

**CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

Análise de tecnologias de reconhecimento para quebra de CAPTCHAs

VITOR DERUBE DOS SANTOS

Marília, 2016

**CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

Análise de tecnologias de reconhecimento para quebra de CAPTCHAs

Monografia apresentada ao Centro
Universitário Eurípides de Marília
como parte dos requisitos
necessários para a obtenção do
grau de Bacharel em Ciência da
Computação.

Orientador: Prof. Rodolfo
Chiaromonte



CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
MANTIDO PELA FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"


BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO


Vítor Derobe dos Santos

Análise de tecnologias de reconhecimento para quebra de CAPTCHAs.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em
Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de
Bacharel em Ciência da Computação.

Nota: 10,0 (Dez)

Orientador: Rodolfo Barros Chiaramonte 

1º. Examinador: Mauricio Duarte 

2º. Examinador: Luan Cardoso dos Santos 

Marília, 06 de dezembro de 2016.

Dedico este trabalho a todas as pessoas importantes que me apoiaram durante estes 4 anos de faculdade, especialmente minha família e amigos.

AGRADECIMENTOS

Agradeço a Deus por estar presente em todos os momentos de minha vida e me conceder a capacidade para alcançar meus sonhos e objetivos.

A todas as pessoas próximas a mim que contribuíram de diversas formas para a minha formação, destacando os meus fiéis amigos César Torralvo Alves e Gabriel Caires Guimarães que estiveram comigo durante os quatro anos de faculdade.

A meu pai Adauto Silva dos Santos, minha mãe Iléia Josiane Derobe dos Santos e meu irmão Gustavo Derobe dos Santos por todo o incentivo e apoio concedidos durante toda a minha vida.

Ao Centro Universitário Eurípides de Marília e seu corpo docente pelas diversas oportunidades de crescimento acadêmico e ensinamentos profissionais respectivamente, especialmente ao meu orientador Prof. Me. Rodolfo Barros Chiamonte por acreditar em meu desempenho em diversas atividades acadêmicas além do trabalho de conclusão de curso, meus eternos agradecimentos.

E a Boa Vista Serviços por me conceder uma oportunidade de estágio onde muitos conhecimentos foram adquiridos.

“A alegria está na luta, na tentativa, no sofrimento envolvido e não na vitória propriamente dita”.
(Mahatma Gandhi)

RESUMO

A segurança das informações na internet sempre foi uma preocupação global, com o passar dos anos muitas tecnologias diferentes surgiram para complementar a segurança e dificultar a vida dos hackers e scripts automáticos de navegação na internet. Uma delas é o CAPTCHA (teste de Turing público completamente automatizado para diferenciação entre computadores e humanos) que surgiu com o propósito de diferenciar usuários humanos dos robôs de captação de dados. Os CAPTCHAs podem ser encontrados em toda internet e se tornaram muito populares nos últimos tempos. Este trabalho visa um estudo sobre os CAPTCHAs baseados em texto; mais especificamente o objetivo é o reconhecimento do CAPTCHA utilizado no site do currículo Lattes, através de técnicas de processamento de imagem, segmentação de caracteres e ferramentas de reconhecimento, como o Tesseract, GOCR e a Rede Neural Artificial Rápida.

Palavras-Chave: CAPTCHA; Reconhecimento Óptico de Caracteres; Processamento de Imagens; Rede Neural Artificial.

ABSTRACT

The security of information on the internet always was a global concern, over the years many technologies emerged to complement the security and complicate the work of hackers and automated Web navigation scripts. One of them is the CAPTCHA (Completely Automated Public Turing tests to tell Computers and Humans Apart) that emerged with the idea of distinguish human users from Web crawlers. The CAPTCHAs can be found all over the web and they have become very popular over the last years. This project aims to study about text-based CAPTCHAs; more specifically the goal is to recognize the CAPTCHA of curriculum Lattes website, through image processing technics, characters segmentation technics and recognizing tools, like Tesseract, GOCR and the Fast Artificial Neural Network.

Palavras-Chave: CAPTCHA; Optical Character Recognition; Image Processing; Artificial Neural Network.

LISTA DE ILUSTRAÇÕES

Figura 1 – CAPTCHA utilizado no site da Receita Federal brasileira.....	17
Figura 2 – CAPTCHA utilizado no site da consulta do currículo Lattes.	19
Figura 3 – Exemplos de CAPTCHAs baseados em texto.	20
Figura 4 – Exemplo de CAPTCHA de difícil resolução para humanos.....	21
Figura 5: a, b, c, d – Exemplo de variação do CAPTCHA do RegistroBr.	22
Figura 6 – Exemplo de CAPTCHA baseado em imagem.	22
Figura 7 – Exemplo de CAPTCHA baseado em questões.	23
Figura 8 – Exemplo de CAPTCHA baseado em questões e difíceis para humanos.	24
Figura 9: a – No CAPTCHA sem preenchimento, b – No CAPTCHA preenchido.....	25
Figura 10 – Exemplo do No CAPTCHA reCAPTCHA apresentando uma imagem.	26
Figura 11 – Exemplo do No CAPTCHA reCAPTCHA apresentando uma foto do Google maps.....	27
Figura 12 – Exemplo do reCAPTCHA exibindo um trecho de livro.	27
Figura 13 – Exemplo de núcleos do Morphology.	32
Figura 14 – Exemplo de erosão realizada com o Morphology.....	32
Figura 15 – Exemplo do tratamento de erosão.	33
Figura 16 – Exemplo de dilatação realizada com o Morphology.....	33
Figura 17 – Exemplo do tratamento de dilatação.	33
Figura 18 – Exemplo do tratamento Thinning realizado com o Morphology.	34
Figura 19 – Exemplo da eficiência do algoritmo de Gibbs de-noising.	40
Figura 20 – Pirâmide representando a importância das etapas da quebra de um CAPTCHA baseado em texto.	42
Figura 21: a, b, c, d, e, f, g. – CAPTCHAs do site do currículo Lattes.	43
Figura 22: a, b, c, d, e, f, g. – CAPTCHAs do site do currículo Lattes, após o tratamento de Thresholding e do corte da imagem.	44
Figura 23: a, b, c, d, e, f, g. – CAPTCHAs do site do currículo Lattes, após o tratamento de dilatação.....	45
Figura 24: a, b, c – CAPTCHAs do site do currículo Lattes com o mesmo plano de fundo....	46
Figura 25 – Parte do algoritmo para colorização dos pixels.	46
Figura 26: a, b, c– CAPTCHAs do site do currículo Lattes, após o tratamento de remoção do traço branco.	47
Figura 27: a, b, c – CAPTCHAs do site do currículo Lattes, após o tratamento de remoção de ruidos.	47
Figura 28 – Parte do algoritmo para transformar a imagem em matriz.....	48
Figura 29: a, b, c, d, e, f, g – CAPTCHAs do site do currículo Lattes, após o tratamento de Gibbs de-noising.....	49
Figura 30: a, b– CAPTCHAs do site do currículo Lattes, após o algoritmo de preenchimento vertical.	49
Figura 31 – CAPTCHA do Lattes.	50
Figura 32 – Captura de tela da matriz que representa o CAPTCHA.....	50
Figura 33 – Captura de tela da matriz que representa o CAPTCHA.....	51
Figura 34: a, b, c, d, e, f, g, h, i – Estado dos CAPTCHAs através dos passos de quebra.	51
Figura 35: a, b, c, d, e, f – Representação dos cortes nos CAPTCHAs que falharam.....	52

Figura 36: a, b – Representação dos caracteres em forma de matriz exibidos no terminal do Linux.....	53
Figura 37: a, b, c, d, e, f – Caracteres do CAPTCHA do Lattes segmentados.	55
Figura 38 – CAPTCHA com a letra “W” deformada.....	58
Figura 39 – Comando executado no terminal Linux.	66
Figura 40 – Comando executado no terminal Linux.	66

LISTA DE TABELAS

Tabela 1. Estatísticas da execução do algoritmo de quebra do CAPTCHA do LATTES.....58

LISTA DE GRÁFICOS

Gráfico 1. Taxa de acerto dos CAPTCHAs.	56
--	----

LISTA DE ABREVIATURAS E SIGLAS

CAPTCHA	Completely Automated Public Turing tests to tell Computers and Humans Apart
FANN	Fast Artificial Neural Network
GOOCR	Optical Character Recognition developed under GNU Public License
OCR	Optical Character Recognition
CPF	Cadastro de Pessoas Físicas
CNPJ	Cadastro Nacional da Pessoa Jurídica
DDoS	Distributed Denial-of-Service attack
IA	Inteligência Artificial
RNA	Rede Neural Artificial
HP	Hewlett-Packard
DLL	Dynamic Link Library
GNU	General Public License
PNM	Portable Any Map
PGM	Portable Gray Map
JPEG	Joint Photographic Experts Group
SVM	Support Vector Machine
CNN	Cable News Network
NIH	National Institutes of Health
SOM	Self Organizing Map

Sumário

Sumário.....	14
1 Introdução.....	15
1.1 O Que é CAPTCHA	16
1.2 Motivação para a quebra do CAPTCHA	17
1.3 Objetivos.....	18
1.4 Organização do Trabalho	19
2 Tipos de CAPTCHA	20
2.1 Baseados em Texto	20
2.2 Baseados em Imagem	22
2.3 Baseados em questões.....	23
2.4 Baseado em áudio	24
2.5 No CAPTCHA.....	25
2.6 Considerações finais do capítulo	28
3 Processos da quebra de um CAPTCHA baseado em texto	29
3.1 Tratamento da imagem com ImageMagick	29
3.1.1 Remoção de Fundo (Background removal)	30
3.1.2 Up-sampling	30
3.1.3 Limiarização (Thresholding).....	30
3.1.4 Line removal (Remoção de linhas)	31
3.1.5 Thinning (Desbaste)	34
3.1.6 Algoritmo de Gibbs de-noising	34
3.2 Segmentação dos caracteres.....	35
3.2.1 Flood-filling segmentation	35
3.2.2 Histogram segmentation.....	35
3.2.3 Pattern matching.....	36
3.3 Reconhecimento dos caracteres	36
3.3.1 FANN.....	36
3.3.2 Tesseract.....	37
3.3.3 GOCR.....	38
3.4 Considerações finais do capítulo	38
4 Trabalhos correlatos	39
4.1 Considerações finais do capítulo	41
5 Implementação	42
5.1 Tratamento da imagem	42
5.2 Segmentação dos caracteres.....	50
5.3 Reconhecimento dos caracteres	52
5.3.1 Reconhecimento dos caracteres com a FANN.....	52
5.3.2 Reconhecimento dos caracteres com a OCR Tesseract	55
5.3.2 Reconhecimento dos caracteres com a GOCR.....	55
5.3 Considerações finais do capítulo	56
6 Resultados e contribuições	57
6.1 Considerações finais do capítulo	59
Conclusão	60
Referências Bibliográficas.....	62
Apêndice A – Comandos de execução do ImageMagick	66

1 Introdução

A segurança das informações na internet sempre foi uma preocupação global, com o passar dos anos muitas tecnologias diferentes surgiram para complementar a segurança e dificultar a vida dos hackers e scripts automáticos de navegação na internet - também conhecidos como *Web Crawlers* ou robôs. Um robô é capaz de navegar pela internet numa velocidade maior que usuários humanos, e geralmente são usados para captação de dados da Web (Catanese, 2011). O seu uso porém não é bem visto por algumas aplicações na internet, um uso massivo de um robô que realiza várias consultas em um site com uma velocidade muito rápida, pode ocasionar em uma falha no servidor por exemplo, por isso os sites buscam uma maneira de evitar que esses scripts de navegações automáticas naveguem livremente em seus domínios (Mahato, et al. 2015).

Uma das tecnologias de segurança Web que se tornou muito popular é o CAPTCHA (teste de Turing público completamente automatizado para diferenciação entre computadores e humanos) (Von Ahn, et al. 2004), que possui exatamente o propósito de diferenciar usuários humanos dos robôs que trafegam na rede normalmente (Banday, et al. 2009), os CAPTCHAs podem ser encontrados em toda internet, se tornaram muito populares nos últimos tempos e passaram por várias versões no decorrer dos anos, eles podem ser encontrados em salas de chat online, mecanismos de pesquisa (vide Google, Bing), sistemas que utilizam *logins* e preenchimentos de formulários, serviços de criações de contas de e-mail, blogs, envio de mensagens, serviços com downloads automáticos, detecção de ataques *phishing*, entre outros (Banday, et al. 2009).

No Brasil não é diferente, muitos websites e serviços utilizam o CAPTCHA como proteção de suas informações, como o site da Receita Federal brasileira que o utiliza como segurança na consulta de CPFs e CNPJ em seus serviços públicos, o site da plataforma Lattes é outro que utiliza o CAPTCHA para barrar as consultas automatizadas em sua página de pesquisa por currículos acadêmicos.

O uso do CAPTCHA realmente funciona, ele diminui ou anula completamente o acesso de robôs aos sites (Morein, 2008), por um outro lado, os usuários humanos comuns não enxergam com bons olhos terem que provar a todo momento que realmente são humanos diante de um CAPTCHA durante sua navegação na internet, esse problema é retratado com mais detalhes posteriormente no decorrer deste documento.

1.1 O Que é CAPTCHA

Existem vários tipos de CAPTCHAs, o mais utilizado é o tipo baseado em texto, que consiste em uma imagem com letras e números embaralhados ou uma palavra ou frase distorcida e com sujeiras que dificultam a leitura dos computadores, porém aos olhos humanos as letras são - ou pelo menos devem ser - facilmente identificadas, esta imagem é gerada automaticamente através de programas geradores de CAPTCHA (Banday, et al. 2009).

A palavra CAPTCHA é uma abreviação de *Completely Automated Public Turing tests to tell Computers and Humans Apart* (testes de Turing públicos e completamente automáticos para distinguir computadores de humanos), de acordo com Luis von Ahn, Manuel Blum e John Langford o fato do CAPTCHA ser público não significa exatamente ser *Open Source*, mas por uma questão de segurança o código e os dados usados para gerar o CAPTCHA devem ser disponibilizados publicamente, somente a aleatoriedade de suas letras e números deve ser privada. Nenhum programa de computador deve conseguir passar pelo seu teste, nem mesmo aquele que o gera e sabe exatamente como é seu funcionamento (Von Ahn, et al. 2004).

A letra “T” para “Turing” da palavra CAPTCHA é devido a este ser semelhante ao original teste de Turing, criado em 1950 por Alan Turing (Turing, 1950), o teste consiste em analisar o quanto um computador consegue se passar por um humano. O teste funciona da seguinte forma: Uma série de perguntas são feitas para dois jogadores, uma máquina e um humano, o juiz (humano) deve analisar o comportamento dos dois jogadores e então distinguir qual é o humano e qual é o computador. A diferença para o CAPTCHA é o fato do juiz neste caso ser o computador, vide “*Automated Turing Test to tell Computers and Humans Apart*” (Von Ahn, et al. 2004).

Jeff Yan e Ahmad Salah El Ahmad destacam em seu artigo *Usability of CAPTCHAs Or usability issues in CAPTCHA design* (2008, p.66) que a usabilidade do CAPTCHA é muito importante, de acordo com eles são três princípios de que um bom CAPTCHA deve possuir: Precisão, tempo de resposta e percepção de dificuldade. A precisão que o usuário deve possuir ao resolver um CAPTCHA deve ser o suficiente para que ele não precise resolver múltiplas vezes até obter êxito, e o tempo de solução do desafio apresentado por esse deve ser o mais rápido o possível, de maneira a evitar estresse no usuário. A percepção de dificuldade do CAPTCHA deve ser mínima, o usuário deve estar satisfeito subjetivamente de resolver tal desafio proposto.

A Figura 1 apresenta o CAPTCHA utilizado no site da Receita Federal do Brasil.

Receita Federal do Brasil
MINISTÉRIO DA FAZENDA

Perguntas Frequentes | Contato | Serviços | Da

■ **Comprovante de Situação Cadastral no CPF**

Preencha os campos abaixo com os dados solicitados.

CPF:

Data de Nascimento:

Digite os caracteres acima:

Cancelar OK

O comprovante gerado não fornece informações sobre a situação econômica, financeira ou fiscal do titular do CPF, limita-se tão somente a comprovar a situação cadastral no CPF.

Consultar Limpar

Figura 1 – CAPTCHA utilizado no site da Receita Federal brasileira.

(Fonte: <https://www.receita.fazenda.gov.br/Aplicacoes/SSL/ATCTA/CPF/ConsultaPublica.asp>, acesso em 18/06/2016.)

A utilização de um CAPTCHA neste site é extremamente importante para manter a estabilidade do serviço disponibilizado pela Receita Federal do Brasil, a informação “Comprovante de Situação Cadastral no CPF” conforme está na imagem, pode ser alvo de muitas consultas automatizadas através de robôs de captação de dados, podendo ocasionar em vários problemas como a queda do serviço, conforme explica William G. Morein em seu artigo *Using Graphic Turing Tests To Counter Automated DDoS Attacks Against Web Servers* (2008, p.66), o excesso de consultas em um servidor web, ato conhecido como ataque de negação de serviço (*DDoS Attack*), pode ocasionar na queda deste servidor, mesmo o objetivo dos robôs não sendo este.

1.2 Motivação para a quebra do CAPTCHA

No artigo desenvolvido por Luis von Ahn, Manuel Blum e John Langford (2004, p.65), o CAPTCHA é classificado como um problema de Inteligência Artificial, e para que uma pessoa maliciosa consiga passar por um CAPTCHA, seja para votar seguidas vezes em uma votação aberta na internet, cadastrar-se em milhares de contas de e-mail, ou consultar um CPF no site da Receita Federal brasileira, este desafio de IA deve ser solucionado, havendo assim um benefício nesta quebra de segurança: Um avanço na área de Inteligência Artificial ou Redes

Neurais Artificiais. Essas áreas da computação estão muito mais avançadas do que na época dos criadores do CAPTCHA, podendo não existir um avanço caso um CAPTCHA seja quebrado, mesmo assim o benefício do estudo e aquisição de conhecimento através destas tecnologias e suas técnicas pode ser proveitoso.

Uma outra vantagem que há na quebra de um CAPTCHA é a detecção e correção de falhas no sistema de segurança, que nem sempre são feitos da maneira correta, em alguns casos um simples tratamento de imagem é capaz de “limpar” toda a sujeira de inserida para dificultar a resolução do desafio da imagem, deixando os caracteres completamente visíveis, o que facilita a ação dos programas capazes de reconhecer textos em imagens - as OCRs (Optical Character Recognition). Essa é uma das falhas que podem ser encontradas em CAPTCHAs, outras falhas serão retratadas no decorrer deste documento.

Além dos motivos citados, é interessante provar com a quebra de um CAPTCHA que este já se encontra defasado em questão de segurança contra robôs de navegação automática.

1.3 Objetivos

Neste trabalho será realizado um estudo sobre os CAPTCHAs, como resolvê-los com técnicas de computação, e superar o desafio proposto pelo conceito de que o CAPTCHA é solúvel apenas para humanos e extremamente difícil para um computador.

Através do estudo de técnicas de tratamento de imagem e ferramentas para reconhecimento de caracteres e padrões, tem-se por objetivo resolver o CAPTCHA utilizado no site do currículo Lattes (demonstrado na Figura 2), assim expondo seus pontos fracos e suas falhas de segurança. Todo o processo da quebra é demonstrado neste documento.

Espera-se contribuir com um estudo relacionado as tecnologias utilizadas para a quebra do CAPTCHA neste projeto, no fim do trabalho um capítulo é dedicado para apresentar e analisar os resultados obtidos com a metodologia utilizada.



Figura 2 – CAPTCHA utilizado no site da consulta do currículo Lattes.

(Fonte: <http://buscatextual.cnpq.br/buscatextual/visualizacv.do>, acesso em 18/06/2016.)

1.4 Organização do Trabalho

O trabalho está organizado em seis capítulos: Introdução, Tipos de CAPTCHA, Processos da quebra de um CAPTCHA, Trabalhos correlatos, Implementação e Resultados. Abaixo uma relação dos capítulos:

Introdução: Apresentação do conteúdo do trabalho e de sua problemática, descrição da motivação e dos objetivos.

Tipos de CAPTCHA: Descrição dos tipos de CAPTCHA encontrados em uso na Web.

Processos da quebra de um CAPTCHA: Estudo dos processos da quebra de um CAPTCHA, apresentação das tecnologias e técnicas de tratamento de imagem, segmentação de caracteres e reconhecimento de caracteres.

Trabalhos correlatos: Análise de trabalhos correlatos feitos por pesquisadores e comparação com os métodos utilizados.

Implementação: Descrição do processo da quebra do CAPTCHA do site do currículo Lattes.

Resultados: Análise dos resultados obtidos com a metodologia utilizada e dos objetivos alcançados.

2 Tipos de CAPTCHA

São vários os tipos de CAPTCHA que podem ser encontrados em uso na internet, entre eles pode-se destacar os tipos: Baseados em texto, baseados em imagem, baseados em áudio, baseados em vídeo e baseados em quebra cabeça (Singh, et al. 2014).

Este capítulo descreve os tipos de CAPTCHA que são mais comuns em uma navegação na internet, que são os baseados em texto, em imagem, em questões, em áudio e o CAPTCHA mais recente desenvolvido pela Google, o No CAPTCHA reCAPTCHA.

2.1 Baseados em Texto

Os CAPTCHAS baseados em texto são imagens que contém caracteres distorcidos e que são fáceis de serem identificados pelos humanos, porém difíceis para os computadores devido ao seu alto nível de distorção, alguns ainda apresentam sujeiras e degradações na qualidade da imagem, o que dificulta ainda mais a leitura pelos computadores (Banday, et al. 2009). Para decodificar este tipo de CAPTCHA, o usuário humano deve digitar as letras e números que aparecem na imagem de forma correta.

A Figura 3 contém exemplos de como são os CAPTCHAs baseados em texto, com diferentes níveis de dificuldade.

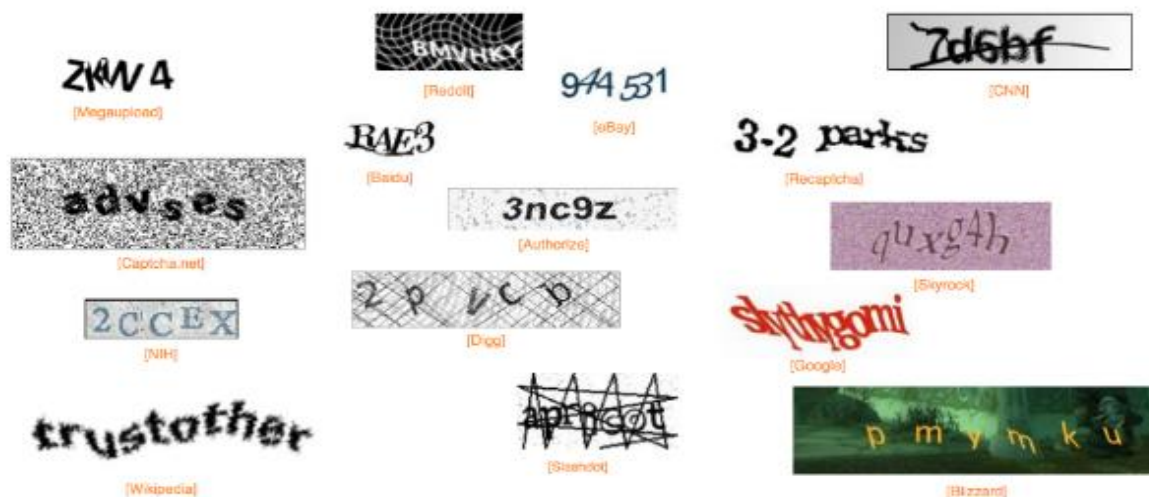


Figura 3 – Exemplos de CAPTCHAs baseados em texto.

(Fonte: <http://images.pcworld.com/images/article/2011/11/chaptchas-5232021.jpg>, acesso em 18/06/2016, imagem editada pelo próprio autor.)

A usabilidade dos CAPTCHAs deve ser menos complicada o possível, um usuário não deve ter que se esforçar muito para resolver um que encontrar em uma navegação web, levar

muito tempo resolvendo um também não é agradável (Yan, 2008). Os CAPTCHAs de texto são os mais utilizados por serem considerados rápidos de se resolver (Bursztein, 2014), basta digitar os caracteres demonstrados na imagem do CAPTCHA, porém com a evolução da computação e seus recursos, este tipo de CAPTCHA passou a ser quebrado facilmente por computadores (Makris, et al. 2014), e devido a isto o seu nível de dificuldade foi aumentando de modo a interromper os ataques automáticos, isso acarretou numa dificuldade maior para os humanos devido as distorções nas letras que ficaram difíceis de serem compreendidas, e sujeiras em excesso foram adicionadas (Banday, et al. 2009).

Na Figura 4 a palavra à esquerda do CAPTCHA é difícil de ser identificada devido as letras estarem muito juntas umas das outras (Yan, 2008).



Figura 4 – Exemplo de CAPTCHA de difícil resolução para humanos.

(Fonte: <https://codemyviews.com/blog/should-we-kill-the-captcha>, acesso em 18/06/2016.)

Alguns sites possuem variações nos CAPTCHAs apresentados ao usuário, o site de consultas de domínios brasileiros “<https://registro.br>”, possui no mínimo dezoito variações de CAPTCHAs. De acordo com uma pesquisa feita pelo próprio autor deste documento - no caso foram realizadas várias consultas ao site para obter o máximo de CAPTCHAs possível - foram encontradas dezoito variações, a partir da centésima consulta, nenhum modelo novo foi encontrado. Todas essas variações para dificultar a quebra por um possível hacker, para se obter um resultado bom de quebra desta segurança, no mínimo dezoito CAPTCHAs com características diferentes deveriam ser tratados, segmentados e ter seus caracteres conhecidos, além do fato de ser necessário descobrir o tipo de CAPTCHA que o site está usando no momento na consulta.

A Figura 5 contém quatro exemplos de variação dos CAPTCHA utilizado no site RegistroBr.



Figura 5: a, b, c, d – Exemplo de variação do CAPTCHA do RegistroBr.

(Fonte: <https://registro.br/cgi-bin/whois/?qr=&c>, acesso em 18/06/2016.)

2.2 Baseados em Imagem

Os CAPTCHAs baseados em imagem não contêm texto nenhum além de sua questão que explica o que deve ser feito para resolvê-lo, todos são basicamente imagens que representam algum objeto ou paisagem (Banday, et al. 2009). Na Figura 6 para resolver o CAPTCHA proposto deve-se clicar na imagem que contém uma flor, dentre uma grade com 9 imagens aleatórias.



Figura 6 – Exemplo de CAPTCHA baseado em imagem.

(Fonte: <https://dab1nmslvvntp.cloudfront.net/wp-content/uploads/2014/05/139961548102-Image-CAPTCHA.png>, acesso em 18/06/2016.)

A vantagem deste tipo de CAPTCHA é a sua facilidade de ser resolvido por humanos, um estudo feito por Monica Chew and J. D. Tyga (2004, p.65), revela que a taxa de acerto por usuários comuns é maior dentre os outros CAPTCHAS. A principal dificuldade em se implementar um tipo deste de CAPTCHA é que são necessárias muitas imagens para dificultar a ação dos algoritmos computacionais, caso o número de imagem seja muito limitado um computador é capaz de referenciar cada imagem através de seu código binário ou gerando uma espécie de código hash para classificá-las (Fritsch, 2010).

2.3 Baseados em questões

Os CAPTCHA baseados em questão são desafios propostos ao usuário em que não basta identificar o que está na tela e escrever no campo de texto, este tipo deve ser interpretado pelo usuário (Mahato, et al. 2015). Nas Figuras 7 e 8 o CAPTCHA apresenta um problema matemático, o usuário - ou robô - deve ser capaz de interpreta-lo para conseguir acessar os dados, duas perguntas são feitas de forma gráfica e representadas em formato de dados: “5 + 3 =?” e também “4 + 6 =?”.

O benefício deste tipo de CAPTCHA é a facilidade de quebra para os usuários humanos, uma simples questão como a da Figura 7 é simples é rápida de se resolver, poupando esforço e tempo, porém algumas questões podem ser exageradas e difíceis demais, vide Figura 8.

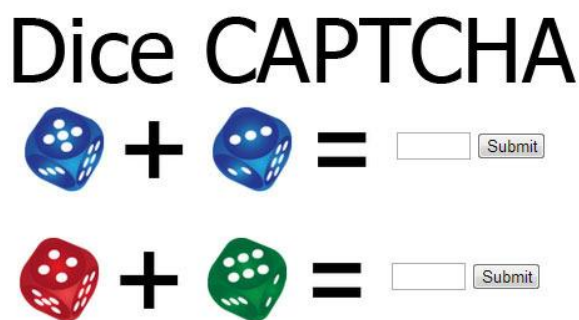


Figura 7 – Exemplo de CAPTCHA baseado em questões.

(Fonte: <https://is301atnsc.files.wordpress.com/2014/06/title.jpg>, acesso em 18/06/2016.)

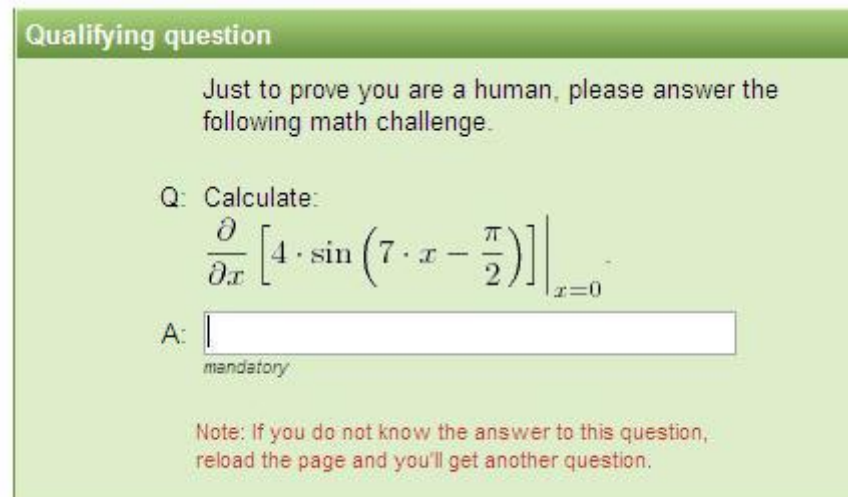


Figura 8 – Exemplo de CAPTCHA baseado em questões e difíceis para humanos.

(Fonte: <http://random.irb.hr/signup.php>, acesso em 18/06/2016.)

Como apresentado na Figura 8, um CAPTCHA deste jeito tem os seus princípios de usabilidade praticamente invertidos, difíceis para humanos e fáceis para um computador.

2.4 Baseado em áudio

Os CAPTCHAS de áudio são executados com um clique em um botão da tela, uma voz diz os caracteres do CAPTCHA para o usuário, que deve digitá-los no campo de texto correspondente, o primeiro CAPTCHA audível foi criado por Nancy Chan para as pessoas com deficiência visual (Banday, et al. 2009).

Este tipo de CAPTCHA pode ser quebrado facilmente através da comparação dos arquivos binários dos áudios, caso seja somente uma voz que grave os caracteres, para evitar este problema alguns CAPTCHAs deste modelo estão utilizando várias vozes para gravar os caracteres e estão adicionando barulhos ou chiados não lineares ao áudio, assim duas faixas de áudio são executadas, a voz dizendo os caracteres e o chiado, deste modo o binário dos arquivos nunca será igual (Banday, et al. 2009).

Muitos sites estão utilizando este tipo de segurança como uma alternativa de solução, por exemplo: Se o CAPTCHA baseado em texto que o site utiliza estiver muito difícil de ser resolvido, há uma opção que disponibiliza o mesmo CAPTCHA com os mesmos caracteres mas em forma de áudio, alguns usuários podem preferir a versão audível.

2.5 No CAPTCHA

Levando em consideração a usabilidade o No CAPTCHA reCAPTCHA é o melhor dentre os outros, criado pelo Google, ele revolucionou a forma dos CAPTCHAS, neste tipo nenhum desafio deve ser resolvido, basta um clique em um *checkbox* comum e o sistema desenvolvido pela empresa é capaz de discernir se o usuário é humano ou robô.

A Figura 9 representa o funcionamento do No CAPTCHA reCAPTCHA.

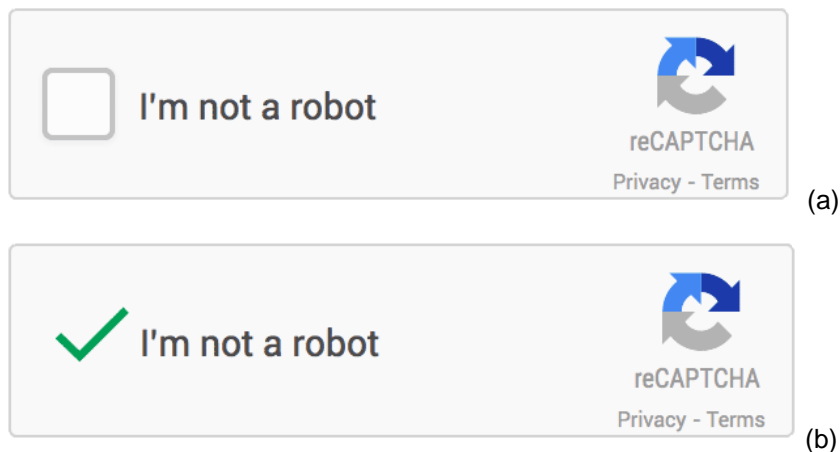


Figura 9: a – No CAPTCHA sem preenchimento, b – No CAPTCHA preenchido.

(Fonte: <https://www.google.com/recaptcha/intro/index.html>, acesso em 18/06/2016)

Para fazer a validação se o usuário é humano ou não, o sistema implementado pelo Google checa vários aspectos de navegação, como o histórico do usuário na internet, a versão do navegador utilizado, a quantidade de consultas realizadas pelo mesmo usuário naquele domínio, entre outras, o artigo escrito por Suphanee Sivakorn, Jason Polakis, e Angelos D. Keromytis (2016, p.66) possui mais informações sobre a validação.

Caso as validações não consigam classificar o usuário como um humano, o reCAPTCHA pode apresentar uma versão com imagens ou texto para a resolução, como demonstrado na Figura 10 em que o usuário deve selecionar todas as imagens que possuem uma fachada de loja.

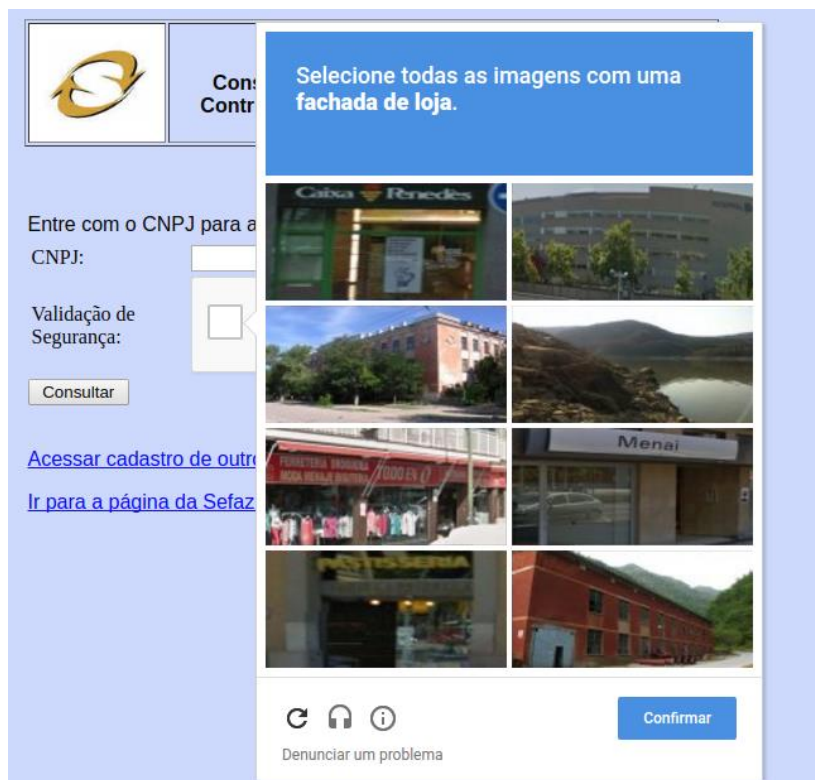


Figura 10 – Exemplo do No CAPTCHA reCAPTCHA apresentando uma imagem.

(Fonte: Fonte: <http://online.sefaz.am.gov.br/sintegra/index.asp>, acesso em 18/06/2016)

Este CAPTCHA tem se demonstrado muito eficiente quanto a segurança contra robôs automatizados e também em questões de usabilidade, as imagens utilizadas pelo Google vêm direto do Google Imagens e possuem uma enorme variação, além de serem fáceis para o usuário reconhecê-las corretamente, na Figura 11 o No CAPTCHA reCAPTCHA utiliza um número de uma casa como um CAPTCHA, imagem vinda do serviço Google Maps, novamente são extremamente fáceis para humanos mas extremamente difíceis para o computador devido ao fato de ser uma foto real.

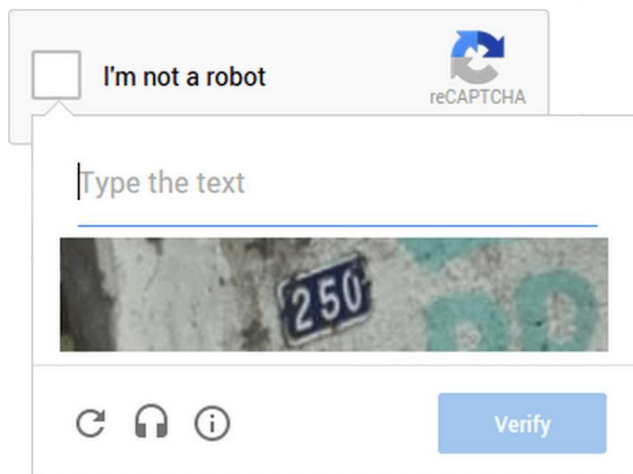


Figura 11 – Exemplo do No CAPTCHA reCAPTCHA apresentando uma foto do Google maps.

(Fonte: <http://www.jrossetto.com.br/news/google-lanca-nova-ferramenta-contra-robos-are-you-a-robot-introducing-no-captcha-recaptcha/>, acesso em 18/06/2016)

Segundo o Google os CAPTCHAs que eles utilizam além de servir como segurança para serviços web, eles ainda são úteis como digitalização de texto, anotação de imagens e também constroem conjuntos de dados para o *Machine Learning* da empresa, ou seja, as imagens selecionadas pelo usuário durante a solução do No CAPTCHA reCAPTCHA, os números das casas e os trechos de livros (Figura 12) são todos reaproveitados em algo útil posteriormente. (reCAPTCHA, 2014)



Figura 12 – Exemplo do reCAPTCHA exibindo um trecho de livro.

(Fonte: <http://www.finallybinary.com/content/images/2015/11/recaptchaBook2.PNG>, acesso em 01/11/2016)

Este CAPTCHA ainda contém a opção de áudio, caso o usuário possua alguma deficiência visual ou prefira a opção audível.

2.6 Considerações finais do capítulo

Os tipos de CAPTCHA que podem ser encontrados na internet são abundantes e suas características se diferem de várias maneiras, pode-se destacar a forma em que o CAPTCHA evoluiu com o passar dos anos, quando ele foi criado era apenas palavras deformadas, e conforme a Web evoluiu, os CAPTCHAs também avançaram, vieram os baseados em imagens e os CAPTCHAs audíveis, os quebra cabeças e os CAPTCHAs iterativos com cliques do mouse - que não foram descritos neste trabalho por serem pouco utilizados -porém o CAPTCHA mais avançado hoje é o No CAPTCHA reCAPTCHA, é o mais seguro entre os outros e o mais fácil de ser identificado por usuários humanos (Sivakorn, 2016).

As técnicas descritas nos capítulos a seguir somente possuem uso nos CAPTCHAs baseados em texto, em qualquer outro tipo elas não funcionarão corretamente.

3 Processos da quebra de um CAPTCHA baseado em texto

De acordo com Christos Makris e Christopher Town (2014, p.65), o processo da quebra de um CAPTCHA baseado em texto possui três etapas: Pré-processamento da imagem (Tratamento da imagem), segmentação dos caracteres e reconhecimento dos caracteres. Este capítulo aborda todas as três etapas, descrevendo as principais técnicas de cada uma e posteriormente no capítulo cinco, as técnicas aqui demonstradas serão utilizadas para quebrar o CAPTCHA utilizado no currículo Lattes conforme descrito no primeiro capítulo.

Para definir o sucesso do ataque a um CAPTCHA é levado em consideração a taxa de acerto em que o *script* automático consegue reconhecer todos os caracteres, dependendo do custo do ataque os *scripts* não podem ter uma taxa de acerto maior que 1 em 10.000 (0,01%), esta taxa foi obtida analisando o preço dos serviços que contratam humanos para quebra de CAPTCHA (Chellapilla, et al. 2005), porém em ambos os trabalhos de Christos Makris e Elie Bursztein, esta taxa é considerada ambiciosa, podendo ser alcançada até mesmo com ataques randômicos nos CAPTCHAs, portanto a porcentagem utilizada por eles é de 1%, considerada mais adequada (Bursztein, et al. 2011), (Makris, et al. 2014). Para a quebra do CAPTCHA do Lattes a métrica é ultrapassar 1% de taxa de acerto.

Bin B. Zhu e seus companheiros de pesquisa (2010, p.67) definem as métricas do sucesso do ataque a um CAPTCHA como: A probabilidade de sucesso da quebra do mesmo (conforme já analisado) e o tempo necessário para a quebra, este deve ser menor que 30 segundos, que é o tempo médio que um humano leva para resolvê-lo, portanto todo o processo desenvolvido para a quebra do CAPTCHA do Lattes não deve ultrapassar 30 segundos.

3.1 Tratamento da imagem com ImageMagick

O pré-processamento de uma imagem é necessário para quase todos os CAPTCHAs, o objetivo desta etapa é deixar a imagem mais limpa, remover o máximo de artefatos que atrapalham a leitura dos caracteres com uma OCR, e também dificultam o processo de segmentação (Makris, et al. 2014), e para análise desta etapa será utilizado o ImageMagick.

O ImageMagick é um software para criação, edição, composição e conversão de imagens, ele é capaz de ler e escrever imagens em mais de 200 formatos como PNG, JPEG, GIF, TIFF, PDF e SVG. Entre suas capacidades de edição de imagens se encontra a possibilidade de alteração de tamanho, rotação, distorção, aplicação de efeitos especiais, desenho de linhas e polígonos entre outras funcionalidades (ImageMagick, 1999).

O software possui suporte para várias linguagens, como JAVA, C, C++, Ruby, Python, etc. O uso do ImageMagick neste trabalho se deve ao fato de ser uma poderosa ferramenta para edição de imagens, todos os processamentos necessários para a quebra de um CAPTCHA podem ser executados através dele, além disso a interface do ImageMagick para a linguagem Ruby funciona excepcionalmente bem, e a linguagem Ruby é utilizada para todos os algoritmos deste trabalho.

As principais técnicas de processamento de imagem estão descritas a seguir.

3.1.1 Remoção de Fundo (Background removal)

A remoção do fundo da imagem é muito útil caso o CAPTCHA utilize as cores como seu principal meio de defesa contra os ataques (Makris, et al. 2014), basicamente o *Background removal* consiste em deixar a cor do plano de fundo diferente do resto dos elementos que compõem a imagem como os caracteres e também as suas irregularidades.

No CAPTCHA do Lattes as cores são um mecanismo de defesa contra o reconhecimento dos caracteres e conseqüentemente a técnica de remoção de fundo deve ser utilizada, são várias as maneiras em que um plano de fundo de uma imagem pode ser alterado, entre elas o *Threshold* que será explicado posteriormente neste capítulo.

3.1.2 Up-sampling

Esta técnica divide cada pixel da imagem em vários pixels, permitindo assim um controle maior sobre a segmentação da imagem e melhorando o efeito de outros tratamentos que serão realizados posteriormente (Makris, et al. 2014).

Basicamente o tamanho da imagem é alterado para um tamanho maior, para um melhor controle dos pixels nos tratamentos, essa técnica se torna mais importante em imagens muito pequenas, como é o caso de alguns CAPTCHAs.

3.1.3 Limiarização (Thresholding)

A limiarização ou binarização da imagem também é necessária para a maioria dos CAPTCHAs, a imagem é passada por um tratamento que a deixa somente com duas cores: Branca e preta (0 e 1 respectivamente) (Gao, et al. 2013), deste modo cada pixel ou será completamente branco ou completamente preto. O objetivo deste tratamento é destacar os caracteres das sujeiras da imagem, vários exemplos serão demonstrados no capítulo 5.

Para aplicar o *Threshold* com o ImageMagick, é necessário definir um valor inteiro ou em porcentagem para servir como o “limite” entre os pixels da imagem que devem ser pretos ou brancos, por exemplo definindo o valor como 70%, todos os pixels com a intensidade da cor branca acima de 70% terão 100% de cor branca, já os abaixo deste valor terão 0% da cor branca, ou seja será um pixel preto (Threshold, 2016).

Um exemplo de *Threshold* pode ser conferido no capítulo 5 onde a técnica é aplicada no CAPTCHA do Lattes.

3.1.4 Line removal (Remoção de linhas)

Este tratamento consiste em remover as linhas e sujeiras que não são consideradas caracteres na imagem (Makris, et al. 2014), são opcionais para a quebra do CAPTCHA, sendo que muitas vezes os tratamentos realizados anteriormente já são suficientes para limpar a imagem, ou para destacar os caracteres dos demais elementos.

Linhas mais finas que os caracteres do CAPTCHA são removidas através de tratamentos de erosão e dilatação, a erosão remove as pequenas sujeiras da imagem, porém como o tratamento é aplicado na imagem inteira do CAPTCHA, os caracteres em si sofrem com a erosão também, diminuindo “espessura” de sua fonte, a dilatação então, faz com que o a fonte volte ao seu estado anterior antes da erosão (Makris, et al. 2014), ou pelo menos chegue perto disto.

Linhas mais grossas são mais complicadas de serem removidas, como elas são consideradas possíveis caracteres, é necessário ter um cuidado maior, um algoritmo de comparação e detecção de linhas deve ser usado nestes casos (Makris, et al. 2014).

Para realizar a técnica de *Line removal* no ImageMagick utiliza-se o tratamento *Morphology*, este tratamento modifica a imagem em várias maneiras de acordo com os “vizinhos” dos pixels que cercam outros pixels da mesma cor, como este tratamento é aplicado depois do *Threshold*, os únicos pixels existentes são os brancos e os pretos, de acordo com a cor em que os caracteres estão condicionados é onde o *Morphology* é aplicado, a partir disto este tratamento é capaz de tanto como expandir ou de contrair os pixels (Morphology, 2010).

O tratamento realizado com pixels vizinhos tem como base os núcleos, que definem quais pixels são considerados vizinhos, alguns núcleos estão demonstrados na Figura 13.

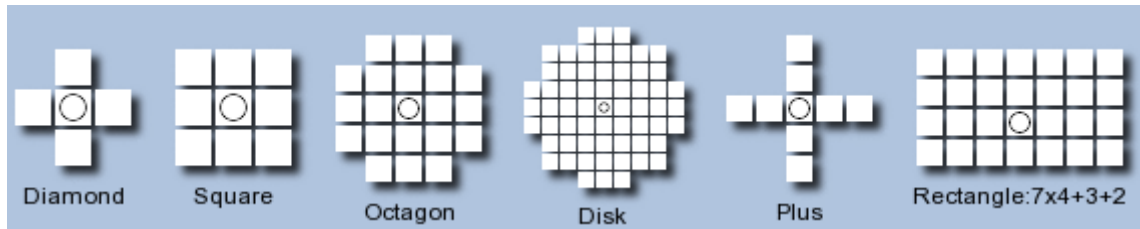


Figura 13 – Exemplo de núcleos do Morphology.

(Fonte: <http://www.imagemagick.org/Usage/morphology/>, acesso em 01/11/2016)

Utilizando o tipo de núcleo *Plus* por exemplo, os pixels tratados como vizinhos serão os que estiverem em formato do sinal de mais (+) adjacentes ao pixel central, assim os tratamentos terão efeito somente nos pixels que estiverem nesta posição, os núcleos também podem variar de tamanho conforme o necessário, podendo conter mais ou menos pixels, quanto menos pixels ele tiver, mais preciso será o tratamento, pois áreas menores serão afetadas.

O tratamento *Morphology Erode* causa uma erosão na imagem, transformando alguns pixels do primeiro plano da imagem – os caracteres no caso dos CAPTCHAs - na cor do fundo da imagem, os pixels transformados dependem do núcleo escolhido e seu tamanho. Na Figura 14 a o *Morphology Erode* foi utilizado na imagem à esquerda, o núcleo escolhido foi o *Octagon* (Figura 13), o resultado do tratamento está a direita, nota-se que alguns pixels brancos foram transformados em pretos, sendo estes todos os pixels em que os vizinhos não formam um *Octagon* completo. (Morphology, 2010).

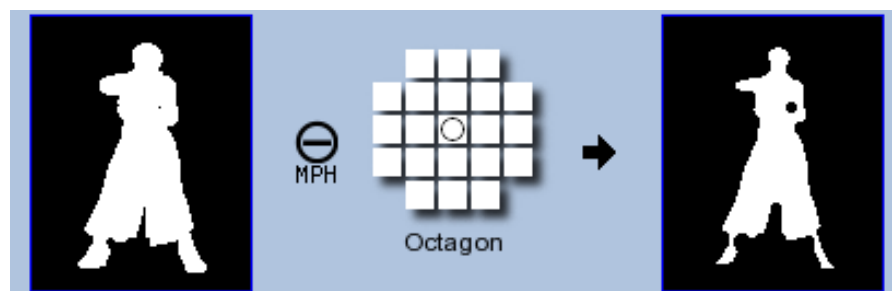


Figura 14 – Exemplo de erosão realizada com o Morphology.

(Fonte: <http://www.imagemagick.org/Usage/morphology/>, acesso em 01/11/2016)

A Figura 15 demonstra claramente uma erosão, o núcleo utilizado é o *Square* (Figura 13). Percebe-se que todos os pixels cujo os vizinhos não formam um *Square* completo são transformados em 0.

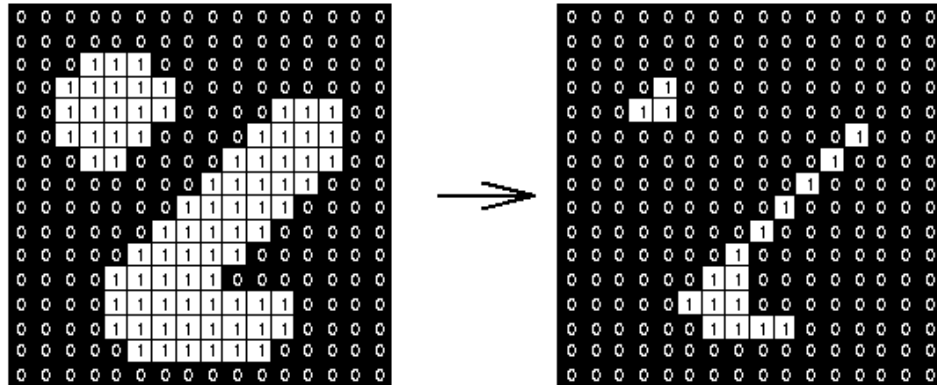


Figura 15 – Exemplo do tratamento de erosão.

(Fonte: <https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm>)

O *Morphology Dilate* possui o efeito contrário do *Morphology Erode*, que é a expansão dos pixels, todos estes que não se comparam ao núcleo escolhido e seu tamanho são expandidos com mais pixels brancos ao seu redor, vide a Figura 16.

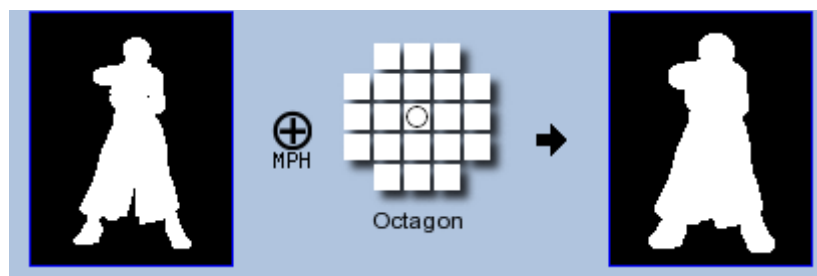


Figura 16 – Exemplo de dilatação realizada com o Morphology.

(Fonte: <http://www.imagemagick.org/Usage/morphology/>, acesso em 01/11/2016)

A Figura 17 demonstra mais claramente o processo inverso a erosão.

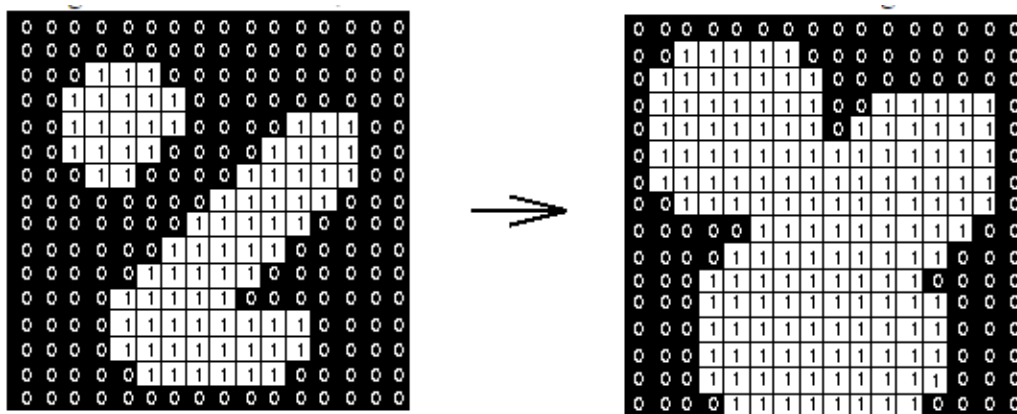


Figura 17 – Exemplo do tratamento de dilatação.

(Fonte: <https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm>)

3.1.5 Thinning (Desbaste)

O processo de *Thinning* é utilizado em alguns tratamentos para a segmentação dos caracteres, é muito comum os CAPTCHAs conterem algumas letras que se sobrepõem para que justamente dificulte a segmentação, o *Thinning* reduz a espessura do texto dos CAPTCHAs, deixando somente o “esqueleto” das mesmas (Makris, et al. 2014).

O *Morphology Thinning* é o tratamento utilizado no ImageMagick para a transformar a imagem em um esqueleto, a aplicação desta técnica é importante para prover uma descrição de imagens muito complexas (Morphology, 2010), a Figura 18 demonstra a aplicação de um *Thinning* na imagem à esquerda, o resultado está a direita:



Figura 18 – Exemplo do tratamento Thinning realizado com o Morphology.

(Fonte: <http://www.imagemagick.org/Usage/morphology/>, acesso em 01/11/2016)

3.1.6 Algoritmo de Gibbs de-noising

O algoritmo de Gibbs de-noising é uma técnica muito eficiente para remoção de itens não desejados em imagens (sujeiras), seu uso se mostrou mais eficaz em CAPTCHAs do que técnica de erosão descrita neste capítulo (Bursztein, et al. 2011).

Este algoritmo é iterativo, ou seja varre a imagem pixel por pixel em um loop e computa a energia de cada pixel baseado nos pixels ao redor, isso é feito mais de uma vez (a quantidade é definida no algoritmo), transformando um pixel do primeiro plano da imagem na cor de fundo quando a energia deste pixel for menor que um limite definido como “energia de Gibbs”, a energia de um pixel é computada com base no valor da cor em escala de cinza de seus 8 pixels vizinhos e dividida por 8, se este valor não ultrapassa o limite de energia definido, a cor daquele pixel se transforma na cor do fundo da imagem (Bursztein, et al. 2011).

Com várias iterações deste algoritmo as linhas e sujeiras que estão contidas entre os caracteres vão sumindo, é um método que funciona principalmente em linhas mais finas, conforme pode ser observado na Figura 19 (pg. 39).

3.2 Segmentação dos caracteres

Durante a segmentação, os caracteres são diferenciados do plano de fundo e são separados em blocos de apenas um caractere cada, para que esta etapa da quebra de um CAPTCHA tenha uma maior chance de sucesso é preciso que a etapa anterior (Tratamento da imagem), tenha limpado a imagem da maioria de suas sujeiras (Makris, et al. 2014).

O maior problema que impede uma perfeita segmentação de caracteres são os que estão grudados uns aos outros, é muito comum em um CAPTCHA que alguns caracteres se sobreponham aos demais, esta é a principal técnica anti-segmentação utilizadas pelos idealizadores de CAPTCHA, além é claro das linhas, formas ou outros tipos de irregularidades que se confundem com as letras (Makris, et al. 2014).

Alguns algoritmos usados em segmentação, estão descritos nos subcapítulos a seguir.

3.2.1 Flood-filling segmentation

Com a binarização da imagem realizada na etapa anterior (Tratamento da imagem), o algoritmo *Flood-filling segmentation* compara todos os pixels da imagem com os seus pixels vizinhos, aqueles que possuírem a cor que represente o caractere – geralmente branco ou preto - e seu pixel vizinho também for da mesma cor, são agrupados e classificado como um possível caractere do CAPTCHA (Makris, et al. 2014).

Geralmente estes grupos de pixels agrupados devem ultrapassar uma certa quantidade para serem considerados caracteres e não sujeiras, caso sejam poucos eles são descartados. Para este algoritmo funcionar é necessário um excelente resultado na etapa de tratamento da imagem, caso os caracteres estejam com muitas falhas ou existem muitos traços de sujeiras na imagem, os grupos de pixels poderão dividir um caractere ao meio ou então acabar juntando dois ou mais caracteres em um grupo só.

3.2.2 Histogram segmentation

Neste método de segmentação o número de pixels pretos - considerados caracteres devido a binarização do CAPTCHA - em cada coluna de pixel da imagem é contado, a coluna com menos pixels é classificada como um possível ponto de corte, acreditando assim que este ponto seja a intersecção entre um caractere e outro (Makris, et al. 2014), porém é preciso tomar cuidado com esta técnica, devido ao fato que nem sempre este ponto é uma intersecção de caracteres, o que levaria a uma segmentação equivocada do CAPTCHA, para diminuir este erro

é necessário saber o número de caracteres que o CAPTCHA contém, e levar em conta o número de pixels das colunas vizinhas também. Muitos CAPTCHAS possuem um número aleatório de caracteres, o que dificultaria o sucesso desta técnica.

3.2.3 Pattern matching

O algoritmo procura por padrões de letras e números nos pixels definidos como caracteres (0 ou 1), por exemplo o pingo no “i” e no “j”, a forma curva do “s” e do “5”, a intersecção em cruz dos caracteres “f”, “x”, entre outros (Makris, et al. 2014).

Os pixels identificados com os padrões são removidos da imagem e classificados de acordo com sua forma, este algoritmo é iterativo, percorre toda a imagem em busca dos padrões, finalizando após todos serem identificados (Makris, et al. 2014).

Ahmad El Ahmad, Jeff Yan e Mohamad Tayara (2011, p.68) utilizaram o *Pattern matching* como método de segmentação de um CAPTCHA utilizado pelo Google em 2011, os padrões de letras e números aplicados no algoritmo desenvolvido por eles foram: Forma de ponto (“i”, “j”), forma de loop (“a”, “b”, “o”), forma de cruz (“t”, “f”) e forma de S (“s”). O artigo desenvolvido por eles contém mais informações sobre como cada algoritmo de identificação dos padrões foi desenvolvido.

3.3 Reconhecimento dos caracteres

O resultado da fase de segmentação são os caracteres extraídos da imagem para o seu reconhecimento (Makris, et al. 2014), o sucesso desta etapa irá depender do desempenho das etapas anteriores, quanto mais limpa a imagem e mais perfeita a segmentação melhor será o reconhecimento.

Para reconhecer um caractere muitas técnicas e ferramentas podem ser utilizadas, desde métodos mais simples como a comparação por matriz dos pixels até redes neurais artificiais treinadas para o reconhecimento (Makris, et al. 2014).

3.3.1 FANN

A FANN (Fast Artificial Neural Network) é uma biblioteca de rede neural artificial de código aberto escrita em C que implementa redes neurais de multicamadas, possui suporte para mais de 20 linguagens de programação como o C#, o JAVA, C++, Ruby, Python, etc (FANN, 2016).

Redes Neurais Artificiais não são inteligentes, mas são muito eficientes em reconhecimento de padrões e também possuem alta capacidade de treinamento, são ótimas para generalizações em conjuntos de dados de treinamentos (Steffen, 2003).

Utilizando vários CAPTCHAs baseados em texto como um conjunto de treinamento para uma RNA (Rede Neural Artificial) por exemplo, com o objetivo de classificar seus caracteres, após a rede ser treinada com este conjunto específico ela é capaz de identificar padrões em CAPTCHAs que não foram utilizados no treinamento, mesmo estes sendo ligeiramente diferentes.

São essas características da Rede Neural Artificial que serão utilizadas neste trabalho, a sua capacidade para classificação de padrões, o objetivo é treiná-la com os caracteres extraídos dos CAPTCHAs após a fase de tratamento de imagem, e segmentação dos caracteres, sendo assim cada caractere extraído servirá como treinamento para a RNA, na qual sua função é identificar os padrões de cada letra ou número do CAPTCHA e ser capaz de identificar um CAPTCHA que não tenha sido utilizado no treinamento posteriormente.

A aplicação da FANN consiste em duas fases: Treinamento e execução. Na fase de treinamento a rede é treinada com vários tipos de entradas, com o objetivo de identificar os padrões destes dados de entrada, podendo assim classificá-los. Na fase de execução a FANN retorna uma saída específica, de acordo com os dados enviados para a execução, esta saída é baseada na fase de treinamento onde os padrões foram identificados (Steffen, 2003).

O treinamento utilizado pela FANN é o *Backpropagation*, que funciona da seguinte maneira: Após a propagação dos dados de entrada através da rede, o erro calculado é propagado de volta pela rede para que os pesos dos neurônios possam se ajustar e o erro possa ser minimizado, esse processo se repete até que o erro quadrático médio (*Mean Square Error*) de toda a rede atinja um certo limite estabelecido previamente, o artigo feito por Nissen Steffen contém mais informações sobre o *Backpropagation* e o funcionamento da FANN (Steffen, 2003).

3.3.2 Tesseract

Tesseract é um motor para reconhecimento ótico de caracteres criado pela Hewlett-Packard Laboratories Bristol (HP) entre 1985 e 1994, em 2005 ele se tornou um software livre, e desde de 2006 vem sendo mantida pela Google (Tesseract, 2016).

Reconhecimento ótico de caracteres é a conversão de documentos físicos escaneados em documentos digitais que possam ser editados por um computador, estes documentos podem

ser escritos à mão ou não, o papel da OCR (Reconhecimento ótico de caracteres) é identificar as palavras contidas nestes documentos escaneados automaticamente, sendo útil para a digitalização de livros e documentos importantes (Patel, et al. 2012).

Existem várias OCRs mas o Tesseract se destaca entre as de código aberto, escrito em C++ o Tesseract pode ser integrado em aplicações de outras linguagens através da *Dynamic Link Library* (DLL), para o seu uso na quebra do CAPTCHA do Lattes o Tesseract v3.03 foi instalado no sistema operacional Ubuntu 14.04 e foi utilizado uma biblioteca da linguagem Ruby “RTesseract” para que o código escrito em Ruby versão 2.3 possa interagir com o Tesseract instalado.

3.3.3 GOOCR

GOOCR (*Optical Character Recognition* sob a licença pública GNU) é uma ferramenta para reconhecimento ótico de caracteres assim como o Tesseract, é um software de código aberto desenvolvido sob a licença GNU, a GOOCR é capaz de ler imagens em vários formatos como PNM e PGM, sua utilização é simples e rápida não necessitando nenhum treinamento previamente (Dhiman, Singh, 2013).

A GOOCR foi instalada na mesma máquina e no mesmo Sistema Operacional que o Tesseract para realizar o reconhecimento do CAPTCHA do Lattes, o Ubuntu 14.04 e a versão utilizada foi a GOOCR v0.50. Para trabalhar com esta ferramenta usa-se o terminal do Linux, a imagem é definida como parâmetro de entrada (geralmente em formatos comuns como JPEG), a GOOCR faz a conversão da imagem em PNM e então após o processamento do software, a saída dos caracteres da imagem são em formato de texto (Dhiman, Singh, 2013).

3.4 Considerações finais do capítulo

Através de todas as técnicas e tecnologias descritas neste capítulo é que os caracteres de um CAPTCHA baseado em texto podem ser reconhecidos, ainda existem outras técnicas não explicadas mas que podem ser encontradas em trabalhos correlatos. Dentre as técnicas de tratamento de imagem destacam-se as utilizadas para a quebra do CAPTCHA do Lattes: *Thresholding*, *Morphology Dilate* e o algoritmo Gibbs de-noising; a técnica de segmentação dos caracteres utilizada foi o *Histogram Segmentation* e todas as ferramentas de reconhecimento foram testadas com os CAPTCHAs tratados e segmentados. O capítulo 5 contém todas as etapas da quebra do CAPTCHA do Lattes.

4 Trabalhos correlatos

Existem vários trabalhos que avaliam os CAPTCHAs contra ataques e propõem novos CAPTCHAs mais eficientes, ou testam certas ferramentas que podem ser úteis para se quebrar um CAPTCHA. O trabalho descrito por Christos Makris e Christopher Town (2014, p.65) denominado “*Character Segmentation for Automatic CAPTCHA Solving*” analisou a segurança de seis CAPTCHAs contra testes de quebra e os resultados obtidos foram excelentes.

Os CAPTCHAs analisados eram utilizados no Hotmail, Wikipedia, Slashdot, BotDetect's Wavy chess e o reCAPTCHA (2014). Depois dos ataques feitos, uma biblioteca de segmentação genérica de CAPTCHA foi implementada com o objetivo de testar qualquer CAPTCHA baseado em texto.

Christos Makris e Christopher Town apresentaram algumas técnicas de tratamento de imagem e segmentação de caracteres utilizados para a quebra do CAPTCHA do Lattes conforme descrito no capítulo 5, técnicas que também foram incluídas no algoritmo de quebra criado por eles. Os tratamentos utilizados por eles na fase de pré-processamento das imagens foram: *Up-sampling*, *Blurring*, *Thresholding* e *Thinning* e alguns algoritmos específicos para os CAPTCHAs do Slashdot e BotDetect's Wavy Chess.

Já as técnicas de segmentação de caracteres utilizadas por Christos Makris e Christopher Town foram: *Flood-filling*, *Histogram segmentation* e um algoritmo específico para o caso do reCAPTCHA. Na etapa de Reconhecimento de caracteres os autores não utilizaram uma Rede Neural Artificial devido ao fato de ser necessário grandes quantidades de amostras para treinamento da rede, a ferramenta utilizada foi um classificador SVM (Support Vector Machine), e o framework de classificação de imagens *Zernike Moments*.

Os resultados dos ataques foram ótimos levando em consideração que apenas 1% de quebra é necessário para um CAPTCHA ser considerado ineficiente contra um algoritmo, os resultados foram: Wikipédia 67%, Slashdot 26%, BotDetect 81%, Hotmail 50%, reCAPTCHA 0.11%.

O reCAPTCHA foi completamente eficiente contra as técnicas de quebra de CAPTCHA utilizadas pelos autores, provando ser um dos CAPTCHAs mais seguros atualmente.

O motivo que levou a este trabalho ser analisado foram as quantidades de técnicas apresentadas, tanto de tratamento e segmentação, que provaram ser eficientes para ataques aos CAPTCHAs baseados em texto atuais.

Elie Bursztein, Matthieu Martin e John C. Mitchell desenvolveram o trabalho *Text-based CAPTCHA Strengths and Weaknesses* (2011, p.67), que consiste em um estudo dos CAPTCHAs baseados em texto com várias distorções nos caracteres e com técnicas de anti-

segmentação. O número de CAPTCHAs avaliados neste trabalho foi de 15, na qual 13 foram classificados como inseguros.

Os passos seguidos pelos autores para a quebra dos CAPTCHAs foram diferentes dos passos abordados neste trabalho para a quebra do CAPTCHA do Lattes, os passos foram: Pré-processamento da imagem, segmentação dos caracteres, pós-segmentação dos caracteres, reconhecimento dos caracteres e pós-processamento. Segundo os autores a fase de pós-segmentação pode limpar a saída da segmentação normalizando o tamanho das imagens que são geradas após este processo. A fase de pós-processamento pode aumentar a chance de reconhecimento dos caracteres aplicando uma verificação de ortografia nos CAPTCHAs que utilizam palavras existentes nos dicionários atuais, como o CAPTCHA do Slashdot.

Os CAPTCHAs analisados foram dos sites: Authorize, Baidu, Blizzard, Captcha.net, CNN, Digg, eBay, Google, Megaupload, NIH, reCAPTCHA, Reddit, Skyrock, Slashdot, e Wikipedia. A Figura 3 do capítulo 2 (p.19) contém todos estes 15 CAPTCHAs legendados especificamente para cada site.

As técnicas de anti-reconhecimento dos CAPTCHAs são: Múltiplas fontes, diferentes Charsets, vários tamanhos da fonte, distorção, borrões irregulares nos caracteres, rotação dos caracteres, ondulação dos caracteres. Já as técnicas de anti-segmentação são as seguintes: Plano de fundo complexo da imagem, linhas irregulares que sobrepõem os caracteres e colapso de caracteres (sobreposição).

Bursztein, Matthieu Martin e John C. Mitchell (2011, p.67) descrevem em vários capítulos quais das técnicas de anti-segmentação e anti-reconhecimento citadas são eficientes e quais não são e devem ser evitadas por CAPTCHAs ativos na web.

O algoritmo de Gibbs de-noising foi apresentado pelos autores como uma ótima opção para remoção de linhas indesejadas nos CAPTCHAs, o algoritmo foi testado no CAPTCHA do site Captcha.net e o resultado é demonstrado na Figura 19.

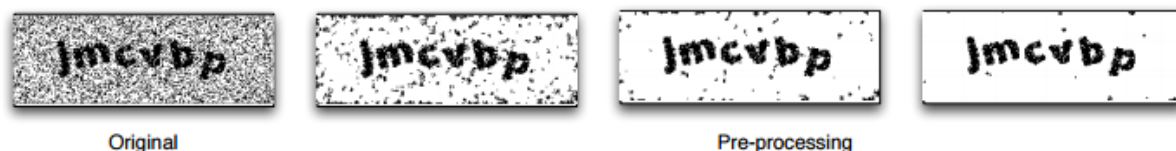


Figura 19 – Exemplo da eficiência do algoritmo de Gibbs de-noising.

(Fonte: Bursztein, et al. 2011)

O resultado alcançado pelo algoritmo Gibbs de-noising foi o motivo dele ter sido incluído neste trabalho para a quebra do CAPTCHA do site Lattes.

O percentual de acerto nos CAPTCHAs analisados foram de: 1% até 10% (Baidu, Skyrock), 10% até 24% (CNN, Digg), 25% até 49% (eBay, Reddit, Slashdot, Wikipedia), 50% ou maior para (Authorize, Blizzard, Captcha.net, Megaupload, NIH). Os CAPTCHAs do Google e o reCAPTCHA foram completamente eficientes em resistir aos ataques.

4.1 Considerações finais do capítulo

Os CAPTCHAs são de extrema importância para as aplicações web como é notável através do número de pesquisas em que eles são testados, postos a prova por várias técnicas e quebrados.

A computação avançou a um nível em que as ferramentas de segurança se tornam obsoletas muito prematuramente, inúmeros sites na web utilizam CAPTCHAs já quebrados pelos pesquisadores ou por hackers, é importante que o estudo dos CAPTCHAs não cesse e que este mecanismo de segurança se torne cada vez mais sofisticado e seguro.

5 Implementação

A seguir serão demonstradas as etapas executadas para a quebra do CAPTCHA do Lattes, cada CAPTCHA deve ser tratado de maneira única, nem sempre as técnicas usadas neste pode funcionar em outros, mesmo sendo de certa forma parecidos. Cada passo apresentado neste trabalho foi executado com base em pesquisas de trabalhos correlatos e todas as técnicas estão devidamente comentadas no capítulo três.

5.1 Tratamento da imagem

A fase de Tratamento pode ser considerada como a base de todo o processo até o reconhecimento do CAPTCHA, se tudo não funcionar como o esperado nesta fase, as outras se tornarão muito complicadas, um tratamento mal feito pode levar a uma péssima segmentação dos caracteres e conseqüentemente, a um reconhecimento ruim.

A Figura 20 representa a importância dos passos de quebra de um CAPTCHA baseado em texto.

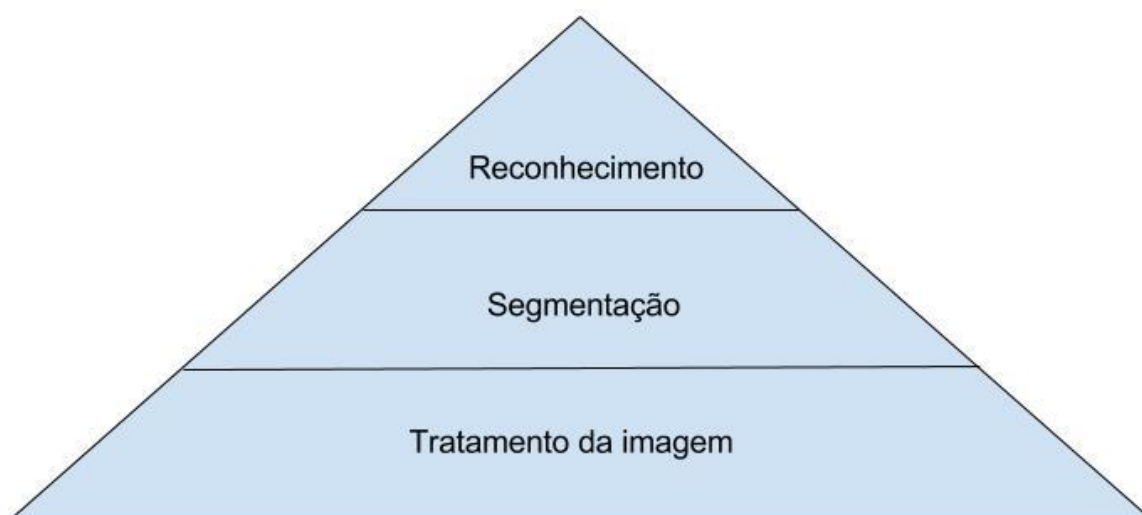


Figura 20 – Pirâmide representando a importância das etapas da quebra de um CAPTCHA baseado em texto.

(Fonte: Próprio autor)

São inúmeras as possibilidades de tratamentos que podem ser utilizados, as principais e mais utilizadas foram descritas no capítulo 3. Devido a este alto número de técnicas conhecidas essa fase da quebra de um CAPTCHA acaba se tornando a mais trabalhosa, muitas vezes um tratamento não se sai tão bem como o esperado e é necessário testá-lo novamente com pequenos

ajustes e modificações que podem fazer toda a diferença, basicamente nesta etapa cada técnica é testada empiricamente até o resultado ser satisfatório para a próxima etapa da quebra.

O CAPTCHA utilizado no site do currículo Lattes possui um grau de dificuldade alto devido as suas características de segurança, entre as quais podemos destacar os diferentes planos de fundo que a imagem tem e os pequenos traços e pontos na mesma cor dos caracteres.

A Figura 21 demonstra todos os planos de fundo diferentes do CAPTCHA do Lattes, são 7 no total.

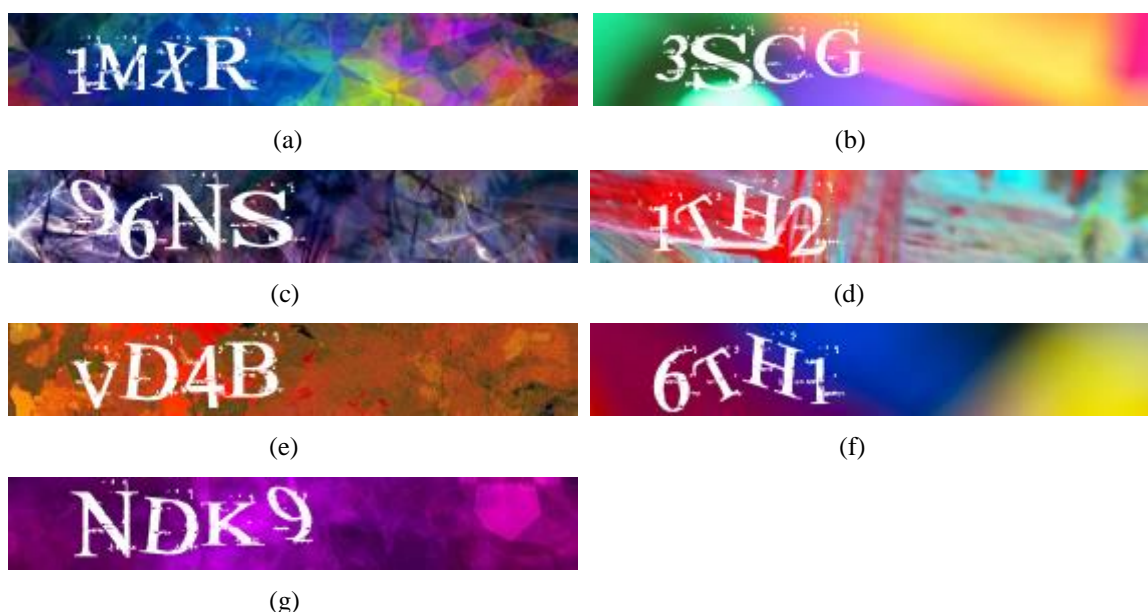


Figura 21: a, b, c, d, e, f, g. – CAPTCHAs do site do currículo Lattes.

(Fonte: <http://buscatextual.cnpq.br/buscatextual/visualizacv.do>, acesso em 18/06/2016.)

Com tantas variações de cores no fundo da imagem a sua remoção se torna difícil, porém o CAPTCHA tem um erro grave, a fonte sempre é da cor branca, assim uma técnica de limiarização (*Thresholding*), é capaz de destacar facilmente as letras do resto da imagem.

Observando o CAPTCHA é possível notar que os caracteres estão sempre à esquerda, ocupando quase sempre a mesma posição na imagem, então é interessante cortar o espaço vazio da imagem para evitar possíveis sujeiras inesperadas.

O tamanho da imagem é de 284 pixels de comprimento e 46 pixels de altura, após o tratamento utilizado para o corte o seu tamanho passa a ser 150 pixels de comprimento e os mesmos 46 pixels de altura. O software ImageMagick foi utilizado para cortar a imagem, e também para a limiarização.

O valor limite do Threshold utilizado foi de 87% - descoberto empiricamente - um valor alto que aproveita do fato de todos os caracteres já serem brancos, assim todos os pixels que

tenha a intensidade da cor branca maior que 87% serão 100% brancos e os que estejam abaixo deste limite serão completamente pretos.

A Figura 22 contém os CAPTCHAs após o uso da limiarização e do corte do espaço não ocupado por caracteres:

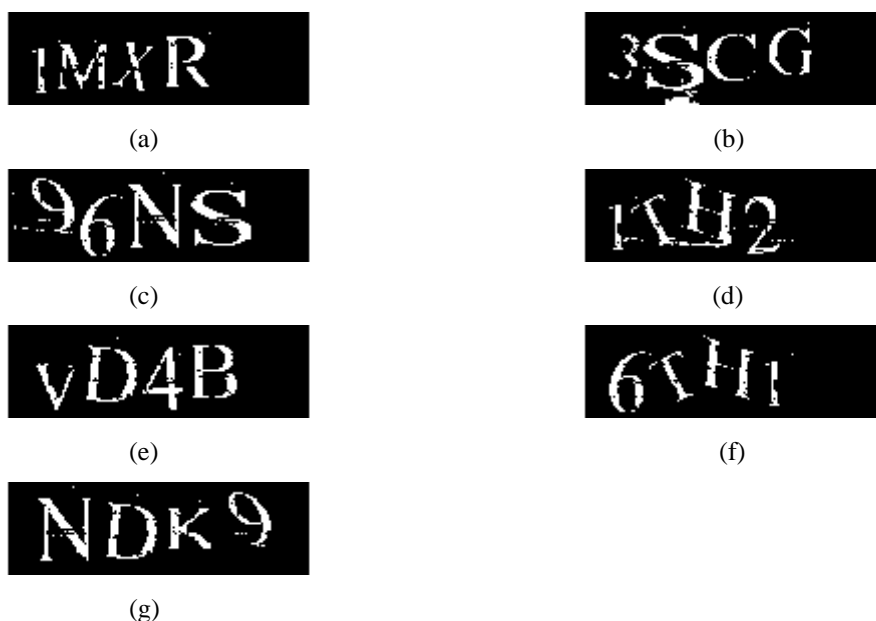


Figura 22: a, b, c, d, e, f, g. – CAPTCHAs do site do currículo Lattes, após o tratamento de Thresholding e do corte da imagem.

(Fonte: Próprio autor.)

O resultado após os tratamentos executados é satisfatório, todo o fundo da imagem foi removido e os caracteres estão destacados, alguns traços indesejados permaneceram principalmente nas Figuras 22.b, 22.c e 22.d, estas imagens possuem ruídos da mesma cor dos caracteres assim acabam sendo destacadas também com o uso do *Thresholding*, o lado positivo disto é que eles estão sempre na mesma posição então é possível colorir os pixels destas posições de forma a minimizar estas sujeiras, mas antes a técnica deve ser aplicada *Morphology Dilate* para “engrossar” os caracteres e preencher alguns espaços que acabaram sendo afetados pelo *Thresholding*, nesta técnica será utilizado o *Morphology Dilate* do ImageMagick.

A figura 23 demonstra as imagens após a dilatação.

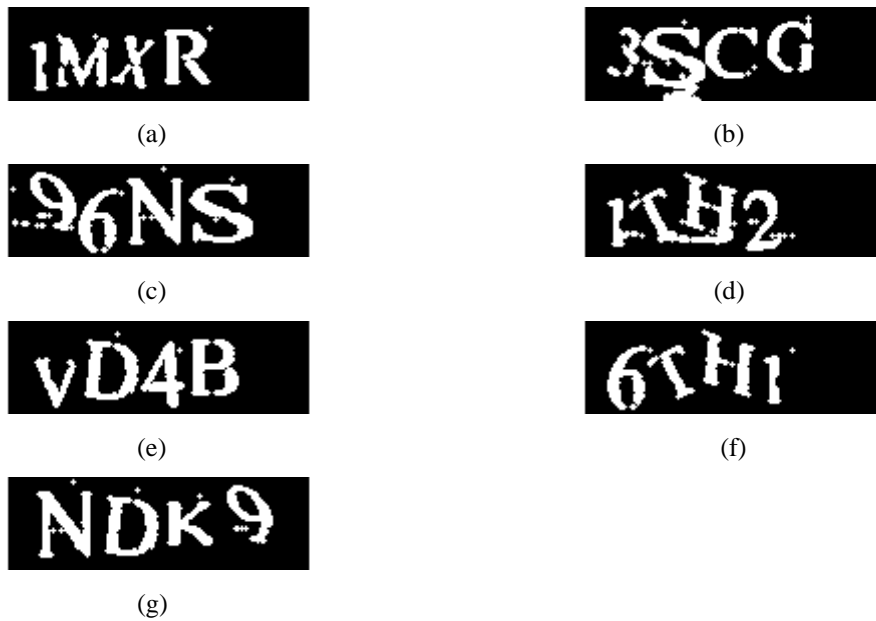


Figura 23: a, b, c, d, e, f, g. – CAPTCHAs do site do currículo Lattes, após o tratamento de dilatação.

(Fonte: Próprio autor.)

Com a dilatação os caracteres ficaram mais “preenchidos”, alguns buracos nas letras e números não existem mais.

Posteriormente os CAPTCHAs que possuem os ruídos mais explícitos, conforme pode ser visto nas figuras: 23.b, 23.c e 23.d, devem ser tratados para terem os seus ruídos removidos. Para isto a cor dos pixels que contém sujeira será transformada na cor do fundo da imagem, como as sujeiras aparecem sempre na mesma posição, esta técnica pode ser realizada sem problemas, o único problema é que alguns caracteres são grandes e assim podem ocupar o mesmo pixel que o ruído, deste modo parte de um caractere será removido também, mas isso será resolvido em um outro método explicado posteriormente.

Para explicar o algoritmo que realizará a transformação dos pixels com ruídos, a Figura 24 contém 3 CAPTCHAs com o mesmo plano de fundo, o traço de ruído pode ser percebido logo abaixo do segundo caractere de cada CAPTCHA, este é o mesmo ruído contido na Figura 23.d.

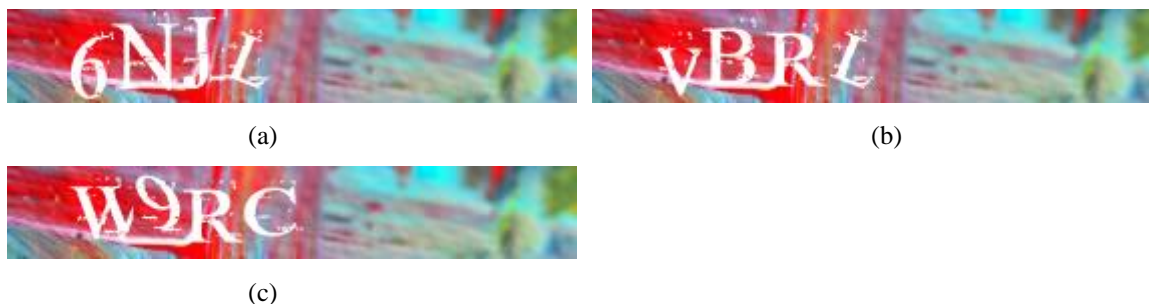


Figura 24: a, b, c – CAPTCHAs do site do currículo Lattes com o mesmo plano de fundo.

(Fonte: <http://buscatextual.cnpq.br/buscatextual/visualizacv.do>, acesso em 18/06/2016.)

O traço branco que atravessa os caracteres está contido em todos os CAPTCHAs com este plano de fundo da Figura 24. Na Figura 24.a o traço atravessa os caracteres “6”, “N” e “j”, na 24.b atravessa o “V”, “B” e o “R”, na 24.c ele só interfere no “W” e no “R”, o “9” está acima e não é alcançado por ele.

Para a remoção do traço branco, é necessário colorir os pixels de sua posição com a cor preta (mesma cor do fundo da imagem). Utilizando um algoritmo desenvolvido em Ruby em conjunto com o software ImageMagick, os pixels na posição 22 de comprimento até a 70, e nas coordenadas de altura 34, 36 e 38 são transformados na cor preta do fundo da imagem (Figura 25), assim de certa forma um pouco do ruído que existia ali antes foi removido, assim como parte de alguns caracteres.

```
(22..70).each do |n|
  img = img.color_point(n, 34, 'black')
  img = img.color_point(n, 36, 'black')
  img = img.color_point(n, 38, 'black')
end
```

Figura 25 – Parte do algoritmo para colorização dos pixels.

(Fonte: Próprio autor.)

A Figura 26 demonstra os CAPTCHAs após a aplicação do algoritmo de remoção do traço branco nas Figuras 24.a, 24.b e 24.c. Percebe-se que na posição em que o traço branco se encontrava, agora contém alguns tracejados com a cor preta, o traço branco não foi removido completamente porque o algoritmo de Gibbs de-noising removerá todo o seu restante posteriormente.

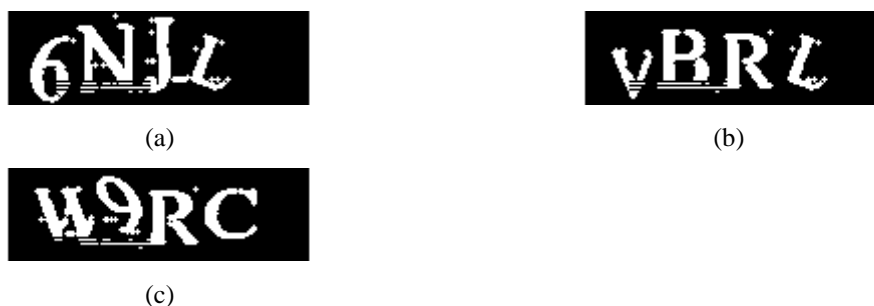


Figura 26: a, b, c– CAPTCHAs do site do currículo Lattes, após o tratamento de remoção do traço branco.

(Fonte: Próprio autor.)

Adaptando o algoritmo de remoção do traço branco para as coordenadas dos pixels com ruídos das Figuras 23.b, 23.c e 23.d, obtém-se o resultado (Figura 27). Nota-se o quanto os ruídos contidos abaixo do caractere “S” na Figura 27.a diminuiu e também abaixo dos caracteres “T” e “H” na Figura 27.c. Para aplicar este algoritmo nos CAPTCHAs com diferentes planos de fundo, foi implementado uma solução que compara as cores dos pixels de cada CAPTCHA, assim para cada plano de fundo o algoritmo é capaz de selecionar as coordenadas corretas das sujeiras.

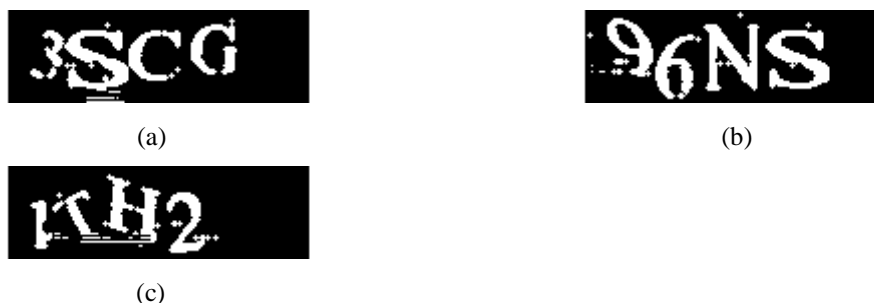


Figura 27: a, b, c – CAPTCHAs do site do currículo Lattes, após o tratamento de remoção de ruídos.

(Fonte: Próprio autor.)

O algoritmo usado é basicamente o mesmo da Figura 25, somente as coordenadas dos pixels que alteram.

O algoritmo de Gibbs de-noising (Geman, et al. 1984) é utilizado para remover o restante da sujeira, para isto a imagem deve ser transformada em uma matriz, as partes brancas da imagem são representados pelo número 1, indicando ser um possível caractere, o restante preto é representado pelo número 0 (Figura 28). Após isto o algoritmo de Gibbs de-noising é aplicado na matriz.

```

def self.image_to_matrix(img)
  pixels = []
  img.each_pixel do |pixel, c, r|
    pixels.push(pixel)
  end

  arr = Array.new(IMG_WIDTH) { |a| [] }

  pixels.each_with_index do |p, idx|
    i = idx % (IMG_WIDTH)
    if p.red < 10000
      arr[i] << 0
    else
      arr[i] << 1
    end
  end

  Matrix[*arr.transpose]
end

```

Figura 28 – Parte do algoritmo para transformar a imagem em matriz.

(Fonte: Próprio autor.)

O algoritmo Gibbs de-noising foi implementado com a linguagem ruby com o propósito de limpar as sujeiras ainda restantes dos CAPTCHAs, ele funciona conforme explicado no capítulo 3, porém como a imagem passou pelo tratamento Threshold os pixels foram binarizados, passaram a ser apenas 0 ou 1 (preto ou branco). Então ao calcular a energia de Gibbs de um pixel com a soma dos valores dos pixels vizinhos, esta energia fica armazenado no pixel para a próxima iteração do algoritmo e é somado com um novo valor de energia de Gibbs, dessa vez maior devido a iteração passada, estes passos se repetem por 10 iterações.

O limite de energia de Gibbs utilizado foi de 5000000, portanto no fim de todas as iterações, os pixels que tiverem o seu valor menor que 5000000 passam a ter a cor do plano de fundo da imagem (representado pelo número 0), dessa forma algumas sujeiras foram removidas das imagens, o valor de 5000000 foi escolhido empiricamente, após testes com valores maiores o algoritmo estava removendo partes dos caracteres e prejudicando a segmentação dos mesmos.

A Figura 29 demonstra as imagens após o algoritmo de Gibbs de-noising.

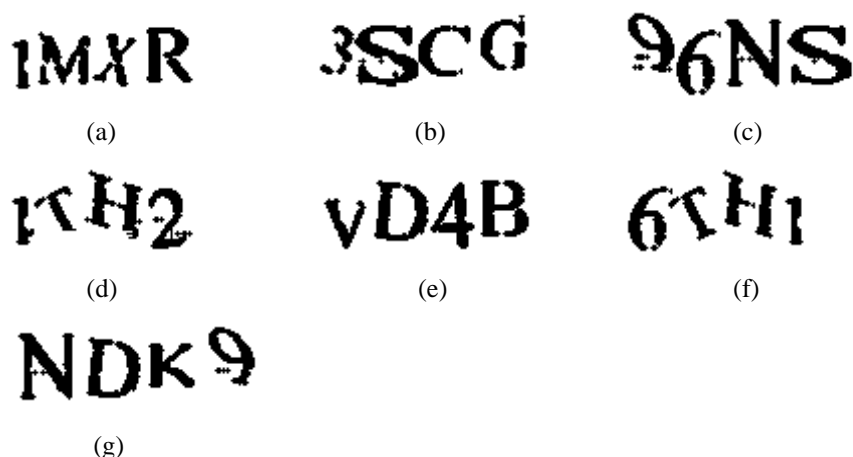


Figura 29: a, b, c, d, e, f, g – CAPTCHAs do site do currículo Lattes, após o tratamento de Gibbs de-noising.

(Fonte: Próprio autor.)

A matriz de saída do algoritmo de Gibbs de-noising é transformada em imagem novamente com o ImageMagick, somente a ordem das cores de fundo que foram alteradas por preferência do autor, mas o princípio foi o mesmo da transformação em matriz, os números 1 se tornaram pixels pretos, e os 0 se tornaram pixels brancos.

Este é o resultado final da fase de tratamento da imagem, o resultado é excelente, os caracteres estão completamente destacados, e o nível de ruído é mínimo.

Para os CAPTCHAs que passaram pelo algoritmo de remoção de sujeiras (Figura 25) um outro algoritmo desenvolvido é aplicado após a fase de Gibbs de-noising, este algoritmo preenche verticalmente os pixels alterados por causa da remoção do traço branco.

A Figura 30 demonstra perfeitamente o preenchimento do número “6”.

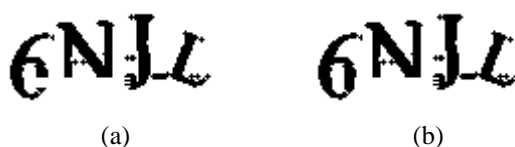


Figura 30: a, b– CAPTCHAs do site do currículo Lattes, após o algoritmo de preenchimento vertical.

(Fonte: Próprio autor.)

Observe que após a aplicação do algoritmo o número 6 foi reestabelecido a sua forma original, este algoritmo compara as distancias verticais dos pixels apenas nas regiões onde o traço branco foi removido, se a distância for menor que 6 pixels, este intervalo é preenchido com valores 1 (pixels pretos).

5.2 Segmentação dos caracteres

O melhor método para segmentar esse CAPTCHA é o *Histogram segmentation* (explicado no capítulo 3) devido aos caracteres estarem a uma distância coerente um dos outros, possibilitando cortes precisos nas imagens, através do cálculo dos histogramas de cada coluna na matriz binária que representa o CAPTCHA, a coluna com menos números 1 é considerada um ponto de corte, assim após 3 cortes os 4 caracteres do CAPTCHA são separados.

A importância da fase de tratamento da imagem pode ser observada aqui, se muitas sujeiras estivessem presentes entre os caracteres, a segmentação se tornaria muito difícil devido a confusão de um valor 1 ser caractere ou não.

As Figuras 32 e 33 são capturas de telas editadas que demonstram o CAPTCHA da Figura 31 em forma de matriz após todo o tratamento já ter sido realizado. Os valores 1 estão contornados para uma melhor visualização. Observa-se claramente as colunas de segmentação entre os caracteres “V”, “D”, “4” e “B”.



Figura 31 – CAPTCHA do Lattes.

(Fonte: <http://buscatextual.cnpq.br/buscatextual/visualizacv.do>, acesso em 18/06/2016.)

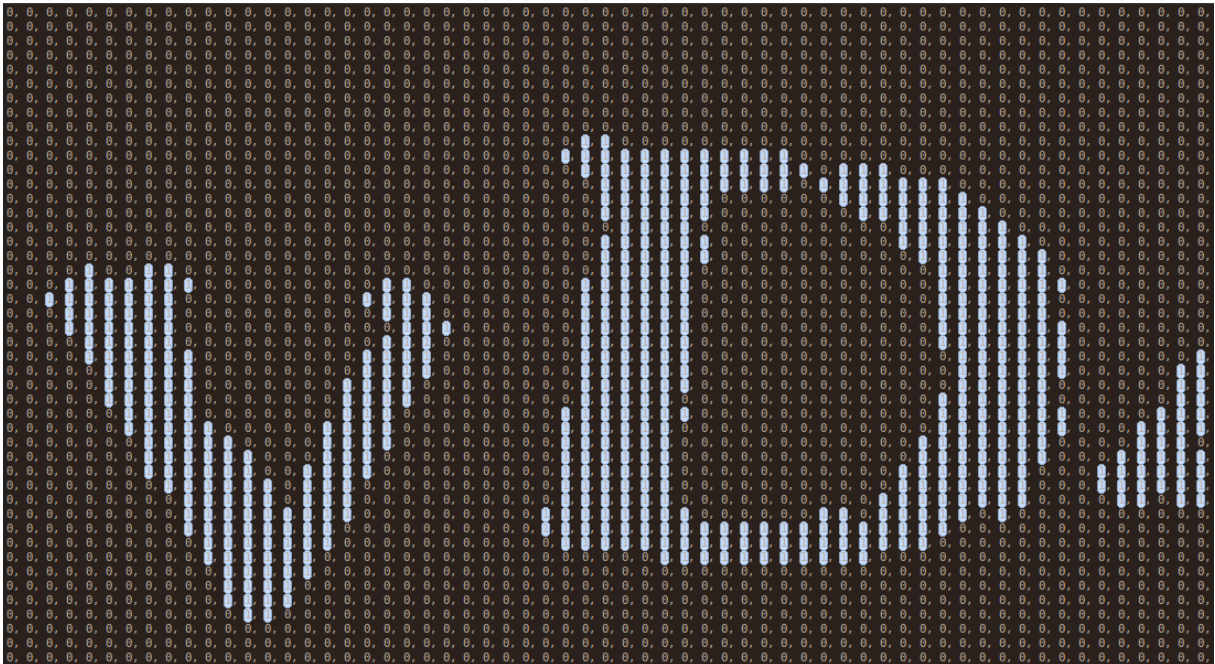


Figura 32 – Captura de tela da matriz que representa o CAPTCHA.

(Fonte: Próprio autor.)



Figura 33 – Captura de tela da matriz que representa o CAPTCHA.

(Fonte: Próprio autor.)

Após os CAPTCHAs serem segmentados verticalmente, são realizados os cortes horizontais na imagem, estes cortes são feitos da mesma maneira que os verticais, mas os histogramas calculados agora são os das linhas da matriz, aquela com menos valores “1” são cortadas, na parte de cima e de baixo da imagem.

A Figura 34 contém exemplos do estado dos CAPTCHAs pelas etapas de quebra, desde sua forma original até a fase de segmentação, as linhas vermelhas demonstram os cortes feitos com o algoritmo de segmentação.

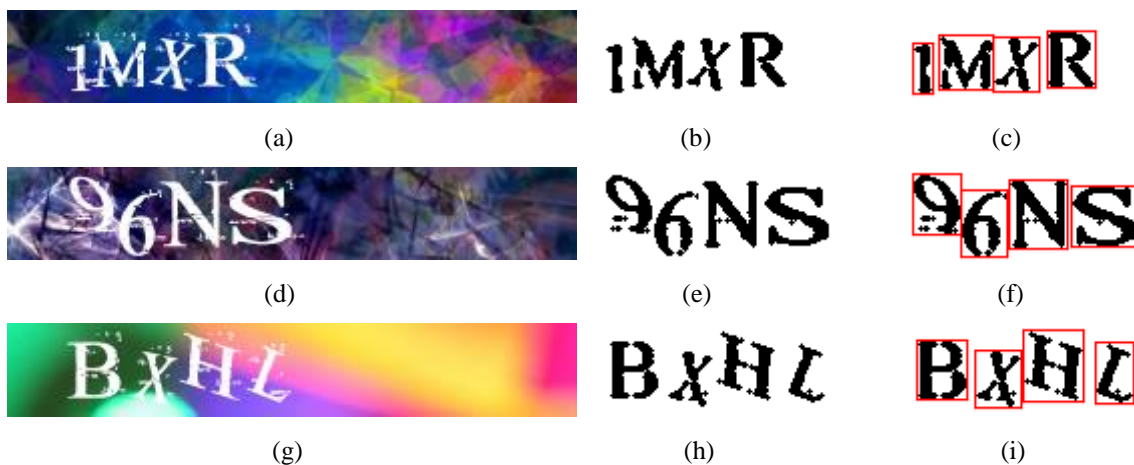


Figura 34: a, b, c, d, e, f, g, h, i – Estado dos CAPTCHAs através dos passos de quebra.

(Fonte: Próprio autor.)

Em alguns casos os CAPTCHAS não foram segmentados corretamente, como os exemplos apresentados na Figura 35, no primeiro caso o caractere “T” se encontra muito próximo do caractere “9”, e os tratamentos utilizados não conseguiram separá-los, a segmentação falhou conforme pode ser visto. No segundo caso um traço não pertencente aos caracteres se encontra entre o “J” e o “C”, e o “H” acabou sendo corrompido com o tratamento, ficando dividido em duas partes, assim o algoritmo de segmentação considerou o “H” como sendo dois caracteres e o “J” e o “C” como somente um devido ao cálculo dos histogramas.



Figura 35: a, b, c, d, e, f – Representação dos cortes nos CAPTCHAs que falharam.

(Fonte: Próprio autor.)

5.3 Reconhecimento dos caracteres

O reconhecimento dos caracteres foi feito com três ferramentas, a RNA FANN e as OCRs: Tesseract e GOCR. Todas sendo de código aberto (*open source*), para tal feito foi analisado em pesquisas correlatas as principais ferramentas utilizadas pelos pesquisadores para o reconhecimento dos caracteres de um CAPTCHA, o Tesseract foi utilizado por M. Korakakis em seu trabalho intitulado como *Automated CAPTCHA Solving: An Empirical Comparison of Selected Techniques*. Já Chanathip Namprem utiliza a GOCR para a etapa de reconhecimento em seu trabalho: *Mitigating Dictionary Attacks with Text-Graphics Character CAPTCHAs*. A rede neural FANN foi escolhida pelo autor com base na pesquisa de Jeff Yan e Ahmad Salah El Ahmad - *A Low-cost Attack on a Microsoft CAPTCHA* (2008, p.68) - onde o uso de redes neurais artificiais é indicado para o reconhecimento dos caracteres de um CAPTCHA baseado em texto.

5.3.1 Reconhecimento dos caracteres com a FANN

A primeira ferramenta experimentada foi a FANN, uma rede neural escrita em C e que possui suporte para várias linguagens (FANN, 2016), como a linguagem Ruby já foi usada nas

outras etapas desta pesquisa, foi utilizado a versão da FANN para a linguagem Ruby denominada “ruby-fann”, uma interface desenvolvida por Steven Miers e mantida *open source* para o acesso de todos em forma de *gem*, funcionando como uma biblioteca para a linguagem Ruby.

Para iniciar o treinamento 700 CAPTCHAS foram selecionados, sendo 100 de cada plano de fundo que o CAPTCHA possui, todos estes CAPTCHAs passam pelas etapas anteriores, sendo elas o tratamento e a segmentação da imagem, um algoritmo foi desenvolvido para manter todo o processo de forma automatizada.

Assim após todas etapas de quebra cada caractere do CAPTCHA é nomeado de maneira manual, o algoritmo apresenta a matriz do caractere segmentado na tela e o usuário digita a resposta. Desta forma um vetor armazena a matriz e a resposta, esse vetor indicará para a FANN os dados de entrada e qual a saída esperada para estes dados, assim é realizado o treinamento da FANN supervisionado.

A Figura 36 demonstra os caracteres sendo exibidos no terminal do Linux para o reconhecimento do usuário, a matriz exibida aqui é o resultado da segmentação dos caracteres na fase anterior de quebra, os valores “0” foram substituídos por “.” e os valores “1” por “#” para uma melhor visualização do usuário que está treinando a Rede Neural Artificial.

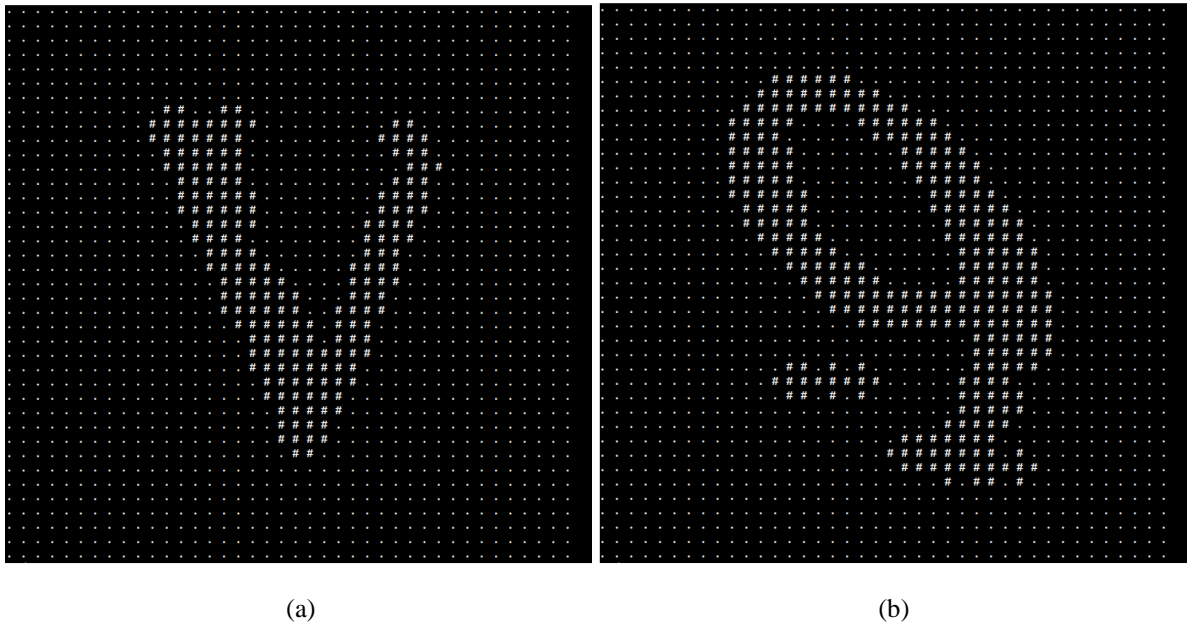


Figura 36: a, b – Representação dos caracteres em forma de matriz exibidos no terminal do Linux.

(Fonte: Próprio autor.)

Foram utilizadas três camadas de neurônios na FANN para o processo de

reconhecimento dos caracteres: A camada de entrada, uma camada escondida e a camada de saída. O número de neurônios da camada de entrada utilizados para a FANN foi de 1600, que é o número de elementos da matriz segmentada possui para cada caractere, a matriz possui a dimensão de 40x40.

A quantidade de neurônios da camada escondida foi escolhida empiricamente, após vários valores testados, a quantidade adequada (com maior taxa de reconhecimento dos CAPTCHAs) foi de 90 neurônios. A camada de saída possui um neurônio para cada caractere que o CAPTCHA pode possuir, são 28 no total e estes caracteres são: “b”, “c”, “d”, “f”, “g”, “h”, “j”, “k”, “l”, “m”, “n”, “p”, “r”, “s”, “t”, “v”, “x”, “w”, “z”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8” e “9”. É importante destacar que os caracteres não descritos neste parágrafo não são utilizados pelo CAPTCHA.

A função de ativação para a camada escondida e a camada de saída da Rede Neural Artificial utilizada foi a *Sigmoid Symmetric* (Sigmoidal) porém com os valores de saídas podendo variar de -1 a 1 e inclinação (*steepness*) de 0,0001. A *Sigmoid Symmetric* foi escolhida devido a sua precisão ser maior para a aproximação das características dos caracteres, com o seu valor de saída podendo variar de -1 a 1 os cálculos dos pesos dos neurônios são mais precisos. A inclinação da função de ativação da FANN indica o quão suave é a conversão dos valores de saída do neurônio de -1 a 1, com o valor baixo de 0,0001 o treinamento da rede pode demorar mais, porém também tende a ser mais apurado com os valores fracionais.

Com a função de ativação escolhida, os pesos iniciais dos neurônios de entrada são definidos de acordo com a matriz do parâmetro de entrada, os valores 0 passam a ser -1 e os valores 1 continuam sendo 1.

O número de épocas que o treinamento passou foi de 1000, este valor também foi escolhido empiricamente, sendo que se poucas épocas fossem utilizadas a RNA pode não conseguir classificar os padrões dos caracteres, e com muitas épocas a RNA se torna muito específica para cada parâmetro de entrada.

Com essas configurações da FANN e com os valores de entrada e saída já definidos, a RNA é treinada até as 1000 épocas terminarem ou até o Erro Quadrático Médio (Mean Square Error) ser de 0,1 ou menos, a gem ruby-fann realiza o treinamento automaticamente após toda a configuração de seus parâmetros serem feitas. Com o fim do treinamento, um arquivo com a extensão “.net” é criado contendo todas as informações da Rede Neural Artificial gerada, este arquivo é usado para a execução da rede. Esta execução é realizada com o parâmetro de entrada (a matriz do CAPTCHA segmentado), na qual a rede reconhece qual é o caractere contido nesta matriz.

5.3.2 Reconhecimento dos caracteres com a OCR Tesseract

A utilização da OCR Tesseract para o reconhecimento dos caracteres é bem mais simples do que a FANN, não necessitando nenhum treinamento para ser realizado previamente, porém o Tesseract não irá utilizar a matriz dos caracteres segmentados para o reconhecimento, o seu parâmetro de entrada é a própria imagem do CAPTCHA, portanto após a segmentação dos caracteres na etapa anterior de quebra, cada matriz segmentada é transformada em imagem novamente com o uso da ferramenta ImageMagick.

A Figura 37 demonstra alguns caracteres após a fase de tratamento de imagem, segmentação, dos caracteres e já transformados em imagem novamente.

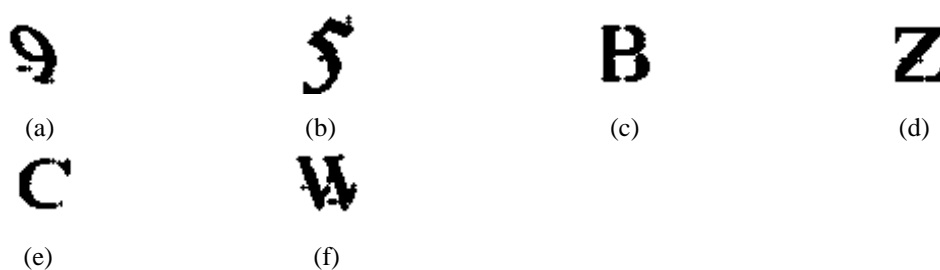


Figura 37: a, b, c, d, e, f – Caracteres do CAPTCHA do Lattes segmentados.

(Fonte: Próprio autor.)

Com os parâmetros de entrada definidos, o software Tesseract é capaz de identificar os caracteres contidos em cada imagem, a gem `rtesseract` realiza a iteração do código Ruby com a ferramenta instalada no sistema operacional Ubuntu v14.04. Como o Tesseract é uma ferramenta para reconhecimento de palavras, é necessário passar o parâmetro “`-psm 8`” para a execução do software, desta forma ele busca identificar apenas uma letra ou número ao invés de uma palavra completa.

5.3.2 Reconhecimento dos caracteres com a GOCR

Com uma maneira de utilização parecida com a OCR Tesseract, a GOCR também reconhece os caracteres contido nas imagens, portanto as matrizes segmentadas são transformadas em imagens da mesma maneira feita para o reconhecimento com Tesseract. Um exemplo pode ser conferido na Figura 37.

Não foi utilizado nenhuma gem para realizar a iteração do software GOCR instalado no Sistema Operacional Ubuntu v14.04 com o código feito em Ruby, para chamar a execução do software foi utilizado o terminal Linux diretamente, passando a imagem com o caractere

segmentado como parâmetro de entrada, a saída é indicada diretamente no terminal em execução.

5.3 Considerações finais do capítulo

Todo o processo realizado neste capítulo foi feito em um computador com o processador Intel Core i5-4200U com 1.60GHz de frequência e 2.60GHz com aceleração de CPU, memória RAM de 6GB e com Sistema Operacional Ubuntu v14.04.

Cada etapa de quebra do CAPTCHA foi analisada e desenvolvida de acordo com trabalhos correlatos feitos por outros atores da área de pesquisa, o CAPTCHA do Lattes foi selecionado para a quebra por ser um CAPTCHA onde todos os conceitos estudados de tratamento de imagem, segmentação de caracteres e reconhecimento de caracteres pudessem ser testados e postos a prova.

O resultado do reconhecimento dos caracteres feitos neste capítulo está descrito no capítulo 6.

6 Resultados e contribuições

Após todas as etapas do processo da quebra do CAPTCHA a solução desenvolvida se mostrou muito eficiente, os resultados foram medidos com 1000 CAPTCHAs do site do currículo Lattes, nenhum deles participaram como treinamento para a FANN, e todos passaram pelo mesmo Tratamento de imagem, mesma segmentação de caracteres e por todas as ferramentas de reconhecimento da mesma maneira (FANN, Tesseract e GOCR).

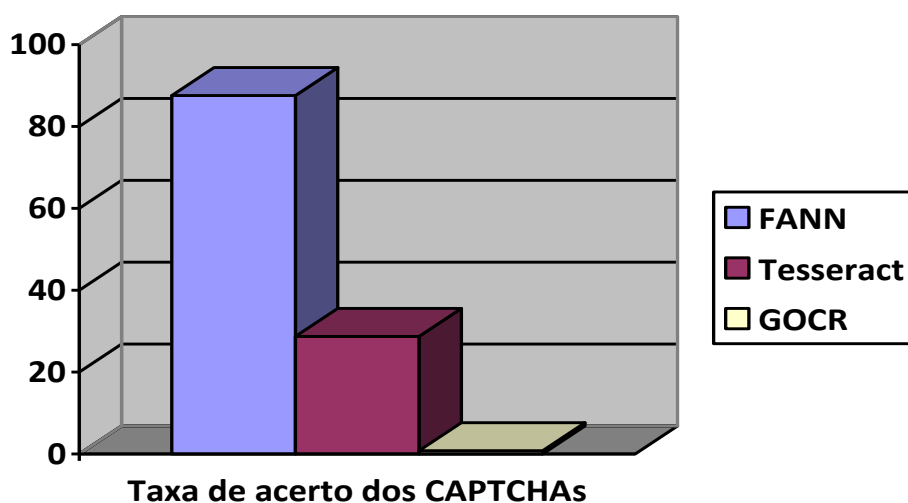
Como citado no capítulo 3 o mínimo de acerto necessário para um CAPTCHA ser considerado quebrado é de 1% (Makris, et al. 2014), qualquer acerto acima disto seria satisfatório, porém quanto maior a porcentagem mais prejudicial para o site seria o ataque de robôs de captação de dados automatizados.

Com a aplicação da fase de reconhecimento dos caracteres com a FANN o resultado obtido foi de 87,5% de acerto, um resultado excelente de acordo como era esperado de uma rede neural com o treinamento aplicado, os CAPTCHAs completamente reconhecidos foram 875 dos 1000 testados.

O Tesseract teve 28,7% de acerto, 287 acertos dos 1000 CAPTCHAs no total, resultado completamente satisfatório em níveis de ataques a um CAPTCHA.

Já a GOCR obteve 0,8% de reconhecimento dos CAPTCHAs, com apenas 8 acertos, não atingindo o mínimo de 1% definido pelos pesquisadores (Makris, et. al 2014).

Gráfico 1 - Taxa de acerto dos CAPTCHAs



(Fonte: Próprio autor.)

O resultado de 87,5% - obtido com a quebra do CAPTCHA do Lattes - comparado com a quebra dos CAPTCHAs descritas nos trabalhos correlatos (capítulo 4), a taxa de acerto é excelente. Christos Makris e Christopher Town obtiveram os seguintes resultados: Wikipédia 67%, Slashdot 26%, BotDetect 81%, Hotmail 50%, reCAPTCHA 0.11% (2014, p.65). Elie Bursztein, Matthieu Martin e John C. Mitchell conseguiram: Baidu 5%, Skyrock 2%, CNN 16%, Digg 20%, eBay 43%, Reddit 42%, Slashdot 35%, Wikipedia 25%, Authorize 66%, Blizzard 70%, Captcha.net 73%, Megaupload 93%, NIH 72% (2011, p.66). O nível de dificuldade de cada CAPTCHA é diferente, cada um contém suas defesas anti-segmentação e anti-reconhecimento.

Claramente o que afetou a performance das OCRs foi a inclinação dos caracteres e o fato das letras não serem exatamente como devem ser, o “W” por exemplo não está completo, conforme pode ser observado na Figura 38:



Figura 38 – CAPTCHA com a letra “W” deformada

(Fonte: Próprio autor.)

O Tesseract conseguiu se sair bem apesar das deformidades dos caracteres, mostrando ser uma ferramenta mais eficiente do que a GOCR levando em conta o que foi apresentado neste trabalho.

O teste com as ferramentas FANN, Tesseract e GOCR apresentam o mesmo resultado sempre que executados com os mesmos 1000 CAPTCHAs, não havendo variação nenhuma.

O tempo de médio de execução dos 1000 CAPTCHAS com a FANN é de 385 milissegundos, muito abaixo do limite de 30 segundos (Zhu, et al. 2010), o pior tempo de execução foi de 597 milissegundos. Já com o Tesseract o tempo médio de execução é de 1,144 segundos e o pior tempo registrado foi de 1,914 segundos. Com a GOCR a média de tempo ficou 454ms e a execução mais demorada com 871ms. Portanto os algoritmos desenvolvidos não possuem problema nenhum com o tempo, qualquer ferramenta de reconhecimento utilizada pode manter um tempo muito baixo de execução.

A Tabela 1 contém estatísticas sobre o tempo de execução do processo de quebra, como o desvio padrão.

Tabela 1 - Estatísticas da execução do algoritmo de quebra do CAPTCHA do LATTES

Ferramenta de reconhecimento de caracteres	Tempo médio de execução	Tempo máximo de execução	Desvio padrão
FANN	385ms	597ms	43,760ms
Tesseract	1,144s	1,914s	127,208ms
GOOCR	454ms	871ms	79,617ms

Fonte (Próprio autor).

6.1 Considerações finais do capítulo

As contribuições deste trabalho destinam-se a um estudo das ferramentas e técnicas para processamento de imagens e quebra de CAPTCHA, como tratamentos e métodos de segmentações, que podem ser aplicados em qualquer tipo de imagem e funcionaram desde que os objetivos sejam os mesmos dos descritos neste trabalho e as características das imagens sejam parecidas com o CAPTCHA do Lattes, as técnicas de quebra de CAPTCHA descritas também podem ser estudadas e utilizadas para outros CAPTCHAs baseados em texto, espera-se uma contribuição para os estudos relacionados a reconhecimento de caracteres com o uso da Fast Artificial Neural Network e também com as OCRs Tesseract e GOOCR.

Foi provado que o CAPTCHA em uso pelo Lattes não é seguro, e que suas características de defesa estão defasadas contra ataques, como recomendação para um aumento de segurança para este CAPTCHA analisado, é indicado uma maior variação de fonte como o CAPTCHA do RegistroBR (Figura 5, p.21), e também nas cores das fontes, com uma cor parecida com o plano de fundo da imagem a técnica de *Thresholding* se torna menos eficiente, um aumento nos ruídos presentes nas imagens também dificultaria a precisão na segmentação dos caracteres. Este trabalho também visa contribuir para a segurança dos CAPTCHAs que podem ser utilizados pelos sites ativos na web.

Conclusão

O objetivo primordial deste projeto era conseguir a quebra do CAPTCHA do site do currículo Lattes através de técnicas de tratamentos de imagem e reconhecimento de caracteres, com o estudo de trabalhos correlatos foi observado que são vários os tipos de CAPTCHAs em uso na web e que a classificação do CAPTCHA do Lattes se encaixa nos tipos baseados em texto, o estudo indicou ainda quais são os passos necessários para a quebra deste tipo de CAPTCHA: Tratamento da imagem, segmentação dos caracteres e reconhecimento dos caracteres.

A partir das informações obtidas, uma análise das técnicas apresentadas pelos trabalhos correlatos foi realizada, cada técnica de tratamento de imagem e segmentação dos caracteres foi testada várias vezes para uma melhor precisão nos resultados futuros, O ImageMagick se mostrou uma ferramenta eficiente e de simples utilização.

A escolha das ferramentas de reconhecimento de caracteres foi baseada em trabalhos correlatos, as mais utilizadas foram selecionadas, testadas e aplicadas de diversas maneiras até a melhor taxa de acerto do CAPTCHA fosse alcançada. A quebra de um CAPTCHA se mostrou complexa e com muitos passos a serem executados, porém o objetivo foi alcançado e foi provado que o CAPTCHA do Lattes não está completamente seguro contra *scripts* automatizados.

Os CAPTCHAs passaram por muitas evoluções durante os últimos anos, mas continua claro que ainda precisam continuar evoluindo devido a computação também haver progredido, as ferramentas descritas neste trabalho são todas de código aberto e podem ser encontradas facilmente na internet, o que pode facilitar a ação de hackers e seus robôs de captação de dados, é de extrema importância que os serviços mais importantes da internet que mais necessitam de CAPTCHA fiquem protegidos contra os robôs, o No CAPTCHA reCAPTCHA é o mais avançado em termos de segurança hoje em dia, porém também já foi quebrado por Suphanee Sivakorn, Jason Polakis e Angelos D. Keromytis (2016, p.66), indicando assim que um CAPTCHA que funcione sem nenhuma falha ainda não existe.

Trabalhos futuros

Para trabalhos futuros tem-se como objetivo as seguintes proposições:

- Testar os métodos de tratamento de imagem e segmentação de caracteres aplicados no CAPTCHA do Lattes em outros CAPTCHAs baseados em texto

em uso na internet.

- Aplicar o reconhecimento dos caracteres com Redes Neurais Artificiais com outras características como a SOM (*Self-Organizing Maps*).
- Desenvolver um CAPTCHA baseado em texto que possa ser seguro contra as técnicas utilizadas neste trabalho e com um nível de usabilidade por humanos aceitável.

Referências Bibliográficas

CATANESE, S. A. **Crawling Facebook for Social Network Analysis Purposes**. 2011, Disponível em: <https://arxiv.org/pdf/1105.6307.pdf>. Acesso em: 18 jun. 2016.

MAHATO, Sandeep, SAXENA, V. P., BHAVSAR, Devendra. **A Survey of Captcha based Web and Application Security Methods and Techniques**. HCTL Open International Journal of Technology Innovations and Research (IJTIR). Volume 14. 2015. Disponível em: http://ijtir.hctl.org/vol14/IJTIR_Article_201504024.pdf. Acesso em: 18 jun. 2016.

VON AHN, Luis, BLUM, Manoel, LANGFORD John. **TELLING HUMANS AND COMPUTERS APART AUTOMATICALLY**. COMMUNICATIONS OF THE ACM. Volume 47, Número 2. 2004. Disponível em: http://www.captcha.net/captcha_cacm.pdf. Acesso em: 18 jun. 2016.

BANDAY, M.T., SHAH, N.A. **A Study of CAPTCHAs for Securing Web Services**. IJSDIA International Journal of Secure Digital Information Age. Volume 1, Número 2. 2009. Disponível em: <http://arxiv.org/pdf/1112.5605.pdf>. Acesso em: 18 jun. 2016.

BURSZTEIN, Elie. **Easy Does It: More Usable CAPTCHAs**. CHI '14 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2014. Disponível em: <http://nlp.stanford.edu/pubs/easy.pdf>. Acesso em: 18 jun. 2016.

MAKRIS, Christos, TOWN, Christopher. **Character Segmentation for Automatic CAPTCHA Solving**. Open Computer Science Journal. Volume 2. 2014. Disponível em: <http://benthamopen.com/FULLTEXT/COMPSCI-1-1>. Acesso em: 18 jun. 2016.

CHEW, Monica, TYGAR, J. D. **Image Recognition CAPTCHAs**. 7th International Information Security Conference. 2004. Disponível em: http://people.eecs.berkeley.edu/~tygar/papers/Image_Recognition_CAPTCHAs/imagecaptcha.pdf. Acesso em: 18 jun. 2016.

FRITSCH, Christoph. **Attacking Image Recognition Captchas A Naive but Effective Approach**. Trust, Privacy and Security in Digital Business. 2010. Disponível em: http://epub.uni-regensburg.de/16872/1/trustbus_1.pdf. Acesso em: 18 jun. 2016.

GAO, Haichang, YAN, Jeff. **The Robustness of Hollow CAPTCHAs**. Proceedings of the

2013 ACM SIGSAC conference on Computer & communications security. 2013. Disponível em: <http://dl.acm.org/citation.cfm?id=2516732>. Acesso em: 18 jun. 2016.

GEMAN, Stuart, GEMAN, Donald. **Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images**. IEEE Transactions on Pattern Analysis and Machine Intelligence. Volume PAMI-6. 1984. Disponível em: <http://www.stat.cmu.edu/~acthomas/724/Geman.pdf>. Acesso em: 01 nov. 2016.

YAN, Jeff, SALAH, Ahmad. **Usability of CAPTCHAs Or usability issues in CAPTCHA design**. Proceedings of the 4th symposium on Usable privacy and security. 2008. Disponível em: <http://dl.acm.org/citation.cfm?id=1408671>. Acesso em: 18 jun. 2016.

MOREIN, William, STAVROU, Angelos, COOK, Debra, KEROMYTIS, Angelos, MISRA, Vishal, RUBENSTEIN, Dan. **Using Graphic Turing Tests To Counter Automated DDoS Attacks Against Web Servers**. Nova York. Proceedings of the 10th ACM conference on Computer and communications security. 2003. Disponível em: <http://dl.acm.org/citation.cfm?id=948114>. Acesso em: 01 nov. 2016.

SINGH, Ved, PAL, Preet. **Survey of Different Types of CAPTCHA**. International Journal of Computer Science and Information Technologies. Volume 5. 2014. Disponível em: <http://research.ijcaonline.org/accnet2016/number1/accnet2257.pdf>. Acesso em: 01 nov. 2016.

SIVAKORN, Suphannee, POLAKIS Jason, KEROMYTIS A. D. **I'm not a human: Breaking the Google reCAPTCHA**. Black Hat ASIA. 2016. Disponível em: <https://www.blackhat.com/docs/asia-16/materials/asia-16-Sivakorn-Im-Not-a-Human-Breaking-the-Google-reCAPTCHA-wp.pdf>.

reCAPTCHA. 2014. Disponível em: <https://developers.google.com/recaptcha/>. Acesso em: 01 nov. 2016.

ZHU, Bin, YAN, Jeff, LI, Qiuji, YANG, Chao, LIU, Jia, XU, Ning, YI, Meng, CAI, Kaiwei. **Attacks and design of image recognition CAPTCHAs**. Proceedings of the 17th ACM conference on Computer and communications security. 2010.

Threshold. 2016. Disponível em: <http://www.imagemagick.org/script/command-line-options.php#threshold>. Acesso em: 01 nov. 2016.

Morphology. 2010. Disponível em: <http://www.imagemagick.org/Usage/morphology/>. Acesso em: 01 nov. 2016.

BURSZTEIN, Elie, MARTIN, Matthieu, MITCHELL, John. **Text-based CAPTCHA Strengths and Weaknesses**. Proceedings of the 18th ACM conference on Computer and communications security. 2011. Disponível em: <http://dl.acm.org/citation.cfm?id=2046724>. Acesso em: 01 nov. 2016.

Tesseract. 2016. Disponível em: <https://github.com/tesseract-ocr/tesseract/>. Acesso em: 01 nov. 2016.

DHIMAN, Shivani, SINGH, A.J. **Tesseract Vs Gocr A Comparative Study**. International Journal of Recent Technology and Engineering. Volume 2. 2013. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.676.9685&rep=rep1&type=pdf>. Acesso em: 01 nov. 2016.

STEFFEN, Nissen. **Implementation of a Fast Artificial Neural Network Library (FANN)**. Department of Computer Science University of Copenhagen (DIKU). 2003.

FANN. 2016. Disponível em: <http://leenissen.dk/fann/wp/>. Acesso em: 01 nov. 2016.

AHMAD, Ahmad, YAN, Jeff, TAYARA, Mohamad. **The Robustness of Google CAPTCHAs**. Newcastle University, UK, School of Computing Science Technical Report Series. 2011.

CHELLAPILLA, Kumar, LARSON, Kevin, SIMARD, Patrice, CZERWINSKI, Mary. **Building Segmentation Based Human-friendly Human Interaction Proofs (HIPs)**. WA, USA, Microsoft Research, One Microsoft Way, Redmond. 2005.

PATEL, Chirag, PATEL, Atul, PATEL Dharmendra. **Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study**. International Journal of Computer Applications. Volume 55. 2012.

TURING, A. M. **COMPUTING MACHINERY AND INTELLIGENCE**. 1950. Disponível

em: <http://www.csee.umbc.edu/courses/471/papers/turing.pdf>. Acesso em: 18 jun. 2016.

AHMAD, A. S. YAN, Jeff. **A Low-cost Attack on a Microsoft CAPTCHA**. Proceedings of the 15th ACM conference on Computer and communications security. 2008. Disponível em: <http://dl.acm.org/citation.cfm?id=1455839>. Acesso em: 01 nov. 2016.

Apêndice A – Comandos de execução do ImageMagick

A Figura 39 demonstra o comando executado no terminal do Linux para execução do tratamento de *Thresholding* e corte com a ferramenta ImageMagick nos CAPTCHAs.

A Figura 40 contém o comando *Morphology Dilate* executado no terminal Linux com o ImageMagick nos CAPTCHAs após o tratamento de *Thresholding*.

```
:~$ convert imagem.png -threshold 87% -crop 150x46+20+0 pos_tratamento.gif
```

Figura 39 – Comando executado no terminal Linux.

(Fonte: Próprio autor.)

```
:~$ convert pos_tratamento.gif -morphology Dilate Diamond pos_dilatacao.gif
```

Figura 40 – Comando executado no terminal Linux.

(Fonte: Próprio autor.)