliares, como o tipo do autômato presente na imagem e a quantidade de estados que o autômato possui. Para isto foi utilizada uma CNN. Após todo o processo de construção e treinamento da rede, é necessário utilizar apenas o modelo gerado. A abordagem utilizada para a construção do modelo é apresentada na Subseção 4.1.1.

O segundo processo foi o estudo de técnicas de identificação de círculos em imagens. Como já citado, na Subseção 2.3.2, três métodos foram analisados, sendo: a transformada de Hough, o *Learning Automata* e o *Haar Cascade*. O desenvolvimento deste estudo é apresentado na Subseção 4.1.2.

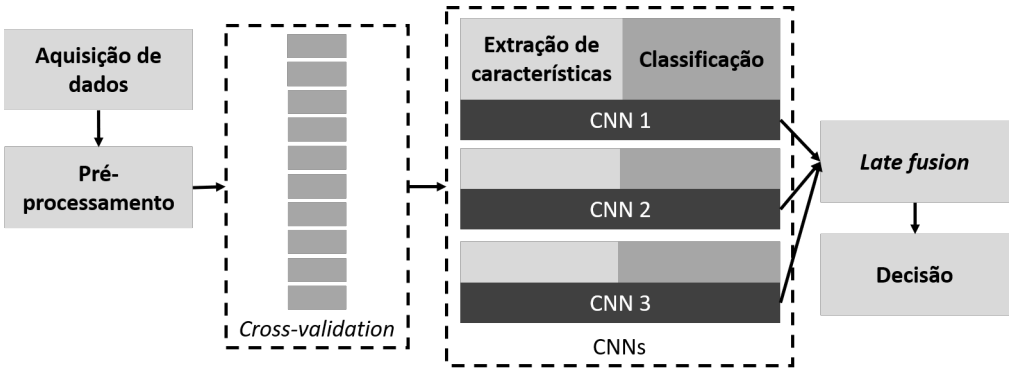
4.1.1 Classificação Automática de imagens de autômatos utilizando CNN

A implementação de um conjunto de CNNs para a identificação de algumas das características dos diagramas de autômatos contém o propósito de auxiliar a extração das demais informações das imagens. Duas abordagens foram estudadas, sendo elas:

- reconhecimento do tipo de autômato, que é dividido nas classes: Autômato Finito, Autômato de Pilha e Máquina de Turing;
- reconhecimento da quantidade de estados do autômato, dividido em oito classes de reconhecimento. A primeira indica um estado de reconhecimento, a segunda indica dois estados, e assim subsequentemente, até a última, que indica oito ou mais estados.

Cada uma das abordagens foi tratada individualmente. Contudo, os mesmos passos foram seguidos para as duas. O método utilizado na elaboração das CNNs é apresentado pela Figura 4.1.

Figura 4.1: Metodologia utilizada na criação de modelos de reconhecimento nas abordagens: tipo de autômato e quantidade de estados do autômato



Fonte: (PRÓPRIA, 2019)

Na Subseção 4.1.1.1 a seguir é descrita a criação de um banco de imagens de autômatos conjuntamente com o pré-processamento realizado. Na sequência, na Subseção 4.1.1.2, os detalhes da implementação realizada são apresentados.

4.1.1.1 Banco de imagens de Autômatos

Para analisar a performance de uma CNN é necessária uma quantidade significativa de imagens. Por isso, foi criado um *database* com imagens retiradas da Internet de Autômatos Finitos, Autômatos de Pilha e Máquinas de Turing.

Analisando as imagens coletadas, verificou-se que havia apenas um tipo de representação de transição de Autômatos Finitos. Inicialmente, foram coletadas 640 imagens de AF, que são balanceadas pelo número de estados, conforme mostra a Tabela 4.1.

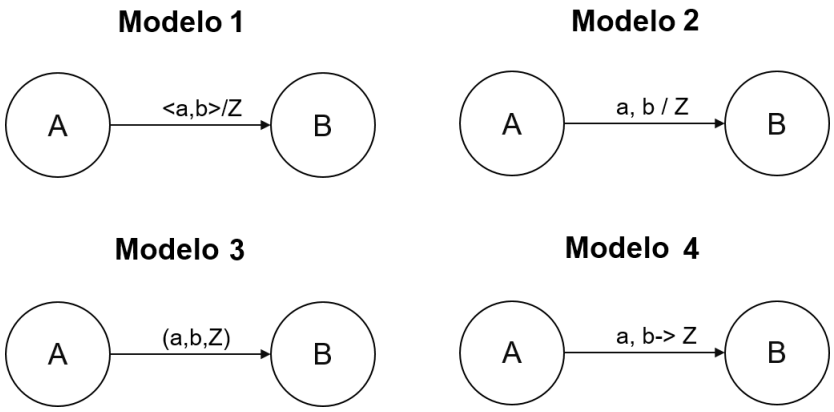
Tabela 4.1: Balanceamento de imagens para Autômatos Finitos

Número de estados	Número de imagens
1	80
2	80
3	80
4	80
5	80
6	80
7	80
8 ou mais	80
Total	**640**

Fonte: (PRÓPRIA, 2019)

Já para os Autômatos de Pilha há mais de uma maneira de representar as descrições das transições, que variam entre autores, e alguns modelos foram observados nas imagens coletadas. A Figura 4.2 mostra os quatro modelos selecionados, sendo que eles foram os que apresentaram a maior quantidade de exemplos encontrados na Internet.

Figura 4.2: Modelos de transições de Autômatos de Pilha utilizados



Fonte: (PRÓPRIA, 2019)

Além de balancear o número de imagens de cada modelo, foi necessário igualar o número de estados do autômato. Diferentemente dos Autômatos Finitos, foi encontrada uma boa quantidade de exemplos de Autômatos de Pilha de no máximo quatro estados. Na Tabela 4.2, é possível observar o balanceamento realizado.

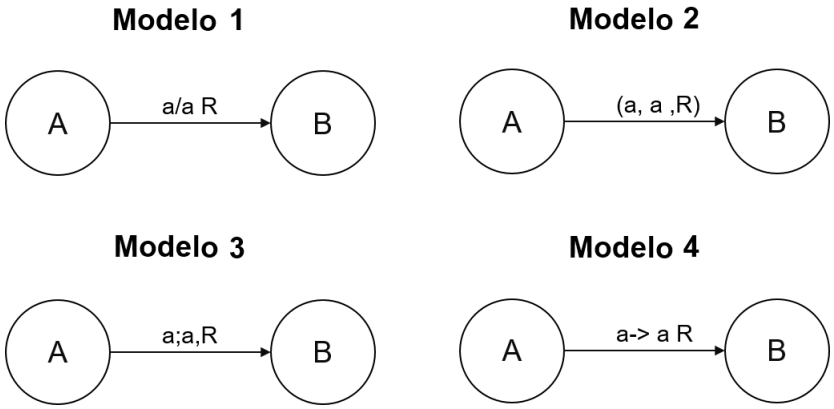
Tabela 4.2: Balanceamento de imagens para Autômatos de Pilha

Número de estados	**Modelos**				**Número de imagens**
	1	**2**	**3**	**4**	
1	40	40	40	40	160
2	40	40	40	40	160
3	40	40	40	40	160
4 ou mais	40	40	40	40	160
Total	160	160	160	160	**640**

Fonte: (PRÓPRIA, 2019)

Da mesma forma que os Autômatos de Pilha, há mais de uma maneira de representar a descrição de uma transição na Máquina de Turing. Os modelos selecionados são mostrados na Figura 4.3.

Figura 4.3: Modelos de transições de Máquinas de Turing utilizados



Fonte: (PRÓPRIA, 2019)

Para a Máquina de Turing, observou-se que exemplos de apenas um estado não existem. Portanto, o balanceamento foi realizado iniciando em 2 estados e considerando os modelos de transições, conforme a Tabela 4.3.

Tabela 4.3: Balanceamento de imagens para Máquinas de Turing

Número de estados	**Modelos**				**Número de imagens**
	1	**2**	**3**	**4**	
2	20	20	20	20	80
3	20	20	20	20	80
4	20	20	20	20	80
5	20	20	20	20	80
6	20	20	20	20	80
7	20	20	20	20	80
8	20	20	20	20	80
9 ou mais	20	20	20	20	80
Total	160	160	160	160	**640**

Fonte: (PRÓPRIA, 2019)

A configuração inicial do banco de imagens foi de 640 imagens de cada tipo de autômato, somando um total de 1920. Como tais imagens foram coletadas da Internet, elas não tinham, inicialmente, um formato adequado considerando as premissas do trabalho. Assim, foi realizado um processamento para que se tornassem quadrangulares, de forma a manter uma padronização. A dimensão das imagens no conjunto de dados é 64×64 *pixels*.

Em ambos os casos de classificação abordados, não foi necessário conhecer as informações de cores das imagens, de modo que, para fins de menor processamento, as imagens

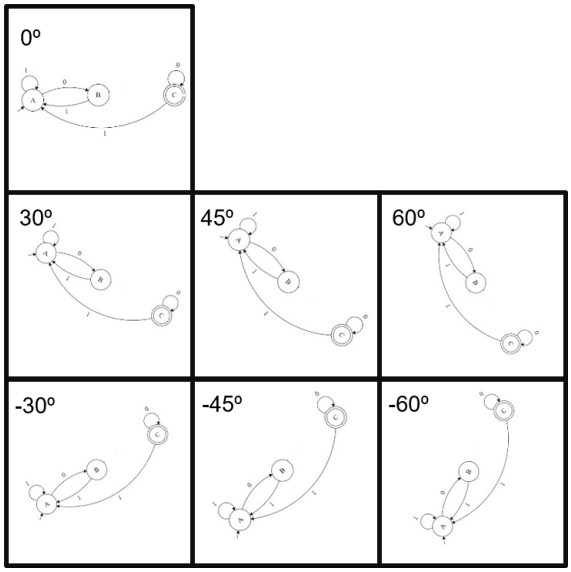
foram transformadas em escala de cinza.

Para melhores desempenhos, as CNNs geralmente usam bancos de dados com uma quantidade elevada de imagens. Considerando que inicialmente havia apenas 1920 imagens, foi necessário aumentar a quantidade de dados existentes para melhorar o desempenho. O processo de aumento de dados pode ser realizado modificando imagens já presentes no banco de dados coletados e criando uma nova imagem com alguma diferença da original, este processo é conhecido como *data augmentation*.

Ao avaliar as transformações típicas usadas em *data augmentation* (zoom *in* / *out*, alterar as condições de iluminação e introduzir ruído), percebeu-se que o resultado que seria obtido com tais transformações não expressavam uma imagem que pudesse ser caracterizada como um automato ou máquina de Turing válidos. Assim, estas transformações ditas típicas não foram consideradas.

Outro método comumente usado é a rotação. Em alguns casos, é possível criar novos exemplos usando qualquer grau de rotação. Neste trabalho, consideram-se seis rotações válidas que mantêm as características buscadas nas imagens: 30^o, -30^o, 45^o, -45^o, 60^o e -60^o. Na Figura 4.4, mostra-se um exemplo de *data augmentation*, considerando a rotação.

Figura 4.4: Exemplo de *data augmentation*



Fonte: (PRÓPRIA, 2019)

Após a aplicação do método de rotação, o banco de dados passou a ter 13440 imagens, 3840 para cada tipo de autômato. Esta operação permitiu que o banco de dados fosse balanceado para a primeira abordagem. No entanto, ao dividir o banco de dados para a segunda abordagem

(reconhecimento do número de estados) em 8 classes, não foi possível manter tal equilíbrio, conforme mostrado na Tabela 4.4.

Tabela 4.4: Classificação pelo número de estados do autômato

Número de estados	Número de imagens
1	1680
2	2240
3	2240
4	2240
5	1120
6	1120
7	1120
8 ou mais	1680
Total	**13440**

Fonte: (PRÓPRIA, 2019)

Então, para realizar o balanceamento da segunda abordagem, as classes foram construídas com 1120 imagens, resultando em um banco de dados de 8960. As imagens coletadas da Internet, que formam o *database* com as 1920 imagens originais coletadas, denominado AD[1] (Automata Database), está disponível *online*.

4.1.1.2 Implementação das CNNs

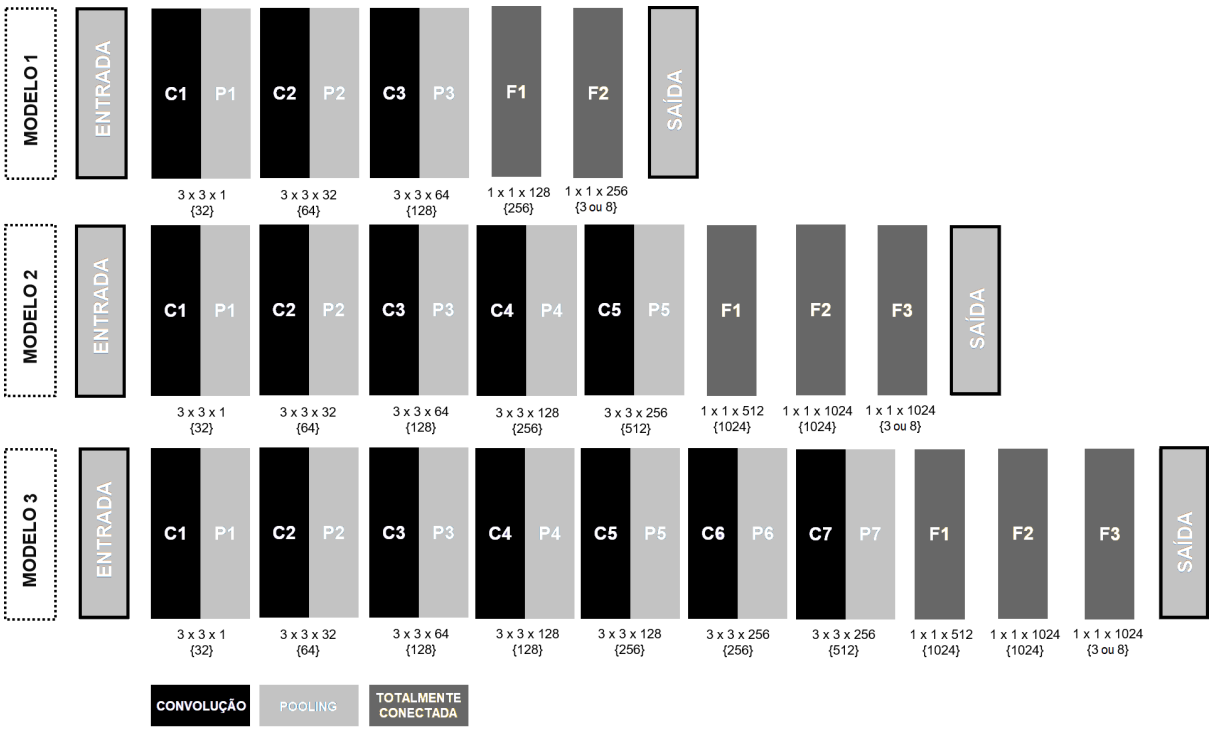
Três modelos de CNNs foram implementados. Elas diferem umas da outras pelo número de camadas ocultas. Para diferenciar um tipo de autômato de outro poucos detalhes são necessários. Assim, realizou-se um aumento na quantidade de camadas de convolução, permitindo maior capacidade de armazenamento de padrões que podem ser generalizados pela rede neural. Todas as variações implementadas recebem como entrada uma imagem de $64 \times 64 \times 1$ *pixels*, pois estamos usando imagens em escala de cinza.

Considerando as duas abordagens avaliadas, reconhecimento de tipo de autômato e reconhecimento do número de estados do autômato, todas as versões da CNN foram adaptadas ao número de classes dos problemas em questão, sendo três para a primeira abordagem e oito para a segunda abordagem.

A Figura 4.5 mostra as três arquiteturas construídas, na qual o número abaixo de cada camada representa o tamanho dos filtros e o número entre chaves representa a quantidade de filtros usados em cada etapa.

[1] https://sites.google.com/view/automatadatabase

Figura 4.5: Arquiteturas dos modelos de CNN utilizadas



Fonte: (PRÓPRIA, 2019)

A base para a construção dos modelos foi a arquitetura AlexNet (KRIZHEVSKY et al., 2012), isto porque ela é uma rede adequada as configurações do computador que havia disponível para treinamento da rede. As arquiteturas criadas apresentam uma camada de *pooling* após cada camada de convolução, o que difere do modelo original, que apresenta esse processo apenas em algumas camadas. O tamanho da camada de *pooling* utilizada é 2×2, o mesmo em todos os casos. Ao aplicar um *pooling* com este valor, as imagens resultantes diminuem de tamanho pela metade em altura e largura, o que consequentemente reduz o custo da computação.

Outro fator é que os filtros usados em cada camada têm tamanho 3×3. O mesmo valor foi escolhido para simplificar o modelo e a quantidade de hiperparâmetros, e também para adicionar não-linearidade. A seguir tem-se o detalhamento de cada modelo:

- modelo 1 - arquitetura com cinco camadas: esse modelo consiste em três camadas de convolução (cada uma acompanhada por uma camada de *pooling* e da aplicação da função *ReLU*) e de duas camadas totalmente conectadas. No final do processamento da primeira camada de convolução, que usa 32 filtros de tamanho 3×3, as imagens de saída têm tamanho 32×32. Para a Camada 2, foram utilizados 64 filtros de tamanho 3×3,

resultando em imagens de saída 16 $\times$ 16. Da mesma forma, o processo ocorre para a terceira camada com 128 filtros e as imagens de saída são 8 $\times$ 8. As duas camadas totalmente conectadas usam 256 neurônios. No entanto, a função ReLU é aplicada apenas no final da primeira camada;

- modelo 2 - arquitetura com oito camadas: consiste em cinco camadas de convolução, na qual cada uma é seguida por uma camada de *pooling* e três camadas totalmente conectadas, sendo que somente a primeira tem o uso da função *ReLU*. O aumento de camadas ocultas fornece maior capacidade de armazenamento de padrões. Como uma das abordagens foi descobrir o tipo de autômato, mais padrões permitem uma melhor diferenciação das imagens, pois poucos detalhes diferenciam os tipos de autômatos. Nessa arquitetura, o número de filtros duplica de camada para camada, a partir de 32 até 512 filtros e cada uma das três camadas totalmente conectadas usa 1024 neurônios;

- modelo 3 - arquitetura com dez camadas: este modelo consiste em sete camadas de convolução e três camadas totalmente conectadas. Neste caso, maximiza-se o número de camadas ocultas até o tamanho da imagem de saída ser 1 $\times$ 1, permitindo um maior armazenamento de padrões. Um ponto a se destacar é que, nos modelos anteriores, para cada camada de convolução havia o dobro da quantidade de filtros do que na camada anterior. No entanto, em alguns casos, neste modelo, a quantidade de filtros é mantida. Isso ocorre entre as camadas 3 e 4, mantendo o número de filtros em 128 e entre as camadas 5 e 6 em que o número de filtros é 256.

O treinamento dos modelos e os resultados são apresentados na Seção 6.1.1.

4.1.2 Estudo de métodos de localização de círculos

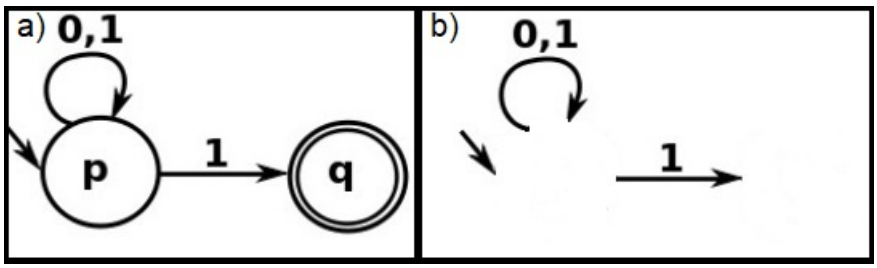
Como visto na Seção 2.1, os objetos que compõem um autômato podem apresentar variações de formato de acordo com diferentes sistemas e autores. Considerando esse fato, partiu-se do princípio que os objetos que simbolizam estados possuem representações semelhantes nas diversas versões de imagens de diagramas de estados de um autômato, sendo raríssimos os casos em que eles não são representados por círculos.

O desafio deste trabalho consiste na correta segmentação de cada objeto, por isso, o método proposto parte do objeto considerado o mais semelhante nas representações de imagens de autômatos, a saber, os estados.

A segmentação correta dos círculos, que representa os estados, permite a separação de

praticamente todos os demais objetos na imagem que estavam ligados a algum outro objeto, como pode-se observar na Figura 4.6 .

Figura 4.6: Exemplo de segmentação dos estados (b) para a imagem original (a)



Fonte: (PRÓPRIA, 2019)

Devido à importância de localizar os círculos corretamente na imagem foi realizado um estudo sobre algumas técnicas, como: *Learning Automata*, Hough e *Haar Cascade*. O objetivo foi analisar estes três métodos e então escolher um deles para fazer parte do MannAR. Nas próximas seções são apresentados detalhes sobre o desenvolvimento de cada técnica. Porém, os resultados sobre os testes realizados com cada uma são apresentados na Subseção 6.1.2.

4.1.2.1 *Learning Automata* Adaptado

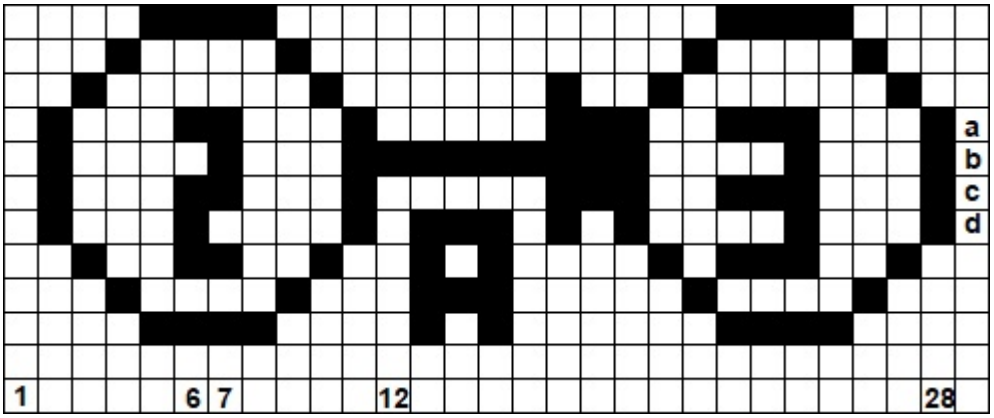
Buscando a otimização da técnica *Learning Automata*, optou-se pela realização de algumas modificações, pois a versão original sugere que sejam analisados 5% dos *pixels* da imagem, sendo que para os pontos selecionados devem ser verificadas a probabilidade da junção de três pontos serem um círculo para todas as possibilidades de combinações de três em três *pixels*.

No método LA são geradas muitas possibilidades, na qual não existem ou são muito pequenas as probabilidades dos conjuntos de pontos serem um círculo. Assim, buscando realizar o teste do círculo apenas se houver chance do conjunto de *pixels* analisados pertencerem a um círculo, foi realizada uma análise para cada coluna individualmente. Inicialmente, cada coluna foi representada por uma lista de pontos de objetos. Depois, foi definida uma distância mínima entre os pontos, que representa o menor diâmetro de círculo procurado.

Na sequência, deve ser realizada uma combinação dois a dois dos *pixels* da coluna, excluindo aqueles em que a distância seja menor que o valor estabelecido. Para cada possibilidade de círculo encontrada, deve ser realizado o trecho do método LA chamado de reforço de sinal, estabelecendo assim um percentual do par de círculo analisado pertencerem a

um círculo que realmente exista. Na Figura 4.7 é possível observar um exemplo, na qual cada quadrado representa um *pixel*. A distância mínima entre dois *pixels* pretos deve ser maior que dois, ou seja, o menor diâmetro possível para um círculo procurado nesta imagem será três.

Figura 4.7: Autômato Finito utilizado na exemplificação do processo LA



Fonte: (PRÓPRIA, 2019)

Na primeira parte do algoritmo devem ser criadas as listas de pontos, sendo uma para cada coluna. Exemplificando, a lista da coluna doze tem apenas um *pixel*, já a lista da coluna vinte e oito tem quatro. Na segunda etapa são analisadas apenas as listas que tiverem mais de um *pixel*. Voltando ao exemplo, a lista da coluna doze não seria analisada. No caso da lista da coluna vinte e oito, quando forem realizadas as combinações, apenas os *pixels a* e *d* possuem distância maior que dois entre eles, portanto, para essa coluna é gerada apenas uma possibilidade de círculo. O Algoritmo 1 apresenta esse processo.

Nas linhas de um a oito é criada uma lista de *pixels* pretos para cada coluna. Da linha nove até a vinte e seis é realizado o processo de percorrer cada uma das listas criadas e a geração de um círculo usando combinação dois a dois com todos os elementos da lista, se a distância entre eles for maior que a pré-estabelecida.

O pior caso deste algoritmo é quando a imagem é totalmente preta e o diâmetro mínimo procurado para os círculos é zero. Para a primeira etapa, que cria as listas, a complexidade é de $O(l \times c)$, na qual l é o número de linhas e c o número de colunas. Para a segunda etapa, que gera os círculos, no pior caso, todas as colunas são percorridas e a quantidade de *pixels* em cada coluna é o número de linhas. Assim, fazendo a combinação dois a dois, a complexidade desta etapa é $O(l^2)$ para cada coluna, totalizando $O(c \times l^2)$. Considerou-se que a etapa de cálculo do reforço de sinal é constante.

Ao final, temos que a complexidade do algoritmo é $O(c \times l^2)$. Porém, como os *pixels*

presentes nas imagens de autômatos são esparsos, o pior caso, só irá acontecer se uma imagem incorreta foi inserida no sistema.

Algorithm 1 Detecção de possíveis círculos utilizando o método LA com adaptações

Require: *img* {Imagem com aplicação do método Canny}
Ensure: *circles* {Lista de possíveis círculos}

```
1:  minRadius ← 20
2:  for all col c in img do
3:      for all rol r in img do
4:          if img[r][c] = 255 (branco) then
5:              list[c] ← r
6:          end if
7:      end for
8:  end for
9:  for c = 0 to |list| do
10:     currentList ← list[c]
11:     if |currentList| > 1 then
12:         for i = 0 to |currentList| do
13:             for i + 1 = 0 to |currentList| do
14:                 dist = currentList[j] − currentList[i]
15:                 if dist > minDist then
16:                     raio ← dist/2
17:                     y0 ← c
18:                     x0 ← currentList[i] + raio
19:                     circle ← (x0, y0, r)
20:                     beta ← reinforcementSignal(img, circle)
21:                     circles ← (circle, beta)
22:                 end if
23:             end for
24:         end for
25:     end if
26: end for
```

Ao final, todos os círculos que possuem o valor de *beta* menor do que $0,5$, ou seja, os que possuem menos de 50% de chance de ser um círculo, são descartados. Por fim, são excluídos os círculos que são muito próximos. Por exemplo, na Figura 4.7, a análise do primeiro com o último ponto das colunas seis e sete podem gerar valores de betas bem parecidos, portanto, têm-se dois possíveis círculos para apenas um círculo real. Assim, círculos que possuem os centros com distância menor do que a distância mínima são considerados repetidos e descartados.

4.1.2.2 Hough para círculos e *Haar Cascade*

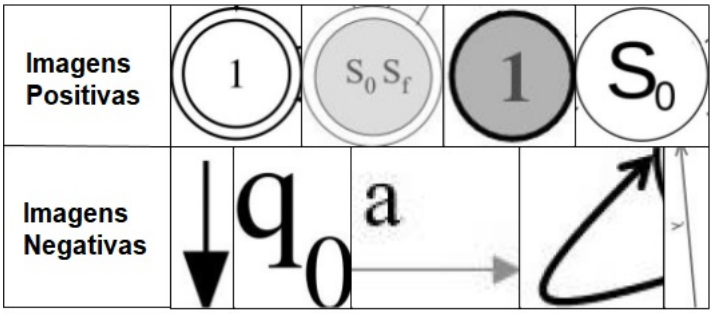
Para as técnicas Hough e *Haar Cascade* foram utilizados os métodos descritos respectivamente nas Subseções 2.3.2.1 e 2.3.2.3. Para os dois métodos, considera-se de importância a escolha correta dos parâmetros. No caso do Hough, por exemplo, são vários parâmetros envolvidos, a saber: relação inversa da resolução do acumulador para a resolução da imagem ("dp"), qual a distância mínima que um círculo deve estar do outro considerando os pontos centrais ("minDist"), raio mínimo e máximo que devem ser buscados ("minRadius" e "maxRadius"), valor a ser considerado no *upper thresholding* do método Canny ("param1"), valor do *thresholding* para detecção do centro do círculo ("param2"). No caso deste último parâmetro, quanto menor o limite, mais círculos serão detectados (incluindo círculos falsos). Quanto maior o limite, mais círculos serão potencialmente retornados.

Para que a definição dos parâmetros não dependesse de cada imagem foi utilizado o modelo de CNN, conforme descrito na Subseção 4.1.1, que resulta na quantidade de círculos que a imagem apresenta. Assim, o método Hough retorna vários possíveis círculos, isto é, uma lista de círculos ordenada do mais provável para o menos provável. Contudo, apenas a quantidade informada pelo modelo de CNN é considerada como resposta. Por exemplo, se o modelo informou que existem três estados na imagem, a resposta consiste nos três primeiros círculos da lista - os mais prováveis de acordo com o método.

Já para o *Haar Cascade*, alguns parâmetros que devem ser considerados no momento do treinamento são: número de estágios ("numStages"), taxa de acerto mínima esperada em cada estágio ("minHitRate"), taxa máxima de alarme falso desejado para cada estágio do classificador ("maxFalseAlarmRate"), número de imagens positivas e negativas que devem ser utilizadas em cada estágio ("numPos" e "numNeg"), tamanho das amostras utilizadas em *pixels* ("w" e "h").

Outro ponto é que para a versão que utiliza *Haar Cascade* foi elaborado, por meio do banco de dados apresentado na Seção 4.1.1.1, um grupo de imagens positivas e negativas, sendo 534 imagens positivas e 2337 imagens negativas. Alguns exemplos das imagens geradas estão na Figura 4.8.

Figura 4.8: Exemplos de imagens positivas e negativas



Fonte: (PRÓPRIA, 2019)

4.2 VISÃO GERAL DO MÉTODO DE RECONHECIMENTO PROPOSTO

Uma imagem de um autômato é composta por vários objetos. Assim, para realizar o reconhecimento é necessário identificar não só os objetos como também as suas relações. Portanto, neste trabalho, os seguintes dados precisam ser extraídos para que a informação completa da imagem seja obtida:

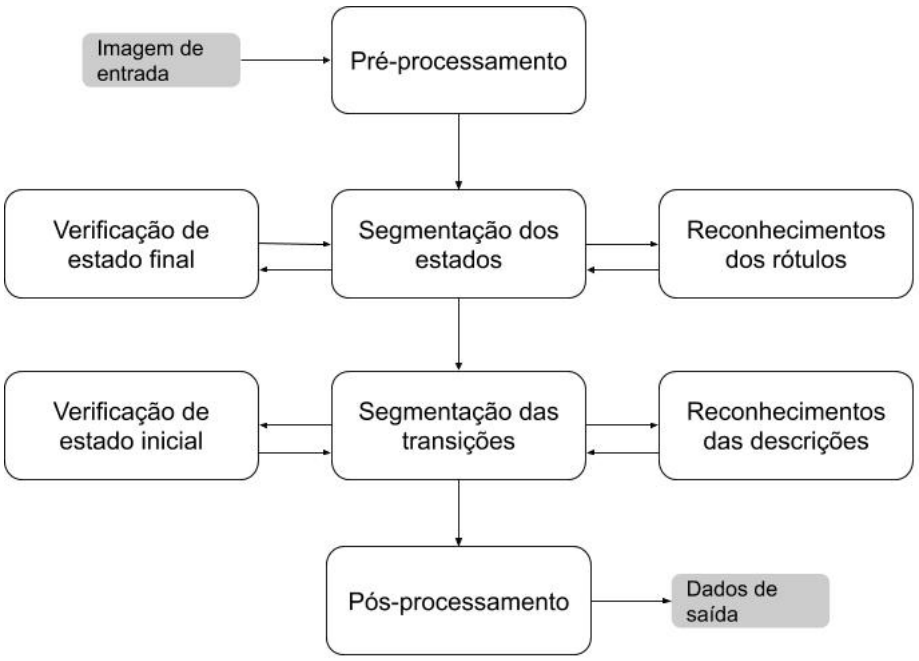
- objetos círculos: que caracterizam os estados ou símbolo de estado final;
- objetos setas: que caracterizam transições entre estados ou símbolo de estado inicial;
- objetos texto: que estão presentes nos rótulos que identificam os estados e nas descrições das transições;
- relações entre setas e círculos: caracterizam uma transição entre dois estados ou um *loop*, ou ainda, indica os estados iniciais;
- relações entre texto e círculos: caracterizam os rótulos dos estados;
- relações entre texto e setas: caracterizam as descrições das transições;
- tipo do autômato: as opções consideradas são Autômato Finito, Autômato de Pilha e Máquina de Turing.

Para que estes dados e informações sejam obtidos foram definidas as seguintes etapas: pré-processamento da imagem, segmentação dos estados, segmentação das transição

e pós-processamento. Na etapa de segmentação dos estados também são reconhecidas outras informações sobre eles, como o rótulo do estado e se o estado é final ou não.

O mesmo processo ocorre de forma semelhante na segmentação das transições, na qual, além de reconhecer as transições, são identificados suas descrições e o símbolo de estado inicial. A Figura 4.9 representa a visão geral do método proposto, sendo que cada etapa é detalhada nas seções subsequentes.

Figura 4.9: Visão geral do método proposto



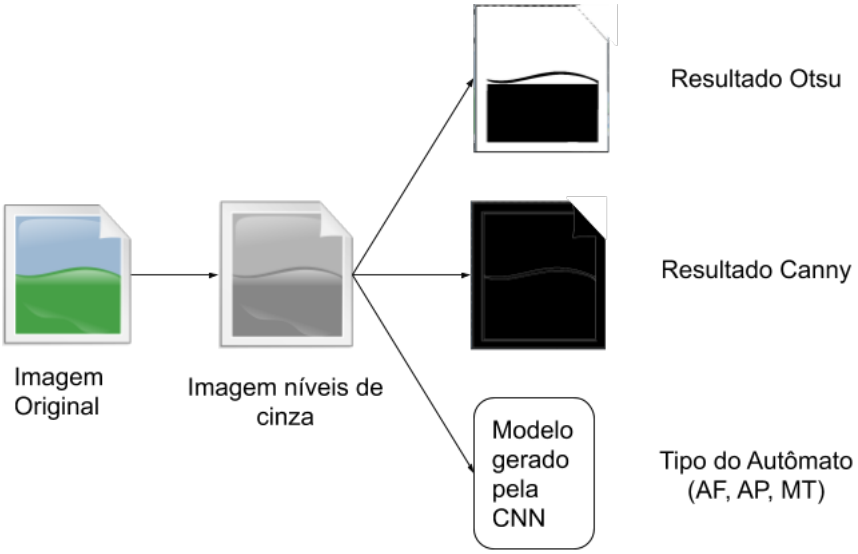
Fonte: (PRÓPRIA, 2019)

4.3 PRÉ-PROCESSAMENTO

O pré-processamento consiste em preparar a imagem para a segmentação dos estados e das transições. Depois do recebimento da imagem pelo sistema, ela é convertida para níveis de cinza para facilitar o processo de segmentação dos objetos e diminuir o processamento de dados.

A imagem gerada é enviada como entrada no modelo de identificação de tipo do autômato (Subseção 4.1.1) e também são geradas duas versões dessa imagem, sendo uma binária e outra com destaque nas bordas. Na Figura 4.10 apresenta-se o diagrama das etapas do pré-processamento.

Figura 4.10: Diagrama das etapas do pré-processamento



Fonte: (PRÓPRIA, 2019)

Como pode-se observar, na Figura 4.10, são geradas duas versões da imagem original. A primeira versão representa a imagem na sua forma binária. Para determinar corretamente o limiar de transformação do *thresolding* é utilizado o método Otsu.

A segunda representa a imagem com suas bordas destacadas, ou seja, a aplicação do método Canny. Posteriormente a esses procedimentos, a imagem em escala de cinza é enviada para o modelo gerado por um conjunto de CNNs, a fim de obter informações que auxiliem na segmentação dos objetos.

4.4 SEGMENTAÇÃO DOS ESTADOS

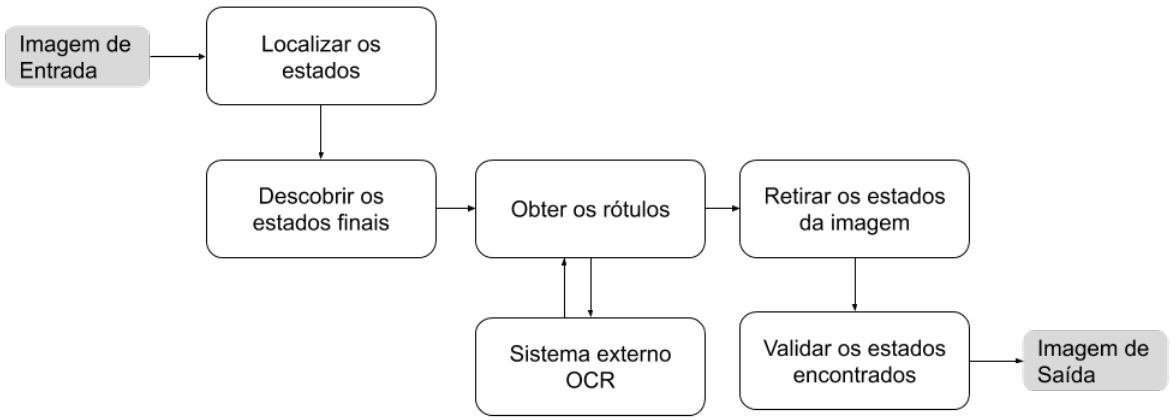
Como já comentado na Seção 4.2, o principal desafio do método é a segmentação correta dos objetos. O primeiro componente que deve ser segmentado é o que compreende os estados e, assim, providenciar que os demais componentes sejam separados. Nesta etapa de segmentação, a entrada de dados consiste na imagem do autômato após o pré-processamento e a saída deve ser a imagem do autômato sem os estados e seus atributos. Cada estado encontrado é representado por um conjunto de atributos, a saber:

- centro do círculo: representado por um par de valores (x, y);
- raio: armazena o tamanho do raio considerando uma valoração inteira.

- rótulo do estado: armazena uma *string* que representa o nome do estado;
- estado inicial: atributo booleano que armazena *True* caso o estado seja inicial e *False* caso não seja;
- estado final/aceitação: atributo booleano que armazena *True* caso o estado seja final e *False* caso não seja.

A etapa de segmentação dos estados consiste nas seguintes sub-etapas: descobrir a localização dos estados, verificação de estado final, obtenção do rótulo e validação dos estados localizados. Na Figura 4.11 apresenta-se o diagrama de etapas da segmentação de estados.

Figura 4.11: Visão geral da segmentação de estados



Fonte: (PRÓPRIA, 2019)

A primeira etapa de localização dos círculos é a única que possui três versões estudadas, pois é o cerne do método e sua eficácia atinge o método como um todo. Cada uma das etapas é detalhada nas próximas Seções.

4.4.1 Localização dos círculos

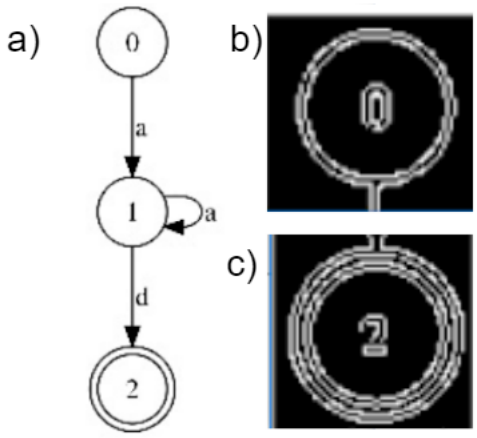
O primeiro passo para conseguir construir a lista de estados é descobrir onde estão os círculos na imagem. A localização dos estados na imagem podem ser realizada por algumas técnicas (detalhadas na Subseção 2.3.2) como: Transformada de Hough para círculos, *Haar Cascade* e *Learning Automata*. Com o estudo realizado (detalhado na Subseção 4.1.2), que possui os resultados dos testes apresentados na Subseção 6.1.2, o método *Learning Automata* foi utilizado como parte do método MannAR devido ao seu melhor desempenho.

A saída dessa primeira etapa deve ser uma lista de possíveis círculos, sendo que cada um é representado pelos atributos $c = (x, y, r)$, na qual x e y representam o ponto central do círculo e r é o raio. Cada círculo também apresenta o seu valor de *beta* que corresponde ao porcentagem de um círculo candidato ser realmente um círculo válido.

4.4.2 Detecção de estados finais

Após a detecção das regiões que possuem círculos, cada uma delas deve ser analisada para verificar se o círculo detectado é um estado final/aceitação. Na Figura 4.12 é possível verificar a diferença de um estado comum (b) e de um estado final (c) para a imagem original (a) após a aplicação do método Canny.

Figura 4.12: Diferença entre um estado comum (b) e um final (c) após aplicação do método Canny para a imagem original (a)



Fonte: (PRÓPRIA, 2019)

Um estado final é composto por dois círculos, porém, ao realizar a detecção de objetos, apenas um círculo é detectado naquela região. Ao final dessa etapa não é possível saber se o objeto identificado corresponde ao círculo interno ou externo. Assim, a primeira etapa do método de verificação de estado final consiste em delimitar a região de busca em volta do círculo encontrado. Para isso, é calculado a posição dos pontos de borda mais distantes do centro do círculo que são representados pelas equações a seguir:

$$a = (x0 - raioMaior) \tag{25}$$

$$b = (x0 + raioMaior) \tag{26}$$

$$c = (y0 - raioMaior) \tag{27}$$

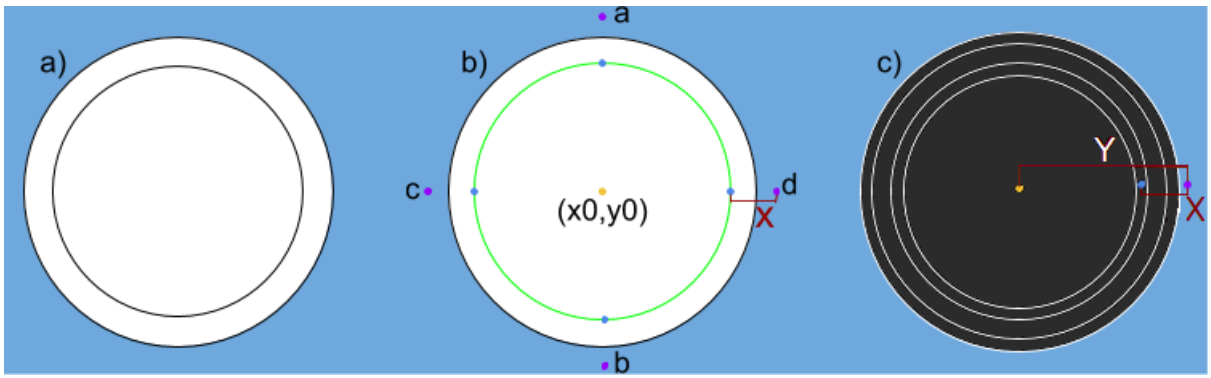
$$d = (y0 + raioMaior) \tag{28}$$

Sendo que o par $(x0, y0)$ representa o ponto central do círculo e a variável *raioMaior* representa o valor do raio do círculo acrescido de um valor X, desta forma é possível realizar a busca em uma região maior do que a do círculo original.

Na Figura 4.13 é apresentado um estado final representado pela imagem (a). Na parte (b) é apresentado o círculo identificado, na qual, neste caso, foi descoberto o círculo interno. Os pontos de borda, no círculo interno, representam os pontos mais longe do centro (considerando as quatro direções, norte, sul, leste, oeste).

Ao realizar a soma do valor X nas equações, tem-se como resultado os pontos a, b, c *e* d, que são representados pelos pontos externos aos círculos. Na parte (c) tem-se a imagem após a aplicação do método Canny, na qual, para cada círculo, foi gerada a borda interna e externa.

Figura 4.13: Exemplo de detecção do estado final



Fonte: (PRÓPRIA, 2019)

O valor de X foi definido como sendo $30\% + 1$ do valor do raio do círculo. A distância identificada por Y na parte (c) da Figura 4.13 representa a área analisada no método de identificação de estados finais. Para cada direção é construído um vetor das posições dos *pixels* brancos encontrados na região delimitada (região Y). Como para cada círculo original são obtidas duas bordas, os vetores que possuem menos de quatro posições não são analisados.

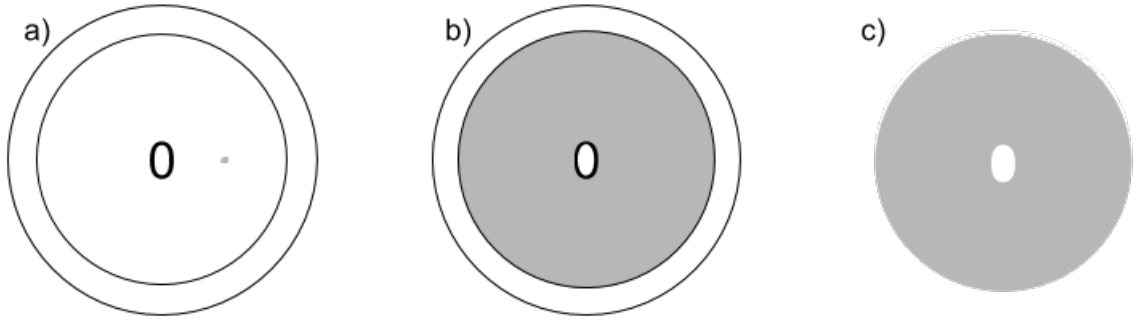
Se houver pelo menos dois lados onde os vetores possuam quatro ou mais *pixels* que estejam perto entre si (considerado uma distância de dois a três *pixels*), considera-se que o círculo analisado é um estado final. O mesmo processo deve se repetir para todos os círculos encontrados no primeiro passo.

4.4.3 Exclusão dos círculos da imagem

Para realizar a exclusão dos estados da imagem original percorre-se cada círculo encontrado novamente. Nessa etapa realiza-se a segmentação da parte interior do círculo. Para isso são necessários os seguintes passos, que são observáveis na Figura 4.14:

1. escolha de um *pixel* branco aleatório interno ao círculo, destacado em cinza para melhor visualização na parte (a) da Figura 4.14;
2. busca dos *pixels* brancos alcançáveis (parte (b) da Figura 4.14), pelo processo de busca em largura (BFS) (CORMEN et al., 2009);
3. separação da parte segmentada em uma nova imagem (parte (c) da Figura 4.14).

Figura 4.14: Processo de segmentação do círculo



Fonte: (PRÓPRIA, 2019)

O *pixel* inicial é escolhido de forma aleatória, entre os *pixels* que estão a uma distância maior do que metade do tamanho do raio, afim de evitar a seleção de um *pixel* interno a um rótulo. Caso o *pixel* sorteado seja preto, um novo sorteio é realizado até que o *pixel* escolhido seja branco. Após a busca em largura, todos os *pixels* identificados são copiados para uma nova imagem.

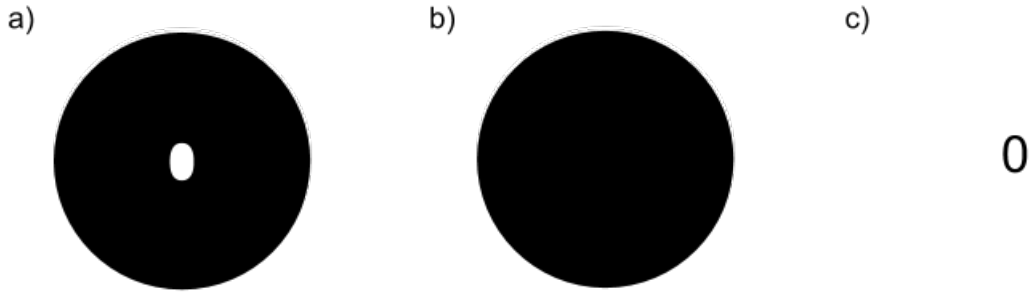
Para que todo o círculo seja preenchido, incluindo áreas internas aos rótulos, é realizado o processo de abertura do objeto, que é, basicamente, um processo de erosão seguido de uma dilatação. Para isto são necessários os seguintes passos, observáveis na Figura 4.15.

1. realização de *thresolding* para que imagem volte a ser binária (parte (a) da Figura 4.15);
2. realização do processo de abertura (parte (b) da Figura 4.15);

3. segmentação do rótulo (parte (c) da Figura 4.15).

Para realizar a segmentação do rótulo, é necessário criar uma imagem branca do mesmo tamanho da original. Na sequência, verifica-se a imagem segmentada *pixel* a *pixel*. Se o *pixel* analisado for preto, então é atribuído, no *pixel* da nova imagem, o valor do correspondente da imagem original. Ao final, tem-se apenas o rótulo do estado na imagem final.

Figura 4.15: Exemplo de segmentação de um rótulo de um estado



Fonte: (PRÓPRIA, 2019)

A imagem que contém apenas o rótulo é enviada para um sistema externo de OCR (*Optical Character Recognition*), que reconhece a imagem e a transforma em texto. Por fim, o mesmo círculo (b) da Figura 4.15 é utilizado para a exclusão do círculo correspondente na imagem original, como mostra a Figura 4.16. Para isso, são realizados os seguintes passos, observáveis na Figura 4.16:

1. inversão da imagem, transformando o que é branco em preto e o que é preto em branco (processo realizado na parte b da Figura 4.15);
2. aplicação do processo de dilatação no círculo segmentado (parte b da Figura 4.16);
3. soma do resultado na imagem original (parte c da Figura 4.16).

Figura 4.16: Exemplo de exclusão de círculo para a imagem original (a)

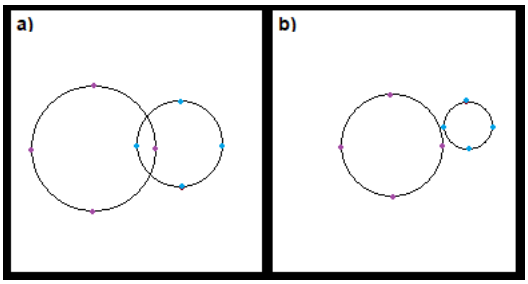
Fonte: (PRÓPRIA, 2019)

4.4.4 Validação dos círculos identificados

Devido à semelhança de alguns *loops* com círculos, em certos casos eles acabaram sendo identificados como estados. Para ajudar na descoberta de quais *loops* foram classificados como estados, são utilizadas as bordas de cada círculo, verificando se elas se cruzam ou estão muito próximas.

A partir do raio e do centro do círculo é possível calcular quais são os pontos mais distantes, como já visto na Subseção 4.4.2, por meio das equações 25, 26, 27 e 28. Com os valores, é possível cruzar as informações com as dos demais círculos e então descobrir quais círculos se cruzam ou estão muito próximos, o que caracteriza um *loop*, como mostra a Figura 4.17.

Figura 4.17: Exemplo de *loops* reconhecidos como estados



Fonte: (PRÓPRIA, 2019)

Na Figura 4.17 é apresentado dois exemplos de casos que *loops* foram identificados como estados, sendo o primeiro de dois círculos que se cruzam na parte (a) e o segundo de dois círculos muito próximos na parte (b). Ao identificar uma dessas duas situações, é necessário identificar qual deles é o *loop* e qual é o estado. Para isso, utiliza-se um sistema de pontuação, apresentado na Tabela 4.5, que se baseia em atributos já extraídos anteriormente, a saber: estado final e rótulo.

Atributo	Pontuação
Estado final	2
Rótulo != ‘ ’	1

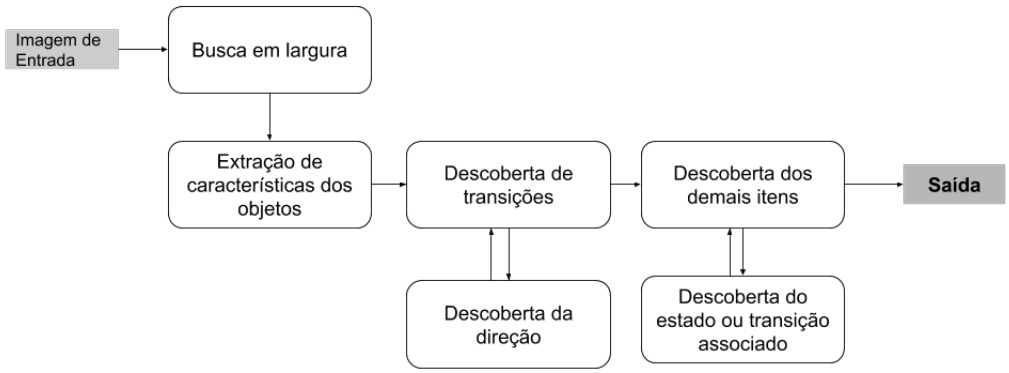
Tabela 4.5: Pontuação para classificação de *loops* e estados

Uma pontuação menor foi atribuída para os rótulos dos estados devido à maior probabilidade de erro em sua identificação, pois há a dependência de um sistema OCR externo. Quando a pontuação para os dois círculos é a mesma, é considerado como *loop* o círculo que possuir a menor área interna.

4.5 SEGMENTAÇÃO DAS TRANSIÇÕES

Com a retirada dos círculos da imagem, é possível dar início ao estágio de segmentação das transições. A entrada dessa etapa deve ser a imagem do autômato sem a presença dos estados e a lista de círculos gerada na fase anterior. Como saída, espera-se uma lista de todas as transições e uma lista de estados iniciais. A Figura 4.18 apresenta a visão geral do procedimento de segmentação das transições.

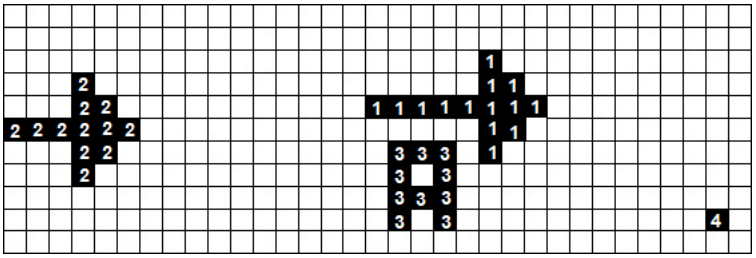
Figura 4.18: Visão geral da segmentação de transições



Fonte: (PRÓPRIA, 2019)

Cada transição é representada pelo par (círculo de saída, círculo de entrada), pelo sua descrição e pelo seu número de objeto, sendo que este último pode ser gerado percorrendo toda a imagem com uma busca em largura e enumerando cada agrupamento de *pixels* pretos encontrado, como mostra a Figura 4.19 .

Figura 4.19: Exemplo da aplicação do método de busca em largura



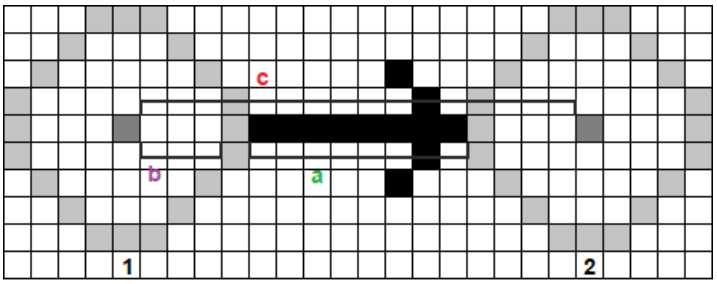
Fonte: (PRÓPRIA, 2019)

Com essa estratégia é possível retirar outras características de cada objeto da imagem, como as que foram apresentadas na Seção 2.3.4, a saber: área, quantidade de *pixels* pretos que compõe o objeto, excentricidade e solidez.

Utilizando as características descritas, os objetos podem ser classificados em: caractere, estado inicial, transição ou ruído. A primeira etapa consiste na separação das transições que estão relacionadas com dois círculos.

Para isto, é necessário verificar se o objeto analisado é maior horizontalmente ou verticalmente. Depois, encontram-se quais os círculos mais próximos dos pontos mais à direita e mais à esquerda - caso o objeto seja maior horizontalmente - ou encontram-se com os círculos mais próximos dos pontos mais alto e baixo - caso o objeto seja maior verticalmente. Se o resultado consistir em dois círculos diferentes, tem-se uma possível transição e então é necessário analisar o tamanho do objeto em questão. A Figura 4.20 a seguir exemplifica este processo.

Figura 4.20: Exemplo de identificação de transições que sejam entre dois estados



Fonte: (PRÓPRIA, 2019)

Observa-se, na figura, dois círculos já reconhecidos representados em cinza. Apesar dos centros estarem representados na imagem, neste momento do processo eles já devem ter sido "apagados". Considerando cada quadrado como um *pixel*, o tamanho do objeto analisado, que é a flecha, é maior horizontalmente. Assim, cada uma das extremidades é mais perto de um círculo diferente. O próximo passo, é comparar o tamanho do objeto com a distância entre os centros dos círculos, para que apenas flechas sejam identificadas e caracteres sejam descartados.

Neste caso, a distância entre os centros é de 16 *pixels* (representado pela linha *c*), e o tamanho da flecha é de oito *pixels* (representado pela linha *a*). Outro ponto é que cada círculo possui raio de três *pixels* (representado pela linha *b*). Portanto, a distância entre a subtração dos centros com os valores dos raios deve ser próxima do tamanho do objeto analisado. No exemplo, temos: $dist - r1 - r2 = 16 - 3 - 3 = 10$. Considera-se uma margem de erro pelas bordas que não foram descontadas. Tem-se, então, que o resultado é próximo do tamanho do objeto, que é oito.

Na sequência, é realizada a identificação do lado da flecha. O método utilizado é o apresentado por Babalola (2015), que realiza a divisão do objeto em quatro partes, e então

compara a soma de *pixels* das extremidades, sendo que o lado que obtiver mais *pixels* é considerado a direção da flecha.

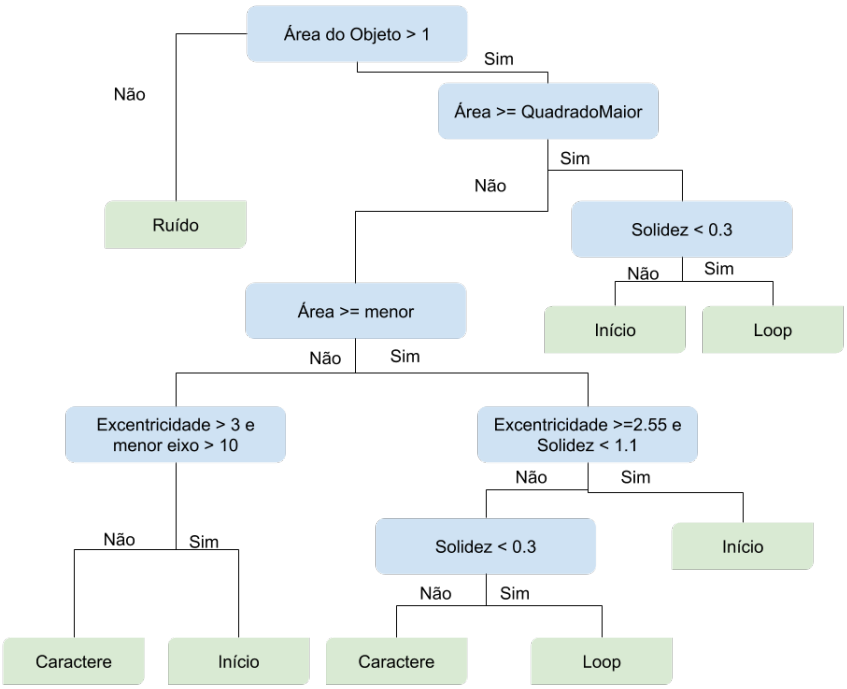
Após esta etapa, têm-se ainda na imagem caracteres, loops, uma ou mais flechas que representam os estados iniciais e ruídos. Para a classificação dos demais itens são calculadas a média e mediana da soma de todas as áreas dos objetos restantes. Com esses valores devem ser definidas as variáveis "menor" e "QuadradoMaior" conforme as Equações 29 e 30, respectivamente:

$$menor = \begin{cases} \text{média} & \text{se média } <= \text{mediana} \\ \text{mediana} & \text{caso contrário} \end{cases} \tag{29}$$

$$QuadradoMaior = \begin{cases} \text{média}^2 & \text{se média } >= \text{mediana} \\ \text{mediana}^2 & \text{caso contrário} \end{cases} \tag{30}$$

A base para realizar a classificações dos demais itens é o trabalho de Babalola (2015), com a diferença de que o autor classifica em linha, curvas, texto e círculos. Já neste trabalho, se classifica em: *loops*, caracteres, símbolo de estado inicial e ruídos. Os demais itens podem ser classificados de acordo com a árvore de decisão, apresentada na Figura 4.21.

Figura 4.21: Árvore de decisão para classificar *loops*, caracteres, estado inicial e ruídos



Fonte: (PRÓPRIA, 2019)

Por fim, cada objeto classificado é associado a um estado ou transição. Para o símbolo de início e *loops* busca-se o círculo mais próximo. Após a inclusão de todos os *loops* na lista de transições, é associada à cada símbolo encontrado a transição mais próxima. Para isto, é realizada uma análise espacial da figura, sendo que a transição mais próxima é buscada da seguinte forma:

1. criação de uma imagem em branco do tamanho da imagem original;
2. adição de todas as transições na imagem criada;
3. para cada símbolo, adiciona-se ele na na imagem criada e percorre-se simultaneamente a 8-vizinhança do objeto, sendo que ao encontrar outro objeto, que necessariamente será uma transição, o símbolo é associado a ele.

Ao final deste processo, todos os elementos da imagem já foram segmentados e devido a análise espacial ser realizada simultaneamente, cada objeto já possui um significado completo. Por exemplo, ao invés de ter identificado apenas que o objeto é uma a transição, ela já possui seu estado de origem, destino e texto associados.

4.6 PÓS-PROCESSAMENTO

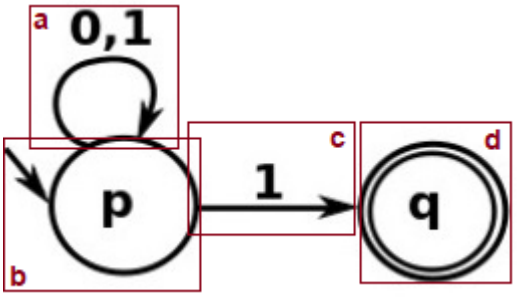
No pós-processamento é realizada a identificação do alfabeto do autômato. Como já se sabe o tipo do autômato, cada um é enviado a um processamento diferente. Para o Autômato Finito basta percorrer todas as descrições das transições e dos *loops*, sendo que quando há mais de um símbolo na transição eles estão separados por ",".

Já para o Autômato de Pilha, como existem várias formas de representar a transição, é necessário primeiramente identificar qual modelo está sendo utilizado. Assim, é possível identificar o alfabeto do autômato e o alfabeto da pilha. O processo inicial é semelhante para a Máquina de Turing, inicialmente identifica-se qual modelo de transição está sendo utilizado e então o alfabeto do autômato e o alfabeto da fita podem ser descobertos.

Nesse processo também é construída uma pequena descrição para os objetos encontrados, por exemplo, ao identificar um estado com o rótulo A, a descrição construída será "Estado A". A Figura 4.22 apresenta exemplos de áreas que são geradas uma descrição.

Os modelos utilizados no método foram os mesmos apresentados na Seção 4.1.1.1, que aborda a criação do database de autômatos. Porém, mais modelos podem ser adicionados.

Figura 4.22: Exemplo de áreas que podem ter uma descrição gerada



Fonte: (PRÓPRIA, 2019)

Na sequência, são apresentadas as descrições que podem ser geradas para as áreas identificados pelos retângulos, a, b, c e d.

- área a: transição do estado p para o estado p com os símbolos 0,1;
- área b: estado inicial p;
- área c: transição do estado p para o estado q com o símbolo 1;
- área d: estado final q.

A descrição de cada área e as características relacionadas aos autômatos, bem como os alfabetos, podem ser úteis na fase de modelagem do autômato extraído no formato alternativo desejado.

4.7 CONSIDERAÇÕES FINAIS

Este tópico apresentou o método desenvolvido para o reconhecimento de imagens de autômato. Como a descoberta dos estados é essencial para o bom funcionamento do método, optou-se por testar três formas diferentes de encontrar os círculos. A forma final foi decidida de acordo com os desempenhos dos testes que são apresentados no Tópico 5.7.

Apesar do processo empregado na implementação das CNNs utilizadas, para descoberta do tipo de autômato e quantidade de estados, terem sido apresentadas nos processos preliminares ao método, faz parte do método MannAR apenas os modelos de classificação gerados.

O método proposto particiona a segmentação em dois módulos gerais , sendo o primeiro a segmentação de estados e o segundo a segmentação das transições. Cada um desses módulos identifica também os atributos e realiza a análise espacial do objeto na imagem, buscando já prover um significado.

O método também proporciona que cada parte seja construída como um módulo proporcionando que, desde que a entrada necessária seja fornecida, o módulo funcione de forma independente.

5

O PROTÓTIPO MANNAR PARA AUTÔMATOS FINITOS

Este Tópico apresenta a descrição do desenvolvimento de um protótipo Web utilizando o método de reconhecimento de imagens de autômatos apresentado no Tópico 4.

A primeira versão do protótipo envolve os Autômatos Finitos. As versões destinadas a APs e MTs não estão contempladas no escopo desta dissertação considerando o tempo e o esforço de desenvolvimento. Também verificou-se a necessidade da realização de um estudo mais aprofundado sobre versões alternativas de representação de Autômatos de Pilha e Máquinas de Turing em um formato textual.

A primeira versão do protótipo foi denominado MannAR-FA (*Manna Automata Recognition - Finite Automata*). Sua construção foi norteada pelas diretrizes de desenvolvimento acessível da W3C que foram apresentadas na Subseção 2.2.1.

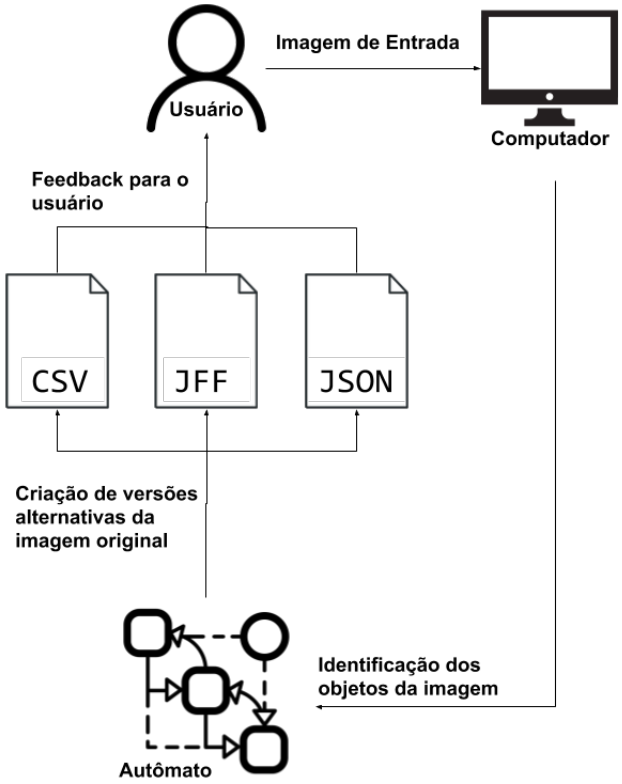
Na próxima seção apresenta-se a visão geral do protótipo, destacando as características e o escopo do sistema. Depois aborda-se o usuário, a interface, o servidor, o método e as versões alternativas de resposta do sistema MannAR-FA.

5.1 VISÃO GERAL DO PROTÓTIPO MANNAR-FA

O protótipo MannAR-FA consiste em um sistema Web, que permitirá ao usuário enviar uma imagem de Autômato Finito. Esta imagem será convertida para três possíveis versões

alternativas A Figura 5.1 apresenta a visão geral do protótipo.

Figura 5.1: Visão geral do protótipo MannAR-FA



Fonte: (PRÓPRIA, 2019)

Conforme observa-se na Figura 5.1 , o usuário envia uma imagem de um Autômato Finito e na sequência o sistema identifica os estados e as relações entre eles e então transforma os dados obtidos em representações alternativas, sendo elas:

- arquivo em extensão .csv: arquivo texto que possui os dados separados por vírgulas. É possível abrir este arquivo em editores de texto e planilhas;
- arquivo em extensão .jff: arquivo xml compatível com o software JFLAP (descrito na subseção 3.2.1);
- arquivo em extensão .json: arquivo json compatível com sistema háptico MannaHap que está sendo desenvolvido em parceria com esta pesquisa.

O sistema MannAR-FA é uma troca de informações entre os elementos usuário, interface do método de reconhecimento, o método em si, servidor que armazena o método e versões alternativas do autômato geradas pelo método. Tais elementos são descritos a seguir:

- usuário: pessoa com deficiência visual, ou não, que deseja obter uma versão alternativa de uma imagem de Autômato Finito.
- interface do método de reconhecimento: página Web que permite ao usuário enviar a imagem da qual se deseja obter uma versão alternativa, além de visualizar os resultados obtidos;
- servidor de processamento: local que hospeda o método de reconhecimento em si, recebendo a imagem enviada pela interface e retornando os resultados para a mesma;
- método de reconhecimento: núcleo do sistema que realiza a conversão de uma imagem de autômato para versões alternativas;
- versão alternativa de um AF: representação de um autômato fornecida pelo método (.csv, .jff e .json). É necessário um *software* externo que seja compatível com algum dos formatos disponíveis para visualização da resposta.

Nas próximas Seções são detalhados cada um dos elementos supracitados, sendo possível uma melhor compreensão sobre as características que compõem o protótipo.

5.2 O USUÁRIO DO MANNAR-FA

O usuário do protótipo MannAR-FA é qualquer pessoa que deseja transformar a imagem de um Autômato Finito para uma das versões alternativas disponíveis. Apesar disso, o foco do sistema são usuários com deficiência visual.

Segundo a classificação ICD-9-CM da *National Center for Health Statistics* (2009), existem sete níveis de deficiência visual, sendo que as três categorias classificadas com menor acuidade visual - baixa visão profunda, próximo a cegueira e cegueira total - necessitam de auxílios adicionais, na qual se adequa o MannAR-FA. Desse modo, pessoas que se enquadram a essas categorias são usuários em potencial do sistema.

Não há restrições de idade para o uso da aplicação, apenas espera-se que o usuário consiga compreender o que são imagens e figuras, como círculos e flechas.

Para uma melhor compreensão e manipulação do sistema, espera-se que o usuário possua experiência básica na utilização de computadores e esteja familiarizado com os conceitos teóricos de Autômato Finito.

5.3 INTERFACE DO MANNAR-FA

A interface do sistema MannAR-FA é parte integrante de um sistema Web. Ela fornece os recursos necessários para que o usuário utilize o sistema, como por exemplo: página para o envio da imagem de um autômato a ser reconhecida, página com informações gerais do sistema e contato. A principal funcionalidade do sistema está na página inicial, que possibilita ao usuário o envio da imagem que ele deseja transformar em uma versão alternativa para o servidor de processamento. O desenvolvimento da interface foi realizado em HTML (*HyperText Markup Language*) e CSS (*Cascading Style Sheets*).

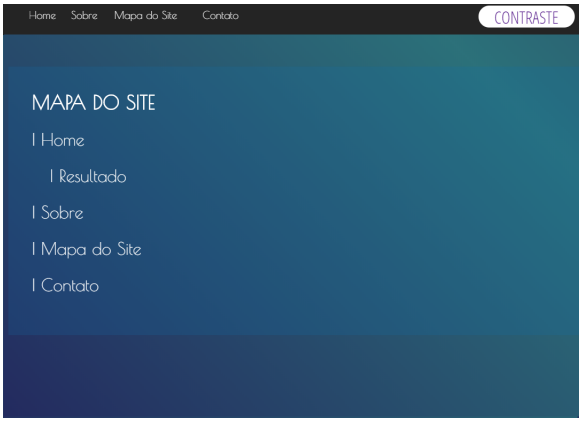
A principal interação do usuário é realizada na página inicial (parte (a) da Figura 5.2), na qual é possível enviar a imagem desejada. Após esse processo, é exibida uma tela com os resultados gerados (parte (b) da Figura 5.2) pelo método de reconhecimento de autômatos.

Figura 5.2: Interface das páginas inicial e resultados do sistema MannAR-FA

Fonte: (PRÓPRIA, 2019)

Como o foco principal do sistema são usuários com deficiência visual, seguiram-se no desenvolvimento as diretrizes estabelecidas pela WCAG. Uma delas recomenda que sejam fornecidas maneiras alternativas de ajudar o usuário a determinar onde ele está na página. Então, adicionou-se à página mapa do site. Um mapa do site é uma lista hierárquica com as páginas existentes no sistema, conforme mostra a Figura 5.3.

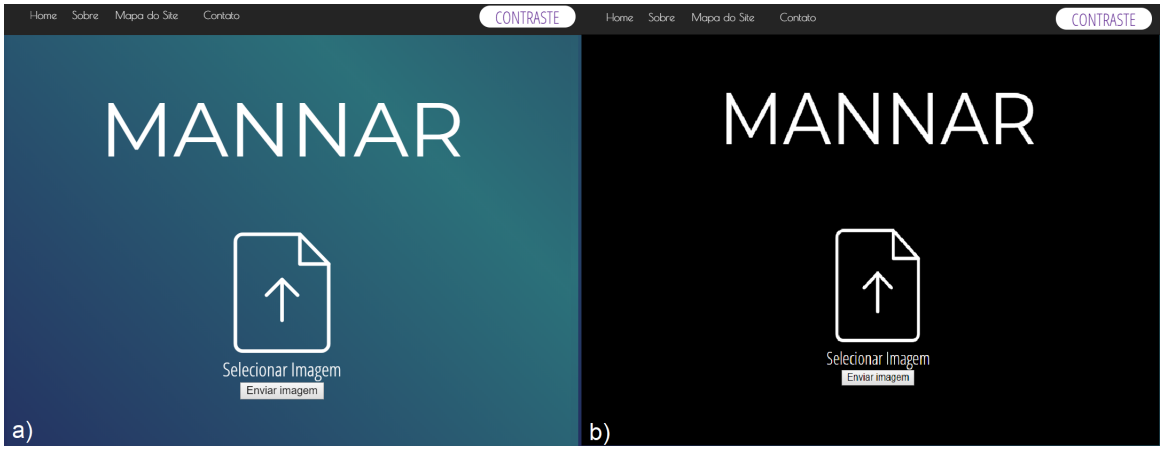
Figura 5.3: Mapa do site contendo as páginas do sistema MannAR-FA



Fonte: (PRÓPRIA, 2019)

Outra função adicionada foi a opção de alto contraste, que realiza a troca do CSS para uma versão na qual a visualização dos elementos da página é facilitada pelo sistema de cores utilizado. Na Figura 5.4 apresenta-se a diferença entre a página original (parte (a)) e a página utilizando o layout com alto contraste (parte (b)).

Figura 5.4: Diferença entre a página inicial com e sem alto contraste



Fonte: (PRÓPRIA, 2019)

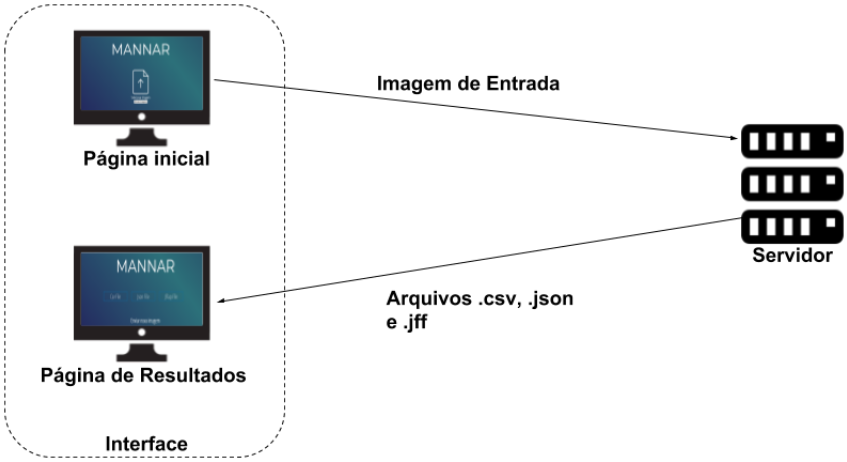
Durante a construção das páginas buscou-se utilizar um layout simples para que os programas leitores de tela funcionassem de forma adequada. Também foram adicionadas às páginas “Contato” e “Sobre”, que detalham informações a respeito do sistema, no geral, como autoria, propósito e informações sobre a utilização do sistema.

5.4 SERVIDOR DE PROCESSAMENTO DO MANNAR-AF

A comunicação da interface com o método de reconhecimento é realizada por meio de uma arquitetura cliente-servidor. O servidor foi implementado utilizando JSON, Python e a biblioteca Tornado[1], que possibilita a criação de aplicações Web.

O usuário interage com o sistema por meio da interface. Como mostra a Figura 5.5, a imagem de autômato que se deseja reconhecer é enviada utilizando a página inicial do sistema.

Figura 5.5: Arquitetura cliente-servidor utilizada no sistema MannAR-FA



Fonte: (PRÓPRIA, 2019)

Após o envio da imagem, o servidor que a recebe realiza o processamento necessário para que todos os elementos da imagem sejam reconhecidos. Com a obtenção da resposta, o usuário é direcionado para a página de resultados, na qual ele tem acesso aos arquivos gerados como resposta pelo sistema.

5.5 MÉTODO DE RECONHECIMENTO DE AF

O protótipo do MannAR-FA implementa o método descrito no Tópico 4. O protótipo foi construído na linguagem Python na versão 3.6.4 com o auxílio das seguintes bibliotecas:

- OpenCv[2] (*Open Source Computer Vision Library*) - versão 3.3.1: apoia o desenvolvimento de aplicações no campo da visão computacional;

[1] https://www.tornadoweb.org/en/stable/
[2] https://opencv.org/

- Numpy[3] - versão 1.14.0: permite o uso de *arrays* e matrizes multidimensionais;
- Networkx[4] - versão 2.1: permite a criação e manipulação de gráficos e redes.

A implementação do método constitui as etapas de pré-processamento, segmentação de estados, segmentação de transições e pós-processamento.

Devido ao fato do protótipo ser apenas para Autômatos Finitos, não foi utilizado o modelo de CNN construído. Então, o pré-processamento constituiu-se apenas da aplicação dos métodos Otsu e Canny na imagem original.

Para representar o autômato a ser identificado na imagem, utilizou-se a biblioteca *Networkx*, que possui métodos para a criação de um grafo dirigido, cuja estrutura modelou o autômato. Assim, para cada estado identificado foi construído um nó com os seguintes atributos:

1. nome: código de identificação do estado;
2. rótulo do estado: nome do estado obtido da imagem original;
3. estado inicial: indica se o estado é inicial ou não;
4. estado final/aceitação: indica se o estado é final ou não;
5. centro do círculo: par ordenado de *pixels* que indica o centro do estado na imagem original;
6. raio: raio do círculo que representa o estado;
7. descrição: texto que descreve o estado;
8. imagem: imagem do objeto, representada por uma matriz de *pixels*. Considera-se o retângulo envolvente do objeto como a imagem a ser salva;
9. coordenadaInicial: coordenada do *pixel*, na imagem original, que representa o início da região do objeto.

O atributo 1 é um código único para cada estado. Os atributos de 2 a 7 foram implementados de acordo com a necessidade do método de reconhecimento. Já os atributos 8 e 9 foram necessários para realização da conversão do autômato para o formato JSON, que é um dos formatos alternativos escolhidos.

[3] http://www.numpy.org/
[4] https://networkx.github.io/

Para representar as transições com a biblioteca *Networkx*, foi construída uma aresta com os seguintes atributos:

- círculo saída: estado de saída relacionado à transição;
- círculo entrada: estado de entrada relacionado à transição;
- descrição: texto que descreve a transição;
- imagem: imagem do objeto, representada por uma matriz de *pixels*. Considera-se o retângulo envolvente do objeto como a imagem a ser salva;
- coordenadaInicial: coordenada do *pixel*, na imagem original, que representa o início da região do objeto.

De forma semelhante a criação dos nós, os atributos 1 a 4 foram implementados de acordo com a necessidade do método de reconhecimento e os atributos 5 e 6 são demandados pelo formato JSON, um dos modelos escolhidos como representação alternativa da imagem do AF.

Para gerar os *labels*, tanto do estado como da transição, utilizou-se um sistema externo de OCR. No protótipo foi empregada a biblioteca *Python-tesseract*[5] (*pytesseract*), que é específica para a utilização de OCR no Python. Ela permite o acesso a *Engine Tesseract-OCR* do Google.

Foram escolhidas três formas de resposta final (arquivos .csv, . jff e .json), sendo que para cada uma foi criado um módulo diferente para geração do arquivo de resposta, que são detalhados na Seção 5.6.

5.6 VERSÕES ALTERNATIVAS DE UM AF

Cada usuário possui uma necessidade específica. Mesmo com as mesmas deficiências, existem variações na forma de aprendizado e preferência de cada um. Foram escolhidas três versões alternativas de um AF para serem disponibilizadas como resultado, a saber: arquivo .csv, arquivo .jff e arquivo .json.

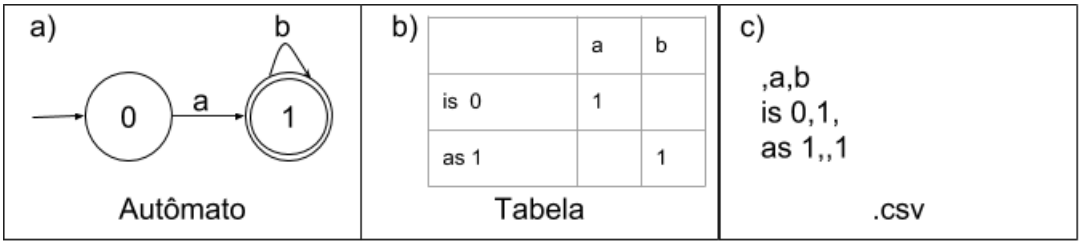
Cada um possui um propósito. A primeira é um arquivo .csv que possui duas funções: permitir que o usuário possa ter acesso ao resultado utilizando ferramentas que ele já possui no

[5]https://pypi.org/project/pytesseract/

computador, como editores de planilhas e até mesmo um bloco de notas; e facilitar a importação por outras ferramentas.

Para implementar este módulo foi utilizada a biblioteca *csv* no Python, que permite a manipulação de arquivos com este formato. O arquivo gerado contém o AF no formato de tabela, com as adições das siglas "as" (*accept state*) para estado final e "is" (*initial state*) para estado inicial. A Figura 5.6 apresenta o autômato finito (a), no formato de tabela (b), e no formato csv (c).

Figura 5.6: Exemplo de arquivo csv (c) para a tabela (b) do autômato (a)



Fonte: (PRÓPRIA, 2019)

Cada linha do arquivo .csv representa uma linha da tabela, na qual cada coluna é separada por uma vírgula.

O segundo formato escolhido foi o .jff, que é o formato utilizado pela ferramenta JFLAP. Disponibilizar o arquivo neste formato permite que usuários com deficiência visual total possam manipular o autômato, se utilizada a versão com acessibilidade. Porém, dependendo do grau da deficiência visual, é possível que seja utilizada também a versão tradicional da ferramenta.

Outra função é que qualquer pessoa, seja com deficiência visual ou não, pode utilizar esse formato para estudar de forma autônoma. O JFLAP possui funcionalidades de simulação do autômato, conversão de não determinístico para determinístico, entre outras. Esse formato permite que a ferramenta atenda aos requisitos do Design Universal, pois permite uma utilidade da mesma para qualquer pessoa.

O arquivo .jff é um arquivo XML no formato exigido pela ferramenta Jflap, necessitando das seguintes informações:

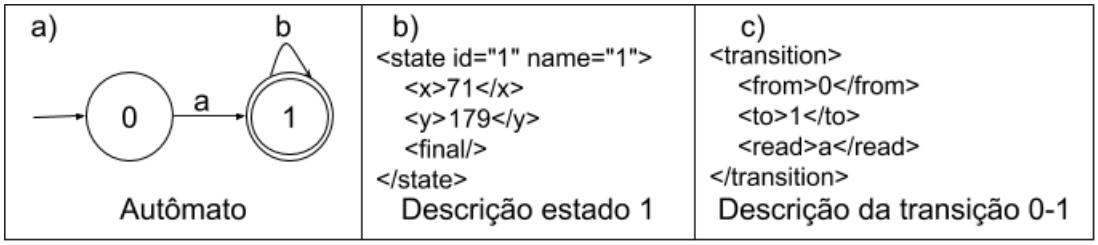
- tipo do autômato: representado pela sigla FA (*finite automata*);
- lista de autômatos: cada autômato possui obrigatoriamente um id único, um nome, que é o rótulo do estado e a posição central do estado representado pelo par ordenado (x,y).

Além disso, pode possuir os atributos *initial* e/ou *final* para indicar que o estado é inicial e/ou final, respectivamente;

- lista de transições: cada item da lista possui o estado de saída e o estado de entrada da transição, representado pelo id único estabelecido na lista de estados e a descrição da transição.

A implementação deste módulo foi feita em Python utilizando a biblioteca *Document*, que possui funcionalidades para a criação de arquivos XML. O arquivo gerado é compatível com a última versão estável do JFLAP, que é a 7.1. Na Figura 5.7 tem-se a representação de um estado (parte b) e de uma transição (parte c), retirados do autômato na parte "a", no formato .jff.

Figura 5.7: Exemplo da descrição em arquivo jff para um estado (b) e uma transição (c) retiradas do autômato (a)



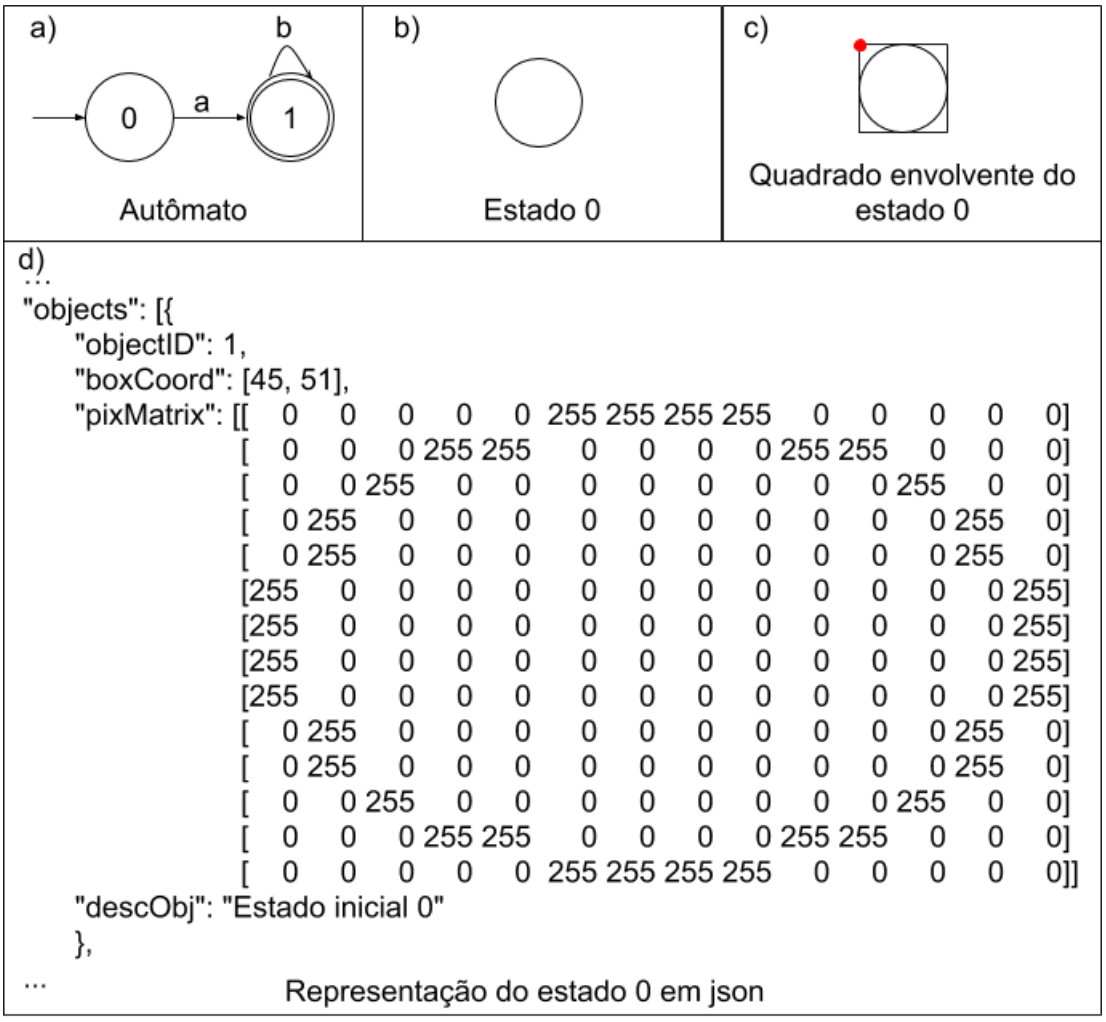
Fonte: (PRÓPRIA, 2019)

O último formato de arquivo disponibilizado é o .json, com as especificações do projeto MannaHap, conforme detalhado na Seção 3.3. A implementação deste módulo foi realizada utilizando Python com a biblioteca JSON.

Para este formato são informados os dados gerais da imagem como quantidade de colunas, linhas e uma lista de objetos da imagem. Cada objeto possui os seguintes elementos: um id único; coordenada de origem; matriz de pixel; e descrição do objeto.

A Figura 5.8 apresenta, como exemplo, a representação do estado 0 (b), da imagem original (a), no formato .json (d). Também apresenta-se o quadrado envolvente do estado 0, que é a área representada no item matriz de *pixels* do objeto (atributo "pixMatrix") no arquivo json, na qual o ponto representa a coordenada inicial que é atribuída ao atributo "boxCoord".

Figura 5.8: Exemplo de descrição do estado 0 (b), retirado da imagem original (a), utilizando o seu quadrado envolvente (c), em json (d)



Fonte: (PRÓPRIA, 2019)

Disponibilizar o arquivo neste formato permite que o usuário, caso tenha acesso à tecnologia MannaHap, complemente a experiência da visualização do autômato também por meio do tato.

5.7 CONSIDERAÇÕES FINAIS

Neste Tópico foram apresentados os conceitos, as características gerais e a concepção do protótipo MannAR-FA, que utiliza o método proposto no Tópico 4.

A primeira versão do protótipo contempla as imagens digitais de Autômatos Finitos Porém, após a realização deste estudo, APs e MTs poderão ser inseridos no sistema, pois o

mesmo já dá suporte a identificação destes reconhecedores de linguagens formais.

O protótipo construído é um sistema Web devido ao fato de existirem diretrizes de acessibilidade bem definidas pela comunidade direcionadas ao desenvolvimento acessível.

Cada usuário possui habilidades, experiências e necessidades diferentes. Tal fato influenciou a escolha de disponibilizar mais de um formato do autômato reconhecido como resposta. O primeiro formato, que é o arquivo .csv, visa a independência do usuário, dando liberdade para o acesso ao conteúdo utilizando seu software de preferência e também permite que outros sistemas utilizem a resposta fornecida com facilidade.

O segundo formato, que é o arquivo .jff (arquivo da ferramenta JFLAP), é utilizado para que o sistema desenvolvido seja compatível com os princípios do Design Universal, já que a ferramenta JFLAP possibilita simulações e conversões de autômato que podem ser úteis para o estudo de qualquer pessoa, sendo ela com ou sem deficiência.

Por fim, o terceiro formato abordado é o arquivo .json, que possibilita que, desde que o usuário tenha a acesso a ferramenta MannaHap, o entendimento do autômato posso ser complementado por meio do tato. Apesar destas escolhas, o sistema permite que sejam inseridos outros modelos de resposta.

6

RESULTADOS E DISCUSSÕES

Este Tópico apresenta os resultados de cada etapa do trabalho desenvolvido. Eles estão divididos em três segmentos, a saber: resultados relacionados aos processos preliminares de elaboração do método MannAR, resultados de avaliação do método MannAR e resultados de avaliação do protótipo MannAR-FA.

Os resultados relacionados ao processo de elaboração do método visam apoiar as decisões tomadas para a construção da versão final do MannAR. Assim, na Seção 6.1, apresentam-se os resultados do estudo utilizando CNN e da comparação entre os métodos de localização de círculos.

Na Seção 6.2, apresentam-se os resultados que envolvem a avaliação do método MannAR. Para isto, são realizados testes quantitativos, que avaliam se o método acertou os itens: estados, transições, estados finais, estados iniciais e *loops* em um conjunto de imagens de AF. Na sequência, são apresentados testes com usuários reais que visam dois objetivos. O primeiro é avaliar o resultado do método, ou seja, verificar se um usuário consegue entender corretamente a principal versão alternativa do método que é o arquivo .csv. A segunda é permitir que os usuários conheçam melhor o MannAR para que possam dar sugestões de melhorias no método. Ao final, apresenta-se uma comparação do método MannAR com o método proposto por Babalola (2015).

Por fim, na Seção 6.3, são apresentados os resultados da avaliação da ferramenta MannAR-FA. Inicialmente, é apresentada a avaliação automática dos critérios da W3C para o código da ferramenta. Na sequência, apresenta-se o teste com usuários reais que foram: teste das recomendações da W3C e testes de usabilidade. Na última subseção, é apresentado o

resultado de integração do MannAR-FA com a ferramenta JFLAP.

6.1 RESULTADOS DE PROCESSOS PRELIMINARES PARA DESENVOLVIMENTO DO MÉTODO MANNAR

Os resultados relacionados ao processo de elaboração do método de reconhecimento foram divididos em duas etapas. A primeira etapa realizada, na construção de um método que fosse capaz de identificar AFs, APs e MTs foi o estudo da classificação de imagens utilizando CNN. O processo de treinamento e resultados são discutidos na Subseção 6.1.1.

Na segunda etapa foi realizada uma comparação entre os métodos de localização de círculo Hough, *Learning Automata* e *Haar Cascade*. Os resultados do desempenho de cada um, discussões e a escolha de qual deles foi utilizado na construção da versão final do método e protótipo são apresentadas na Subseção 6.1.2.

6.1.1 Classificação utilizando CNN

Seguindo a metodologia abordada na Figura 4.1, o banco de dados apresentado na Subseção 4.1.1.1 foi dividido em 10 *folds* iguais, sendo 10×1344 imagens para a abordagem de tipos de autômato e 10×896 imagens para a abordagem de quantidade de estados do autômato. Assim, o processo de teste foi realizado 10 vezes, sendo que em cada vez um *fold* diferente foi utilizado como teste e todos os outros foram usados no treinamento. A resposta final é a média de acertos de todos os testes realizados.

Como este é o primeiro teste realizado com este banco de dados optou-se por treinar a rede a partir do zero. O computador utilizado tinha as seguintes configurações: Intel Core i5-7400, 8GB DDR4 2133Mhz, GPU Nvidia 1050Ti e SSD 120GB. O tamanho do *batch*, ou seja, quantas imagens a rede treina de cada vez, foi definido como 32, para que o computador não travasse ou ficasse muito lento, porque ficou sem memória RAM. Algumas outras especificações utilizadas no treinamento foram:

- os modelos foram treinados para o mesmo número de épocas (foram utilizadas 45);
- a taxa de aprendizado utilizada foi de 10^{-4};
- todos os *bias* foram setados em 0,05 e os pesos foram inicializados aleatoriamente utilizando uma distribuição normal, na qual o desvio padrão foi de 0,05;

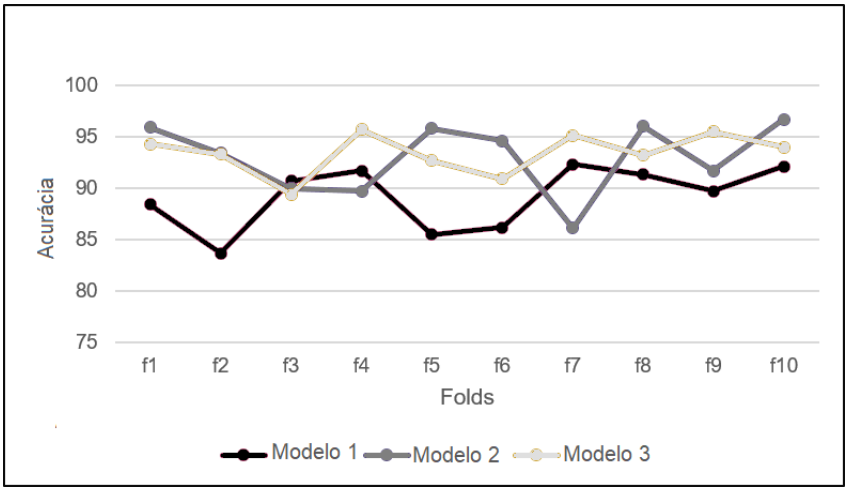
- em todas as sessões de treinamento, 10% da base de treinamento foi separada para validação.

Visando potencializar a acurácia do modelo criado foi utilizada a técnica *Late Fusion*. Foram testadas todas as combinações de CNN utilizadas, a técnica de *Late Fusion* foi feita pela combinação de CNNs considerando todas as combinações de pares e também usando os três modelos de CNN criados. Em todos os casos, as regras soma, produto, máximo e mínimo são usadas. Como também usamos validação cruzada, a combinação dos resultados em cada dobra é feita e, em seguida, as precisões médias são calculadas.

Assim, nas duas abordagens utilizadas, sendo elas tipo de autômato e quantidade de estados do autômato, foram utilizados os mesmos processos, como segue: validação cruzada (10 *folds*), o teste com os três modelos CNN (Modelo 1 - cinco camadas; Modelo 2 - oito camadas; Modelo 3 - dez camadas) e a combinação de resultados usando *Late Fusion*.

Para a base utilizada no reconhecimento de tipo do autômato, que teve três classes, a precisão média dos *folds* para cada uma das três classificações resultou em uma precisão final de 89,16% para a arquitetura com cinco camadas, 92,99% para a arquitetura com oito camadas e 93,41% para a arquitetura com dez camadas. O desempenho da execução de cada *fold* para cada versão de CNN pode ser visto na Figura 6.1

Figura 6.1: Performance da execução de cada *fold* para a abordagem tipo de autômato



Fonte: (PRÓPRIA, 2019)

No gráfico é possível observar que não houve uma rede que se destacou, mas o Modelo 3 foi o que obteve melhor desempenho. Para melhorar os resultados, as regras de soma, produto,

máximo e mínimo foram testadas como *Late Fusion*. Na Tabela 6.1 é possível observar a comparação dos modelos e a fusão entre eles.

Tabela 6.1: Resultado da acurácia dos classificadores utilizados no reconhecimento do tipo do autômato

Classificadores	**Regras do *Late fusion***	**Acurácia (%)**
Modelo 1	-	89.16 ± 3.07
Modelo 2	-	92.99 ± 3.51
Modelo 3	-	$\mathbf{93.41 \pm 2.02}$
Late fusion modelos 1 e 2	Soma	96.34 ± 1.02
	Produto	96.45 ± 0.94
	Máx	96.27 ± 1.06
	Min	$\mathbf{96.48 \pm 0.92}$
Late fusion modelos 1 e 3	Soma	96.13 ± 0.95
	Produto	96.26 ± 0.87
	Máx	96.05 ± 0.89
	Min	$\mathbf{96.31 \pm 0.84}$
Late fusion modelos 2 e 3	Soma	96.34 ± 1.21
	Produto	96.46 ± 1.15
	Máx	96.31 ± 1.27
	Min	$\mathbf{96.47 \pm 1.17}$
Late fusion modelos 1, 2 e 3	Soma	97.02 ± 0.61
	Produto	$\mathbf{97.12 \pm 0.45}$
	Máx	96.19 ± 1.81
	Min	96.87 ± 0.61

Fonte: (PRÓPRIA, 2019)

Todas as opções testadas obtiveram desempenho similar. O melhor resultado foi 97,12% para o *Late Fusion* das três CNNs pela regra do produto, sendo que esse resultado é muito próximo ao obtido no *Late Fusion* pela regra da soma. Entretanto, o desvio padrão da regra do produto foi menor que o da regra por soma, mostrando uma maior homogeneidade dos resultados.

Na Tabela 6.2 é mostrado a matriz de confusão da fusão do melhor resultado (*Late Fusion* usando a regra produto). A matriz de confusão foi construída usando a soma de todas as matrizes geradas em cada *fold*. A classe em que houve mais acertos foi Máquina de Turing e a que obteve menos foi Autômato Finito.

Tabela 6.2: Matriz de confusão para o melhor resultado do reconhecimento de tipo de autômato

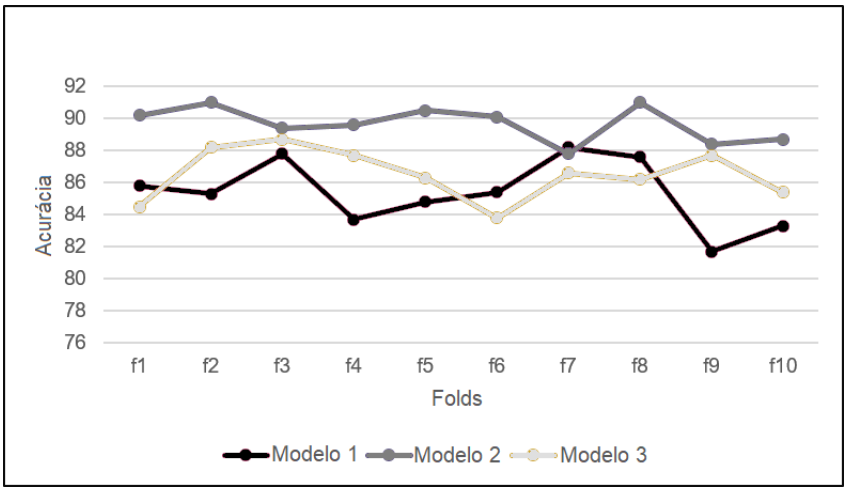
	AF	**AP**	**MT**	**Total**
FA	4241	90	149	4480
PDA	101	4355	24	4480
TM	16	5	4459	4480
Total	4358	4450	4632	**13440**

Fonte: (PRÓPRIA, 2019)

Considerando o problema do reconhecimento do número de estados do autômato, que possuía oito classes, as acurácias foram: 85,36% para arquitetura de cinco camadas, 89,67% para a arquitetura de oito camadas e 86,51% para a arquitetura de dez camadas.

Ao contrário do caso anterior, que nenhum modelo se destacou, o modelo de oito camadas obteve um desempenho superior em quase todas os *folds*, como pode ser observado na Figura 6.2, que representa a acurácia média da execução de cada *fold* para cada modelo de CNN.

Figura 6.2: Performance da execução de cada *fold* para a abordagem quantidade de estados do autômato



Fonte: (PRÓPRIA, 2019)

A Tabela 6.3 apresenta a matriz de confusão para o melhor resultado que foi do *Late Fusion* que foi pela regra do produto. O menor número de acertos foi para a classe de quatro estados e o que teve maior número de acertos foi nos autômatos com um estado.

Tabela 6.3: Matriz de Confusão para o melhor resultado do reconhecimento de números de estados de um autômato

	1 estado	**2 estados**	**3 estados**	**4 estados**	**5 estados**	**6 estados**	**7 estados**	**8 ou mais estados**	**Total**
1 estado	1064	14	4	7	6	3	17	5	1120
2 estados	38	1015	13	20	6	15	11	2	1120
3 estados	10	23	1044	5	7	12	8	11	1120
4 estados	4	28	19	993	32	20	12	12	1120
5 estados	4	6	13	10	1029	27	11	20	1120
6 estados	7	7	12	14	23	1008	24	25	1120
7 estados	9	11	12	9	14	28	1019	18	1120
8 ou mais estados	3	4	4	6	8	26	37	1032	1120
Total	1139	1108	1121	1064	1125	1139	1139	1125	**8960**

Fonte: (PRÓPRIA, 2019)

A Tabela 6.4 mostra a comparação entre os resultados das acurácias dos modelos de classificadores utilizados e as acurácias das regras usadas como *Late Fusion*.

Tabela 6.4: Resultado da acurácia dos classificadores utilizados para o reconhecimento de número de estados do autômato

Classificadores	**Regras do *Late fusion***	**Acurácia (%)**
Modelo 1	-	85.36 ± 2.11
Modelo 2	-	$\mathbf{86.67 \pm 1.10}$
Modelo 3	-	86.51 ± 1.61
Late fusion modelos 1 e 2	Soma	90.61 ± 0.61
	Produto	$\mathbf{90.74 \pm 0.62}$
	Máx	90.45 ± 0.68
	Min	90.55 ± 0.49
Late fusion modelos 1 e 3	Soma	89.31 ± 1.06
	Produto	89.36 ± 0.73
	Máx	89.06 ± 1.31
	Min	$\mathbf{89,40 \pm 0.97}$
Late fusion modelos 2 e 3	Soma	90.91 ± 0.70
	Produto	$\mathbf{91.05 \pm 0.60}$
	Máx	90.74 ± 0.67
	Min	91.02 ± 0.70
Late fusion modelos 1, 2 e 3	Soma	91.56 ± 0.71
	Produto	$\mathbf{91.69 \pm 0.85}$
	Máx	90.93 ± 0.75
	Min	90.94 ± 0.73

Fonte: (PRÓPRIA, 2019)

O processo de *Late Fusion* com as regras soma, produto, regras do máximo e do mínimo foram realizadas da mesma forma que no método anterior. Novamente, todos os resultados obtidos foram semelhantes, e a regra do produto no *Late Fusion* das três CNNs obteve o melhor desempenho com precisão de 91,69%, seguida pela regra da soma com 91,56%. Nesse caso, embora a regra do produto tenha obtido o melhor resultado, seu desvio padrão é maior do que o obtido pela regra da soma.

O modelo CNN de melhor desempenho foi diferente em cada abordagem. O modelo 3, que representa a arquitetura de 10 camadas, obteve um melhor desempenho na classificação do tipo autômato e o modelo 2, que representa a arquitetura de 8 camadas, obteve um melhor desempenho na classificação do número de estados do autômato. Apesar da pequena diferença entre os resultados, em geral o Modelo 2 foi considerado o de melhor desempenho, pois mesmo quando perdeu para o modelo 3, a diferença foi menor que 1%.

O processo de *Late Fusion* obteve melhores resultados em todos os métodos realizados, mas quando calcula-se a diferença do melhor resultado do *Late Fusion* com o melhor resultado entre os modelos de CNN, o desempenho na classificação do número de estados foi relativamente menor, cerca de 2%, quando comparado ao desempenho obtido na classificação do tipo de autômato que foi de quase 8%.

Outro ponto a ser observado, é que a classificação do número de estados parece ser intuitivamente mais simples de ser realizada por uma CNN do que a classificação de tipos de autômato, mas em geral os resultados foram menores. Acredita-se que os fatores que forneceram esse resultado estão relacionados com o número de classes do problema que foram relativamente maiores para a abordagem de quantidade de estados do autômato. Além disso, o número de imagens usadas foram diminuídas para o balanceamento de classes.

Em geral, os resultados foram considerados satisfatórios, permitindo o uso dos modelos criados nas demais etapas do desenvolvimento deste trabalho.

6.1.2 Comparação entre métodos de localização de círculos

Para avaliar a etapa de reconhecimento de estados do método MannAR, foram realizados experimentos considerando cada método (Transformada de Hough, *Haar Cascade* e LA) distintamente. O método *Haar Cascade* necessita que o banco de dados disponível seja dividido em treinamento e teste. Neste trabalho optou-se pela divisão 50%-50%, assim o treinamento utilizou metade das imagens disponíveis (escolhidas aleatoriamente) no banco de dados de Autômato descrito na Subseção 4.1.1.1, restando 960 imagens para experimentos.

A verificação de cada um dos testes foi realizada de forma manual, pois mesmo que o resultado de uma imagem com três estados fossem três estados, poderiam haver falsos positivos. Portanto, um estado poderia não ser reconhecido, enquanto um outro local poderia ser considerado como estado.

Assim, o total identificado estaria correto, porém a localização dos círculos não. Devido a este fator apenas 40 imagens foram escolhidas aleatoriamente de cada classe, totalizando 320 imagens, sendo que as mesmas imagens foram usadas nos testes de cada método. Os resultados experimentais da transformada de Hough são apresentados na Tabela 6.5.

Tabela 6.5: Resultados dos experimentos do reconhecimento de estados utilizando a transformada de Hough

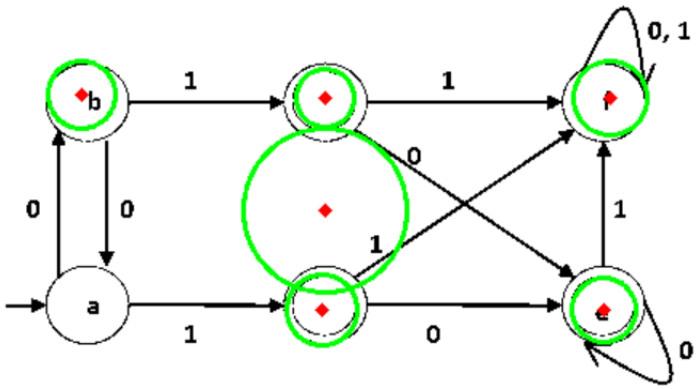
Classes	Acertos	Acurácia(%)
1 estado	31	77.50
2 estados	26	65.00
3 estados	25	62.50
4 estados	15	37.5
5 estados	7	17.5
6 estados	5	12.5
7 estados	1	2.5
8 ou mais estados	1	2.5
Média	13.87	34.68

Fonte: (PRÓPRIA, 2019)

O melhor reconhecimento foi obtido no autômato de 1 estado com uma precisão de 77,5%. Embora os resultados pareçam promissores para figuras com menos estados, à medida que aumenta o número de círculos nas imagens, aumenta a complexidade do problema, diminuindo a taxa de reconhecimento.

Mesmo sabendo o número de estados, os melhores círculos gerados pelo método não correspondem completamente aos círculos reais, como revela a Figura 6.3. Outra característica notada é que o método não reconheceu apenas um estado em 50% das imagens com reconhecimento incorreto.

Figura 6.3: Exemplo de reconhecimento incorreto dos estados



Fonte: (PRÓPRIA, 2019)

Em relação ao método *Haar Cascade*, o modelo de classificação foi gerado utilizando a biblioteca OpenCv. Alguns parâmetros utilizados foram: treinamento com 20 estágios; taxa de acerto mínimo para cada estágio classificador de 0,999; e memória buffer alocada para realizar o treinamento de 1024 Mb. A Tabela 6.6 mostra os resultados de reconhecimento de estados usando o modelo gerado no treinamento.

Tabela 6.6: Resultados dos experimentos do reconhecimento de estados usando o método *Haar Cascade*

Classes	Acertos	Acurácia(%)
1 estado	30	75.00
2 estados	29	72.50
3 estados	20	50.00
4 estados	19	47.50
5 estados	14	35.00
6 estados	20	50.0
7 estados	7	17.50
8 ou mais estados	17	42.50
Média	19.50	48.75

Fonte: (PRÓPRIA, 2019)

É possível notar que os resultados melhoraram quando comparados com a transforma de Hough, exceto para classes de 1 estado e 3 estados. No entanto, a taxa de reconhecimento foi baixa para algumas classes, como no caso das classes de 5 e 7 estados.

Também é observado que, neste caso, a quantidade de estados de imagem não aumentou diretamente a complexidade do problema, como aconteceu na transformada de Hough. Por exemplo, a classe de 7 estados teve 17,5% de acurácia, enquanto a classe de 8

estados teve 42,5%.

Nos experimentos do método *Learning Automata*, todos os círculos que obtiveram o valor de sinal de reforço maior que 75% foram considerados como parte do resultado. A Tabela 6.7 apresenta os resultados obtidos. Nota-se que a taxa de acurácia do LA nas classes de 4 a 8 estados é maior que a dos métodos Hough e *Haar Cascade*. No entanto, o método LA obteve uma taxa de acurácia menor em algumas classes, como no caso das classes de 1 a 3 estados.

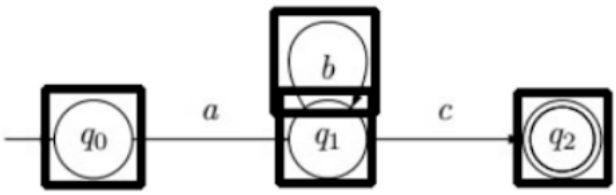
Tabela 6.7: Resultados dos experimentos para o reconhecimento de estados com o método *Learning Automata*

Classes	Acertos	Acurácia(%)
1 estado	21	52.50
2 estados	25	62.50
3 estados	17	42.50
4 estados	19	47.50
5 estados	16	40.00
6 estados	26	65.00
7 estados	20	50.00
8 ou mais estados	18	45.00
Média	20.25	50.62

Fonte: (PRÓPRIA, 2019)

Também observou-se que os resultados foram mais homogêneos quando comparados aos obtidos no *Haar Cascade*. Outra característica observada é que muitos resultados possuem *loops* identificados como círculos, como mostrado na Figura 6.4.

Figura 6.4: Exemplo do reconhecimento de um *loop*



Fonte: (PRÓPRIA, 2019)

Nesses casos, nenhuma outra parte da imagem, diferente de um algum *loop*, é considerada um círculo. Para melhorar a precisão dos métodos, utilizou-se um algoritmo que separa os círculos dos *loops* usando as arestas de interseção, ou seja, buscam-se por círculos que se cruzam ou estão muito próximos. Os resultados são apresentados na Tabela 6.8, considerando

os métodos *Haar Cascade* e LA.

Tabela 6.8: Resultados dos experimentos do reconhecimento de estados separando círculos e *loops*

Classes	Haar Cascade		Learning Automata	
	Acertos	Acurácia(%)	Acertos	Acurácia(%)
1 estado	33	82.50	34	85.00
2 estados	34	85.50	34	85.00
3 estados	25	62.50	34	85.00
4 estados	23	57.50	32	80.00
5 estados	16	40.00	30	75.00
6 estados	21	52.50	34	85.00
7 estados	10	25.00	29	72.50
8 ou mais estados	17	42.50	21	52.50
Média	22.37	50.62	31	77.50

Fonte: (PRÓPRIA, 2019)

É possível observar que separar círculos de *loops* aumentou substancialmente a taxa de precisão. Em 100% das classes, as taxas de reconhecimento foram melhores ou iguais em comparação com os primeiros experimentos. O método LA obteve as maiores taxas de precisão para todas as classes, sendo o melhor método em nossos experimentos.

Para classes de 1, 2, 3 e 6 estados, LA atingiu 85 % de precisão. Examinando todas as classes, a média de reconhecimento foi de 77,5 %, o que é um bom resultado considerando que o reconhecimento do estado é uma tarefa complexa. Além disso, usamos um grande número de imagens quando comparado com trabalhos relacionados. Devido a este fator, escolheu-se utilizar o método LA, com separação entre círculos e *loops*, como parte do método de reconhecimento de imagens de autômatos.

6.2 AVALIAÇÃO DO MÉTODO MANNAR

A avaliação do método MannAR está dividida duas etapas. A primeira realiza um estudo de caso utilizando os AFs da Base de Autômatos. Os resultados são apresentados na Subseção 6.2.1.

Na segunda, na Subseção 6.2.2, é realizada uma comparação do método MannAR com o método proposto por Babalola (2015). Em todos os testes realizados nesta seção foi utilizada a ferramenta MannAR-FA, porém o objetivo inicialmente é apenas a avaliação do método.

6.2.1 Estudo de caso utilizando os AFs da Base de Autômatos

Para realização dos testes do método de reconhecimento de autômatos foi construído o protótipo MannAR-FA. Cada teste exigiu uma verificação manual dos resultados. Devido ao tempo demandado, foram selecionadas 10 imagens de AF de cada classe aleatoriamente (considerando a divisão por quantidade de estados), totalizando 80 imagens. Este conjunto de imagem foi denominado base AF80. As imagens selecionadas estão disponíveis em nossa página sobre o projeto MannAR[1].

Nesta etapa foi realizada uma análise quantitativa dos resultados que são apresentados na Tabela 6.9. Os seguintes critérios foram considerados:

- acerto de todos os estados (AE);
- acerto de quais estados são finais/aceitação (AA);
- acerto de quais estados são iniciais (AI);
- acerto de todas as transições entre estados (AT);
- acerto de todos os *loops* (AL).

Para o conjunto de imagens testadas, o critério acerto de todos os estados (AE) possui o melhor desempenho com 97,5% de acurácia. Já o critério acerto de quais estados são iniciais (AI) obteve um desempenho de 78,75%, sendo dentre todos os menor. Este resultado afeta diretamente o tópico acerto de todas as transições (AT), pois constatou-se que em alguns momentos transições foram classificadas como símbolos de estado inicial, ou estados iniciais foram considerados *loops*. Portanto, a realização de melhorias na classificação deste dois itens pode indicar um aumento significante da acurácia geral de acertos do método. Uma das opções disponíveis é avaliar a classificação de objetos flechas utilizando CNN ou *Haar Cascade*.

Analisando pela perspectiva de critérios a serem avaliados pela divisão de classes, haviam 50 itens para cada uma delas (5 itens $\times$ 10 imagens de cada classe). Assim, é possível observar na Tabela 6.9 que para a classe de 7 estados obteve-se o menor número de acertos, sendo apenas 76%. Já para a classe de 2 estados obteve-se acerto em todos os itens, totalizado 100%. No final, temos que a média de acertos por classe foi de 44,62 itens, o que representa 89,24% de acerto.

[1] https://sites.google.com/view/MannARproject

Tabela 6.9: Resultados quantitativos do MannAR

Classes	AE		AA		AI		AT		AL		Total	
	Acertos	Acurácia(%)	Acertos	Acurácia(%)	Acertos	Acurácia(%)	Acertos	Acurácia(%)	Acertos	Acurácia(%)	**Acertos**	**Acurácia(%)**
1 estado	10	100	10	100	9	90	10	100	9	90	48	96
2 estados	10	100	10	100	10	100	10	100	10	100	50	100
3 estados	10	100	10	100	7	70	8	80	9	90	44	88
4 estados	9	90	9	90	10	100	9	90	9	90	46	92
5 estados	10	100	9	90	8	80	9	90	9	90	45	90
6 estados	10	100	10	100	6	60	8	80	10	100	44	88
7 estados	9	90	7	70	7	70	5	50	10	100	38	76
8 ou mais estados	10	100	9	90	6	60	7	70	10	100	42	84
Média	9,75	97,5	9,25	92,5	7,875	78,75	8,25	82,5	9,5	95	**44,62**	**89,24**

Fonte: (PRÓPRIA, 2019)

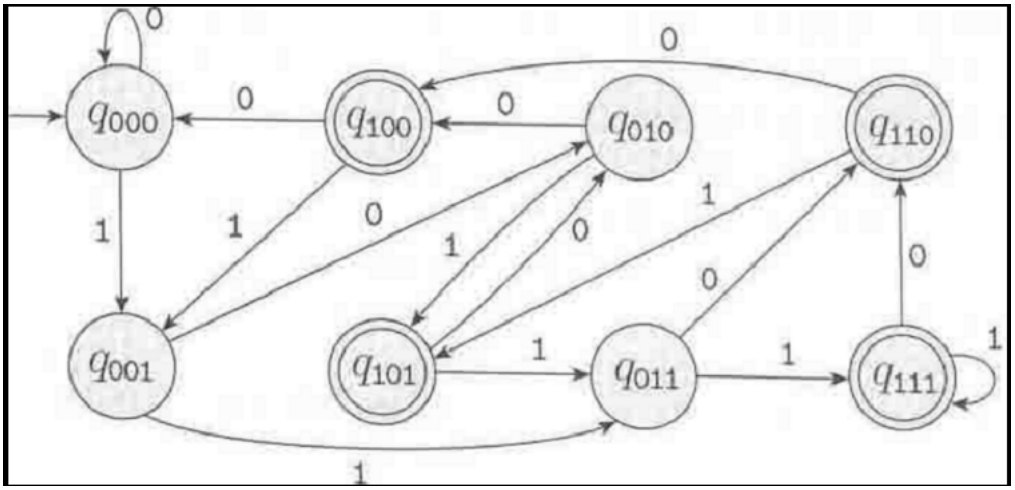
Das 80 imagens analisadas, 58 (72,5%), obtiveram sucesso em todos os itens listados anteriormente, ou seja, contabilizou-se apenas as figuras que foram reconhecidas corretamente em todas as etapas. Para considerar o reconhecimento completo, é necessário incluir o desempenho do OCR utilizado para o reconhecimento dos *labels* dos estados e das descrições das transições. Para isto, foi utilizada a biblioteca do Python denominada *pytesseract*, que é um *wrapper* para a *Engine* Tesseract-OCR[2] da empresa Google. Assim, das 58 imagens consideradas corretas, 12 (20,68%), obtiveram acerto também no reconhecimento de caracteres.

Os resultados quantitativos apresentaram bons resultados para o conjunto de imagens testadas. Porém, também indicam a necessidade de melhorias no reconhecimento do símbolo de estado inicial e principalmente a criação de um OCR específico para o reconhecimento de caracteres individuais em autômatos. Outro fator que possui grande influência no resultado é a qualidade da imagem. No Apêndice A é apresentado o relatório completo de acertos e erros para cada figura presente na base AF80.

6.2.2 Comparação com o trabalho de BABALOLA (2015)

O trabalho apresentado por Babalola (2015) utiliza três imagens para realização dos testes do método apresentado no trabalho. As mesmas imagens foram submetidas no MannAR-FA para efeitos de comparação. Como as imagens originais utilizadas não estavam disponíveis, elas foram retiradas do próprio trabalho do autor. A primeira imagem utilizada é apresentada na Figura 6.5.

Figura 6.5: Autômato Finito de 8 estados FSA1



Fonte: (SIPSER, 2006, apud BABALOLA, 2015)

[2]https://github.com/tesseract-ocr/

Para cada uma das imagens o autor apresenta um resumo dos dados extraídos, sendo eles:

- dimensões do Diagrama (linha, coluna);
- quantidade de pixels de objeto: como já mencionado anteriormente na Subseção 2.3.1.2, uma imagem binária é composto pelo fundo e pelos objetos. No caso das imagens utilizadas aqui, os objetos possuem a cor preta e o fundo a cor branca. Assim, a quantidade de pixels de objeto é a soma total de todos os pixels pretos na imagem. No trabalho de Babalola (2015) este processo é realizado depois da retirada dos objetos identificados como texto;
- número de círculos detectados na imagem: quantidade de círculos identificados, incluindo os círculos que representam estados finais;
- número de arcos detectados na imagem: quantidade de transições, incluindo também os *loops* e o símbolo de estado inicial;
- quantidade de caracteres identificados na imagem: quantidade de caracteres identificados após o processo de extração de texto da imagem;
- número de elementos não identificados: elementos não classificados em nenhuma categoria (círculo, arco, texto);
- número de conexões nó link na imagem: quantidade de pontos que ligam dois objetos identificados.

A Tabela 6.10 apresenta a comparação entre os dados extraídos pelo trabalho de Babalola (2015) e os dados extraídos pelo MannAR-FA para a imagem FSA1.

O trabalho de Babalola (2015) realiza inicialmente a identificação dos objetos textos na imagem. Assim, a divergência apresentada entre a quantidade de pixels de objeto se deve ao fato de que o autor realiza a contagem após a retirada dos objetos de texto. Como este processo não é realizado no MannAR-FA a contagem apresentada é de todos os pixels de objeto na imagem, isto acontece para todas as três imagens testadas.

O resultado de Babalola (2015) para o número de círculos e número de arcos da imagem FSA1, também se diverge da imagem original, sendo que 4 círculos não foram identificados e 2 arcos foram identificados como texto. Por outro lado, no MannAR-FA todos os círculos e arcos foram identificados corretamente.

Tabela 6.10: Comparação do resultado para a imagem FSA1

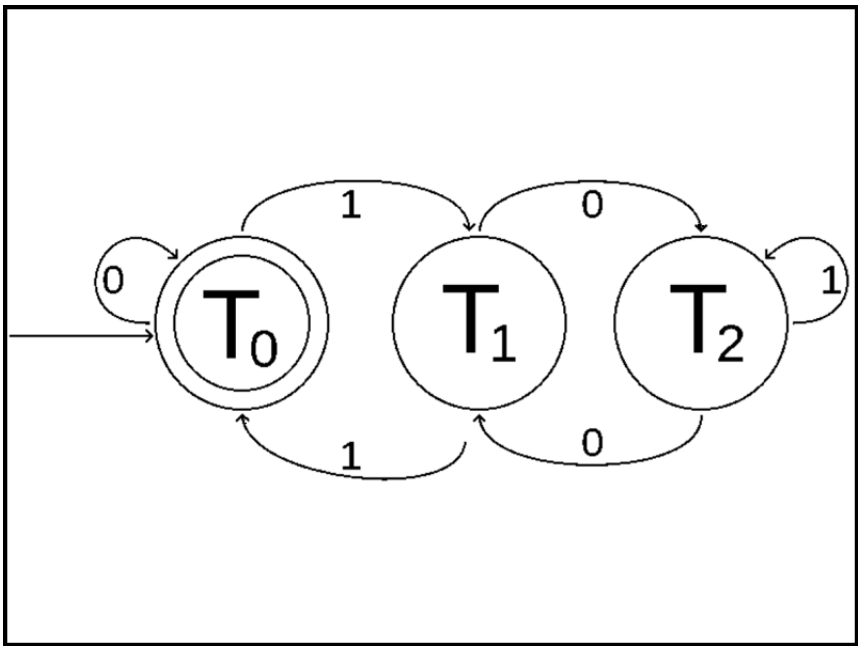
	BABALOLA (2015)	**MannAR-FA**
Dimensões do Diagrama (linha, coluna)	(360,756)	(360,756)
Quantidade de pixels de objeto	4158	14632
Número de círculos detectados na imagem	8	12
Número de arcos detectados na imagem	15	17
Quantidade de caracteres identificados na imagem	2	24
Número de conexões nó-link identificadas	30	-
Número de elementos não identificados	0	0

Fonte: (PRÓPRIA, 2019)

Na sequência, no trabalho de Babalola (2015) apresenta a quantidade de caracteres identificados na imagem, após o processo de retirada de texto, assim o resultado representa neste caso dois arcos que foram identificados como texto. Já para o MannAR-FA a contagem é de quantas regiões foram considerados texto e foram enviados ao OCR. Após, Babalola (2015) apresenta o número de conexões nó-link, ou seja, quantas ligações entre dois objetos foram identificadas. Tal processo não é realizado no MannAR-FA. Por fim, têm-se o número de elementos não identificados que neste caso foi 0 para ambas abordagens.

A Figura 6.6 apresenta a segunda imagem utilizada no teste por Babalola (2015). Ela foi denominada FSA2 e representa um autômato finito de três estados.

Figura 6.6: Autômato Finito de 3 estados FSA2



Fonte: (BABALOLA, 2015)

Na Tabela 6.11 apresenta-se a comparação do método de Babalola (2015) com o MannAR-FA para a imagem FSA2.

Tabela 6.11: Comparação do resultado para a imagem FSA2

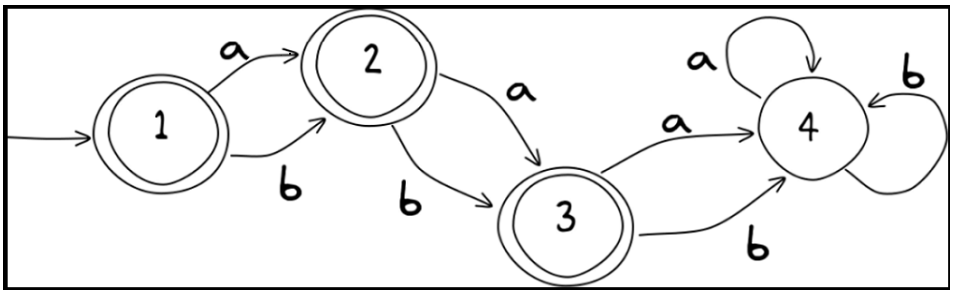
	BABALOLA (2015)	**MannAR-FA**
Dimensões do Diagrama (linha, coluna)	(600,800)	(600,800)
Quantidade de pixels de objeto	3825	8695
Número de círculos detectados na imagem	4	4
Número de arcos detectados na imagem	7	6
Quantidade de caracteres identificados na imagem	12	9
Número de conexões nó-link identificadas	14	-
Número de elementos não identificados	0	1

Fonte: (PRÓPRIA, 2019)

No caso da imagem FSA2, todos os círculos foram identificados corretamente em ambas as abordagens. Já para os arcos, o MannAR-FA não reconheceu o símbolo de estado inicial, enquanto o trabalho de Babalola (2015) reconheceu todos os arcos presentes na imagem. Para o item quantidade de caracteres identificados na imagem, a abordagem de Babalola (2015) identificou 12 caracteres, já o MannAR-FA identificou 9 regiões de texto que foram enviadas ao OCR. Ao final, os dois processos tiveram sucesso no envio das regiões corretas para o reconhecimento dos caracteres.

Na sequência apresenta-se a terceira imagem, denominada FSA3, utilizada nos testes por Babalola (2015). A Figura 6.7 representa um autômato finito de 4 estados.

Figura 6.7: Autômato Finito de 4 estados FSA3



Fonte: (BABALOLA, 2015)

Na Tabela 6.12 apresenta-se a comparação do método de Babalola (2015) com o MannAR-FA para a imagem FSA3.

O método de Babalola (2015) não conseguiu separar um círculo de uma transição.

Tabela 6.12: Comparação do resultado para a imagem FSA3

	BABALOLA (2015)	**MannAR-FA**
Dimensões do Diagrama (linha, coluna)	(304,1013)	(304,1013)
Quantidade de pixels de objeto	4612	8695
Número de círculos detectados na imagem	6	7
Número de arcos detectados na imagem	7	9
Quantidade de caracteres identificados na imagem	13	12
Número de conexões nó-link identificadas	14	-
Número de elementos não identificados	1	0

Fonte: (PRÓPRIA, 2019)

Assim, este objeto composto por um círculo e uma transição foi classificado como um elemento não identificado. O outro arco não identificado não é abordado pelo autor, já que ele apresenta uma imagem com 8 elementos classificados como arcos em seu trabalho, gerando assim uma inconsistência. Já no MannAR-FA todos os círculos e arcos foram detectados corretamente.

O Quadro 6.1 apresenta um resumo do desempenho do método de Babalola (2015) e do MannAR-FA para as três imagens utilizadas como teste por Babalola.

Quadro 6.1: Resumo do desempenho dos métodos Babalola e MannAR-FA

Elementos	**FSA1**		**FSA2**		**FSA3**	
	BABALOLA	MannAR-FA	BABALOLA	MannAR-FA	BABALOLA	MannAR-FA
Círculos	×	✓	✓	✓	×	✓
Arcos	×	✓	✓	×	×	✓
Caracteres	×	✓	✓	✓	×	✓

Fonte: (PRÓPRIA, 2019)

Observando o Quadro 6.1, que considera para cada imagem se o método acertou todos itens correspondentes à cada elemento, é possível notar que o método MannAR possui um desempenho melhor do que o proposto por Babalola (2015). Além disso, o autor não explica algumas inconsistências, como o caso da imagem FSA3, que possui nove arcos. Babalola apresenta como resultado para essa imagem sete arcos explicando apenas o não reconhecimento de um deles.

O autor também apresenta uma imagem com os arcos reconhecidos, na qual se encontram 8 arcos. Assim, não se sabe se houve apenas um erro de digitação na tabela de resultados ou se ocorreu algum outro problema. No Apêndice B é realizada uma comparação mais detalhada entre os dois trabalhos, na qual é possível visualizar quais elementos foram reconhecidos para os itens: círculos, arcos e caracteres nos dois trabalhos.

6.3 AVALIAÇÃO DA FERRAMENTA MANNAR-FA

A avaliação da ferramenta MannAR-FA se divide em duas etapas, sendo elas: avaliação automática das diretrizes WCAG, que realiza uma busca, no conteúdo e no código das páginas a procura de erros que possam ser reconhecidos de forma automática.

Já a segunda etapa apresenta testes realizados com deficientes visuais. Tais testes possuem o objetivo de encontrar erros que só sejam possíveis de perceber pelo uso de forma manual e por pessoas que sejam usuários em potenciais para o sistema. A realização de cada uma dessas etapas é detalhada nas próximas subseções.

6.3.1 Avaliação automática das diretrizes WCAG

A primeira avaliação realizada da ferramenta foi a validação automática do código. Esse modelo de avaliação segue as diretrizes de desenvolvimento acessível para Web descritas na Subseção 2.2.1. Ao todo são quatro princípios que estão divididos em diretrizes.

A avaliação foi realizada subentendo todas as páginas do MannAR-FA na ferramenta *Web Accessibility Checker* [3]. Ela foi escolhida por permitir uma avaliação do sistema página por página. As seguintes páginas compõem a ferramenta MannAR-FA:

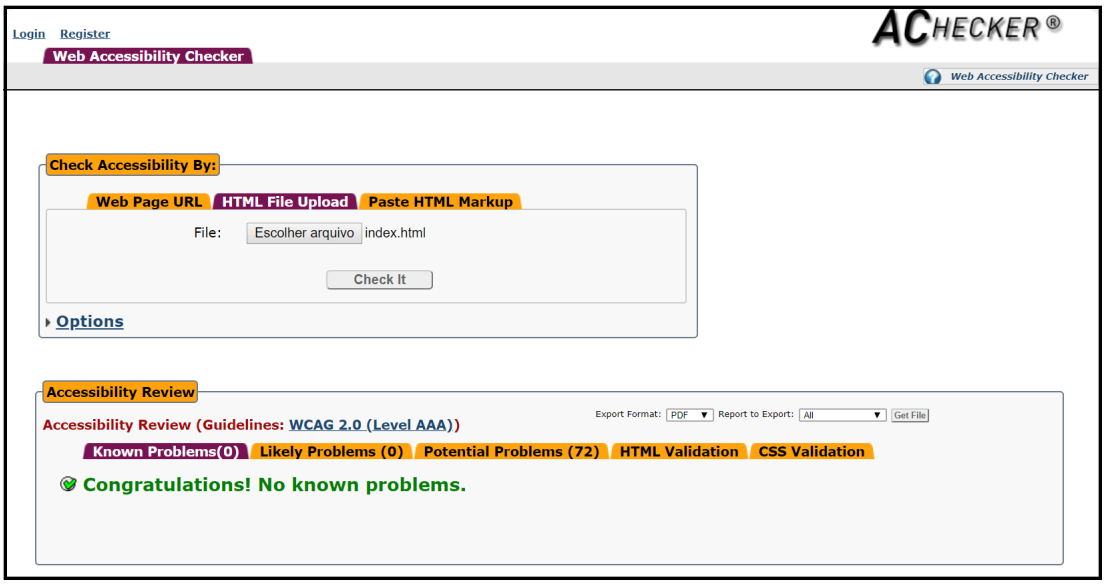
- index: página principal que possibilita o acesso a funcionalidade de enviar uma imagem digital de um AF para conversão para versões alternativas;
- sobre: página com detalhes sobre a ferramenta, como modo de usar e desenvolvedores;
- mapa do site: página com o acesso a todas as páginas do site;
- contato: página com o contato do projeto MannAR;
- resultado: página que contém os resultados de uma conversão. Ela fica disponível apenas após o envio de uma imagem.

Os itens analisados na avaliação automática estão listados no Anexo A. Cada subitem de uma diretriz possui um nível, podendo ser A, AA, ou AAA. Para a página ser considerada nível A, todas os itens A devem ser cumpridos, para ser do nível AA, todos do nível A e AA devem ser cumpridos e assim por diante. O mesmo resultado foi gerado para todas as páginas da ferramenta, sendo que nenhum item foi apontado como não atendido. Portanto, as páginas

[3] https://achecker.ca/checker/index.php

do MannAR-FA foram consideradas como nível AAA. Na Figura 6.8, é possível observar o resultado da página index.

Figura 6.8: Resultado da avaliação automática para a página index do MannAR-FA



Fonte: Web Accessibility Checker. Acesso em: 20/02/2019

Para cada página, a ferramenta utilizada na validação do WCAG também gera uma lista de problemas em potenciais. Fazem parte desta lista itens que não podem ser verificados automaticamente, como por exemplo, se uma imagem possui a descrição correta. Assim, para cada página cada item da lista foi verificado e considerado atendido.

6.3.2 Testes com deficientes visuais

Os testes sobre a ferramenta com deficientes visuais foram divididos em duas etapas. A primeira avalia a adequação da mesma com as diretrizes da WCAG. Os mesmos itens já explorados na subseção anterior, que estão disponíveis no Anexo A, são avaliados aqui. Todavia, a perspectiva agora da analise é do ponto de vista do usuário.

A segunda etapa avalia a usabilidade da ferramenta de acordo com Nielsen. Avaliar a usabilidade do sistema, na perspectiva do deficiente visual, visa a adequação da ferramenta a diferentes preferências e tipos de usuários.

Este projeto, intitulado “Tecnologias Assistivas de Imagens Digitais para Deficientes Visuais”, foi aprovado pelo Comitê Permanente de Ética em Pesquisa com Seres Humanos (COPEP) no dia 20 de dezembro de 2018, com número de parecer de aprovação 3.098.506

(Anexo B).

Este projeto foi desenvolvido em parceria com a pesquisa MannaHap, porém, só são tratados os resultados obtidos pelos testes realizados no MannAR. A mesma metodologia foi estabelecida para a realização dos dois experimentos, sendo ela:

1. recrutamento de voluntários: os voluntários deveriam possuir baixa visão profunda, cegueira parcial ou total;
2. divisão amostral: os voluntários seriam divididos em dois grupos: pessoas que cursam ou cursaram graduação na área de computação, e pessoas que não cursaram;
3. treinamento: se realizaria um treinamento com os voluntários para utilização das tecnologias assistivas desenvolvidas, explicando o funcionamento básico e os objetivos das ferramentas;
4. utilização da tecnologia MannAR: realização dos experimentos envolvendo esta ferramenta;
5. utilização da tecnologia MannaHap: realização de um segundo experimento envolvendo esta ferramenta;
6. aplicação de questionários de avaliação: para cada ferramenta utilizada, seriam aplicados questionários para avaliar os itens propostos na pesquisa;
7. análise dos dados: a análise dos dados obtidos com os experimentos seria realizada de maneira heurística. Baseado nas respostas do questionário, os dados seriam compilados de maneira qualitativa, considerando uma escala de aproveitamento / aceitação de cada um dos itens avaliados.

Foram convidados todos os alunos com algum grau de deficiente visual participantes do Programa Multidisciplinar de Pesquisa e Apoio à Pessoa com Deficiência e Necessidades Educativas Especiais (PROPAE), conforme autorização estabelecida pelo COPEP. Sendo um deles com cegueira total e os outros cinco com baixa visão. Aceitaram participar do testes dois alunos que possuem baixa visão. Apesar de um deles ser do curso de Ciência da Computação e outro não, optou-se por não realizar a divisão previamente estabelecida devido ao baixo número de participantes no experimento. Nas próximas seções tem-se o detalhamento dos questionários realizados.

6.3.2.1 Questionário WCAG para deficientes visuais

Os mesmos princípios avaliados com a validação automática (Anexo A) foram utilizados como questionário na avaliação do MannAR-FA. Ao total foram 128 pontos verificados, sendo que para cada item o usuário tinha três opções: atendido, não atendido e não se aplica.

Os dois participantes utilizaram o sistema operacional Windows e o navegador Google Chrome. Além disso, um deles utilizou o leitor de tela NVDA (*NonVisual Desktop Access*). Para o outro participante foi realizada uma leitura da tela por meio de um voluntário.

Dos 128 pontos analisados, 92,1%, 118 pontos, foram considerados atendidos pelos dois participantes e 7%, nove pontos, foram considerados como itens que não se aplicam também pelos dois participantes. O outro um ponto restantes, 0,9%, foi considerado atendido por um participante e não atendido pelo outro.

O item considerado não atendido pelo participante foi o ponto:

- 3.3.6 Prevenção de Erros (de qualquer tipo): Para as páginas Web que exijam ao usuário o envio de informação, no mínimo, pelo menos um dos seguintes casos é verdadeiro: (Nível AAA)
  - reversível: As ações de envio são reversíveis;
  - verificado: Os dados introduzidos pelo usuário são verificados no que diz respeito a erros de inserção de dados e é dada ao utilizador a possibilidade de os corrigir;
  - confirmado: Está disponível um mecanismo para rever, confirmar e corrigir as informações antes do envio final de dados.

O participante relatou que na página index, onde existe um formulário que permite o envio da imagem a ser reconhecida, não foi possível encontrar facilmente o botão de seleção de imagens. O erro relatado pelo usuário foi:

> "o botão de seleção de imagens foi anunciado apenas como um nível de cabeçalho, não houve indicação de que era um link ou um botão, pensei que era apenas o título de uma seção da página. Por conta disso, continuei indo pra baixo, onde encontrei o botão "Enviar imagem", pensei que esse botão seria responsável por abrir a janela de escolha de imagem, mas na verdade era o botão para enviar a imagem selecionada."

Assim, o usuário não conseguiu avançar no processo e nenhum erro foi gerado. Portanto, o usuário considerou este item não atendido. Este erro ocorreu devido à utilização

de uma imagem como botão. Para correção deste item foi adicionado a tag “tabindex=0” no código HTML que gera a imagem, permitindo que ocorra um foco no botão e o usuário o encontra de forma mais fácil. Também, adicionou-se o aviso de que é necessário selecionar uma imagem para o caso em que o usuário clique diretamente no botão “enviar imagem”.

6.3.2.2 Avaliação da usabilidade de acordo com Nielsen

Para realização da avaliação da usabilidade do MannAR-FA foi elaborado um questionário com base nas heurísticas de Nielsen. O questionário possui 10 itens, sendo eles:

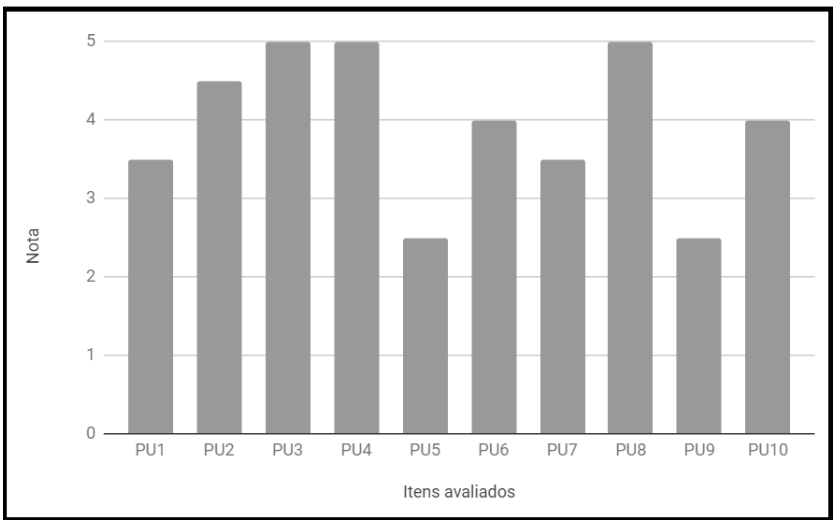
- PU1 - o sistema informa ao usuário o que está acontecendo antes, durante e após a interação;
- PU2 - o sistema possui uma linguagem próxima ao usuário, sem utilizar jargões técnicos ou de difícil compreensão;
- PU3 - o sistema permite ao usuário navegar livremente pelo sistema, podendo desfazer suas ações e retornando para pontos anteriores de interação;
- PU4 - o sistema mantém uma consistência em seus meios de comunicação e interação;
- PU5 - em caso de ações “drásticas”, como a deleção de um arquivo, o sistema sinaliza e pede a confirmação destas ações, prevenindo erros;
- PU6 - o sistema é intuitivo e de fácil utilização, tendo ajudas contextuais e indicando o fluxo de ações a se tomar;
- PU7 - o sistema é flexível em sua utilização para usuários experientes e novatos, ou seja, é aprimorado para usuários já habituados (provendo itens como atalhos de teclado, preenchimento automático, etc.) e informativo para os que o utilizam pela primeira vez;
- PU8 - o sistema possui *design* minimalista, com conteúdo de maneira direta e simples;
- PU9 - o sistema auxilia o usuário a corrigir erros cometidos;
- PU10 - o sistema provê ao usuário toda a ajuda necessária para utilização.

Para as respostas foram utilizadas uma escala numérica com valores entre 1 e 5, sendo que os valores indicam níveis de satisfação com as questões: 1 - discordo totalmente; 2 - discordo parcialmente; 3 - indiferente; 4 - concordo parcialmente; 5 - concordo totalmente.

Além disso, caso o usuário acreditasse que a questão não se aplicava ao sistema, podia marcar o nível 0.

Ao analisar as respostas, nenhum item foi considerado como não aplicável ao sistema. Na Figura 6.9 é apresentada a média dos resultados obtidos na avaliação.

Figura 6.9: Resultado da avaliação da Heurística de Nielsen



Fonte: (PRÓPRIA, 2019)

Para os itens PU2, PU3, PU4, PU6, PU8 e Pu10, o resultado foi considerado satisfatório, pois as respostas são iguais ou superiores a quatro. Já para os itens PU5 e PU9, a média dos resultados foi de 2,5, sinalizando que os itens devem ser melhorados no sistema. Tal fato ocorreu devido ao erro levantando na subseção anterior. Assim, o sistema não permitiu nem a prevenção e nem ajudou a corrigir um erro que aconteceu durante a interação. Esse mesmo problema também afetou o resultado dos itens PU1 e PU7, para os quais a média foi de 3,5.

Contudo, para 60% dos itens a média das respostas foi considerada satisfatória e somente para 20% das respostas houve um média menor do que o nível considerado indiferente. De forma geral, a usabilidade do sistema foi considerada aceitável no contexto de deficientes visuais.

6.3.3 Testes de integração com o JFLAP

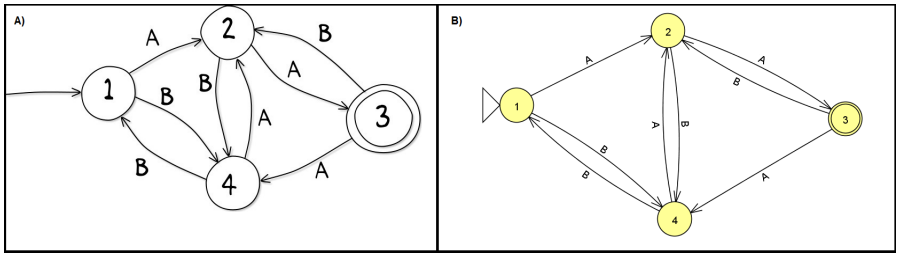
Como destacado na Seção 3.2, a ferramenta JFLAP pode ser utilizada como uma versão alternativa para uma imagem de AF. Para isto, é preciso converter os dados retirados da imagem para um arquivo .jff.

Para verificar o desempenho da integração com o JFLAP foram testadas as 12 imagens que foram corretamente reconhecidas e também obtiveram acerto de todos os caracteres no OCR. Para realização dos testes os arquivos gerados de cada autômato foi aberto no JFLAP. Utilizou-se a versão 7.1, pois é a última considerada estável pelos desenvolvedores.

O arquivo .jff é composto por uma lista de estados e uma lista de transições. Cada estado possui um id, um nome (*label*), e um par (x, y) que representa a posição do ponto central do círculo. Além disso, podem possuir a indicação de ser inicial e/ou de ser final. Já para as transições são necessários apenas os IDs de quais estados fazem parte da transição e a descrição da mesma.

Das 12 imagens testadas, 7 (58,33%) obtiveram sucesso na conversão para o arquivo .jff. Na Figura 6.10 apresenta-se um exemplo de comparação entre a imagem original (A) e a imagem do JFLAP (B) para um AF de 4 estados.

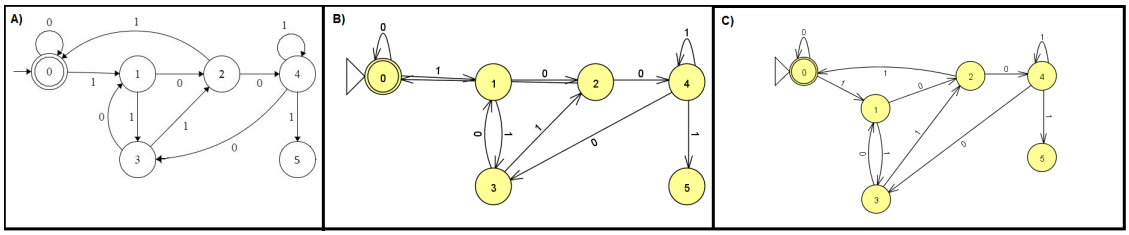
Figura 6.10: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 4-2



Fonte: (PRÓPRIA, 2019)

Em quatro imagens (33,33%) a imagem gerada possui alguma sobreposição de transições. Esse fator ocorre devido a característica do arquivo .jff que não apresenta informações de localização sobre as transições. Um exemplo de sobreposição pode ser visualizado na Figura 6.11.

Figura 6.11: Comparação entre a imagem original (A), a imagem gerada pelo JFLAP (B) e a imagem com os elementos reposicionados para o AF 6-3

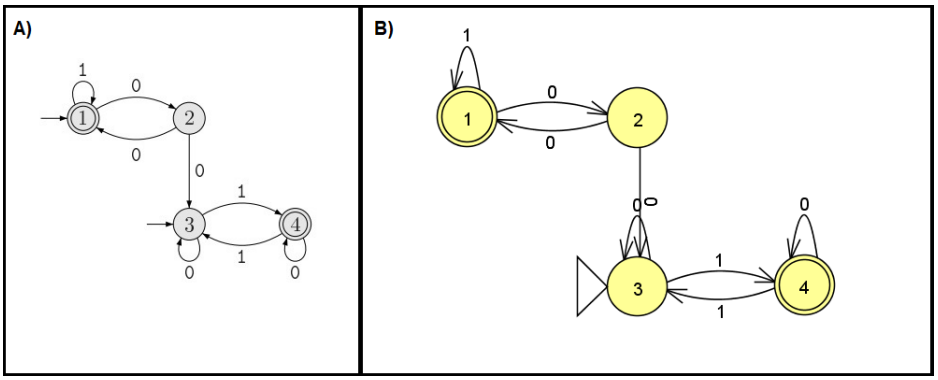


Fonte: (PRÓPRIA, 2019)

Apesar disso, o uso do arquivo .jff gerado pode ser prejudicado apenas nos casos em que pessoas com baixa visão estejam utilizado uma versão do JFLAP sem acessibilidade. Na versão com acessibilidade existem menus extras que permitem a navegação entre os estados e transições, assim a posição de cada um não afeta o desempenho final.

Outro ponto identificado em uma das imagens com sobreposição é que o JFLAP permite apenas um estado inicial. Tal fator ocasionou o erro apresentado na Figura 6.12. Neste caso apenas o último elemento da lista de estados, que possui o atributo inicial, permanece como inicial no resultado final.

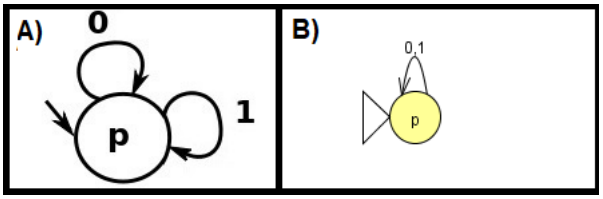
Figura 6.12: Comparação entre a imagem original (A), a imagem gerada pelo JFLAP (B) para o AF 4-10



Fonte: (PRÓPRIA, 2019)

Por fim, identificou-se um autômato (8,33%) que apresentou divergência entre a imagem original e a imagem gerada para o JFLAP. Neste caso, havia mais de um loop para o mesmo estado e na representação final todas as descrições foram representadas no mesmo loop. Apesar da representação visual final ser diferente os autômatos são equivalentes. Um exemplo desse é mostrado na Figura 6.13.

Figura 6.13: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 1-4



Fonte: (PRÓPRIA, 2019)

Considerando os fatores apresentados, o desempenho da conversão foi considerado

satisfatório. O Apêndice C apresenta a comparação entre todas as 12 figuras geradas pelo JFLAP e suas respectivas imagens originais.

6.4 CONSIDERAÇÕES FINAIS

A primeira etapa realizada foi a avaliação do uso de CNNs no reconhecimento de imagens de autômato, com o propósito de utilizar o modelo criado para o desenvolvimento de tecnologias assistivas.

Três modelos simples de CNN foram desenvolvidos para verificar a evolução deste problema, e bons resultados foram obtidos com os experimentos iniciais nas duas abordagens investigadas, permitindo que os modelos fossem incluídos nas próximas etapas de desenvolvimento deste trabalho.

O reconhecimento da imagem de um autômato é uma tarefa não trivial que envolve muitas subtarefas. Neste trabalho, elas foram divididas em três grandes tarefas, o reconhecimento de estados, o reconhecimento de transições, e o reconhecimento das relações entre elas. A primeira foi considerada como central no desenvolvimento do método, pois ao encontrar os círculos corretamente e segmentá-los da imagem é possível separar os demais objetos, facilitando o reconhecimento das demais etapas. Realizamos testes com três métodos diferentes de reconhecimento: transformada de Hough, *Haar Cascade* e *Learning Automata* (LA).

Os experimentos mostram que o método LA teve os melhores resultados, quando considero-se a separação dos círculos e *loops*, com uma precisão de 85% para algumas classes e uma média de reconhecimento de 77,5%. Devido a este fator, este método foi o escolhido para fazer parte da versão final do MannAR.

Na segunda etapa foram realizadas avaliações relacionadas ao método implementado. Para a base AF80, o resultado obtido foi 72,5% de acerto, ou seja, das oitenta imagens de AF, cinquenta e duas apresentaram acerto em todos os critérios avaliados. Nesta etapa, também, realizou-se uma comparação do MannAR-FA com o trabalho de Babalola (2015), na qual obteve-se um desempenho superior do MannAR-FA.

Na terceira e última etapa os experimentos apresentados estão relacionados com a ferramenta. Foram realizados testes de verificação automática das diretrizes da WCAG, nos quais os resultados foram considerados satisfatórios, pois não foram encontrados erros em nenhum *checkpoint*. Na sequência também realizaram-se testes de acessibilidade e usabilidade

com deficientes visuais. Os resultados dos dois testes foram afetados pelo mesmo problema, sendo que um usuário não encontrou facilmente o botão de enviar a imagem. Porém, como apenas um problema foi apontado os resultados desses testes também foram considerados satisfatórios.

Ao final, apresentaram-se também experimentos de integração com a ferramenta JFLAP. A integração com esta ferramenta permite que usuários sem deficiência visual possam aproveitar a ferramenta para estudar autômatos, já que ele provê funcionalidades de simulação de autômatos. Também permite que usuários com deficiência visual tenham acesso a essa funcionalidade e possam editar um autômato caso estejam utilizando uma versão do JFLAP que possua acessibilidade.

Com todos os testes realizados, considerou-se que a ferramenta MannAR-FA segue as diretrizes de desenvolvimento acessível e que o método MannAR possui um bom desempenho quando utilizado com Autômatos Finitos.

7

CONCLUSÃO

Esta dissertação apresentou os referenciais, metodologias, procedimentos e testes realizados na elaboração do objetivo principal deste trabalho que é desenvolver um método, denominado MannAR, que realiza a tradução de imagens de diagramas de estados de transição de mecanismos reconhecedores de linguagens formais para uma versão que possibilita o acesso de deficientes visuais à informação contida nessas imagens.

Visando este objetivo, inicialmente, elaborou-se uma CNN que pudesse distinguir imagens de diagramas de estados de transição e informar qual o tipo do mecanismo representado naquela imagem. É possível utilizar este modelo aplicado na construção do método MannAR. O método neste contexto deve ter precisão na tradução das imagens e o uso da CNN colabora para aumentar a confiabilidade da tradução.

Apesar da tentativa de utilização de um modelo de CNN, também na etapa de localização dos círculos nas imagens, que é considerada o cerne do método MannaR, verificou-se por meio dos testes que a melhor solução para o contexto do trabalho foi a *Learning Automata*.

Com todos os itens preliminares da construção do método definidos, foi possível estabelecer uma versão final do MannAR e então implementar um protótipo que o utilizasse. Foi desenvolvida a primeira versão do protótipo que contempla os AFs, devido ao tempo disponível para o desenvolvimento desta dissertação.

Cada usuário possui experiências e preferências diferentes que podem influenciar o seu processo de aprendizagem. Mesmo dentro do grupo de deficientes visuais, existem diferentes tipos de usuários, pois podem existir diferentes graus de perda de visão. Além disso, existe uma

preocupação de que o sistema seja útil para todas as pessoas que desejam apreender sobre AFs.

Dessa forma, optou-se por disponibilizar três versões alternativas acessíveis diferentes do AF. A primeira traz o AF em formato de tabela disponível em um arquivo .csv. Esta versão visa ser o mais simples possível e é a principal da ferramenta, pois os usuários conseguem acessa-lá com *softwares* em que eles estão acostumados a utilizar e já possuem instalado no computador ou são de fácil acesso *online*. Esta mesma versão permite que outros sistemas que desejam se integrar ao MannAR-FA possuam uma forma de exportar os resultados gerados pelo mesmo.

A segunda versão, que é um arquivo .jff utilizado na ferramenta JFLAP, visa aumentar a autonomia do usuário quando se trata do estudo de AFs. O JFLAP possui a funcionalidade de simulação dos autômatos o que pode auxiliar o processo de aprendizagem. Esta mesma funcionalidade traz para o projeto o conceito de Design Universal, pois com esse formato qualquer aluno pode realizar a conversão de uma imagem de autômato que ele encontrar em livros ou materiais didáticos e utilizar as simulações do JFLAP para compreendê-la melhor.

A terceira versão, que é um arquivo .json, visa a integração da ferramenta com o projeto MannaHap que é desenvolvido em parceria com o MannAR. O MannaHap é um hardware dinâmico que permite a interação do usuário com a imagem por meio do tato com *feedback* auditivo.

Como última etapa deste trabalho foram realizados experimentos relacionados com o método e com a ferramenta. Para os testes quantitativos com o método, utilizou-se uma base de imagens de autômato, denominada AF80. Considerou-se como sucesso a imagem que teve todos os itens reconhecidos, sendo eles: acerto no número de estados, acerto dos estados finais, acertos dos estados inicias, acertos das transições e acertos dos *loops*. Os testes mostram um acerto de 72,5% na base AF80 e revelou que grande parte dos erros foram o não reconhecimento de estados iniciais.

Por outro lado, se considerarmos a quantidade de itens a serem reconhecidos, sendo 50 itens por classe (5 itens por imagem x 10 imagens por classe), a média de acertos na classe chegou a 89,24%. Outro fator, é que nesse teste foi possível perceber que poucas imagens obtiveram acerto total no reconhecimento de caracteres.

Ainda na análise do método, ele foi comparado com o proposto por Babalola (2015). Neste teste 3 imagens foram analisadas, sendo que o MannAR obteve um desempenho igual ou melhor em 8 de 9 itens analisados.

Já na avaliação sobre MannAR-FA, obtiveram-se resultados satisfatórios com a

avaliação da adequação da ferramenta as diretrizes propostas pela WCAG, realizada de forma automática, pois nenhum item foi apontado como não atendido.

Na sequência realizaram-se as avaliações com deficientes visuais, que foram aprovadas pelo Comitê de Ética da UEM. As mesmas diretrizes, utilizadas no teste anterior, também foram aplicadas como questionário aos deficientes visuais que utilizaram a ferramenta MannAR-FA. Durante esse procedimento um dos participantes relatou uma dificuldade em encontrar o botão de selecionar a imagem desejada na página principal da ferramenta MannAR. Este ponto foi o único relatado como um problema em potencial para o sistema.

Assim, tanto o teste da WCAG, quanto o teste de usabilidade, que foi baseado nas Heurísticas de Nielsen, tiveram seus resultados considerados como satisfatórios. O último ponto analisado foi a integração com JFfap, onde em alguns casos ocorreu uma sobreposição de transições, porém na maioria das imagens testadas 58,33% obtiveram sucesso na conversão.

Considerando os processos realizados anteriormente, os resultados deste trabalho são:

- o modelo de CNN, que identifica os tipos de imagens de autômatos, elaborado como auxílio para o desenvolvimento do método MannAR;
- a base de autômatos, composta por imagens digitais de Autômatos Finitos, Autômatos de Pilha e Máquina de Turing, com 1920 imagens;
- o método MannAR;
- a ferramenta MannAR-FA que implementa o método MannAR para AFs e gera três diferentes versões acessíveis da imagem do autômato.

No geral, considerou-se que todos os testes foram bem sucedidos e destacam-se os seguintes pontos de melhorias como trabalhos futuros:

- investigar diferentes abordagens para o reconhecimento do símbolo de estado inicial, visto que esse item obteve a menor taxa de acerto entre os itens avaliados;
- criar um OCR próprio para reconhecer texto nos autômatos;
- expandir a ferramenta para os demais mecanismos reconhecedores de linguagens;
- integrar o MannAR-AF com outras ferramentas acessíveis relacionadas a manipulação de autômatos que possam ser úteis para deficientes visuais;
- realizar testes com um número maior de deficientes visuais;

- investigar a possibilidade de uma imagem de autômato ser reconhecida de forma automática em um pdf, possibilitando também o envio desse formato na ferramenta MannAR-FA.

O reconhecimento completo de um autômato é uma tarefa não trivial, composta por diferentes etapas complexas. De todo modo, esse método possui um desempenho melhor do que seu antecessor. Conclui-se que é viável desenvolver um método que abranja grande quantidade de imagens, pois, em alguns casos, não é possível processar imagens com qualidade extremamente baixas. Porém, acredita-se que em trabalhos futuros seja possível construir uma abordagem que possa funcionar para imagens com qualidade baixa, até um certo ponto, e possa ser generalizado para diferentes tipos de diagramas.

O MannAR é um método que busca seguir os princípios do Desing Universal e atender as necessidades de diferentes tipos de usuários, por meio da integração com ferramentas acessíveis tanto de *software* quanto de *hardware*. Com os resultados obtidos neste trabalho, há um forte indicativo de que o método pode ser um prelúdio para o desenvolvimento de novas pesquisas no campo de reconhecimento de imagens digitais para deficientes visuais.

REFERÊNCIAS

BABALOLA, O. T. **Automatic recognition and interpretation of finite state automata diagrams**. Tese (Doutorado) — Stellenbosch: Stellenbosch University, 2015.

BALIK, S. P. et al. Gsk: universally accessible graph sketching. In: ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION,44,2013. **Proceedings...** New York: ACM, 2013. p. 221–226.

BALIK, S. P. et al. Including blind people in computing through access to graphs. In: INTERNATIONAL ACM SIGACCESS CONFERENCE ON COMPUTERS & ACCESSIBILITY,16,2014. **Proceedings...** New York: ACM, 2014. p. 91–98.

BARROW, H. G.; TENENBAUM, J. M. Computational vision. **Proceedings of the IEEE**, IEEE, v. 69, n. 5, p. 572–595, 1981.

BHOWMICK, A.; HAZARIKA, S. M. An insight into assistive technology for the visually impaired and blind people: state-of-the-art and future trends. **Journal on Multimodal User Interfaces**, Springer, v. 11, n. 2, p. 149–172, 2017.

BRASIL. Constituição da república federativa do brasil. Brasília, 1988. 292 p.

BRASIL. Subsecretaria nacional de promoção dos direitos da pessoa com deficiência. comitê de ajudas técnicas. **Tecnologias Assistivas**. Brasília: CORDE, 2009. 138 p.

CANNY, J. A computational approach to edge detection. **IEEE Transactions on pattern analysis and machine intelligence**, Ieee, n. 6, p. 679–698, 1986.

COHEN, R. F.; MEACHAM, A.; SKAFF, J. Teaching graphs to visually impaired students using an active auditory interface. **ACM SIGCSE Bulletin**, v. 38, n. 1, p. 279–282, 2006.

CORMEN, T. H. et al. **Introduction to algorithms**. [S.l.]: MIT press, 2009.

COSTA, L. da F.; JR, R. M. C. **Shape analysis and classification: theory and practice**. [S.l.]: CRC press, 2010.

CRESCENZI, P.; ROSSI, L.; APOLLARO, G. Making turing machines accessible to blind students. In: ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION. **Proceedings...** New York: ACM, 2012. p. 167–172.

CUEVAS, E. et al. Circle detection on images using learning automata. In: **Artificial Intelligence, Evolutionary Computing and Metaheuristics**. [S.l.]: Springer, 2013. p. 545–570.

CUTTER, M.; MANDUCHI, R. Improving the accessibility of mobile ocr apps via interactive modalities. **ACM Transactions on Accessible Computing (TACCESS)**, ACM, New York, v. 10, n. 4, p. 11, 2017.

DIVERIO, T. A.; MENEZES, P. B. **Teoria da Computação–UFRGS: Máquinas Universais e Computabilidade**. [S.l.]: Bookman Editora, 2009.

DUDAR, O.; HART, P. Use ofthehoughtransform todetectlines and curves in pictures. **Communications of the ACM**, 1972.

FENG, C. Designing wearable mobile device controllers for blind people: A co-design approach. In: ACM SIGACCESS CONFERENCE ON COMPUTERS AND ACCESSIBILITY, 16, 2016. **Proceedings...** New York: ACM, 2016. p. 341–342.

FRANCIONI, J. M.; SMITH, A. C. Computer science accessibility for students with visual disabilities. **ACM SIGCSE Bulletin**, v. 34, n. 1, p. 91–95, 2002.

GOEL, A. et al. Raspberry pi based reader for blind people. 2018.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 3. ed. [S.l.]: Pearson, 2007.

GÖTZELMANN, T. Lucentmaps: 3d printed audiovisual tactile maps for blind and visually impaired people. In: INTERNATIONAL ACM SIGACCESS CONFERENCE ON COMPUTERS AND ACCESSIBILITY, 18, 2016. **Proceedings...** Reno: ACM, 2016. p. 81–90.

GREWE, L. et al. isight: computer vision based system to assist low vision. In: SIGNAL PROCESSING, SENSOR/INFORMATION FUSION, AND TARGET RECOGNITION, 27, 2018. **Proceedings...** Orlando: International Society for Optics and Photonics, 2018. v. 10646, p. 1064613.

GUO, A. et al. Facade: Auto-generating tactile interfaces to appliances. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 18, 2017. **Proceedings ...** Denver: ACM, 2017. p. 5826–5838.

HENRY, S. L.; ABOU-ZAHRA, S.; BREWER, J. The role of accessibility in a universal web. In: WEB FOR ALL CONFERENCE, 11, 2014. **Proceedings...** Seoul: ACM, 2014. Article 17.

HOUGH, P. V. **Method and means for recognizing complex patterns**. [S.l.]: Google Patents, dez. 18 1962. US Patent 3,069,654.

KARASNEH, B.; CHAUDRON, M. R. Extracting uml models from images. In: COMPUTER SCIENCE AND INFORMATION TECHNOLOGY (CSIT), 5, 2013. **Proceedings...** [S.l.]: IEEE, 2013. p. 169–178.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2012. p. 1097–1105.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

LEWIS, P. M. et al. Restoration of vision in blind individuals using bionic devices: A review with a focus on cortical visual prostheses. **Brain Research**, v. 1595, n. 1, p. 51–73, 2015.

LIU, D. et al. Sample-specific late fusion for visual category recognition. In: CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2013. **Proceedings...** Portland: IEEE, 2013.

LUO, Y. H.-L.; CRUZ, L. D. A review and update on the current status of retinal prostheses (bionic eye). **British medical bulletin**, v. 109, n. 1, p. 31–44, 2014.

LUQUE, L. et al. On the inclusion of learners with visual impairment in computing education programs in brazil: practices of educators and perceptions of visually impaired learners. **Journal of the Brazilian Computer Society**, v. 24, n. 1, p. 4, 2018.

LUQUE, L. et al. Model2gether: a tool to support cooperative modeling involving blind people. In: BRAZILIAN CONFERENCE OF SOFTWARE, 7, 2016. **Proceedings...** Maringá, 2016.

MACEDO, M. M. G. de. **Uso da Transformada de Hough na Vetorização de Moldes e outras Aplicações**. Tese (Doutorado) — Master's thesis, Universidade Federal Fluminense, Niterói, RJ, 2005.

MILLER, D. **Can we work together?** Tese (Doutorado) — The University of North Carolina at Chapel Hill, 2009.

MOHAMAD, M.; YAHAYA, W. A. J. W.; WAHID, N. A. The preliminary study of a mobile health application for visual impaired individual. In: INTERNATIONAL CONFERENCE ON EDUCATION AND MULTIMEDIA TECHNOLOGY, 2, 2018. **Proceedings...** Okinawa: ACM, 2018. p. 97–101.

MORENO, V. et al. Automatic classification of web images as uml diagrams. In: SPANISH CONFERENCE ON INFORMATION RETRIEVAL, 4, 2016. **Proceedings...** Granada, 2016. p. 17.

NIELSEN, J. **Usability engineering**. [S.l.]: Elsevier, 1994.

OTSU, N. A thresholding selection method from gray-scale level histograms. **Transactions of Systems, Man and Cybernetics**, IEEE, v. 9, n. 1, p. 62–66, 1979.

PANSANATO, L. T. E. et al. Projeto d4all: acesso e manipulação de diagramas por pessoas com deficiência visual. In: BRAZILIAN SYMPOSIUM ON HUMAN FACTORS IN COMPUTING SYSTEMS,11,2012. **Proceedings...** Cuiabá: Brazilian Computer Society, 2012. p. 33–36.

RAMIREZ, A. R. G. et al. Towards human smart cities: Internet of things for sensory impaired individuals. **Computing**, Springer, v. 99, n. 1, p. 107–126, 2017.

REIS, J. et al. Sistema computacional no auxílio da inclusão da pessoa com deficiência visual no âmbito educacional. In: BRAZILIAN SYMPOSIUM ON COMPUTERS IN EDUCATION (SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO-SBIE), 29, 2018. **Proceedings...** Fortaleza, 2018. p. 943.

SAUTER, V. L. Making data flow diagrams accessible for visually impaired students using excel tables. **Journal of Information Systems Education**, v. 26, n. 1, 2015.

SHILKROT, R. et al. Fingerreader: A finger-worn assistive augmentation. In: **Assistive Augmentation**. [S.l.]: Springer, 2018. p. 151–175.

SILVA, J.; BRAGA, J. C.; DAMACENO, R. Estudo de aplicativos móveis para deficientes visuais no âmbito acadêmico. In: BRAZILIAN SYMPOSIUM ON COMPUTERS IN EDUCATION (SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO-SBIE),26,2015. **Proceedings ...** Maceió, 2015. p. 722.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SIPSER, M. **Introduction to the Theory of Computation**. [S.l.]: Thomson Course Technology Boston, 2006.

SOUSA, K.; MARENGONI, M. Uso de visão computacional em dispositivos moveis para o reconhecimento de faixa de pedestres. In: WORKSHOP DE VISÃO COMPUTACIONAL, 8, 2012. **Anais ...** Goiânia, 2012. p. 1–85.

STALLMANN, M. F. et al. Proofchecker: an accessible environment for automata theory correctness proofs. v. 39, n. 3, p. 48–52, 2007.

STAMFORD, J.; PEACH, B. Scene detection using convolutional neural networks. IET, 2016.

SUN, D.-W. **Computer vision technology for food quality evaluation**. [S.l.]: Academic Press, 2016.

SVAIGEN, A. R.; BINE, L. M. S.; AYLON, L. B. R. An assistive haptic system towards visually impaired computer science learning. In: PERVASIVE TECHNOLOGIES RELATED TO ASSISTIVE ENVIRONMENTS CONFERENCE, 11, 2018. **Proceedings...** Corfu: ACM, 2018. p. 153–156.

SZEGEDY, C. et al. Going deeper with convolutions. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. **Proceedings...** [S.l.], 2015. p. 1–9.

SZELISKI, R. **Computer vision: algorithms and applications**. [S.l.]: Springer Science & Business Media, 2010.

TAPU, R.; MOCANU, B.; ZAHARIA, T. Wearable assistive devices for visually impaired: a state of the art survey. **Pattern Recognition Letters**, Elsevier, 2018.

TEIXEIRA, A. P. P. Acessibilidade digital para a educação inclusiva: desafios e oportunidades. **Diálogo**, n. 27, p. 97–107, 2014.

TORRES, E. F.; MAZZONI, A. A.; ALVES, J. B. da M. A acessibilidade à informação no espaço digital. **Ciência da Informação**, SciELO Brasil, v. 31, n. 3, 2002.

VELÁZQUEZ, R. et al. An outdoor navigation system for blind pedestrians using gps and tactile-foot feedback. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 8, n. 4, p. 578, 2018.

VIEIRA, N. J. **Introdução aos fundamentos da computação: linguagens e máquinas**. [S.l.]: Pioneira Thomson Learning, 2006.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: COMPUTER VISION AND PATTERN RECOGNITION, 2001. CVPR 2001. **Proceedings...** [S.l.], 2001. v. 1, p. I–I.

YUEN, H. et al. Comparative study of hough transform methods for circle finding. **Image and vision computing**, Elsevier, v. 8, n. 1, p. 71–77, 1990.

ZAPIRAIN, B. G. et al. Learning electronics using image processing techniques for describing circuits to blind students. In: IEEE INTERNATIONAL SYMPOSIUM ON SIGNAL PROCESSING AND INFORMATION TECHNOLOGY, 10, 2010. **Proceedings...** Luxor: IEEE, 2010. p. 156–160.

APÊNDICE A – RELATÓRIO DE ACERTOS E ERROS PARA AS IMAGENS DA BASE AF80

Este Apêndice apresenta o relatório de acertos e erros para cada imagem presente na base AF80. Os quadros estão divididos de acordo com a quantidade de estados do autômato, sendo o Quadro A.1 para os autômatos de 1 estado, o Quadro A.2 para os autômatos de 2 estados, o Quadro A.3 para os autômatos de 3 estados, o Quadro A.4 para os autômatos de 4 estados, o Quadro A.5 para os autômatos de 5 estados, o Quadro A.6 para os autômatos de 6 estados, o Quadro A.7 para os autômatos de 7 estados, o Quadro A.8 para os autômatos de 8 ou mais estados. Os seguintes critérios estão presentes nas análises:

- acerto de todos os estados (AE);
- acerto de quais estados são finais/aceitação (AA);
- acerto de quais estados são iniciais (AI);
- acerto de todas as transições entre estados (AT);
- acerto de todos os *loops* (AL).

Quadro A.1: Resultados quantitativos do MannAR para os AFs de 1 estado da base AF80

Autômatos	AE	AA	AI	AT	AL
af1-1	✓	✓	✓	✓	✓
af1-2	✓	✓	✓	✓	✓
af1-3	✓	✓	✓	✓	✓
af1-4	✓	✓	✓	✓	✓
af1-5	✓	✓	✓	✓	✓
af1-6	✓	✓	✓	✓	✓
af1-7	✓	✓	✓	✓	×
af1-8	✓	✓	✓	✓	✓
af1-9	✓	✓	✓	✓	✓
af1-10	✓	✓	×	✓	✓
Total de acertos	10	10	9	10	9

Fonte: (PRÓPRIA, 2019)

Quadro A.2: Resultados quantitativos do MannAR para os AFs de 2 estados da base AF80

Autômatos	AE	AA	AI	AT	AL
af2-1	✓	✓	✓	✓	✓
af2-2	✓	✓	✓	✓	✓
af2-3	✓	✓	✓	✓	✓
af2-4	✓	✓	✓	✓	✓
af2-5	✓	✓	✓	✓	✓
af2-6	✓	✓	✓	✓	✓
af2-7	✓	✓	✓	✓	✓
af2-8	✓	✓	✓	✓	✓
af2-9	✓	✓	✓	✓	✓
af2-10	✓	✓	✓	✓	✓
Total de acertos	10	10	10	10	10

Fonte: (PRÓPRIA, 2019)

Quadro A.3: Resultados quantitativos do MannAR para os AFs de 3 estados da base AF80

Autômatos	AE	AA	AI	AT	AL
af2-1	✓	✓	✓	✓	✓
af2-2	✓	✓	×	✓	✓
af2-3	✓	✓	✓	✓	✓
af2-4	✓	✓	✓	✓	✓
af2-5	✓	✓	✓	✓	✓
af2-6	✓	✓	✓	×	✓
af2-7	✓	✓	✓	✓	✓
af2-8	✓	✓	×	✓	✓
af2-9	✓	✓	×	×	×
af2-10	✓	✓	✓	✓	✓
Total de acertos	10	10	7	8	9

Fonte: (PRÓPRIA, 2019)

Quadro A.4: Resultados quantitativos do MannAR para os AFs de 4 estados da base AF80

Autômatos	AE	AA	AI	AT	AL
af2-1	✓	✓	✓	✓	✓
af2-2	✓	✓	✓	✓	✓
af2-3	✓	✓	✓	✓	✓
af2-4	✓	✓	✓	✓	✓
af2-5	✓	✓	✓	✓	✓
af2-6	✓	✓	✓	✓	✓
af2-7	✓	✓	✓	✓	✓
af2-8	✓	✓	✓	✓	✓
af2-9	×	×	✓	×	×
af2-10	✓	✓	✓	✓	✓
Total de acertos	9	9	10	9	9

Fonte: (PRÓPRIA, 2019)

Quadro A.5: Resultados quantitativos do MannAR para os AFs de 5 estados da base AF80

Autômatos	AE	AA	AI	AT	AL
af2-1	✓	✓	✓	✓	✓
af2-2	✓	✓	✓	✓	✓
af2-3	✓	✓	✓	✓	✓
af2-4	✓	✓	✓	✓	✓
af2-5	✓	×	×	×	×
af2-6	✓	✓	✓	✓	✓
af2-7	✓	✓	✓	✓	✓
af2-8	✓	✓	×	✓	✓
af2-9	✓	✓	✓	✓	✓
af2-10	✓	✓	✓	✓	✓
Total de acertos	10	9	8	9	9

Fonte: (PRÓPRIA, 2019)

Quadro A.6: Resultados quantitativos do MannAR para os AFs de 6 estados da base AF80

Autômatos	AE	AA	AI	AT	AL
af2-1	✓	✓	×	✓	✓
af2-2	✓	✓	✓	✓	✓
af2-3	✓	✓	✓	✓	✓
af2-4	✓	✓	✓	✓	✓
af2-5	✓	✓	✓	✓	✓
af2-6	✓	✓	✓	✓	✓
af2-7	✓	✓	×	×	✓
af2-8	✓	✓	×	×	✓
af2-9	✓	✓	✓	✓	✓
af2-10	✓	✓	×	✓	✓
Total de acertos	10	10	6	8	10

Fonte: (PRÓPRIA, 2019)

Quadro A.7: Resultados quantitativos do MannAR para os AFs de 7 estados da base AF80

Autômatos	AE	AA	AI	AT	AL
af2-1	✓	×	×	×	✓
af2-2	×	×	✓	×	✓
af2-3	✓	✓	✓	✓	✓
af2-4	✓	✓	×	×	✓
af2-5	✓	✓	✓	×	✓
af2-6	✓	×	✓	×	✓
af2-7	✓	✓	✓	✓	✓
af2-8	✓	✓	✓	✓	✓
af2-9	✓	✓	✓	✓	✓
af2-10	✓	✓	×	✓	✓
Total de acertos	9	7	7	5	10

Fonte: (PRÓPRIA, 2019)

Quadro A.8: Resultados quantitativos do MannAR para os AFs de 8 ou mais estados da base AF80

Autômatos	AE	AA	AI	AT	AL
af2-1	✓	✓	✓	✓	✓
af2-2	✓	✓	✓	✓	✓
af2-3	✓	✓	✓	✓	✓
af2-4	✓	✓	×	×	✓
af2-5	✓	✓	✓	✓	✓
af2-6	✓	✓	×	✓	✓
af2-7	✓	×	×	×	✓
af2-8	✓	✓	✓	✓	✓
af2-9	✓	✓	✓	✓	✓
af2-10	✓	✓	×	×	✓
Total de acertos	10	9	6	7	10

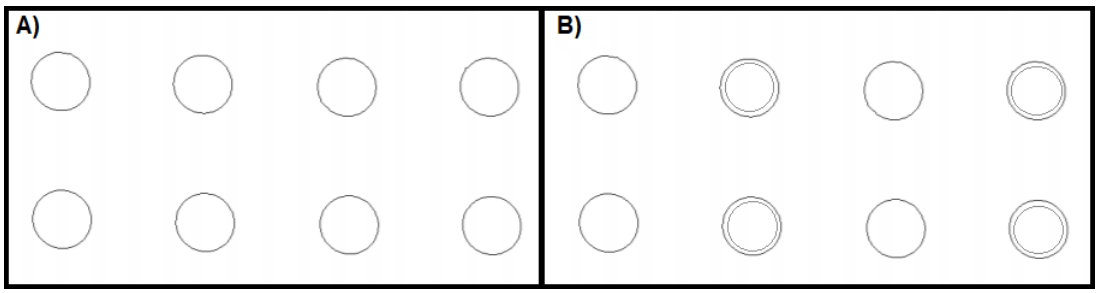
Fonte: (PRÓPRIA, 2019)

APÊNDICE B – COMPARAÇÃO DETALHADA ENTRE OS MÉTODOS BABALOLA (2015) E MANNAR

Este Apêndice apresenta uma comparação entre o MannAR e o método apresentado por Babalola (2015) para os critérios reconhecimento de círculos, arcos e caracteres para as imagens FSA1 (Figura 6.5), FSA2 (Figura 6.6) e FSA3 (Figura 6.7).

A Figura B.1 apresenta a comparação entre os círculos reconhecidos no trabalho de Babalola (2015) (A) e no MannAR (B). Para os círculos são considerados tanto os estados como os círculos que representam os estados finais. Portanto, para a imagem FSA1 são doze círculos que deveriam ser reconhecidos.

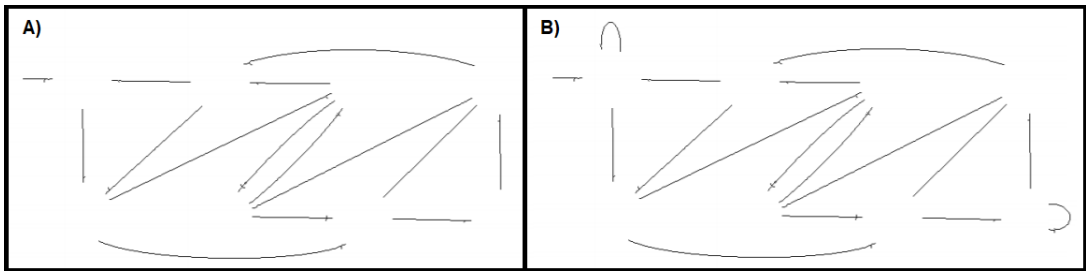
Figura B.1: Comparação entre os círculos reconhecidos por Babalola (2015) (A) e o os reconhecidos pelo MannAR para a imagem FSA1



Fonte: (PRÓPRIA, 2019)

Observa-se que os círculos não reconhecidos por Babalola (2015) são os círculos que representam os estados finais. No método MannAR todos os círculos foram reconhecidos corretamente. Já para os arcos, devem ser consideradas as transições entre dois estados, os *loops* e o símbolo de estado inicial. Portanto, para a imagem FSA1 são 17 elementos que devem ser identificados. Na Figura B.2 apresenta-se a comparação entre o MannAR (B) e o método apresentado por Babalola (2015) (A) para os arcos reconhecidos.

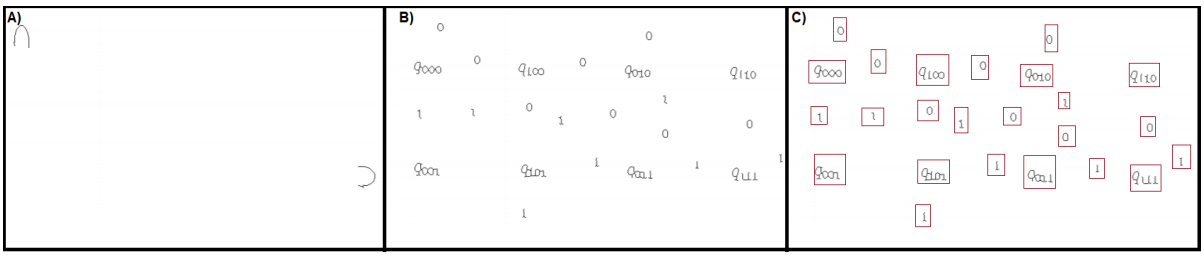
Figura B.2: Comparação entre os arcos reconhecidos por Babalola (2015) (A) e os reconhecidos pelo MannAR para a imagem FSA1



Fonte: (PRÓPRIA, 2019)

No caso dos arcos, o trabalho de Babalola (2015) não reconheceu dois *loops*, sendo que o MannAR segmentou corretamente todos os símbolos considerados arcos. Os dois *loops* foram identificados como caracteres como observa-se na parte A da Figura B.3. Babalola (2015), inicialmente, divide sua imagem em duas camadas, sendo uma de texto e uma de objeto. Após este processo, realiza a classificação dos demais objetos presentes no diagrama.

Figura B.3: Caracteres reconhecidos por Babalola (2015) após a retirada da camada de texto (A), caracteres reconhecidos por Babalola (2015) na camada de texto e os reconhecidos pelo MannAR para a imagem FSA1

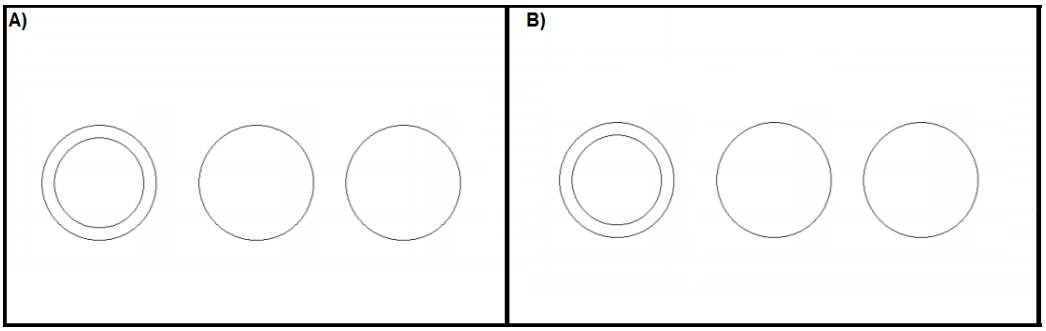


Fonte: (PRÓPRIA, 2019)

Para a camada de texto, Babalola (2015) reconheceu todos os caracteres corretamente (B), porém no processo de classificação dos demais objetos, dois *loops* foram considerados texto (A). Já no sistema MannAR foram identificadas vinte e quatro regiões que possuíam texto que foram enviadas para a realização do OCR (C).

Para a imagem FSA2, o reconhecimento dos círculos foi correto tanto para o método de Babalola (2015) quanto para o MannAR como é mostrado na Figura B.4.

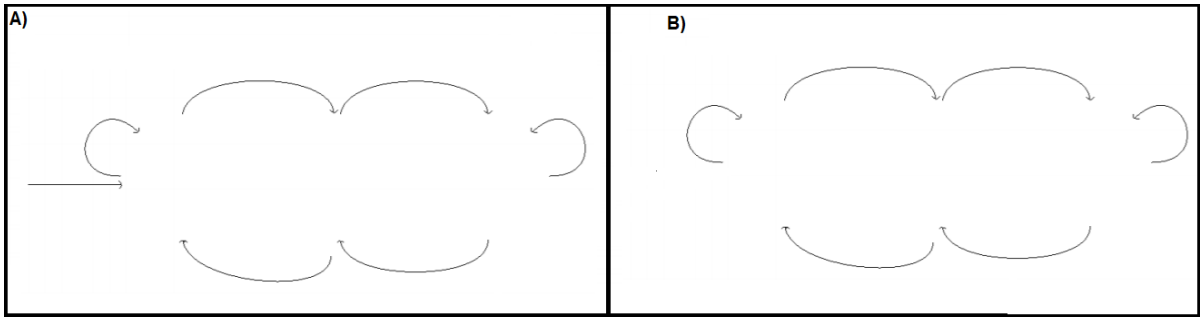
Figura B.4: Comparação entre os círculos reconhecidos por Babalola (2015) (A) e os reconhecidos pelo MannAR para a imagem FSA2



Fonte: (PRÓPRIA, 2019)

Os arcos da imagem FSA2 foram reconhecidos corretamento por Babalola (2015), porém o MannAR não conseguiu reconhecer o símbolo de estado inicial como é apresentado na Figura B.5. Este caso é o único para essas três figuras testadas por Babalola (2015), na qual o resultado não foi gerado corretamente pelo MannAR.

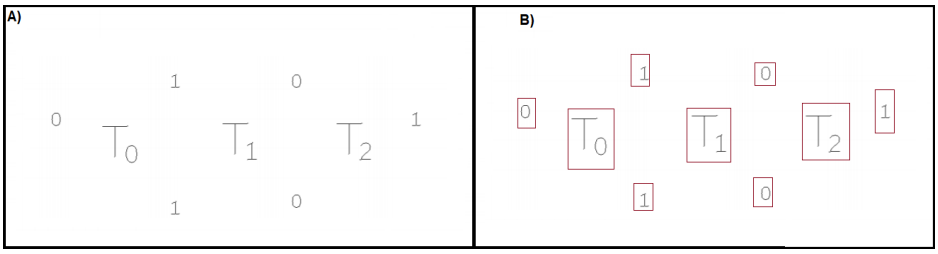
Figura B.5: Comparação entre os arcos reconhecidos por Babalola (2015) (A) e os reconhecidos pelo MannAR para a imagem FSA2



Fonte: (PRÓPRIA, 2019)

Na sequência, comparam-se os caracteres reconhecidos pelos métodos analisados. Os resultados são apresentados na Figura B.6. No MannAR foram reconhecidas nove regiões de texto. Já para o método Babalola, o autor apresenta que foram reconhecidos doze caracteres, porém, não deixa claro se esse resultado foi antes ou depois da divisão entre camadas de texto e objetos que é realizada na abordagem.

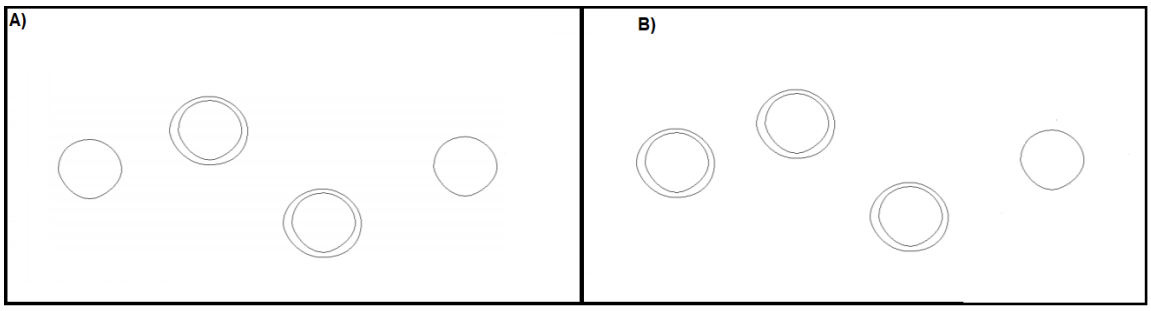
Figura B.6: Caracteres reconhecidos por Babalola (2015) (A) e os reconhecidos pelo MannAR (B) para a imagem FSA2



Fonte: (PRÓPRIA, 2019)

Na Figura B.7 tem-se a comparação entre os círculos reconhecidos por Babalola (A) e os reconhecidos pelo método MannAR (B).

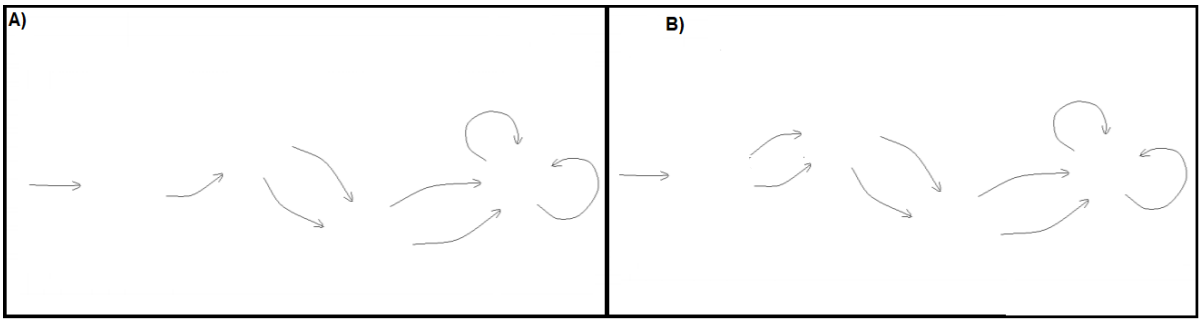
Figura B.7: Comparação entre os círculos reconhecidos por Babalola (2015) (A) e o os reconhecidos pelo MannAR (B) para a imagem FSA3



Fonte: (PRÓPRIA, 2019)

Neste caso, o método de Babalola (2015) não reconheceu um círculo. Tal fato ocasionou o não reconhecimento de uma transição como é possível observar na comparação entre os arcos reconhecidos por Babalola (2015) e o método MannAR, apresentada na Figura B.8.

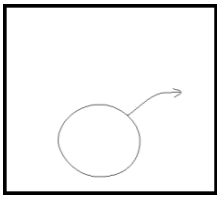
Figura B.8: Comparação entre os arcos reconhecidos por Babalola (2015) (A) e os reconhecidos pelo MannAR para a imagem FSA3



Fonte: (PRÓPRIA, 2019)

Neste caso, ocorre uma inconsistência nos resultados, sendo que em um momento Babalola (2015) apresenta como resultado sete arcos reconhecidos. Tais resultados foram utilizados na Tabela 6.12 que compara os resultados entre os dois métodos. Porém, na imagem de reconhecimento de arcos o autor apresenta oito elementos. Já o método MannAR reconheceu todos os arcos contidos em FSA3. Na sequência, na Figura B.9 são apresentados o círculo e arco não reconhecidos.

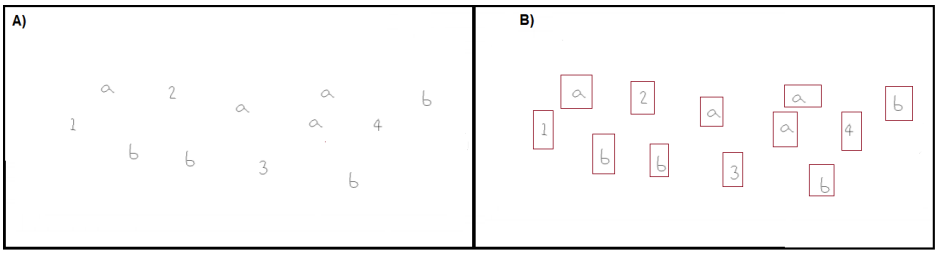
Figura B.9: Objeto não reconhecido por Babalola (2015) para a imagem FSA3



Fonte: (BABALOLA, 2015)

O método de Babalola (2015), não conseguiu realizar a separação dos dois objetos, ocasionando o não reconhecimento de um círculo e de um arco. A seguir, na Figura B.10, comparam-se os resultados para o reconhecimento de caracteres nos dois métodos.

Figura B.10: Caracteres reconhecidos por Babalola (2015) (A) e os reconhecidos pelo MannAR (B) para a imagem FSA3



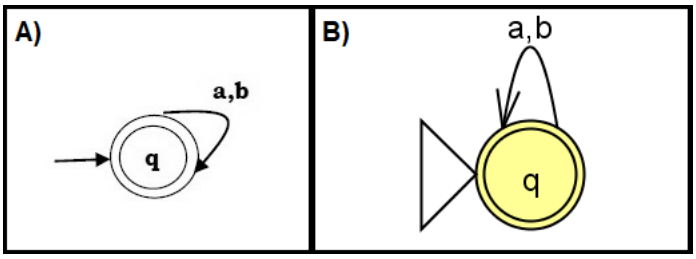
Fonte: (PRÓPRIA, 2019)

Novamente, Babalola (2015) não deixa claro no trabalho se o resultado dos caracteres apresentado na Tabela 6.12 é antes ou depois da separação entre as camadas de texto e de objeto realizada por ele. Neste caso, no resultado apresentado na separação da camada de texto, têm-se doze caracteres, porém, no resultado final Babalola (2015) apresenta que treze caracteres foram reconhecidos. Já no sistema MannAR foram reconhecidas doze áreas de texto.

APÊNDICE C – RESULTADOS DA INTEGRAÇÃO COM O JFLAP

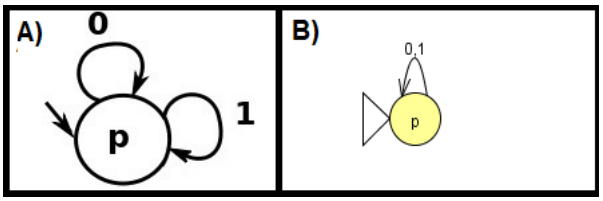
Comparações entre as 12 imagens originais testadas e suas respectivas versões geradas para o JFLAP.

Figura C.1: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 1-1



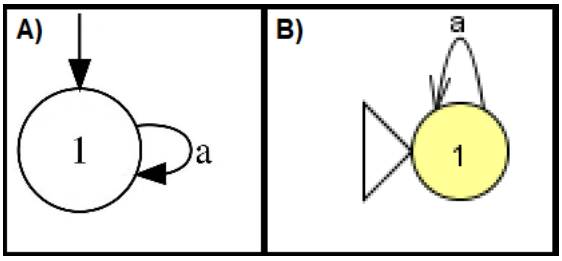
Fonte: (PRÓPRIA, 2019)

Figura C.2: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 1-4



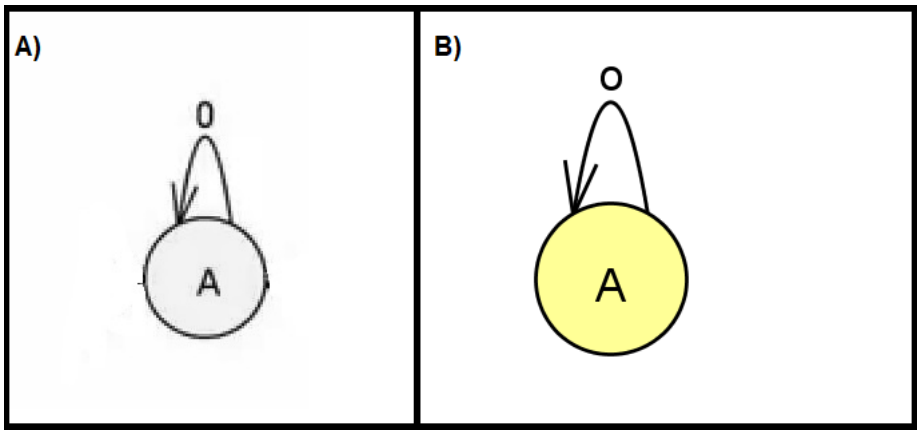
Fonte: (PRÓPRIA, 2019)

Figura C.3: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 1-5



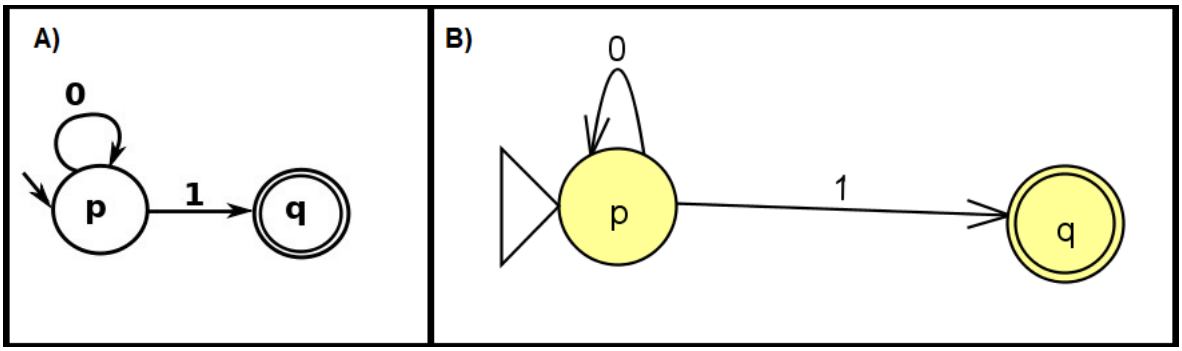
Fonte: (PRÓPRIA, 2019)

Figura C.4: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 1-9



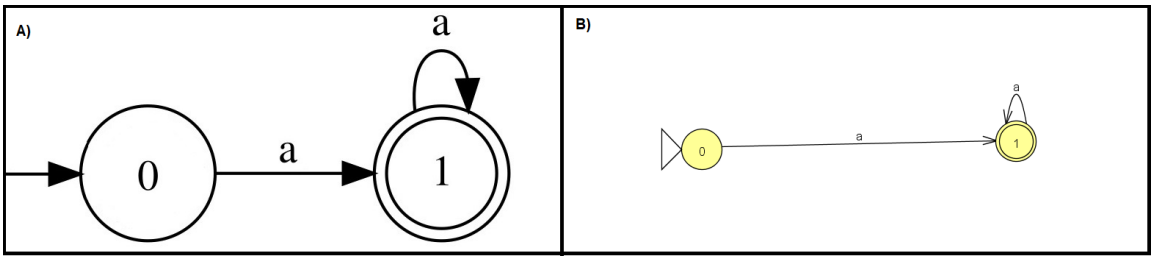
Fonte: (PRÓPRIA, 2019)

Figura C.5: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 2-1



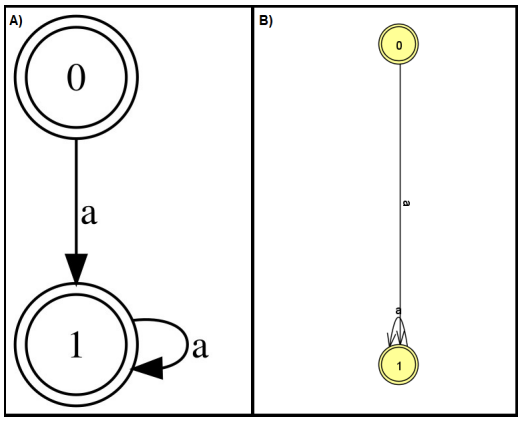
Fonte: (PRÓPRIA, 2019)

Figura C.6: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 2-2



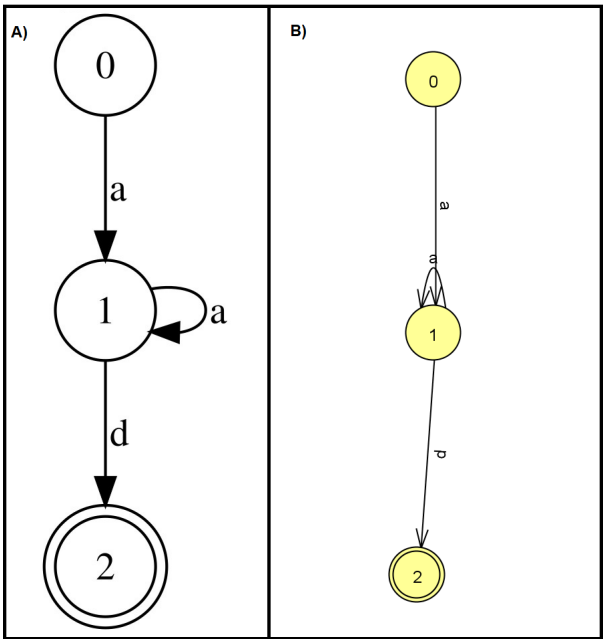
Fonte: (PRÓPRIA, 2019)

Figura C.7: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 1-7



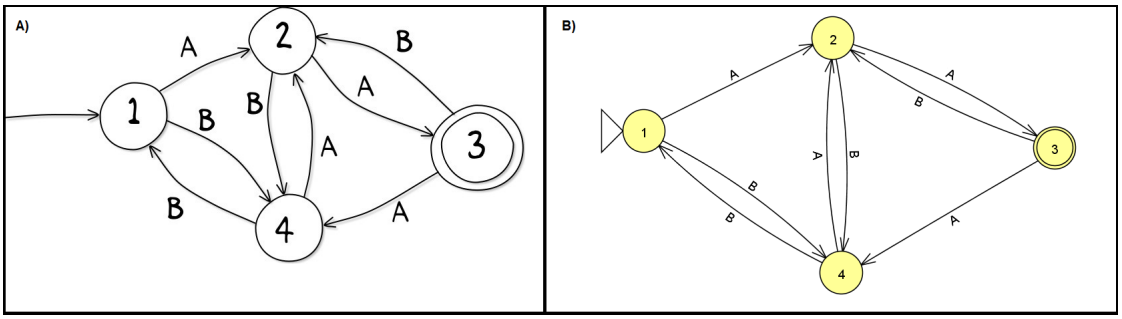
Fonte: (PRÓPRIA, 2019)

Figura C.8: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 3-1



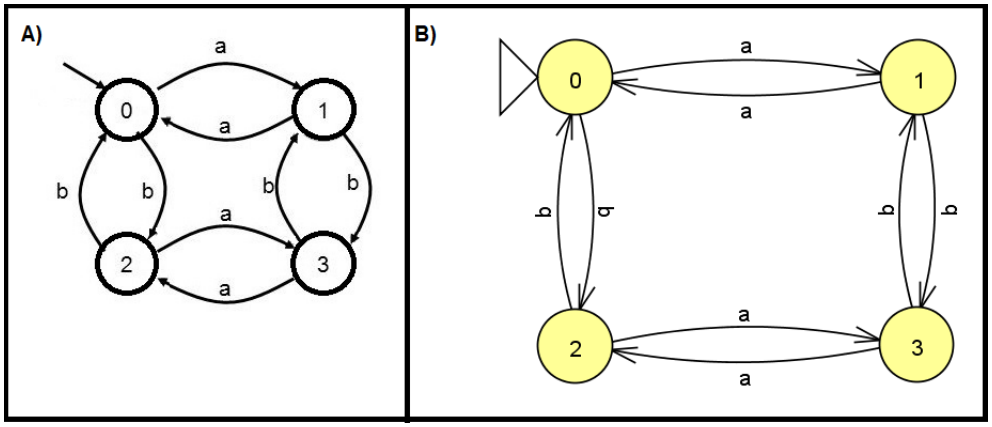
Fonte: (PRÓPRIA, 2019)

Figura C.9: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 4-2



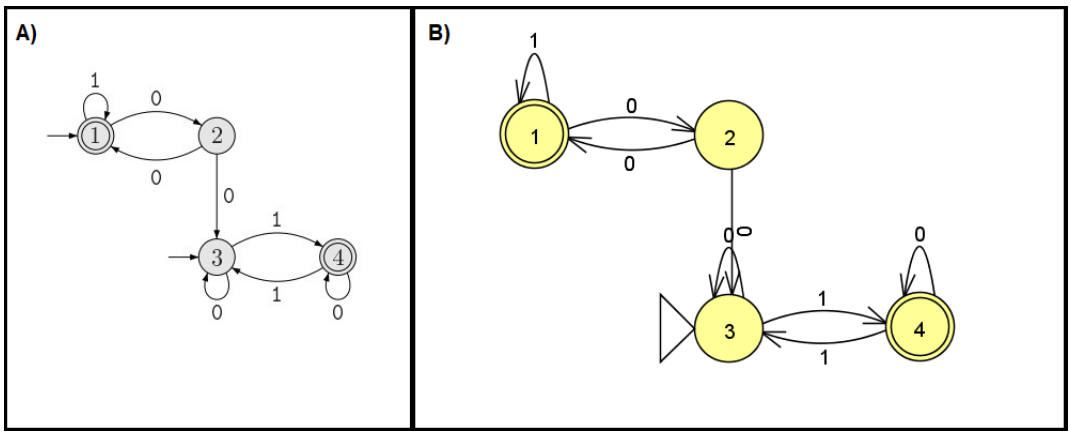
Fonte: (PRÓPRIA, 2019)

Figura C.10: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 4-8



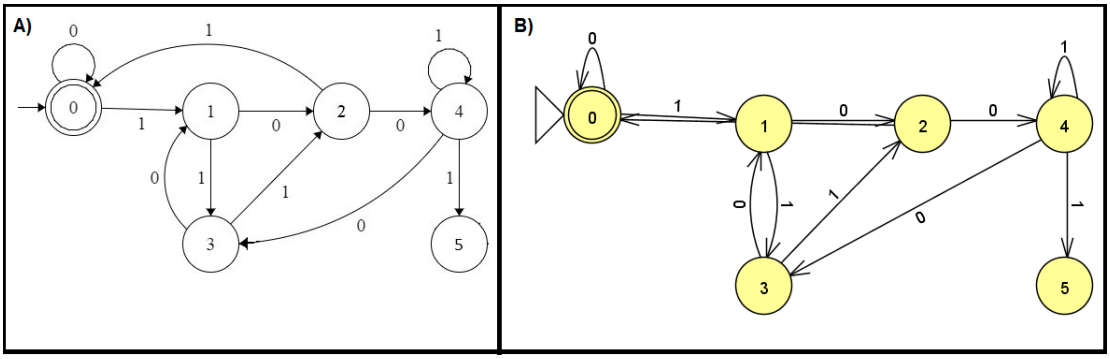
Fonte: (PRÓPRIA, 2019)

Figura C.11: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 4-10



Fonte: (PRÓPRIA, 2019)

Figura C.12: Comparação entre a imagem original (A) e a imagem gerada pelo JFLAP (B) para o AF 6-3



Fonte: (PRÓPRIA, 2019)

ANEXO A – DIRETRIZES DE ACESSIBILIDADE PARA CONTEÚDO WEB (WCAG) 2.0

Este Anexo apresenta as Diretrizes de Acessibilidade para Conteúdo Web, versão 2.0, fornecidas pela W3C. Ao total são quatro princípios e cada um deles está dividido em diretrizes como segue:

- princípio 1: Perceptível - quatro diretrizes;
- princípio 2: Operável - quatro diretrizes;
- princípio 3: Compreensível - três diretrizes;
- princípio 4: Robusto - uma diretriz.

A seguir cada princípio e suas diretrizes são detalhados. A versão apresentada é a versão original em inglês[1], pois não é fornecida uma tradução oficial para pt-br .

Principle 1: Perceivable - Information and user interface components must be presentable to users in ways they can perceive.

- guideline 1.1 Text Alternatives: Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language;
  - 1.1.1 non-text Content: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose, except for the situations listed below. (Level A).
    - controls, Input: If non-text content is a control or accepts user input, then it has a name that describes its purpose. (Refer to Guideline 4.1 for additional requirements for controls and content that accepts user input.);

[1] https://www.w3.org/TR/2008/REC-WCAG20-20081211/

    * time-Based Media: If non-text content is time-based media, then text alternatives at least provide descriptive identification of the non-text content. (Refer to Guideline 1.2 for additional requirements for media.);
    * test: If non-text content is a test or exercise that would be invalid if presented in text, then text alternatives at least provide descriptive identification of the non-text content;
    * sensory: If non-text content is primarily intended to create a specific sensory experience, then text alternatives at least provide descriptive identification of the non-text content;
    * CAPTCHA: If the purpose of non-text content is to confirm that content is being accessed by a person rather than a computer, then text alternatives that identify and describe the purpose of the non-text content are provided, and alternative forms of CAPTCHA using output modes for different types of sensory perception are provided to accommodate different disabilities;
    * decoration, Formatting, Invisible: If non-text content is pure decoration, is used only for visual formatting, or is not presented to users, then it is implemented in a way that it can be ignored by assistive technology.

- guideline 1.2 Time-based Media: Provide alternatives for time-based media;
  - 1.2.1 Audio-only and Video-only (Prerecorded): For prerecorded audio-only and prerecorded video-only media, the following are true, except when the audio or video is a media alternative for text and is clearly labeled as such: (Level A);
    * prerecorded Audio-only: An alternative for time-based media is provided that presents equivalent information for prerecorded audio-only content;
    * prerecorded Video-only: Either an alternative for time-based media or an audio track is provided that presents equivalent information for prerecorded video-only content.
  - 1.2.2 Captions (Prerecorded): Captions are provided for all prerecorded audio content in synchronized media, except when the media is a media alternative for text and is clearly labeled as such. (Level A);
  - 1.2.3 Audio Description or Media Alternative (Prerecorded): An alternative for time-based media or audio description of the prerecorded video content is provided for synchronized media, except when the media is a media alternative for text and is clearly labeled as such. (Level A);

  - 1.2.4 Captions (Live): Captions are provided for all live audio content in synchronized media. (Level AA);
  - 1.2.5 Audio Description (Prerecorded): Audio description is provided for all prerecorded video content in synchronized media. (Level AA);
  - 1.2.6 Sign Language (Prerecorded): Sign language interpretation is provided for all prerecorded audio content in synchronized media. (Level AAA);
  - 1.2.7 Extended Audio Description (Prerecorded): Where pauses in foreground audio are insufficient to allow audio descriptions to convey the sense of the video, extended audio description is provided for all prerecorded video content in synchronized media. (Level AAA);
  - 1.2.8 Media Alternative (Prerecorded): An alternative for time-based media is provided for all prerecorded synchronized media and for all prerecorded video-only media. (Level AAA);
  - 1.2.9 Audio-only (Live): An alternative for time-based media that presents equivalent information for live audio-only content is provided. (Level AAA).

- guideline 1.3 Adaptable: Create content that can be presented in different ways (for example simpler layout) without losing information or structure;
  - 1.3.1 Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A);
  - 1.3.2 Meaningful Sequence: When the sequence in which content is presented affects its meaning, a correct reading sequence can be programmatically determined. (Level A);
  - 1.3.3 Densory Characteristics: Instructions provided for understanding and operating content do not rely solely on sensory characteristics of components such as shape, size, visual location, orientation, or sound. (Level A).

- guideline 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.
  - 1.4.1 Use of Color: Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element. (Level A);

- 1.4.2 Audio Control: If any audio on a Web page plays automatically for more than 3 seconds, either a mechanism is available to pause or stop the audio, or a mechanism is available to control audio volume independently from the overall system volume level. (Level A);
- 1.4.3 Contrast (Minimum): The visual presentation of text and images of text has a contrast ratio of at least 4.5:1, except for the following: (Level AA);
- 1.4.4 Resize text: Except for captions and images of text, text can be resized without assistive technology up to 200 percent without loss of content or functionality. (Level AA);
- 1.4.5 Images of Text: If the technologies being used can achieve the visual presentation, text is used to convey information rather than images of text except for the following: (Level AA);
- 1.4.6 Contrast (Enhanced): The visual presentation of text and images of text has a contrast ratio of at least 7:1, except for the following: (Level AAA);
- 1.4.7 Low or No Background Audio: For prerecorded audio-only content that (1) contains primarily speech in the foreground, (2) is not an audio CAPTCHA or audio logo, and (3) is not vocalization intended to be primarily musical expression such as singing or rapping, at least one of the following is true: (Level AAA);
- 1.4.8 Visual Presentation: For the visual presentation of blocks of text, a mechanism is available to achieve the following: (Level AAA);
- 1.4.9 Images of Text (No Exception): Images of text are only used for pure decoration or where a particular presentation of text is essential to the information being conveyed. (Level AAA).

Principle 2: Operable - User interface components and navigation must be operable.

- guideline 2.1 Keyboard Accessible: Make all functionality available from a keyboard;
  - 2.1.1 Keyboard: All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes, except where the underlying function requires input that depends on the path of the user's movement and not just the endpoints. (Level A);
  - 2.1.2 No Keyboard Trap: If keyboard focus can be moved to a component of the page using a keyboard interface, then focus can be moved away from that component

using only a keyboard interface, and, if it requires more than unmodified arrow or tab keys or other standard exit methods, the user is advised of the method for moving focus away. (Level A);

- 2.1.3 Keyboard (No Exception): All functionality of the content is operable through a keyboard interface without requiring specific timings for individual keystrokes. (Level AAA).

• guideline 2.2 Enough Time: Provide users enough time to read and use content;

- 2.2.1 Timing Adjustable: For each time limit that is set by the content, at least one of the following is true: (Level A);
- 2.2.2 Pause, Stop, Hide: For moving, blinking, scrolling, or auto-updating information, all of the following are true: (Level A);
- 2.2.3 No Timing: Timing is not an essential part of the event or activity presented by the content, except for non-interactive synchronized media and real-time events. (Level AAA);
- 2.2.4 Interruptions: Interruptions can be postponed or suppressed by the user, except interruptions involving an emergency. (Level AAA);
- 2.2.5 Re-authenticating: When an authenticated session expires, the user can continue the activity without loss of data after re-authenticating. (Level AAA).

• guideline 2.3 Seizures: Do not design content in a way that is known to cause seizures;

- 2.3.1 Three Flashes or Below Threshold: Web pages do not contain anything that flashes more than three times in any one second period, or the flash is below the general flash and red flash thresholds. (Level A);
- 2.3.2 Three Flashes: Web pages do not contain anything that flashes more than three times in any one second period. (Level AAA).

• guideline 2.4 Navigable: Provide ways to help users navigate, find content, and determine where they are;

- 2.4.1 Bypass Blocks: A mechanism is available to bypass blocks of content that are repeated on multiple Web pages. (Level A);
- 2.4.2 Page Titled: Web pages have titles that describe topic or purpose. (Level A);
- 2.4.3 Focus Order: If a Web page can be navigated sequentially and the navigation sequences affect meaning or operation, focusable components receive focus in an order that preserves meaning and operability. (Level A);

- 2.4.4 Link Purpose (In Context): The purpose of each link can be determined from the link text alone or from the link text together with its programmatically determined link context, except where the purpose of the link would be ambiguous to users in general. (Level A);
- 2.4.5 Multiple Ways: More than one way is available to locate a Web page within a set of Web pages except where the Web Page is the result of, or a step in, a process. (Level AA);
- 2.4.6 Headings and Labels: Headings and labels describe topic or purpose. (Level AA);
- 2.4.7 Focus Visible: Any keyboard operable user interface has a mode of operation where the keyboard focus indicator is visible. (Level AA);
- 2.4.8 Location: Information about the user's location within a set of Web pages is available. (Level AAA);
- 2.4.9 Link Purpose (Link Only): A mechanism is available to allow the purpose of each link to be identified from link text alone, except where the purpose of the link would be ambiguous to users in general. (Level AAA);
- 2.4.10 Section Headings: Section headings are used to organize the content. (Level AAA).

Principle 3: Understandable - Information and the operation of user interface must be understandable.

- guideline 3.1 Readable: Make text content readable and understandable;
  - 3.1.1 Language of Page: The default human language of each Web page can be programmatically determined. (Level A);
  - 3.1.2 Language of Parts: The human language of each passage or phrase in the content can be programmatically determined except for proper names, technical terms, words of indeterminate language, and words or phrases that have become part of the vernacular of the immediately surrounding text. (Level AA);
  - 3.1.3 Unusual Words: A mechanism is available for identifying specific definitions of words or phrases used in an unusual or restricted way, including idioms and jargon. (Level AAA);
  - 3.1.4 Abbreviations: A mechanism for identifying the expanded form or meaning of abbreviations is available. (Level AAA);

  - 3.1.5 Reading Level: When text requires reading ability more advanced than the lower secondary education level after removal of proper names and titles, supplemental content, or a version that does not require reading ability more advanced than the lower secondary education level, is available. (Level AAA);
  - 3.1.6 Pronunciation: A mechanism is available for identifying specific pronunciation of words where meaning of the words, in context, is ambiguous without knowing the pronunciation. (Level AAA).

- guideline 3.2 Predictable: Make Web pages appear and operate in predictable ways;
  - 3.2.1 On Focus: When any component receives focus, it does not initiate a change of context. (Level A);
  - 3.2.2 On Input: Changing the setting of any user interface component does not automatically cause a change of context unless the user has been advised of the behavior before using the component. (Level A);
  - 3.2.3 Consistent Navigation: Navigational mechanisms that are repeated on multiple Web pages within a set of Web pages occur in the same relative order each time they are repeated, unless a change is initiated by the user. (Level AA);
  - 3.2.4 Consistent Identification: Components that have the same functionality within a set of Web pages are identified consistently. (Level AA);
  - 3.2.5 Change on Request: Changes of context are initiated only by user request or a mechanism is available to turn off such changes. (Level AAA).

- guideline 3.3 Input Assistance: Help users avoid and correct mistakes.
  - 3.3.1 Error Identification: If an input error is automatically detected, the item that is in error is identified and the error is described to the user in text. (Level A);
  - 3.3.2 Labels or Instructions: Labels or instructions are provided when content requires user input. (Level A);
  - 3.3.3 Error Suggestion: If an input error is automatically detected and suggestions for correction are known, then the suggestions are provided to the user, unless it would jeopardize the security or purpose of the content. (Level AA);
  - 3.3.4 Error Prevention (Legal, Financial, Data): For Web pages that cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses, at least one of the following is true: (Level AA);

- 3.3.5 Help: Context-sensitive help is available. (Level AAA);
- 3.3.6 Error Prevention (All): For Web pages that require the user to submit information, at least one of the following is true: (Level AAA).

Principle 4: Robust - Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies.

- guideline 4.1 Compatible: Maximize compatibility with current and future user agents, including assistive technologies.
  - 4.1.1 Parsing: In content implemented using markup languages, elements have complete start and end tags, elements are nested according to their specifications, elements do not contain duplicate attributes, and any IDs are unique, except where the specifications allow these features. (Level A);
  - 4.1.2 Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined; states, properties, and values that can be set by the user can be programmatically set; and notification of changes to these items is available to user agents, including assistive technologies. (Level A).

ANEXO B – PARECER CONSUBSTANCIADO DO COPEP - UEM

UNIVERSIDADE ESTADUAL DE MARINGÁ

PARECER CONSUBSTANCIADO DO CEP

DADOS DO PROJETO DE PESQUISA

Título da Pesquisa: Tecnologias assistivas de imagens digitais para deficientes visuais
Pesquisador: LINNYER BEATRYS RUIZ AYLON
Área Temática:
Versão: 2
CAAE: 02198318.0.0000.0104
Instituição Proponente: CTC - Centro de Tecnologia
Patrocinador Principal: Financiamento Próprio

DADOS DO PARECER

Número do Parecer: 3.098.506

Apresentação do Projeto:
Trata-se de projeto de pesquisa proposto por pesquisador vinculado à Universidade Estadual de Maringá.

Objetivo da Pesquisa:
Objetivo Primário: Investigar se representações alternativas de imagens digitais para deficientes visuais contribuem para uma melhoria de sua interpretação e contextualização. Objetivos Secundários: Investigar se as representações criadas são válidas e auxiliam em disciplinas relacionadas aos cursos de Computação; Avaliar a facilidade de uso dos sistemas criados nesta pesquisa; Avaliar a aceitação e grau de satisfação dos deficientes visuais com as soluções apresentadas.

Avaliação dos Riscos e Benefícios:
Avalia-se que os possíveis riscos a que estarão submetidos os sujeitos da pesquisa serão suportados pelos benefícios apontados.

Comentários e Considerações sobre a Pesquisa:
Pesquisa possui caráter exploratório e qualitativo para avaliar duas tecnologias assistivas para a interpretação de imagens digitais por parte de deficientes visuais: MannAR e MannaHap. No estudo proposto a população será constituída de voluntários que deverão possuir baixa visão profunda, cegueira parcial ou total e serem maiores de 18 anos atendidos pelo Programa Multidisciplinar de

Endereço: Av. Colombo, 5790, UEM-PPG, sala 4
Bairro: Jardim Universitário **CEP:** 87.020-900
UF: PR **Município:** MARINGA
Telefone: (44)3011-4597 **Fax:** (44)3011-4444 **E-mail:** copep@uem.br

Pesquisa e Apoio à Pessoa com Deficiência e Necessidades Educativas Especiais da UEM (PROPAE-UEM). Os voluntários serão divididos em dois grupos: pessoas que cursam ou cursaram graduação na área de computação, e pessoas que não cursaram e receberão um treinamento individual para utilização das tecnologias assistivas desenvolvidas, de modo a garantir que o usuário possua o conhecimento básico de manipulação de cada sistema. Após o treinamento, o usuário fará um primeiro experimento utilizando o MannAR (software que faz o reconhecimento dos objetos da imagem e das relações entre eles para que seja produzida a descrição correta do conteúdo da imagem, que ao final do processo será representada por um texto). O usuário irá submeter ao sistema 3 imagens digitais. Cada imagem enviada resultará em um texto com sua descrição que o usuário poderá ouvir por meio da leitura de tela do computador. O usuário fará um segundo experimento utilizando o MannaHap (hardware tátil que proporciona ao usuário uma forma alternativa de compreender a imagem, sendo esta por meio do toque, para que o usuário “sinta” a imagem em conjunto com uma descrição auditiva que complementa o conteúdo e contexto da mesma). Fornecer percepções táteis ao usuário por meio do toque numa mesa de pinos. A partir do toque, um informação por áudio é gerada, acometendo percepções cinestésicas que permitem ao usuário compreender o conteúdo daquela imagem. Serão submetidos ao sistema para sua apreciação 3 diferentes imagens digitais, na qual ele irá “sentí-la” e obter uma contextualização via áudio. Posteriormente será aplicado um questionário de avaliação baseado nas Heurísticas de usabilidade de Nielsen e no TAM (Technology Acceptance Model), com o objetivo de avaliar as duas tecnologias desenvolvidas. O questionário será dividido em 3 partes: 1) avaliação individual do MannAR; 2) avaliação individual do MannaHap; 3) avaliação geral das duas tecnologias; Declaram ainda que as informações serão utilizadas somente para os fins desta pesquisa, e serão tratadas com o mais absoluto sigilo e confidencialidade, de modo a preservar a identidade do participante e serão armazenados no formato digital, ficando em posse da professora coordenadora da pesquisa.

Considerações sobre os Termos de apresentação obrigatória:

Apresenta a folha de rosto devidamente preenchida e assinada pelo responsável institucional. Apresentam autorização do PROPAE-UEM para a realização do projeto. Descreve gastos sob a responsabilidade dos pesquisadores. O período de abrangência de projeto é de 01/01/19 a 15/02/2019. O TCLE encontra-se numa linguagem clara e contempla as garantias de sigilo e confidencialidade.

Conclusões ou Pendências e Lista de Inadequações:

O Comitê Permanente de Ética em Pesquisa Envolvendo Seres Humanos da Universidade Estadual

de Maringá é de parecer favorável à aprovação do protocolo de pesquisa apresentado.

Considerações Finais a critério do CEP:

Face ao exposto e considerando a normativa ética vigente, este Comitê se manifesta pela aprovação do protocolo de pesquisa em tela.

Este parecer foi elaborado baseado nos documentos abaixo relacionados:

Tipo Documento	Arquivo	Postagem	Autor	Situação
Informações Básicas do Projeto	PB_INFORMAÇÕES_BÁSICAS_DO_PROJETO_1238049.pdf	27/11/2018 11:25:01		Aceito
Outros	carta_comite_alteracoes.pdf	27/11/2018 11:24:14	ALISSON RENAN SVAIGEN	Aceito
Outros	autorizacao_propae.pdf	27/11/2018 11:23:39	ALISSON RENAN SVAIGEN	Aceito
Projeto Detalhado / Brochura Investigador	ProjetoDetalhado.pdf	27/11/2018 11:22:19	ALISSON RENAN SVAIGEN	Aceito
TCLE / Termos de Assentimento / Justificativa de Ausência	TCLEsujeitosdapesquisa.pdf	01/11/2018 14:21:02	ALISSON RENAN SVAIGEN	Aceito
Folha de Rosto	folhaDeRosto_digitalizada.pdf	30/10/2018 09:21:02	ALISSON RENAN SVAIGEN	Aceito

Situação do Parecer:

Aprovado

Necessita Apreciação da CONEP:

Não

MARINGA, 20 de Dezembro de 2018

Assinado por:
Ricardo Cesar Gardiolo
(Coordenador(a))

Endereço: Av. Colombo, 5790, UEM-PPG, sala 4
Bairro: Jardim Universitário **CEP:** 87.020-900
UF: PR **Município:** MARINGA
Telefone: (44)3011-4597 **Fax:** (44)3011-4444 **E-mail:** copep@uem.br