



**António Jorge
Ferreira Ramos**

**Validação da autenticidade de documentos
impressos**



**António Jorge
Ferreira Ramos**

**Validação da autenticidade de documentos
impressos**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor André Zúquete, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri

presidente

Prof. Doutor João Antunes da Silva
professor associado da Universidade de Aveiro

Prof. Doutor João Antunes da Silva
professor associado da Universidade de Aveiro

Prof. Doutor João Antunes da Silva
professor associado da Universidade de Aveiro

Prof. Doutor João Antunes da Silva
professor associado da Universidade de Aveiro

**agradecimentos /
acknowledgements**

Agradeço a ajuda dos orientadores, amigos e familiares pelo incentivo de realizar a tese.

palavras-chave

Marca de água, códigos de barras, documentos impressos, documentos digitais, segurança, verificação de integridade

resumo

Nos dias que correm existe a necessidade de proteger certos documentos digitais contra vazamento de informações. Uma das maneiras de controlar isso é com a marca de água. Uma marca de água coloca informações visíveis ou invisíveis num documento para garantir a sua originalidade e autenticidade. Nesta dissertação irão ser abordados métodos e o desenvolvimento de um método para ser implementado numa empresa.

keywords

Watermark, barcodes, printed documents, digital documents, security, integrity

abstract

Nowadays there's a need to protect certain digital documents to avoid information leaks. One of the ways that can be managed is by using a watermark. A watermark inputs information, either in a visible way or not, on a document to guarantee its originality and authenticity. In this dissertation methods will be approached and analysed and one developed and implemented for a company.

Índice

Índice	ii
Lista de tabelas	iii
Lista de Figuras	iv
1. Introdução.....	1
1.1. Descrição do sistema existente	1
1.2. Objetivos	1
2. Estado da Arte.....	3
2.1. Código de barras	3
2.1.1. Análise de código de barras.....	3
2.2. Texto com marca de água.....	5
2.2.1. Ataque zero com mudança de texto	5
2.2.1.1. Exemplo.....	6
2.2.1.1.1. Vantagens.....	6
2.2.1.1.2. Desvantagens.....	6
2.2.2. Ataque zero sem mudança no texto.....	6
2.2.2.1. Vantagens.....	7
2.2.2.2. Desvantagens.....	7
2.2.3. Documentos texto baseados em Eigenvalues.....	7
2.2.3.1. Testes.....	7
2.2.3.4. Vantagens.....	8
2.2.3.5. Desvantagens.....	8
2.2.4. Line-Shift Coding	8
2.2.4.1. Word-Shift Coding	9
2.2.4.2. Character Coding	9
2.2.4.3. Vantagens dos métodos	9
2.2.4.4. Desvantagens dos métodos	9
2.2.5. ECL zero-based watermarking	9
2.2.5.1. Vantagens	9
2.2.5.2. Desvantagens	9
3. Qrcode	11
4. Código de barras 128.....	13
5. Entropia documento	15
6. Sistema de verificação de documentos impressos ou digitais	17
6.1. Arquitetura	17
6.2. Estrutura do documento	18
6.3. Base de dados	18
6.4. Criação do código de barras	19
6.5. Operações desempenhar pelo utilizador	19
6.6. Processamento do documento	19
6.7. Retificação do documento	21
6.8. Verificação de integridade.....	21
7. Caso de estudo	24
8. Ficheiros digitalizados.....	30
9. Robustez algoritmo	35
9.1. Escala ficheiro	35
10. Conclusão.....	39
10.1. Futuro Trabalho	39
11. Referências	41
Anexo	42
Processamento de ficheiro PDF.....	42
ZXing.NET	44
Descodificação de posições dos códigos de barras	45
Posições aleatórias para marca de água.....	47
Inserção de Metadados.....	48
Mudança de escala do documento	49

Lista de tabelas

Tabela 1 - Estudo de código de barras lineares	3
Tabela 2 - Possíveis escolhas	5
Tabela 3 - Resultados Eigenvalues	8
Tabela 4 - Classificação de erros do Qrcode	11

Lista de Figuras

Figura 1 - Exemplo de um ataque zero	6
Figura 2 - Esquema de geração e extração da marca de água.....	7
Figura 3 - Antes da marca de água	7
Figura 4 - Depois da marca de água.....	7
Figura 5 - Comparação Marca de água	8
Figura 6 - Exemplo line-shift	8
Figura 7 - Exemplo word-shift	9
Figura 8 - Exemplo Character Coding	9
Figura 9 - Funcionamento do algoritmo.....	9
Figura 10 - Exemplo Qrcode	11
Figura 11 - Estrutura Qrcode.....	11
Figura 12 - Código de barra 128.....	13
Figura 13 - Tabela ASCII.....	13
Figura 14 - Arquitetura solução.....	17
Figura 15 - Diagrama da Base de dados.....	18
Figura 16 - Processamento exemplo	19
Figura 17 - Erro ao clicar em aceitar ou rejeitar sem antes de processar o ficheiro	20
Figura 18 - Reconhecimento caracteres	20
Figura 19 - Finalização do processamento	20
Figura 20 - Finalização da marca de água	20
Figura 21 - Mensagem Documento Aceite.....	20
Figura 22 - Mensagem documento já aceite	20
Figura 23 - Abrir Documento.....	21
Figura 24 - Retificação documento.....	21
Figura 25 - Algoritmo que conta número de retas que se intersetam	22
Figura 26 - Algoritmo interseção retas.....	22
Figura 27 - Exemplo verificação integridade	22
Figura 28 - Processamento diagrama de fluxo	24
Figura 29 - Documento exemplar sem marca de água	25
Figura 30 - Tempo de execução	25
Figura 31 - Documento com marca de água.....	25
Figura 32 - Documento	26
Figura 33 - Código de barras.....	26
Figura 34 - Marca de água documento.....	26
Figura 35 - Obtenção de caracteres.....	26
Figura 36 - Representação dos valores dos caracteres	26
Figura 37 - Pontos para verificação de integridade.....	26
Figura 38 - Fluxograma Retificar	26
Figura 39 - Resultado Retificação	27
Figura 40 - Resultado Análise Forense	27
Figura 41 - Scan normal.....	27
Figura 42 - Resultado scan normal	28
Figura 43 - Scan torto	28
Figura 44 - Resultado scan torto	28
Figura 45 - Atualização da tabela watermark e criação da tabela dimensions_document	30
Figura 46 - Documento scan torto	30
Figura 47 - Documento scan composto.....	31
Figura 48 - Resultado verificação de integridade	32
Figura 49 - Resultado integridade documento normal	33
Figura 50 - Documento com 50 % de escala	35
Figura 51 - Ficheiro Normal.....	36
Figura 52 - Resultado integridade.....	36
Figura 53 - Ficheiro com 80% de escala	37
Figura 54 - Resultado ficheiro com 80% de escala.....	37
Figura 55 - Código de extração dos caracteres num ficheiro PDF.....	42
Figura 56 - Obtenção dos valores	42
Figura 57 - Execução do ficheiro jar.....	42
Figura 58 - Exemplo posição de caracter no PDF	43
Figura 59 - Valor representativo na base de dados	43
Figura 60 - Gerar códigos de barras 128 e 39.....	44
Figura 61 - Leitura de código de barras 128.....	44
Figura 62 - Obtenção de posições do código de barras	45
Figura 63 - Guardar imagem auxiliar.....	46
Figura 64 - Representação dos pontos do código de barras.....	46
Figura 65 - Código para escrever posições numa imagem	47
Figura 66 - Resultado.....	47

Figura 67 - Desfasamento pontos	47
Figura 68 - Exemplo metadados	48
Figura 69 - Alterar escala do documento	49
Figura 70 - Cálculo escala do documento	49
Figura 71 - Adaptar cálculo posições.....	49

1. Introdução

No âmbito da tese de Mestrado de Engenharia Informática da Universidade de Aveiro, pretende-se que seja desenvolvido um módulo aplicacional a integrar com uma solução de gestão de informação classificada desenvolvida pela empresa iCreate Consulting.

Esta solução prevê a possibilidade de impressão de documentos classificados, devendo ser implementados mecanismos de marcação dos mesmos no sentido de garantir a sua autenticidade.

1.1. Descrição do sistema existente

Estes documentos vão ser tratados em áreas de segurança física destinadas para o efeito, onde para aceder o utilizador têm que ser pessoas idôneas (de confiança) e acesso a documentos com informação classificada deverá ter um certificado periódico.

Cada utilizador tem acesso a um determinado Posto de Trabalho, e aos documentos que foram destinados a esse mesmo Posto de Trabalho.

Por outro lado, esse utilizador só tem permissão para ver documentos até um nível de classificação para o qual está autorizado: por exemplo se só pode ver documentos até ao nível Confidencial, não terá acesso aos documentos Secretos ou de nível de classificação superior.

Apesar de ter acesso a documentos localizados num determinado posto, deve respeitar o princípio da “Necessidade de Conhecer”, não devendo consultar indiscriminadamente os documentos a que tem acesso.

1.2. Objetivos

Para assegurar a sua autenticidade propõe-se dois métodos de segurança no documento.

O primeiro é criar um código de barras que permita validar de uma forma explícita se o documento teve origem no sistema, que possível terá mais informações que caracterize o documento, nomeadamente marcas de segurança. A ideia deste primeiro é passar uma pistola que leia o código de barras e averiguar se o código de barras é válido e se a informação do código de barras corresponde ao documento.

O segundo é criar uma marca de água segura, única, por documento, gerado por um conjunto de informações do documento, que permitam auxiliar em aspectos mais forenses.

2. Estado da Arte

Este capítulo tem como objetivo agregar informações relativas ao tema da tese, sendo dividido em dois subcapítulos, códigos de barras e *text watermarking*. O subcapítulo código de barras servirá para verificar se um documento é fidedigno, isto é comparar a assinatura do documento com a informação que estará no código de barras. O *text watermarking* verificará se a informação do documento foi modificada ou se é o original através de métodos de marca de água.

2.1. Código de barras

Código de barras é um modo de representar informação num estado visual, que possa ser visível sem de ter a necessidade de escrever texto. É classificado em duas categorias, linear (1D) e 2D.

Um dos requisitos da empresa era que o código de barras não ocupasse muito espaço no documento, tendo espaço livre no cabeçalho e no rodapé. Como a categoria linear ocupa menos espaço em altura, tendo um comprimento variável, dependendo do número de caracteres contidos na informação do mesmo. Existem várias subcategorias nos códigos de barras, para analisar as mesmas criou-se uma tabela (1), que compara os mesmos, escolhendo o ideal para inserir no documento. Esta tabela é constituída por colunas:

- Nome: designação do código de barras;
- Imagens exemplificativas;
- Comprimento - variável ou fixo;
- Uso;
- Necessidade de aparelho especial para ler.

Em alguns códigos de barras existe um espaço reservado que não se pode alterar que se chama *checksum* ou *check digit*, cujo objetivo é verificar se a informação do código de barras foi gerada corretamente.

Existem duas maneiras de ler um código de barras:

1. Máquina própria (leitor de barras de barras)
2. Smartphone (iOS ou android) com uma aplicação que permita a leitura.

Devido as questões de segurança, a empresa proíbe uso de telemóveis pessoais dentro do estabelecimento, o que levou descartar a hipótese de usar os smartphones como leitores. E leitores de barras especiais, já que têm um custo adicional para a empresa. Para solução optou-se por usar um package em C#, *Iron Bar Code* [1] que permite a leitura de códigos de barras em ficheiros PDF ou imagens (PNG), sendo apenas o reconhecimento em zonas sem distorção, ou seja, sem que apareça texto ou imagens por cima do código de barras.

2.1.1. Análise de código de barras

Este subcapítulo tem como objetivo expor a análise de vários códigos de barras lineares, cujos parâmetros de escolha foram comprimento (variável ou fixo), tamanho das barras. A tabela 1 apresenta um conjunto de códigos de barras com as características previamente expostas.

Na tabela 2 encontram-se os possíveis códigos de barras que porventura se possam utilizar. Estes códigos foram selecionados com o critério tamanho máximo de caracteres que este pode guardar, podendo mudar no futuro de acordo com a escolha da empresa. A informação que se pretende guardar nos códigos de barras é a data do documento assinado, o local do posto de trabalho e um identificador único do documento, podendo ser mudado no futuro.

Tabela 1 - Estudo de código de barras lineares

Nome	Imagen	Fixo/ Variável	Tamanho das barras	Uso
Post Code Austrália	 12229111ABA 9	Fixo	4	Correios Austrália
CodaBar	 3 1117 01320 6375	Fixo	2	Librarias
Code 25	 0 1 2 3 4 5 6 7 8 9	Variável	2	Librarias
Code 11	 0123452	Fixo	2	Telefones, já não se usa muito devido à sua antiguidade
Code 32		Fixo	2	Farmácia
Code 39	 A012345678 CODE39	Fixo	2	Vários

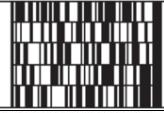
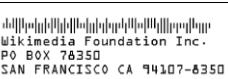
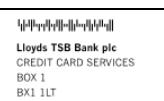
Code 49		Variável	Vários	Vários
Code 93		Variável	Vários	Vários
Code 128		Variável	Vários	Vários
EAN 2		Variável	Vários	Revistas
EAN 5		Variável	Vários	Livros
EAN-8		Variável	Vários	Retalho
GS1-128		Variável	Vários	Vários
GS1 Databar		Variável	Vários	Vários
Intelligent Mail barcode	 Wikimedia Foundation Inc. PO BOX 78350 SAN FRANCISCO CA 94107-8350	Fixo	4	Correios USA
ITF-14		Variável	2	Encomendas
ITF-6		Variável	2	Vários
JAN		Variável	Vários	Usado no Japão
Planet		Variável	Grande/ Pequeno	Correios USA
Plessey	 4 3 2 1 6 Start 43216 Check Stop Digit	Variável	2	Catálogos, revistas, inventários
PostBar		Fixo	4	Correios Canada
PostNET		Fixo	Grande/ Pequeno	Correios USA
RM4SCC/K IX	 Lloyds TSB Bank plc CREDIT CARD SERVICES BOX 1 BX1 1LT	Fixo	4	Correios
RM Mailmark C		Fixo	4	Correios
Universal Product Code		Variável	Vários	Retalho
Telepen	 9 876543210018 ABC-abc-1234	Variável	2	Librarias da Inglaterra

Tabela 2 - Possíveis escolhas

Nome	Imagen	Tamanho máximo de dados	Tipos de dados	Vantagens	Desvantagens
Code 39		43	Letras e números	Ocupa menos espaço.	Não processa caracteres especiais tal como ç.
Code 128		48	Letras, números e caracteres especiais	Faz encoding de strings muito eficiente. Tem checksum	Precisa de espaço reservado no início, fim e check symbols. Tem pouco limite de informação que pode guardar.
Code 93		30	Letras, números e caracteres especiais	Open Source	Não tem checksum

2.2. Texto com marca de água

A *marca de água* é um método que permite a salvaguarda de documentos originais e verificação da autenticidade de documentos ou imagens. É usada nos dias de hoje como por exemplo nas notas europeias, para impedir a impressão de notas não autorizadas. Existem muitos métodos e ataques de marcação de água, neste documento irá ser abordado o método marca de água de texto e os métodos ataques zero devido às necessidades apresentadas pela empresa, contudo poderá mudar consoante o desenvolvimento da tese.

2.2.1. Ataque zero com mudança de texto

O objetivo dos ataques zero é aumentar a fragilidade da marca de água do documento contra ataques de pessoas não autorizadas, normalmente denominados por hackers. Estes ataques tendem a alterar o texto do documento sem alterar a marca de água. Estes ataques normalmente são de inserção, reordenação e remoção de texto.

Uma das soluções propostas pelos autores do artigo [2] foi ataque de substituição que explora a troca de ordem das palavras no texto. Esta solução é constituída por dois elementos:

1. Lista de palavras que se pretende substituir no documento.
2. Lista de palavras contidas no documento depois da substituição.

Em suma, escolhe-se um conjunto de palavras no documento que se quer substituir por novas. Quando uma palavra é selecionada para se substituir todas as ocorrências da mesma são substituídas pela palavra nova. A percentagem do ataque determina a quantidade de modificação do documento. Se a palavra escolhida tiver várias ocorrências no documento, a percentagem vai aumentar.

A percentagem da contagem de palavras no documento é dada pela equação 1. A percentagem do ataque de substituição é dada pela expressão da equação 2.

Existem duas maneiras de escolher a lista de palavras para a substituição, normal e avançada. A normal consiste em selecionar palavras aleatórias do documento. A avançada selecionar palavras com mesmo tamanho do que as palavras da lista de substituição.

Em suma, a marcação de água textual proposto por [2] consiste em 3 passos, selecionar palavras do documento para substituição, selecionar as novas palavras e ocorrência de implementação no texto do documento.

$$WOR(x) = \frac{\text{Number of Occurrence of } x \text{ in Document}}{\text{Total Number of Words in Document}}$$

Equação 1

$$WSR = \frac{\sum WOR(x)}{\text{Total Number of Words in Document}}$$

Equação 2

2.2.1.1.Exemplo

Conforme mencionado no artigo [2] a figura 1, demonstra um exemplo de tipos de substituição que podem ser usados para mascarar informações importantes em um texto. O exemplo demonstra a substituição da palavra "The", que é o pronome mais comum em inglês, por "close" na substituição normal e por "job" na substituição avançada.

A ideia por trás desse método é permitir que palavras-chave ou informações importantes não sejam perdidas, enquanto ainda são mascaradas para outras pessoas que não têm acesso às palavras originais. A substituição de palavras é realizada de tal forma que apenas quem possui o conhecimento necessário para substituir as palavras é capaz de entender o significado original do texto.

Este método pode ser útil em várias situações, como na proteção de informações sensíveis em documentos, comunicações confidenciais, entre outros. No entanto, é importante notar que este método não é infalível.

Original Text: The historic art and architecture of Venice is the key to the city's popularity as a tourist destination, but every two years, from May to October, it also becomes a place of pilgrimage for contemporary art lovers.
Normal Replacement Attack: Close historic art and architecture of Venice is close tech to close city's popularity as a building destination, but every two years, from May to October, it also becomes a defence of pilgrimage for contemporary art lovers.
Advanced Replacement Attack: Job historic art and architecture of Venice is job run to job city's popularity as a brokers destination, but every two years, from May to October, it also becomes a there of pilgrimage for contemporary art lovers.

Figura 1 - Exemplo de um ataque zero

2.2.1.1.Vantagens

Uma das vantagens do método de substituição descrito é permitir a alteração de palavras em um texto por um utilizador fidedigno e, ao mesmo tempo, permitir que o algoritmo saiba em qual zona do documento as alterações foram feitas.

Isso pode ser útil em situações em que uma equipe de revisão precisa fazer alterações em um documento, como em um processo de edição de textos. Ao usar esse método, a equipe de revisão pode fazer as alterações necessárias sem perder informações importantes ou prejudicar a integridade do documento. Além disso, o algoritmo pode identificar facilmente as alterações realizadas e rastreá-las para fins de auditoria e verificação de alterações autorizadas.

2.2.1.2.Desvantagens

Uma das principais desvantagens é que as mudanças no texto podem não fazer sentido para pessoas que não estão familiarizadas com a substituição de palavras usada. Isso pode tornar a leitura do texto difícil e confusa, especialmente se muitas palavras forem substituídas. Se a substituição for realizada de forma excessiva, o texto pode se tornar ininteligível.

Além disso, o método de substituição pode não ser capaz de detetar métodos de cópia de texto, como copiar e colar o texto em um novo documento. Se o novo documento for criado com base no texto original antes da substituição, as informações confidenciais podem ser utilizadas sem que a substituição tenha qualquer efeito.

Outra limitação é que esse método de substituição não garante a segurança completa dos dados, pois é possível que alguém com conhecimento suficiente possa decifrar a substituição e utilizar as informações originais.

2.2.2. Ataque zero sem mudança no texto

A solução proposta no artigo [3] consiste em usar as características do documento para gerar a marca de água com o seguinte layout “autor:watermark:data:tempo” registada pela uma autoridade certificada (CA). O objetivo principal deste método é usar as palavras que tenham um tamanho maior do que quatro caracteres para proteger contra possíveis ataques.

A escolha de palavras maiores do que quatro caracteres é devido a generalidade dos ataques serem direcionados a palavras com tamanho menor do que quatro caracteres.

Para gerar a marca de água, é necessário saber todas as frases de um documento, e assim percorrer cada palavra achando as palavras maiores do que quatro caracteres, quando acabar, guardar o primeiro caracter de cada dessas palavras e por fim gerar a marca de água.

Por exemplo na seguinte frase: “O José gosta muito de ler” a marca de água ficava JGM (José Gosta Muito). Depois de percorrida todas as frases juntasse as marcas de água de cada frase, dando a marca de água final.

A figura 2 demonstra como é que a marca de água é gerada e extraída. Deteta-se um ataque quando o padrão da marca de água tem pelo menos ser 70% igual ao documento comparativo.

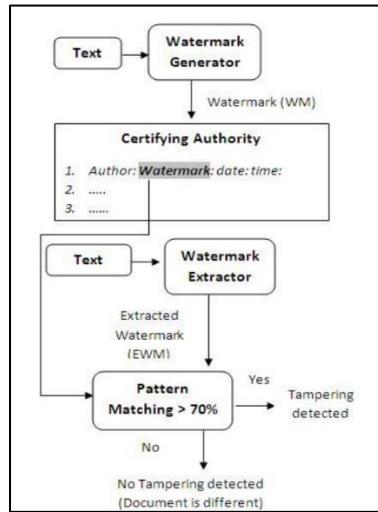


Figura 2 - Esquema de geração e extração da marca de água

2.2.2.1. Vantagens

Tem uma entidade certificadora que vai possuir a marca de água original, que é a única pessoa que pode comparar os documentos, permitindo assim que o documento esteja só na posse de uma pessoa, diminuindo os ataques ou distribuição do documento. A composição da entidade certificadora é similar ao pretendido no código de barras.

2.2.2.2. Desvantagens

Tem testes realizados com ataques aleatórios, como inserir, apagar, reordenar e alterar, que para textos reais os resultados poderão ser diferenciados, levando com que o comportamento do algoritmo seja diferenciado entre documentos, dificultando a sua implementação para a retificação da integridade do documento.

2.2.3. Documentos texto baseados em Eigenvalues

O artigo [4] propõe um algoritmo que guarda as posições de todas as palavras do documento numa matriz e o peso das mesmas em ASCII. Um documento normalmente é constituído por palavras, espaçamento, números e pontuações. Os autores consideram cada contagem para calcular o peso ASCII para gerar o esquema da marca de água com base numa chave privada, que um utilizador fidedigno retifica o documento recebido.

2.2.3.1. Testes

No artigo [4] existe testes realizados a um documento com 47 linhas, não tendo disponibilizado o mesmo, apenas prints de pequenas secções do mesmo.

A figura 3 demonstra um excerto do texto antes de ter a marca de água, e na figura 4 demonstra o texto depois da geração da marca de água. Para simular um ataque foi alterada a palavra “OFF” para “ON” numa zona do texto e pelos resultados binários na figura 5, sendo que em primeiro está o documento original e de seguida o alterado, retira-se que os números são diferentes. Uma pequena alteração da palavra levou à mudança de 11 bits.

Os autores realizaram testes na mudança de vogais, consoantes, caracteres especiais, palavras, números pontuações, e alterações aleatórias no texto, obtendo os resultados da tabela 3, podendo retificar que na coluna “tamper detection” está tudo a 100% que leva a retificar que o algoritmo detetou alterações em todos os casos.

point in time receives a spe
rally speaking, the device 1
made it available. Unauthori
de the tampered version av
nstance, a hacker operating
the image 111 as it is made a
toper with the image 111 as if
a tampered version of the
a variety of means.
140 extracts the verificatio

(a) Before watermarking

Figura 3 - Antes da marca de água

point in time receives a spe
rally speaking, the device 1
made it available. Unauthori
de the tampered version av
nstance, a hacker operating
the image 111 as it is made a
toper with the image 111 as if
a tampered version of the
a variety of means.
140 extracts the verificatio

(b) After watermarking

Figura 4 - Depois da marca de água

Before Tampering Secret key generated is,
0000000000001001011100100000000000000001000000101101
00000000000010111101100010100000000001111001111100
0100000000001110010001001111110110001101000110010
1110000000
After Tampering Secret key generated is,
000000000000100101110011000000000000001000000101101
00000000000010111101100010100000000001111001111100
01000000000011100100010100110110001101000110010
1011000000

Figura 5 - Comparação Marca de água

Tabela 3 - Resultados Eigenvalues

Type of alteration (a single character)	Average eigen value shift	% bit change in secret key	Tamper Detection
Vowels	68.89	12.29	100%
Consonants	66.83	11.67	100%
Words	95.22	14.38	100%
Numerals	169.28	13.96	100%
Punctuations	53.94	10	100%
Random Alterations	127.95	15.63	100%

2.2.3.4. Vantagens

Guarda a posição das palavras do texto numa matriz. Tem uma identidade certificadora que guarda a marca de água e é a única pessoa que pode verificar se o documento é original ou falsificado.

2.2.3.5. Desvantagens

Diffícil de compreensão, a execução do algoritmo leva muito tempo de execução e processamento para textos grandes.

2.2.4. Line-Shift Coding

No artigo [5] expuseram outra forma de fazer marcação de água denominado, *Line-Shift Coding*, que consiste em mover as linhas de texto de um documento para cima ou para baixo, enquanto as linhas adjacentes não são movidas. Na figura 6, demonstra-se um exemplo. Neste exemplo a linha do meio começada por “Effects...”, foi movida para baixo 1/300 inches, que equivale a 0.0084666667 cm. Esta mudança não é perceptível ao olho humano, só através de um OCR, Optical Character Recognition, é que se conseguaria comparar ficheiros para verificar se existe ou não mudança nas linhas.

the Internet aggregates traffic flows from many end systems. Understanding effects of the packet train phenomena on router and IP switch behavior will be essential to optimizing end-to-end efficiency. A range of interesting

the Internet aggregates traffic flows from many end systems. Understanding effects of the packet train phenomena on router and IP switch behavior will be essential to optimizing end-to-end efficiency. A range of interesting

Figura 6 - Exemplo line-shift

2.2.4.1. Word-Shift Coding

No mesmo artigo [5], previamente exposto, expuseram outra forma de fazer marcação de água denominado *Word-Shift Coding*, que consiste em mover as palavras para a esquerda ou para a direita, enquanto as palavras adjacentes não são alteradas. A figura 7, tem-se um exemplo. A segunda linha, contém quatro palavras movidas com espaçamento de 1/150 inch, que equivale a 0.0169333333 cm, enquanto na primeira linha não se altera, a terceira é uma junção das duas anteriores. Como referido no ponto anterior esta mudança não perceptível a olho humano, e para retificar a marca de água recorrer a um OCR.

the Internet aggregates different sessions from many end systems. Understanding
the Internet aggregates different sessions from many end systems. Understanding
the Internet aggregates different sessions from many end systems. Understanding

Figura 7 - Exemplo word-shift

2.2.4.2. Character Coding

No mesmo artigo [5], previamente exposto, deram outra forma similar, referidas anteriormente, mas desta vez consiste em mover o carácter para cima ou baixo, enquanto adjacentes não se alteram. A figura 8, apresenta um exemplo. A primeira letra “e” da palavra “internet” foi movida para baixo 1/600 inches que equivale a 0.0042333333 cm. Como foi referido anteriormente noutros capítulos é necessário um OCR, para retificar as mudanças feitas no texto, isto é dispendioso em termos de tempo, e difícil de percepção.

the Internet aggregates
Internet

Figura 8 - Exemplo Character Coding

2.2.4.3. Vantagens dos métodos

Tem marca de água invisível, ou seja, não é perceptível para os olhos de um humano.

2.2.4.4. Desvantagens dos métodos

Uso de OCR para verificar documento, e ter acesso ao texto para inserir o espaçamento. Erros de impressão, por exemplo faltar letras no documento, pode levar à mal classificação de um documento, por exemplo ser falso, quando se tem a certeza de que o documento é original e não foi modificado.

2.2.5. ECL zero-based watermarking

O artigo [6], propõe um tipo de marcação de água denominado *ECL zero-based watermarking* que consiste num algoritmo que mantém o conteúdo original do texto do documento e constrói a *marca de água* pela seleção de caracteres no documento. A marca de água é guardada num sítio de confiança que se denomina *Certifying Authority*, para quando houver a necessidade de comparar o documento com a *marca de água* para verificação da sua autenticidade, perceber que se o documento é original ou falsificado. Os autores forneceram uma imagem (figura 9) em que se demonstra como funciona o algoritmo.

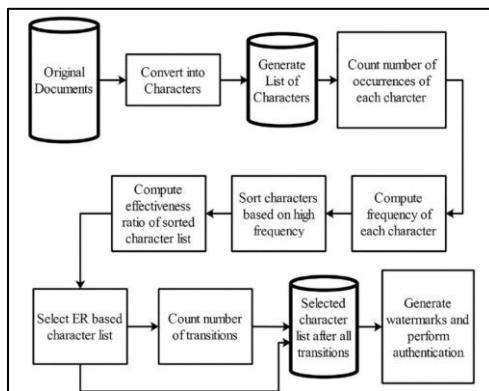


Figura 9 - Funcionamento do algoritmo

2.2.5.1. Vantagens

Tem uma identidade certificadora que garante que o documento original esteja seguro com uma pessoa confiável na empresa.

2.2.5.2. Desvantagens

Falta de testes em cópias de documentos. Pouco efetivo com documentos que tenham um *effectiveness ratio* (ER) menor que 0.1. Effectiveness Ratio determina quais os caracteres para a geração da marca de água.

3. Qrcode

Atualmente vemos qrcode, nas faturas, em cartões de empresas e até mesmo na rua, maior parte das pessoas já viu, mas maior parte pode não saber o processo de leitura ou como surgiu o mesmo, para isso dediquei este capítulo ao qrcode.

O Qrcode (figura 10) surgiu em 1994 pela empresa Denso Wave, onde originalmente foi criado para categorizar peças de automóvel. Os Qrcode podem ter links para páginas web, texto, um endereço geográfico, uma imagem, um vídeo ou contacto telefónico.

Relativamente à estrutura do qrcode é constituído por 3 quadrados de deteção de posição (4.1. Figura 11) que permite a leitura em várias posições do scanner ou de um smartphone com câmara.

Por um padrão alinhamento (4.2. Figura 11) que corrige a distorção do qrcode em superfícies curvadas, o seu número varia consoante a informação contida.

Por padrões de temporização (4.3. Figura 11) que permite obter o tamanho da matriz de dados. Por informações da versão, que indica que versão do qrcode está a ser utilizada (1. Figura 11).

Por informações de formato, que contém informações sobre a tolerância de erros e o padrão da máscara de dados. Por códigos de dados e erros que podem ser do tipo L, M, Q, H, cujos estão apresentados na tabela 6.



Figura 10 - Exemplo Qrcode

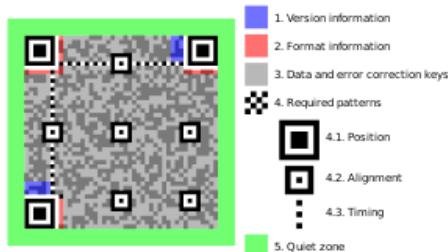


Figura 11 - Estrutura Qrcode

Tabela 4 - Classificação de erros do Qrcode

Nível de correção de erros	Percentagem de área danificada (%)
L (Low) – Baixo	7
M (Medium) – Médio	15
Q (Quartile) - Quartil	25
H (High) – Alto	30

4. Código de barras 128

O código de barras 128 (figura 12) [7] é um tipo de código de barras linear que pode ser usado para codificar uma grande variedade de dados alfanuméricos, incluindo letras, números e caracteres especiais. Ele é chamado de 128 é capaz de codificar 128 caracteres ASCII (figura 13).

Este é muito utilizado em aplicações de logística, como no controle de estoques e na identificação de produtos em supermercados e lojas. Isso ocorre porque ele é capaz de codificar informações como o nome do produto, seu número de série, o código de barras do fabricante e outras informações relevantes em um único código.

É constituído por barras largas e compactas que representam cada caractere, juntamente com barras de início e fim que indicam onde começa e termina a sequência de caracteres. Ele é capaz de codificar uma grande quantidade de informação em um espaço relativamente pequeno, o que o torna uma ferramenta eficiente e econômica para o gerenciamento de inventário e outras aplicações similares.

É um código de barras universalmente reconhecido e amplamente utilizado em todo o mundo, o que leva ser uma opção confiável para empresas que necessitam de um sistema de identificação e rastreamento de produtos rápido e preciso.

Como referido inicialmente, o intuito da dissertação é validação de documentos impressos, para isso é necessário guardar informações acerca do documento em algum lado, para a retificação do original. Como o código de barras consegue agregar informação, como por exemplo um id referenciador para um documento onde por fora esteja uma base de dados que contenha informação do documento, optou-se por utilizar como medida de retificação inserindo-o no rodapé do documento.

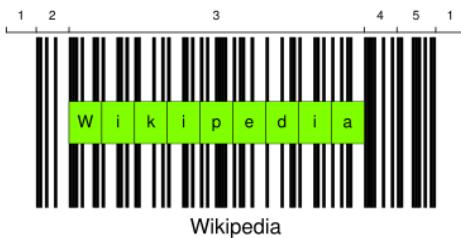


Figura 12 - Código de barra 128

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	'
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	!	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	*	71	47	107	G	103	67	147	g
8	8	8	10	40	28	50	(72	48	110	H	104	68	150	h
9	9	9	11	41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	.	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	,	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	68	8	88	58	130	X	120	78	170	x
25	19	31		57	39	69	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	70	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	71	:	91	5B	133	{	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	l
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	-
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Figura 13 - Tabela ASCII

5. Entropia documento

A presente dissertação analisa estudos de segurança em documentos digitais, é necessário introduzir um conceito fundamental denominada entropia num documento.

A entropia de um documento é uma mediada da incerteza ou desordem da informação contida nele, ou seja, serve para avaliar a segurança de um algoritmo ou sistema criptográfico.

Quanto maior for a entropia maior é dificuldade para a pessoa/hacker descobrir como um algoritmo funciona, o que significa que no desenvolvimento da tese pretendesse que a entropia seja mais pequena possível.

Para atingir esse objetivo é necessário ocultar informações sobre a verificação da integridade e o modo como se gera a marca de água para um documento e desenvolver métodos aleatórios.

6. Sistema de verificação de documentos impressos ou digitais

A primeira intenção para a criação da solução era usar Qrcode, contudo retificou-se que em scans, podiam desaparecer (desvanecer, ou seja, perder a cor) e influenciava na visualização do documento.

Para corrigir estes problemas pensou-se em sugestões como:

- processamento de imagem, que reconhece os Qrcode no texto (através de substituição de cores, por exemplo preto no branco dentro do Qrcode é uma zona crítica, para corrigir é necessário alterar o preto para branco), descartada devido ao elevado tempo de processamento da mudança de cores;
- posições aleatórias no documento, descartada devido a porventura o qrcode ser colocado num sítio não permitido, como por exemplo debaixo de uma imagem, contudo sempre há oportunidade de gerar de novo a marca de água do ficheiro;
- quantidade de Qrcode a colocar no documento;
- dimensão do Qrcode;

Com o desenvolver do algoritmo e o objetivo de traçar retas para a retificação de letras no documento, nos quadrados de posição do Qrcode, optou-se por substituir por uma imagem reduzida com um círculo que contivesse um X, o que levou ao surgimento da questão “se as imagens são visíveis com base nos pontos que se usa para a criação da retificação das intersecções, o hacker não poderá decifrar como se calcula e obtém as mesmas?”, a resposta é sim, consegue. Para corrigir e melhorar a solução removeu-se completamente a imagem, colocando assim os pontos de origem da criação das retas incógnitos ao hacker.

A solução criada envolve duas ações distintas, processamento e retificação do documento.

O processamento do ficheiro envolve utilizar códigos de barras, para a retificação rápida de um documento que seja aceite na base de dados através do utilizador, e posições aleatórias no documento para guardar letras que resultam de intersecções de retas.

A retificação do documento/verificação de integridade/análise forense consiste em determinar se a informação do código de barras corresponder ao documento, se não procedesse a uma análise mais profunda através das intersecções de retas para verificar se as letras que aparecem no sistema são iguais ou não aquelas que aparecem no documento.

6.1. Arquitetura

De acordo com o descrito anteriormente é criada a arquitetura da figura 14, onde surge 3 camadas.

- Servidor destinado à base de dados que irá conter dados armazenados acerca dos ficheiros, códigos de barras, segmentos de reta traçados, posição dos caracteres no ficheiro de input e a geração da marca de água.
- Programa onde acontece o desenvolvimento da criação do código de barras, do processamento do ficheiro para obtenção das posições dos caracteres no documento, da retificação do documento, e verificação de integridade, esta também é responsável pelas conexões entre camadas, desencadeadas pelo utilizador.
- Utilizador é responsável por agregar as ações que o utilizador pode fazer como por exemplo processar o documento ou retificar.

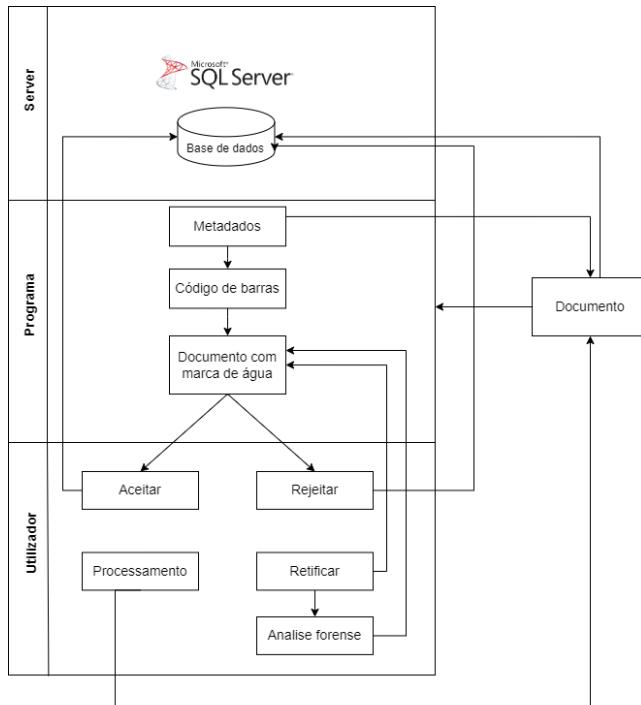


Figura 14 - Arquitetura solução

6.2.Estrutura do documento

Para a utilização de testes do programa foi disponibilizado pela empresa 4 documentos diferentes em formato PDF e digitais. Cabe à empresa inserir manualmente os ficheiros na diretoria correspondente na inicialização do programa e os metadados no programa para tratar os mesmos. É de salientar que os documentos disponibilizados têm uma classificação de segurança mínima pelo que se pode partilhar os resultados do mesmo.

O algoritmo lida com três tipos de documentos:

- Documento sem marca de água;
- Documento com marca de água;
- Documento (scan) com marca de água;

6.3. Base de dados

O intuito de usar base de dados no sistema é garantir segurança sobre as informações do documento através do uso de código de barras, como se referiu anteriormente, o código de barras tem um id aleatório representante das características do documento cujo é inserido na base de dados aquando do seu processamento do documento para a criação da marca de água.

Existem dois tipos de base de dados as relacionais e não relacionais (NoSQL). [8]

As bases de dados relacionais guardam os dados nas tabelas, tendo algumas delas partilha de informação, causando uma relação entre tabelas.

Cada tabela contém colunas que definem a informação que se pode guardar, e linhas que contém a informação.

Normalmente a tabela contém um identificador único que referencia cada linha chamado de chave primária (primary key), caso se queira referenciar os valores a outra tabela utiliza-se a chave estrangeira (foreign key), que obrigatoriamente tem de existir previamente.

A linguagem que se usa para tratar base de dados relacionais é SQL, existem vários programas que permitem correr SQL, como por exemplo mySQL [9], Oracle SQL Developer [10] e Microsoft SQL Server Management Studio 2018 [11].

As bases de dados não relacionais, não usam tabelas relacionais, em vez disso faz cria grupos em se que guarda a informação em informações diferentes.

Como o objetivo do trabalho é sempre perceber que documento é qual, é necessário haver relações entre a marca de água e o documento, então optou-se por usar uma base de dados relacional com o auxílio da ferramenta Microsoft SQL Server Management Studio 2018.

A base de dados SQL foi criada localmente, tendo um utilizador e base de dados dedicado apenas à consulta de informações pelo programa. As informações que se guardam são as características do documento, o código de barras, segmentos de reta traçados entre pontos, posições dos caracteres no documento de entrada (processamento apenas), e a criação da marca de água (aceitação ou rejeição).

Na figura abaixo, está presente um diagrama da base de dados que contém as tabelas usadas e respetivas conexões.

O diagrama é constituído por as seguintes tabelas:

- “document”: guarda características do documento (metados);
- “barcode”: guarda informações relativas ao documento;
- “watermark”: guarda as confirmações do documento com marca de água, se foi aceite ou não, para efeitos de rastreamento;
- “forense_analises”: guarda os segmentos de reta traçados entre dois pontos, o ponto de interseção e a letra que aparece no ponto de interseção para efeitos de verificação de integridade do documento;
- “position_char_file”: guarda as posições dos caracteres no documento.

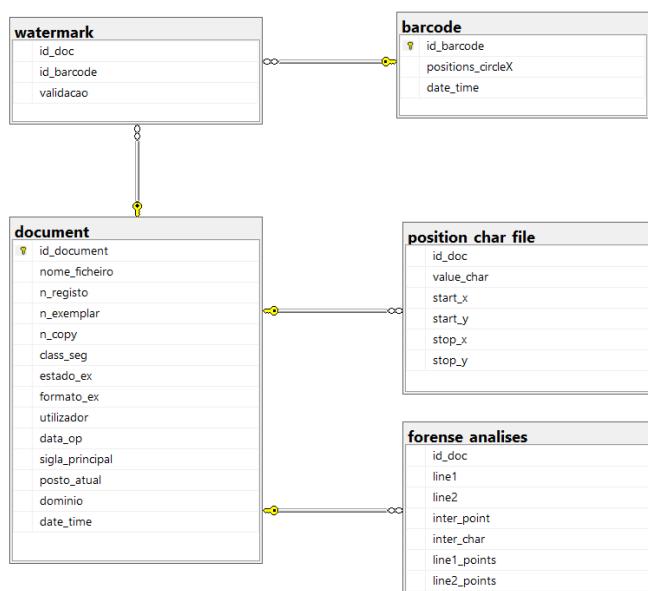


Figura 15 - Diagrama da Base de dados

6.4. Criação do código de barras

Para a criação do código de barras 128 foi utilizado um package para C# denominado Iron Barcode, abordado mais detalhadamente em Anexo.

Como referido anteriormente o código de barras irá ter um id referenciador para o documento e para as posições utilizadas para traçar os segmentos de reta no documento.

O código de barras 128, podendo o seu tamanho ou tipo ser alterado futuramente, irá ser colocado sempre no cabeçalho da primeira página devido a ser uma zona livre do documento, e permitir maior parte das vezes a sua leitura.

6.5. Operações desempenhar pelo utilizador

Um utilizador exerce 4 tipos de funções.

- Escolher entre
 - Processamento: que processa o documento sem a marca de água para originar a marca de água;
 - Retificar: que retifica o documento de input com a marca de água para verificar se é o documento que se apresenta ser.
- Aceitar ou rejeitar o documento com marca de água;
- Comparar a informação do documento com a marca de água no submenu retificar, para determinar a sua autenticidade;
- Verificação de integridade, para determinar a zona da alteração do documento, se por ventura o documento não ser autêntico, através da comparação de informações que aparecem na aplicação versus o documento.

6.6. Processamento do documento

Conforme mencionado anteriormente o utilizador escolhe o ficheiro a processar, o algoritmo só aceita caso o ficheiro não tiver na base de dados e se o nome do mesmo não conter “watermark” que se adiciona ao nome do ficheiro já existente para diferenciar os ficheiros sem e com marca de água, também se adicionou o dia e a hora que se executou o processamento do ficheiro levando ao nome final do ficheiro ser “nome_ficheiro_watermark_dd_mm_yy_ss”, sendo dd:dia, mm:mês, yy:ano, ss:segundos. Antes de processar, caso o utilizador queira visualizar se o ficheiro selecionado é pretendido, o algoritmo mostra o ficheiro numa janela nova com os botões processar, aceitar e rejeitar (figura 16). Caso o utilizador clique em aceitar ou rejeitar o ficheiro sem antes de clicar “processar” aparece o erro da figura 17.

Consoante o clique “processar” o programa vai abrir uma consola que vai executar um ficheiro jar, fechando automaticamente depois da execução, que vai retirar os caracteres do ficheiro e as respetivas posições no documento. Para retirar as posições o algoritmo foi realizado em Java porque em C# não foi encontrado packages que retirassem a posição com certeza dos caracteres no documento (figura 18).

O processamento demora à volta de 2 min, colocando a janela do processamento bloqueada até que as ações de calcular os pontos, obtenção dos pontos de interseção e letras, inserções na base de dados sejam concluídas.

Quando o processamento acaba o utilizador é abordado com uma mensagem da figura 19 para aceitar ou rejeitar o documento que se pode pré visualizar figura 20. Caso o documento seja aceite o utilizador recebe a mensagem “Documento Aceite!” figura 21, porventura se tentar rejeitar ou aceitar de novo o documento é informado de que o documento já foi aceite (figura 22). Caso rejeite o documento é guardado, e o processamento é feito novamente. Caso feche a aplicação sempre poderá consultar o documento que foi gerado para a diretoria que abriu o documento na extensão previamente exposta. A escolha é feita no menu principal quando se escolhe a opção pretendida. É aberto a diretoria predefinida da execução (figura 23) e pode-se mudar consoante o necessário, tendo a noção que ficheiros fora da pasta do algoritmo podem dar origem a erros.



Figura 16 - Processamento exemplo

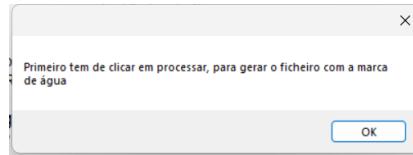


Figura 17 - Erro ao clicar em aceitar ou rejeitar sem antes de processar o ficheiro

```
D|71,29,76,37  
í|79,29,79,37  
á|82,29,85,37  
r|88,29,89,37  
í|92,29,92,37  
o|95,29,98,37  
d|104,29,108,37  
a|111,29,114,37  
R|120,29,125,37  
e|128,29,131,37  
p|134,29,137,37  
ú|140,29,144,37  
b|147,29,151,37  
í|154,29,154,37  
í|157,29,157,37  
c|160,29,163,37  
a|166,29,169,37  
,|172,29,172,37  
í|178,29,181,37
```

Figura 18 - Reconhecimento caracteres

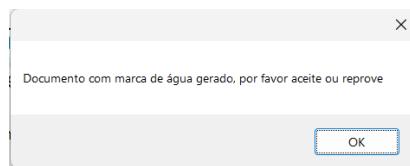


Figura 19 - Finalização do processamento

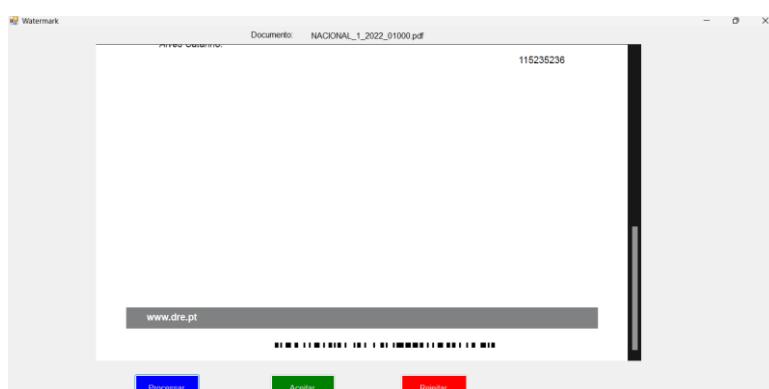


Figura 20 - Finalização da marca de água

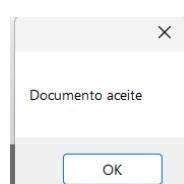


Figura 21 - Mensagem Documento Aceite

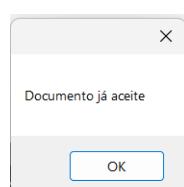


Figura 22 - Mensagem documento já aceite

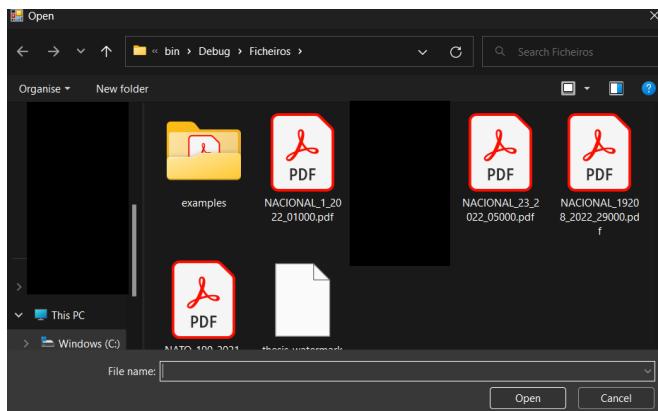


Figura 23 - Abrir Documento

6.7. Retificação do documento

Para retificar um documento é necessário que o documento escolhido para retificação contenha a marca de água e cuja informação esteja na base de dados. Caso esteja, o programa, lê o código de barras 128 e mostra as informações contidas para o utilizador numa janela onde é possível ter a visualização do documento com a marca de água escolhido e algumas informações sobre as características do documento (metadados) (figura 24). Cabe ao utilizador averiguar se o documento e as características demonstradas são diferentes ou iguais, se determinar que são diferentes o utilizador pode proceder à verificação de integridade do documento, referido no ponto 6, determinando as zonas que foram alteradas no documento.



Figura 24 - Retificação documento

6.8. Verificação de integridade

A verificação de integridade ou análise forense, serve para retificar se o documento foi alterado, e em que zonas.

A solução para retificar um documento, é criar 3 segmentos de retas calculados através dos pontos criados nas extremidades da folha 9 (3×3) pontos, totalizando 81(9×9) pontos.

Para saber o número total de retas usou-se a expressão somatória do Gauss [12] (equação 1). O resultado da equação dá igual ao valor no código através do sublinhado a preto na figura 25.

$$n \times \frac{n(n-1)}{2}, n = 9 \leftrightarrow 9 \times \frac{9(9-1)}{2} = 9 \times \frac{9 \times 8}{2} = 9 \times \frac{72}{2} = 9 \times 36 = 324$$

Equação 1 - Gauss

Concluída o traçamento das retas calcula-se o ponto de interseção da mesma [13], com a expressão da figura 26, de seguida com base nas coordenadas dos caracteres obtidos no processamento do ficheiro verifica-se se o ponto pertence ao subconjunto das coordenadas, dando como output a letra correspondida.

Na figura 27 mostra um possível output da verificação da integridade , sendo o círculo amarelo o ponto de interseção e a azul o carácter que está na base de dados, extraído no processamento, sendo que estas apresentações podem ser mudadas no código futuramente, foi escolhida um circulo para ser mais fácil de visualização do que só ter um pixel.

```

62     foreach (KeyValuePair<string, Point> entry in qrcode_points)
63     {
64         string[] val0 = entry.Key.Split('_');
65         foreach (KeyValuePair<string, Point> entry2 in qrcode_points)
66         {
67             string[] val1 = entry2.Key.Split('_');
68             if (!val0[0].Equals(val1[0]))
69             {
70                 qrcode_comb = entry.Key + ":" + entry2.Key;
71                 if (!combs.Contains(entry2.Key + ":" + entry.Key))
72                     combs.Add(qrcode_comb);
73             }
74         }
75     }
76 }
77 @
78 Console.WriteLine("Rects without duplicates " + combs.Count);
79
80 int without_p = 1;

```

The Output window shows the following logs:

```

Show output from: Debug
'WatermarkApp.exe' (CLR v4.0.30319: WatermarkApp.exe): Loaded 'C:\WINDOWS\Microsoft.NET\assembly\GAC_MSIL\PresentationFramework\v4.0_4.0.0.0_c1bd4479d9c084e0\PresentationFramework.dll'.
'WatermarkApp.exe' (CLR v4.0.30319: WatermarkApp.exe): Loaded 'C:\WINDOWS\Microsoft.NET\assembly\GAC_MSIL\System.Xaml\v4.0.4.0.0_96e7d4fbcc67fca1\System.Xaml.dll'.
'WatermarkApp.exe' (CLR v4.0.30319: WatermarkApp.exe): Loaded 'C:\WINDOWS\Microsoft.NET\assembly\GAC_MSIL\SMDiagnostics\v4.0.4.0.0_96e7d4fbcc67fca1\SMDiagnostics.dll'.
'WatermarkApp.exe' (CLR v4.0.30319: WatermarkApp.exe): Loaded 'C:\WINDOWS\Microsoft.NET\assembly\GAC_MSIL\System.ServiceModel.Inter
The thread 0x9e9c has exited with code 0 (0x0).
Rects without duplicates 324
The thread 0x9080 has exited with code 0 (0x0).
The thread 0x3558 has exited with code 0 (0x0).

```

Figura 25 - Algoritmo que conta número de retas que se intersetam

```

private Point Intersection(Point A, Point B, Point C, Point D)
{
    int x1 = A.X;
    int x2 = B.X;
    int x3 = C.X;
    int x4 = D.X;

    int y1 = A.Y;
    int y2 = B.Y;
    int y3 = C.Y;
    int y4 = D.Y;

    double t = (double)((x1 - x3) * (y3 - y4) - (y1 - y3) * (x3 - x4)) / ((x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4));
    double u = (double)((x1 - x3) * (y1 - y2) - (y1 - y3) * (x1 - x2)) / ((x1 - x2) * (y3 - y4) - (y1 - y2) * (x3 - x4));

    Point inter = new Point();
    if ((t >= 0 && t <= 1) && (u >= 0 && u <= 1))
    {
        double x = x3 + u * (x4 - x3);
        double y = y3 + u * (y4 - y3);
        inter = new Point((int)x, (int)y);
    }
    return inter;
}

```

Figura 26 - Algoritmo interseção retas

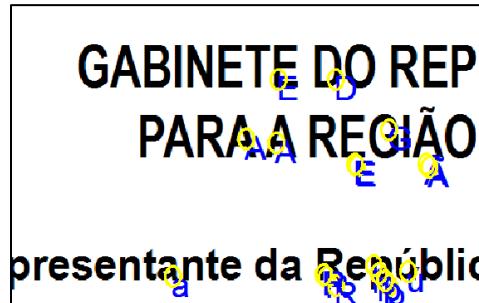


Figura 27 - Exemplo verificação integridade

7. Caso de estudo

Este capítulo é destinado a demonstrar o funcionamento do algoritmo criado num documento exemplar fornecido pela empresa.

Como referido anteriormente existem dois tipos de ações que se pretende desenvolver no algoritmo, processamento de um documento sem marca de água para originar a marca de água e retificação de um documento com marca de água, sendo ele em formato digital ou pelo scan.

O algoritmo para o processamento segue a lógica do fluxograma da figura 31.

O documento escolhido para efeitos de teste, segue-se na figura 32 tendo uma página, denominado “NACIONAL_1_2022_01000”. Depois de 1 min e 25 segundos (figura 33) de processamento descrito no diagrama de fluxo segue-se o documento com a marca de água presente na figura 34.

Na figura 35 encontra-se o resultado do documento inserido na base de dados ao fim da execução do programa.

Na figura 36 tem-se a informação contida no código de barras que é colocado no documento.

Na figura 37 representa a ação do utilizador caso aceite ou rejeite a marca de água de um documento.

Na figura 38 representa um pequeno input da inserção de posições do caracter na base de dados. As colunas start_x, start_y, stop_x, stop_y são os valores representados na figura 39.

Na figura 40 representa uma parte das retas traçadas com base nos pontos dados através do código de barras.

A retificação segue a lógica do fluxograma da figura 41.

O resultado da retificação apresenta-se na figura 42, e caso a informação não seja consistente, realiza-se a verificação de integridade ou análise forense presente na figura 43. O tempo de processamento quer na retificação quer na análise forense é pouco, tendo só o tempo de processamento de mostrar as respetivas janelas com a pre-visualização do documento.

Existindo a possibilidade de os documentos chegarem em formato scaneados, tendo em conta ao documento digitalizado estar contido na base de dados, criou-se dois scans diferentes um com a folha direita (figura 44) e outro com a folha torta (figura 46), sendo que o algoritmo consegue detetar o código de barras num scan normal (figura 44) e realizar a verificação de integridade (figura 45), contudo para o scan torto não consegue, dando um erro de insucesso de leitura (figura 47).

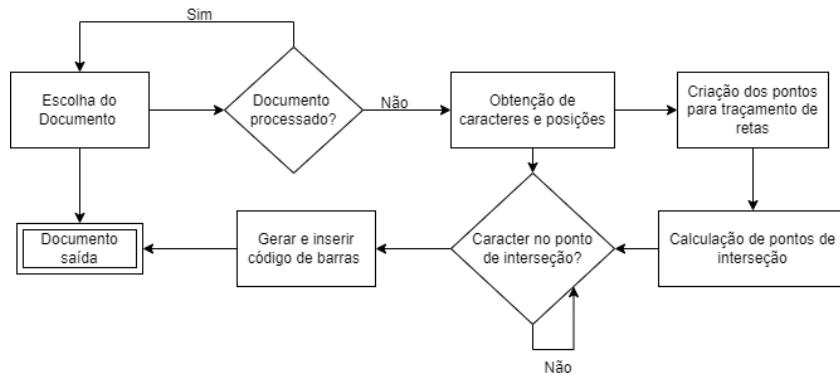


Figura 28 - Processamento diagrama de fluxo

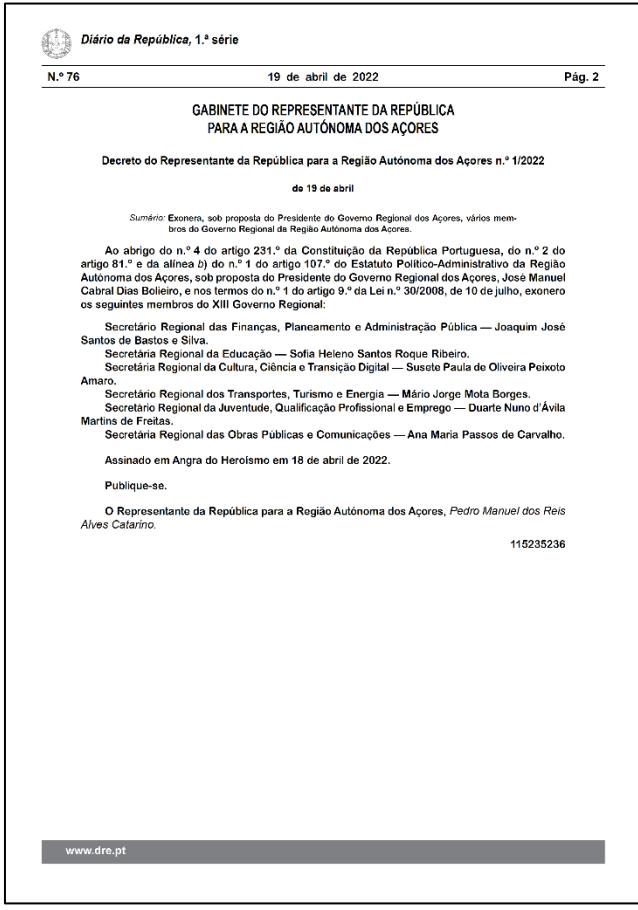


Figura 29 - Documento exemplar sem marca de água

Execution Time 00:01:25.2440000
Figura 30 - Tempo de execução

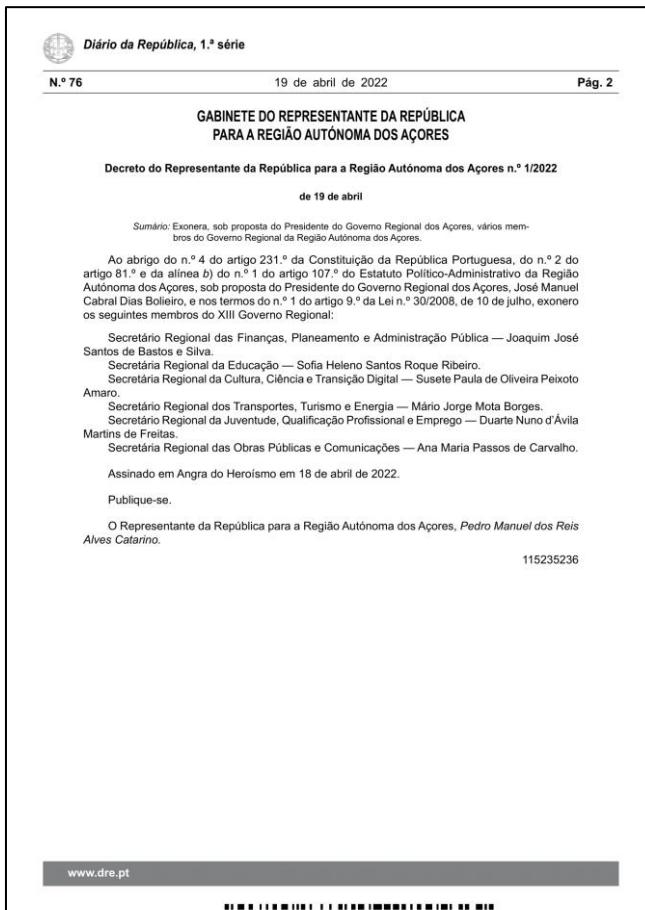


Figura 31 - Documento com marca de água

id_documento	nome_ficheiro	n_registro	n_exemplar	n_copy	class_seg	estado_ex	formato_ex	utilizador	data_op	sigla_principal	posto_atual	dominio
1770408244	NACIONAL_1_2022_01000	1/2022/01000	1	0	S	Arquivado	Eletronico	João Francisco	31/01/2022 15:01:35	Decreto do Representante da República para os Aq...	Registo Central	NACIONAL

Figura 32 - Documento

id_barcode	positions_circleX
1	50,50 297,50 525,50 50,421 297,421 525,421 50,817 297,817 525,817

Figura 33 - Código de barras

id_doc	id_barcode	validacao
1770408244	1	1

Figura 34 - Marca de água documento

id_doc	value_char	start_x	start_y	stop_x	stop_y
1	1770408244	D	71	29	76
2	1770408244	i	79	29	79
3	1770408244	á	82	29	85
4	1770408244	r	88	29	89
5	1770408244	i	92	29	92
6	1770408244	o	95	29	98
7	1770408244	d	104	29	108
8	1770408244	a	111	29	114
9	1770408244	R	120	29	125
10	1770408244	e	128	29	131
11	1770408244	p	134	29	137
12	1770408244	ú	140	29	144
13	1770408244	b	147	29	151
14	1770408244	l	154	29	154
15	1770408244	i	157	29	157
16	1770408244	c	160	29	163

Figura 35 - Obtenção de caracteres

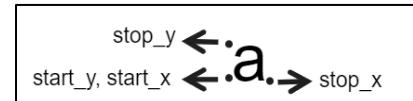


Figura 36 - Representação dos valores dos caracteres

id_doc	line1	line2	inter_point	inter_char	line1_points	line2_points
1	1770408244	point1_l;point5_l	point2_l;point4_b	767,874	G	268,123,1297,1669 1297,123,148,1754
2	1770408244	point1_l;point5_l	point2_l;point7_r	936,1126	s	268,123,1297,1669 1297,123,148,3318
3	1770408244	point1_l;point5_l	point2_l;point8_r	1242,1586	ã	268,123,1297,1669 1297,123,1177,3318
4	1770408244	point1_l;point5_l	point2_l;point8_b	1243,1588	ã	268,123,1297,1669 1297,123,1177,3403
5	1770408244	point1_l;point5_l	point2_b;point7_r	893,1063	p	268,123,1297,1669 1177,208,148,3318
6	1770408244	point1_l;point5_l	point2_b;point8_l	1229,1567	ç	268,123,1297,1669 1177,208,1297,3318
7	1770408244	point1_l;point5_l	point3_r;point4_l	930,1118	s	268,123,1297,1669 2127,123,268,1669
8	1770408244	point1_l;point5_l	point3_r;point7_b	1243,1587	ã	268,123,1297,1669 2127,123,148,3403
9	1770408244	point1_l;point5_r	point2_b;point7_l	991,1183	G	268,123,1177,1669 1177,208,268,3318
10	1770408244	point1_l;point5_r	point3_l;point4_l	890,1182	G	268,123,1177,1669 2247,123,268,1669
11	1770408244	point1_l;point5_r	point3_l;point4_b	888,1178	G	268,123,1177,1669 2247,123,148,1754
12	1770408244	point1_l;point5_r	point3_r;point4_r	853,1118	r	268,123,1177,1669 2127,123,148,1669
13	1770408244	point1_l;point5_r	point3_b;point4_l	889,1180	G	268,123,1177,1669 2127,208,268,1669
14	1770408244	point1_l;point5_r	point3_b;point4_b	887,1176	G	268,123,1177,1669 2127,208,148,1754
15	1770408244	point1_l;point5_b	point2_l;point4_l	736,964	4	268,123,1177,1754 1297,123,268,1669
16	1770408244	point1_l;point5_b	point2_l;point7_r	893,1245	F	268,123,1177,1754 1297,123,148,3318
17	1770408244	point1_l;point5_b	point2_l;point7_r	810,1168	I	268,123,1177,1754 1177,123,148,3403

Figura 37 - Pontos para verificação de integridade

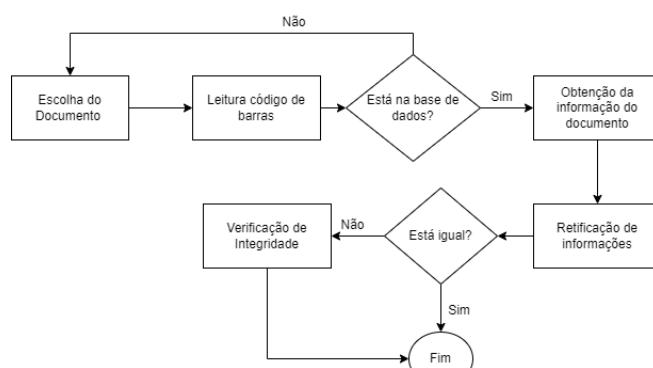


Figura 38 - Fluxograma Retificar



Figura 39 - Resultado Retificação

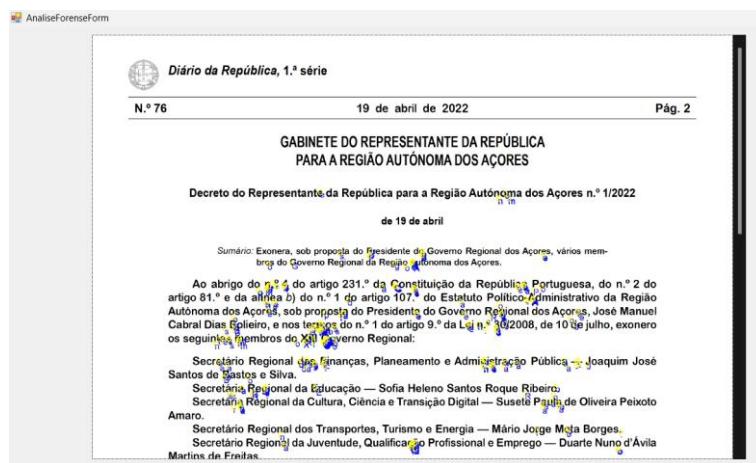


Figura 40 - Resultado Análise Forense



Figura 41 - Scan normal

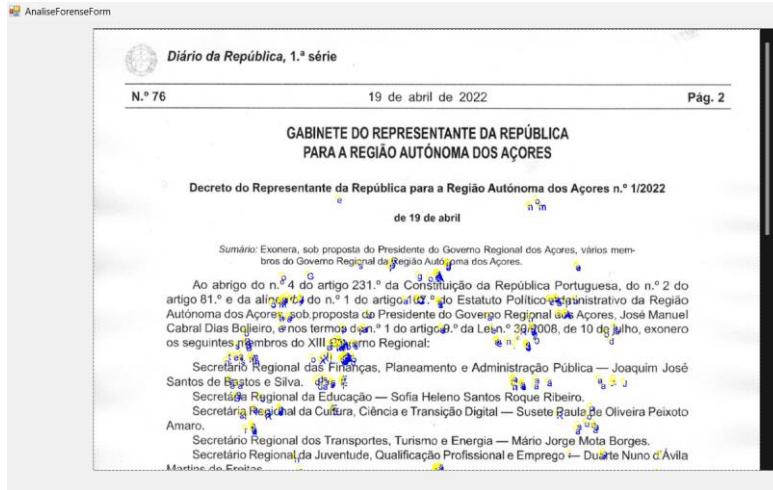


Figura 42 - Resultado scan normal

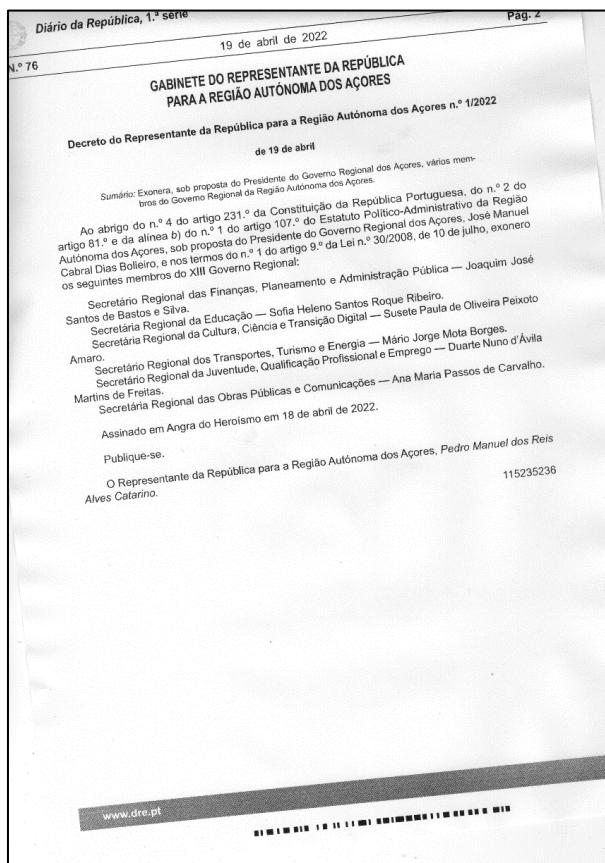


Figura 43 - Scan torto

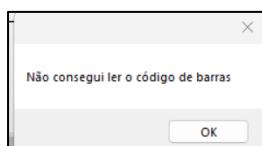


Figura 44 - Resultado scan torto

8. Ficheiros digitalizados

No capítulo anterior, foi apresentado um teste sobre um documento digitalizado que falhou. Caso a digitalização tivesse sido bem-sucedida, as letras estariam no sítio certo. Isto levou a pensar numa estratégia alternativa para compor o documento digitalizado que desse entrada na retificação de forma a garantir a sua retificação.

A estratégia envolve alterar a origem dos pontos pré definidos para serem calculados com base na posição do código de barras no documento e guardar as posições na base de dados do documento original. A tabela responsável por guardar esses valores é “watermark” que tem informações se o documento com marca de água foi ou não aceite (figura 45).

Numa digitalização é possível que um documento seja digitalizado torto, levando à necessidade de aplicar o algoritmo.

A metodologia usada envolve calcular o ângulo da imagem através de setores comparando-os e depois fazer a rotação contrária do ângulo para endireitar o documento, substituindo o documento torto que deu entrada no sistema.

Para garantir o cálculo das novas posições do ficheiro scan estejam corretas no eixo do x e y, criou-se outro código barras do tipo 39 cujo tem a mesma informação do 128, só que o intuito deste é calcular a proporção do y e quanto o documento andou no eixo do y, para calcular o x usa-se o código de barras 128. Porventura verificou-se que certas impressoras quando realizam digitalizações o ficheiro com dimensões diferentes do original(595x842) e é necessário calcular a diferença para ajustar as posições.

A figura 46, demonstra um documento digitalizado relativamente torto, cujo dá entrada no sistema, depois de algum processamento de imagem o resultado apresenta-se na figura 47, mostrando que o algoritmo conseguiu “compor” o documento, pode existir casos que a folha esteja muito torta e o algoritmo não funcionar, mas caso isso aconteça cabe ao utilizador fazer uma digitalização nova.

Através da comparação da figura 48 do resultado da verificação de integridade com o original (figura 49), pode-se afirmar que a discrepância entre pontos é muito pouco sendo a percentagem de eficácia alta, é de salientar que devido a falta de impressoras, só se realizou na impressora Brother mfc7460DN, sendo que os resultados poderão ser diferentes para outras impressoras.

```
CREATE TABLE watermark(
    id_doc INT FOREIGN KEY REFERENCES document(id_document),
    id_barcode INT FOREIGN KEY REFERENCES barcode(id_barcode),
    validacao INT, -- 0 reject, 1 accept
    x INT, -- start x position barcode
    y INT, -- start y position barcode
    x2 INT, -- end x position barcode
    y2 INT, -- end y position barcode
    x_39 INT,
    y_39 INT,
    x2_39 INT,
    y2_39 INT
);

CREATE TABLE dimensions_document(
    id_doc INT FOREIGN KEY REFERENCES document(id_document),
    width int,
    height int,
    width_bmp int,
    height_bmp int
);
```

Figura 45 - Atualização da tabela watermark e criação da tabela dimensions_document

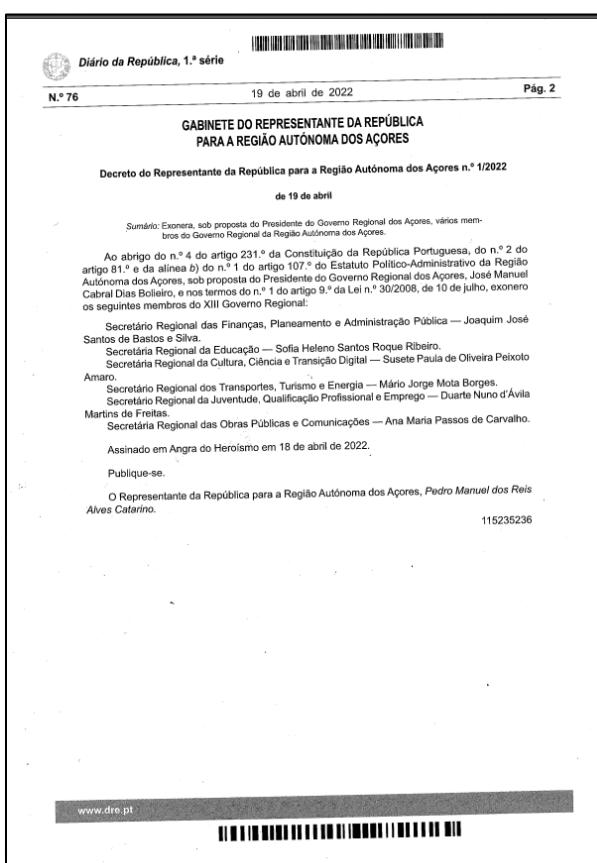


Figura 46 - Documento scan torto

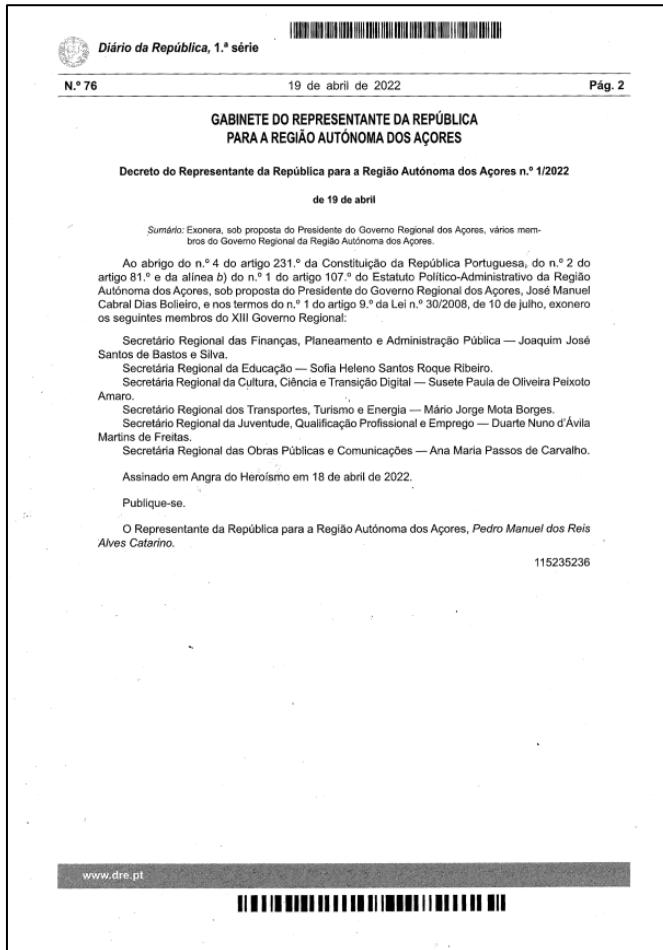


Figura 47 - Documento scan composto

**GABINETE DO REPRESENTANTE DA REPÚBLICA
PARA A REGIÃO AUTÓNOMA DOS AÇORES****Decreto do Representante da República para a Região Autónoma dos Açores n.º 1/2022**

de 19 de abril

Sumário: Emanada, sob proposta do Presidente do Governo Regional dos Açores, vários membros do Governo Regional da Região Autónoma dos Açores.

Ao abrigo do n.º 4 do artigo 231.º da Constituição da República Portuguesa, do n.º 2 do artigo 81.º e da alínea b) do n.º 1 do artigo 107.º do Estatuto Político-Administrativo da Região Autónoma dos Açores, sob proposta do Presidente do Governo Regional dos Açores, José Manuel Carvalho Dias Boavida, e nos termos do n.º 1 do artigo 9.º da Lei n.º 30/2018, de 10 de julho, exófero os seguintes membros do XIII Governo Regional:

Secretário Regional das Finanças, Planeamento e Administração Pública — Joaquim José Santos de Bastos e Silva.

Secretária Regional da Educação — Sofia Helena Santos Roque Ribeiro.

Secretária Regional da Cultura, Ciência e Transição Digital — Susete Paula de Oliveira Peixoto Amaro.

Secretário Regional dos Transportes, Turismo e Energia — Mário Jorge Mota Borges.

Secretário Regional da Juventude, Qualificação Profissional e Emprego — Duarte Nuno d'Ávila Martins de Freitas.

Secretaria Regional das Obras Públicas e Comunicações — Ana Maria Passos de Carvalho.

Assinado em Angra do Heroísmo em 18 de abril de 2022.

Publique-se.

O Representante da República para a Região Autónoma dos Açores, Pedro Manuel dos Reis Alves Catarino.

115235236

www.dre.pt
wd*Figura 48 - Resultado verificação de integridade*



**GABINETE DO REPRESENTANTE DA REPÚBLICA
PARA A REGIÃO AUTÓNOMA DOS AÇORES**

Decreto do Representante da República para a Região Autónoma dos Açores n.º 1/2022

de 19 de abril

Sumário: Exonera, sob proposta do Presidente do Governo Regional dos Açores, vários membros do Governo Regional da Região Autónoma dos Açores.

Ao abrigo do n.º 4 do artigo 231.º da Constituição da República Portuguesa, do n.º 2 do artigo 81.º e da alínea b) do n.º 1 do artigo 107.º do Estatuto Político-Administrativo da Região Autónoma dos Açores, sob proposta do Presidente do Governo Regional dos Açores, José Manuel Gomes Dias Bólio, e nos termos do n.º 1 do artigo 9.º da Lei n.º 30/2008, de 10 de julho, exonera os seguintes membros do XIII Governo Regional:

Secretário Regional das Finanças, Planeamento e Administração Pública — Joaquim José Santos de Bastos e Silva.

Secretária Regional da Educação — Sofia Helena Santos Roque Ribeiro.

Secretária Regional da Cultura, Ciência e Transição Digital — Susete Paula de Oliveira Peixoto Amaro.

Secretário Regional dos Transportes, Turismo e Energia — Mário Jorge Mota Borges.

Secretário Regional da Juventude, Qualificação Profissional e Emprego — Duarte Nuno d'Ávila Martins de Freitas.

Secretaria Regional das Obras Públicas e Comunicações — Ana Maria Passos de Carvalho.

Assinado em Angra do Heroísmo em 18 de abril de 2022.

Publique-se.

O Representante da República para a Região Autónoma dos Açores, Pedro Manuel dos Reis Alves Catarino.

115235236

www.dre.pt



Figura 49 - Resultado integridade documento normal

9. Robustez algoritmo

Para desenvolver um algoritmo é necessário garantir que ele funcione em múltiplos casos, ou seja, que seja robusto. O algoritmo desenvolvido envolve operações em ficheiros e estes podem ser alvo de alterações como por exemplo mudança de escala do ficheiro para impressão e a folha pode ser colocada torta no digitalizador. Neste capítulo abordara-se o comportamento do algoritmo em situações de mudança de escala e alteração no ângulo do ficheiro.

9.1. Escala ficheiro

A escala ideal para testar o algoritmo é entre 80 a 95%, sendo que menor de 50 % o ficheiro torna-se ilegível a olho nu (figura 50), sendo necessário o uso de ferramenta de leitura de PDF para realizar zoom.

O ficheiro utilizado para alteração da escala está apresentado na figura 51 , tendo o resultado da verificação na figura .

Para alterar a escala do ficheiro utilizou-se o package iTextSharp do C#, que permite acesso a modificações de ficheiros seguros que contenham password, o código utilizado irá ser abordado mais detalhadamente em anexo bem como o código para obter a escala do ficheiro.

A figura 53 apresenta-se um print do ficheiro 51 com uma escala de 80%, sendo que o resultado obtido pelo algoritmo presente na figura 54, pode-se retificar através de comparação entre a figura 52 e 54 que o algoritmo conseguiu adaptar as posições no ficheiro com escala de 80%.

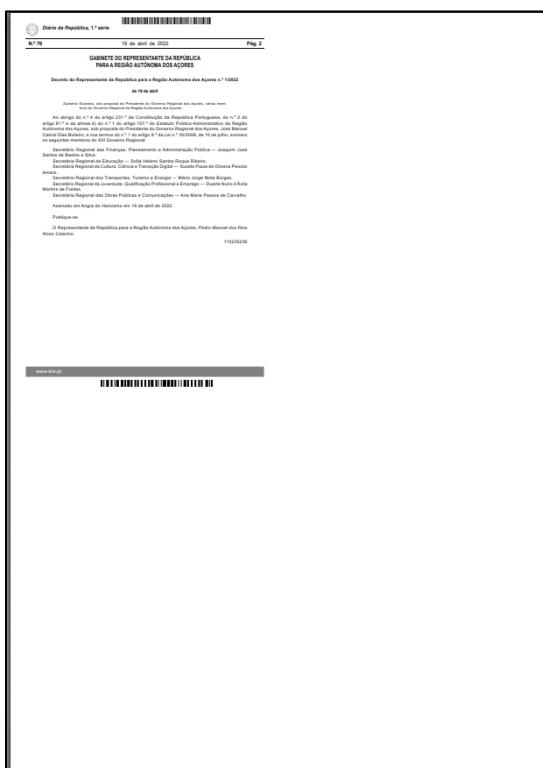


Figura 50 - Documento com 50 % de escala

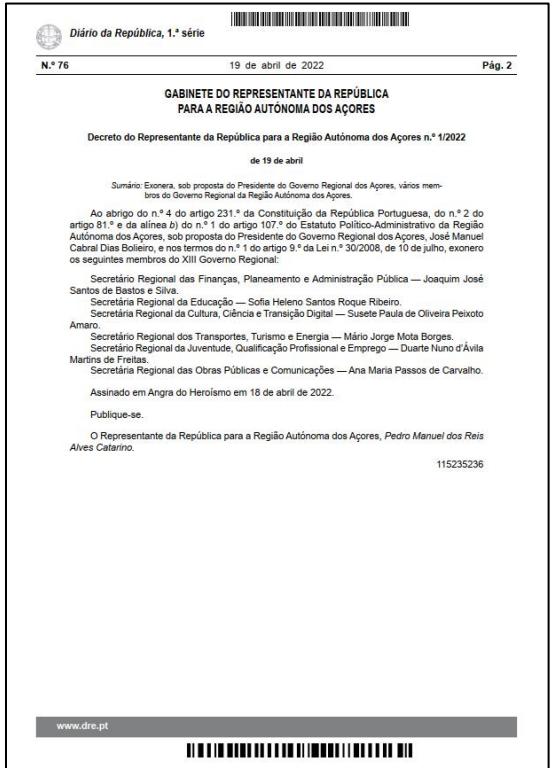


Figura 51 - Ficheiro Normal

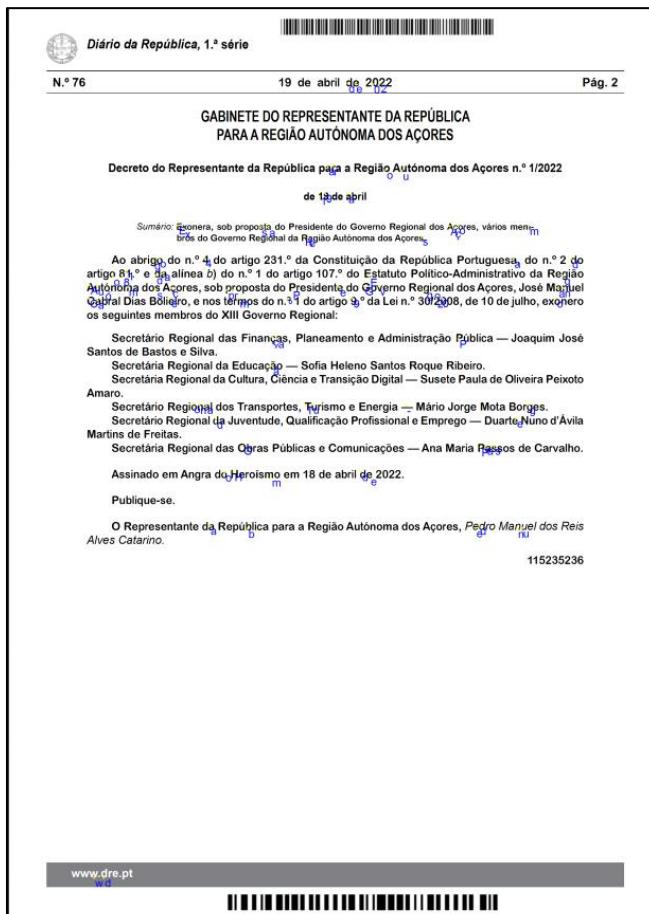


Figura 52 - Resultado integridade

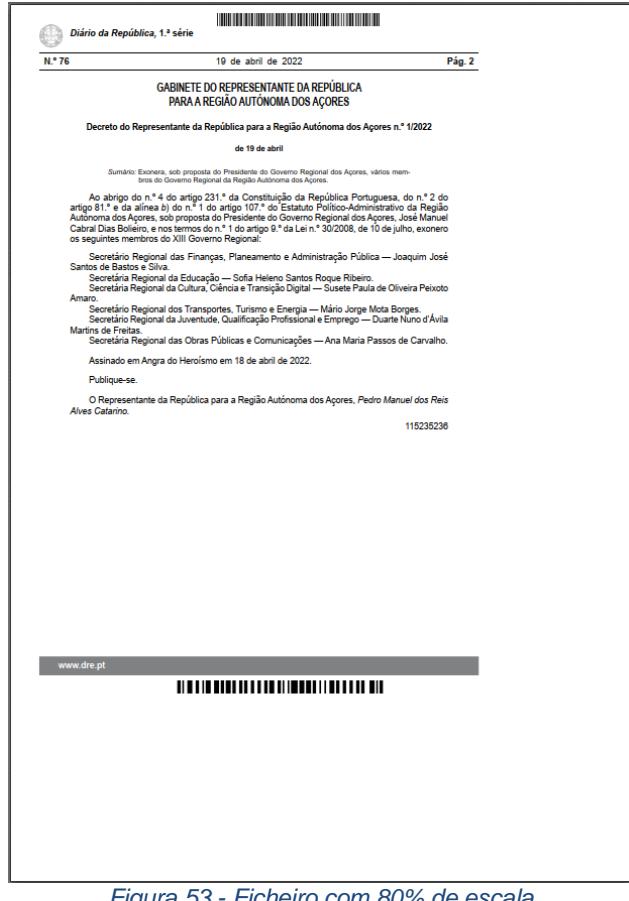


Figura 53 - Ficheiro com 80% de escala

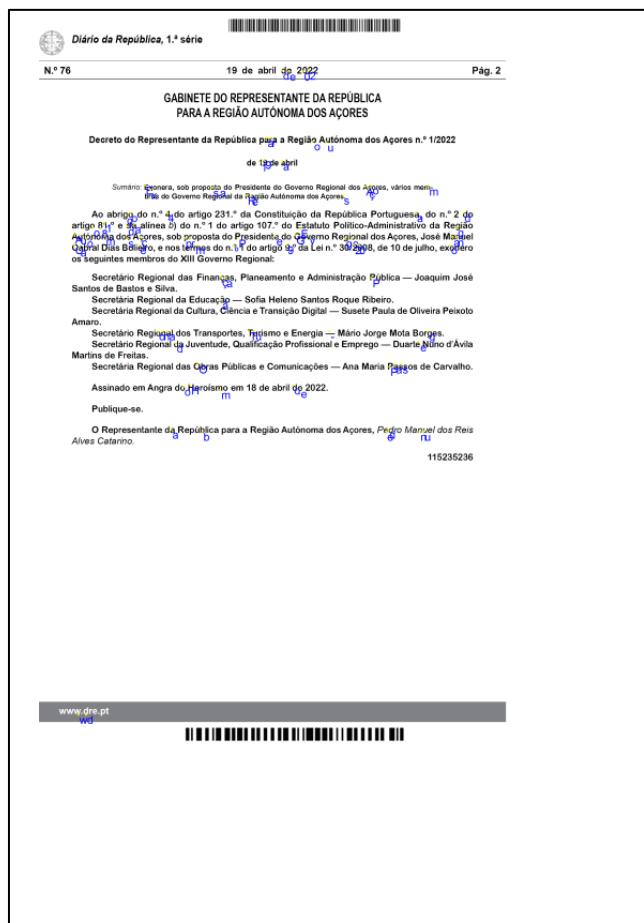


Figura 54 - Resultado ficheiro com 80% de escala

10. Conclusão

O intuito da dissertação é criar um método de autenticidade para documentos, quer digitais quer eletrónicos. Para isso pensou-se em marcas de água, mas a grande questão está como a usar?

Inicialmente pensou-se em usar marcas de água sobre texto, contudo a informação de documentos pode ser confidencial, o que leva a informação contida não ser possível modificada. A solução para resolver o problema foi usar código de barras para controlo de integridade e qrcode para verificação de integridade, usando os quadrados de posição como pontos de referência para traçar retas. Como os qrcode posteriormente não continham nenhuma informação e estavam a ocupar espaço visual no documento optou-se por removê-los, diminuindo assim a possibilidade de os hackers conseguirem perceber a origem dos pontos.

O sistema desenvolvido final permite assim concluir os objetivos iniciais de criação de um código de barras para validação rápida de um documento e criação de marca de água segura capaz de garantir que no documento contenha certas “fingerprints”, que permitam auxiliar os utilizadores em aspetos mais forenses para determinar as zonas do documento que foram alteradas. Apesar de o algoritmo desenvolvido funcionar apenas para a primeira página do documento, para fins demonstrativos, ele pode se modificar no futuro caso se pretenda, levando o aumento de processamento e reconhecimento.

10.1. Futuro Trabalho

11. Referências

- [1] “C# .NET Barcode Quickstart Guide, Code Examples | Iron Barcode.” <https://ironsoftware.com/csharp/barcode/examples/barcode-quickstart/> (accessed Jul. 26, 2022).
- [2] M. B. Mohd, S. Mohd, R. Tanzila, and S. A. Rehman, “Replacement Attack: A New Zero Text Watermarking Attack,” *3D Res.*, vol. 8, 2017, doi: 10.1007/s13319-017-0118-y.
- [3] Z. Jalil, A. M. Mirza, and H. Jabeen, “Word length based zero-watermarking algorithm for tamper detection in text documents,” *ICCET 2010 - 2010 Int. Conf. Comput. Eng. Technol. Proc.*, vol. 6, no. May, 2010, doi: 10.1109/ICCET.2010.5486185.
- [4] T. Rethika, I. Prathap, R. Anitha, and S. V. Raghavan, “A novel approach to watermark text documents based on eigen values,” *2009 Int. Conf. Netw. Serv. Secur. N2S 2009*, no. c, pp. 1–5, 2009.
- [5] J. T. Brassil, S. Low, and N. F. Maxemchuk, “Copyright protection for the electronic distribution of text documents,” *Proc. IEEE*, vol. 87, no. 7, pp. 1181–1196, 1999, doi: 10.1109/5.771071.
- [6] T. Saba, M. Bashardoust, H. Kolivand, M. S. M. Rahim, A. Rehman, and M. A. Khan, “Enhancing fragility of zero-based text watermarking utilizing effective characters list,” *Multimed. Tools Appl.*, vol. 79, no. 1–2, pp. 341–354, Jan. 2020, doi: 10.1007/S11042-019-08084-0/TABLES/5.
- [7] “Code 128 - Wikipedia.” https://it.wikipedia.org/wiki/Code_128 (accessed Jan. 19, 2023).
- [8] “Relational Vs. Non-Relational Databases | MongoDB | MongoDB.” <https://www.mongodb.com/compare/relational-vs-non-relational-databases> (accessed Jan. 19, 2023).
- [9] “MySQL.” <https://www.mysql.com/> (accessed Jan. 19, 2023).
- [10] “Oracle SQL Developer Downloads.” <https://www.oracle.com/database/sqldeveloper/technologies/download/> (accessed Jan. 19, 2023).
- [11] “SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Learn.” <https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16> (accessed Jan. 19, 2023).
- [12] “The Story of Gauss - National Council of Teachers of Mathematics.” <https://www.nctm.org/Publications/TCM-blog/Blog/The-Story-of-Gauss/> (accessed Feb. 08, 2023).
- [13] “Line–line intersection - Wikipedia.” https://en.wikipedia.org/wiki/Line–line_intersection (accessed Oct. 16, 2022).
- [14] “Apache PDFBox | A Java PDF Library.” <https://pdfbox.apache.org/> (accessed Set. 17, 2022).
- [15] “About Us | IronSoftware.com.” <https://ironsoftware.com/about-us/> (accessed Jul. 26, 2022).
- [16] “C# Tesseract OCR In 1 Line of Code | Iron OCR.” <https://ironsoftware.com/csharp/ocr/examples/simple-csharp-ocr-tesseract/> (accessed Jul. 26, 2022).

Anexo

Processamento de ficheiro PDF

Apesar de existirem alguns packages em c# retirassem as posições dos caracteres bem como os mesmos, não era suficiente, já que se precisava de um intervalo de valores donde começa e acaba a letra para determinar se o ponto de interseção pertence ou não à letra para ser usado na verificação de integridade, optou-se por utilizar o package Apache PDFBox [14]desenvolvido em java que conseguia, dar as posições do começo e fim de cada letra bem como a letra respetiva.

Para diminuir o tempo de processamento e para efeitos de teste apenas se lê a primeira página do documento (figura 49), sendo possível depois alterar para todas as páginas, contudo o tempo de processamento aumenta também. Para os valores lidos serem acedidos em C# criou-se um ficheiro temporário que vai guardar as seguintes características “character|start_x,start_y,stop_x,stop_y” como se representa na figura 50.

Contudo é necessário compilar o código em java e criar um ficheiro Jar com os respetivos packages dependentes que permita a execução em C#, para isso abriu-se uma consola e executou-se um comando presente na figura 51.

Para demonstrar que o algoritmo funciona, usou-se um leitor PDF denominado “PDF-Xchange Editor” que permite ver as posições do rato no documento como demonstra na figura 52, a seta aponta para onde está o rato, dando os valores (71, 38), na base de dados presente na figura 53, tem se os valores de “start_x” a 71 e o “stop_y” a 37.

```
PDDocument document = null;
file_name = args[0];
String[] f = file_name.split("\\.pdf");
File fi = new File( pathname + f[0]+".pos.txt");

if(fi.exists())
{
    fi.delete();
}
else {
    fi.createNewFile();
}

String fileName = args[0];
try {
    document = PDDocument.load( new File(fileName) );
    PDFTextStripper stripper = new PositionCharacter();
    stripper.setSortByPosition( true );
    stripper.setStartPage( 0 );
    stripper.setEndPage( 1 );

    Writer dummy = new OutputStreamWriter(new ByteArrayOutputStream());
    stripper.writeText(document, dummy);
}
finally {
    if( document != null ) {
        document.close();
    }
}
```

Figura 55 - Código de extração dos caracteres num ficheiro PDF

```
protected void writeString(String string, List<TextPosition> textPositions) throws IOException {
try{
    String[] f = file_name.split("\\.pdf");
    FileWriter file = new FileWriter( fileName + f[0]+".pos.txt", append: true );
    String ch;

    for (TextPosition text : textPositions) {
        ch = text.getUnicode();

        //remove ?
        if (ch.equals("-"))
            ch = "?";
        else if (ch.equals("`"))
            ch = "'";
        else if (ch.equals("`"))
            ch = "''";

        if(!ch.isBlank() && !ch.isEmpty()) // remove spaces
            file.write( ":" + ch + ":" + Math.round(text.getX()) + "," + Math.round(Math.abs(text.getHeight()) - text.getY()))
                + "," + Math.round(text.getEndX() - text.getWidthOfSpace()) + "," + Math.round(text.getY()) + "\n");
    }
    file.close();
}
catch (IOException e){
    System.out.println("An error occurred");
    e.printStackTrace();
}
}
```

Figura 56 - Obtenção dos valores

```
private void Get_PositionChar()
{
    Process process_file = new Process();
    process_file.StartInfo.UseShellExecute = false;
    process_file.StartInfo.RedirectStandardOutput = true;
    process_file.StartInfo.FileName = "java";
    process_file.StartInfo.Arguments = "-jar " + "" + jar_file + "" + " " + "" + file_name + "";
    process_file.Start();
    process_file.WaitForExit();
}
```

Figura 57 - Execução do ficheiro jar

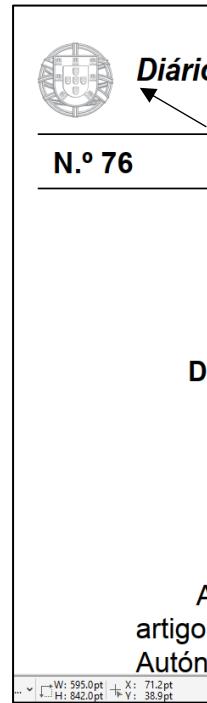


Figura 58 - Exemplo posição de caracter no PDF

Results					
	id_doc	value_char	start_x	start_y	stop_x
1	308082904	D	71	29	76

Figura 59 - Valor representativo na base de dados

ZXing.NET

ZXing.NET é uma livraria open-source desenvolvida em Java, que permite descodificar e gerar códigos de barras (lineares e 2 dimensões). Esta permite ler e codificar os códigos de barras utilizados na dissertação que são eles o 128 e 39, também permite obter as posições dos códigos de barras na imagem.

O código para gerar o código de barras apresenta-se na figura 60 e a descodificação na figura 61, cujo tem só a leitura do código de barras 128, já que este tem informações necessárias que permite retirar dados da base de dados.

```
1 reference
public void Generate_barcode(int id_barcode)
{
    string data_barcode = id_doc.ToString() + ";" + id_barcode.ToString();
    var writer = new BarcodeWriter
    {
        Format = BarcodeFormat.CODE_128,
        Options = new EncodingOptions
        {
            Height = resizedBarcode,
            Width = 250,
            Margin = 0
        }
    };

    var barcodeBitmap = writer.Write(data_barcode);
    barcodeBitmap.Save(filename + commom.extension_barcode);
}

1 reference
public void Generate_barcode_39(int id_barcode)
{
    string data_barcode = id_doc.ToString();
    var writer = new BarcodeWriter
    {
        Format = BarcodeFormat.CODE_39,
        Options = new EncodingOptions
        {
            Height = 15,
            Width = 50,
            Margin = 0
        }
    };

    var barcodeBitmap = writer.Write(data_barcode);
    barcodeBitmap.Save(filename + "_code39.png");
}
```

Figura 60 - Gerar códigos de barras 128 e 39

```
Bitmap bmp = new Bitmap(img_file);
var reader = new BarcodeReader
{
    Options = new DecodingOptions
    {
        PossibleFormats = new List<BarcodeFormat> { BarcodeFormat.CODE_128 },
        TryHarder = true
    }
};

var result = reader.Decode(bmp);
if (result != null)
    return result.Text;
else
    trackerServices.WriteFile("erro na leitura do código de barras");
    bmp.Dispose();
```

Figura 61 - Leitura de código de barras 128

Descodificação de posições dos códigos de barras

Para confirmar se o package ZXing.Net retirava bem as posições do código de barras (figura x), criou-se uma imagem auxiliar donde se escreverá as posições retornadas (figura x). É de salientar que os valores foram ajustados para coincidir com as pontas do código de barras, sendo que para os documentos fornecidos, determinou bem as posições. A imagem x representa um exemplo de um documento com os códigos de barras 128 e 39 onde as bolas amarelas representam as posições do código de barras 128 e rosa do 39.

```
Bitmap bmp = new Bitmap(img_file);
var reader128 = new BarcodeReader()
{
    Options = new DecodingOptions
    {
        PossibleFormats = new List<BarcodeFormat> { BarcodeFormat.CODE_128 },
        TryHarder = true
    }
};

var reader39 = new Bytescout.BarCodeReader.Reader();
reader39.BarcodeTypesToFind.Code39 = true;
reader39.MaxNumberOfBarcodesPerPage = 1;

var result2 = reader39.ReadFrom(bmp);

var barcodeResult128 = reader128.Decodes(bmp);
var result = barcodeResult128.ResultPoints;

List<Point> list128 = new List<Point>();
List<Point> list39 = new List<Point>();
Point p1_barcode128 = new Point();
Point p2_barcode128 = new Point();
Point p1_barcode39 = new Point();

if (result != null && result2 != null)
{
    foreach (var point in result)
    {
        int x = (int)point.X * width / bmp.Width;
        int y = (int)point.Y * height / bmp.Height;
        list128.Add(new Point(x, y));
    }
    foreach (var point2 in result2)
    {
        int x = (int)point2.Rect.X * width / bmp.Width;
        int y = (int)point2.Rect.Y * height / bmp.Height;
        list39.Add(new Point(x, y));
    }
}

for(int i = 0; i < list128.Count; i++)
{
    p1_barcode128 = list128[0];
    p2_barcode128 = list128[1];
}

for(int i = 0; i < list39.Count; i++)
{
    p1_barcode39 = list39[0];
}
else
{
    trackerServices.WriteLine("erro ao obter as posições do código de barras");
    return "";
}

x_barcode_pos = p1_barcode128.X - 10;
y_barcode_pos = p1_barcode128.Y - 2;
x2_barcode_pos = p2_barcode128.X + 16;
y2_barcode_pos = p2_barcode128.Y - 2 + 15;

x_39 = p1_barcode39.X + 1;
y_39 = p1_barcode39.Y;
x2_39 = p1_barcode39.X + 195;
y2_39 = p1_barcode39.Y + 15;

if (file_name.Contains("scan"))
{
    x_barcode_pos = p1_barcode128.X;
    y_barcode_pos = p1_barcode128.Y;
    x2_barcode_pos = p2_barcode128.X;
    y2_barcode_pos = p2_barcode128.Y + 15;
}

x_39 = p1_barcode39.X;
y_39 = p1_barcode39.Y;
x2_39 = p1_barcode39.X + 195;
y2_39 = p1_barcode39.Y + 15;

bmp.Dispose();
```

Figura 62 - Obtenção de posições do código de barras

```

using (Bitmap bmp = new Bitmap(img_file))
{
    using (Graphics g = Graphics.FromImage(bmp))
    {
        SolidBrush drawBrush = new SolidBrush(Color.Chocolate);
        Font drawFont = new Font("Arial", 10);

        int p_x = p1_dig.X * bmp.Width / common.width;
        int p_y = p1_dig.Y * bmp.Height / common.height;
        int p2_x = p2_dig.X * bmp.Width / common.width;
        int p2_y = p2_dig.Y * bmp.Height / common.height;
        Point p1_l_u_r = new Point(p_x, p_y);
        Point p1_r_u_r = new Point(p2_x, p_y);
        Point p1_r_b_r = new Point(p2_x, p2_y);
        Point p1_l_b_r = new Point(p_x, p2_y);

        g.DrawArc(yellow, p1_l_u_r.X, p1_l_u_r.Y, w_arc, h_arc, startAngle, sweepAngle);
        g.DrawArc(yellow, p1_r_u_r.X, p1_r_u_r.Y, w_arc, h_arc, startAngle, sweepAngle);
        g.DrawArc(yellow, p1_l_b_r.X, p1_l_b_r.Y, w_arc, h_arc, startAngle, sweepAngle);
        g.DrawArc(yellow, p1_r_b_r.X, p1_r_b_r.Y, w_arc, h_arc, startAngle, sweepAngle);

        int p_x_39_dig = p1_39_dig.X * bmp.Width / common.width;
        int p_y_39_dig = p1_39_dig.Y * bmp.Height / common.height;
        int p2_x_39_dig = p2_39_dig.X * bmp.Width / common.width;
        int p2_y_39_dig = p2_39_dig.Y * bmp.Height / common.height;
        Point p1_l_u_39_dig = new Point(p_x_39_dig, p_y_39_dig);
        Point p1_r_u_39_dig = new Point(p2_x_39_dig, p_y_39_dig);
        Point p1_r_b_39_dig = new Point(p2_x_39_dig, p2_y_39_dig);
        Point p1_l_b_39_dig = new Point(p_x_39_dig, p2_y_39_dig);

        g.DrawArc(pink, p1_l_u_39_dig.X, p1_l_u_39_dig.Y, w_arc, h_arc, startAngle, sweepAngle);
        g.DrawArc(pink, p1_r_u_39_dig.X, p1_r_u_39_dig.Y, w_arc, h_arc, startAngle, sweepAngle);
        g.DrawArc(pink, p1_l_b_39_dig.X, p1_l_b_39_dig.Y, w_arc, h_arc, startAngle, sweepAngle);
        g.DrawArc(pink, p1_r_b_39_dig.X, p1_r_b_39_dig.Y, w_arc, h_arc, startAngle, sweepAngle);
        g.Dispose();
    }
    string[] filename = file_name.Split(new[] { ".pdf" }, StringSplitOptions.None);
    bmp.Save(filename[0] + "_pos_barcode.png", System.Drawing.Imaging.ImageFormat.Png);
    bmp.Dispose();
}

```

Figura 63 - Guardar imagem auxiliar

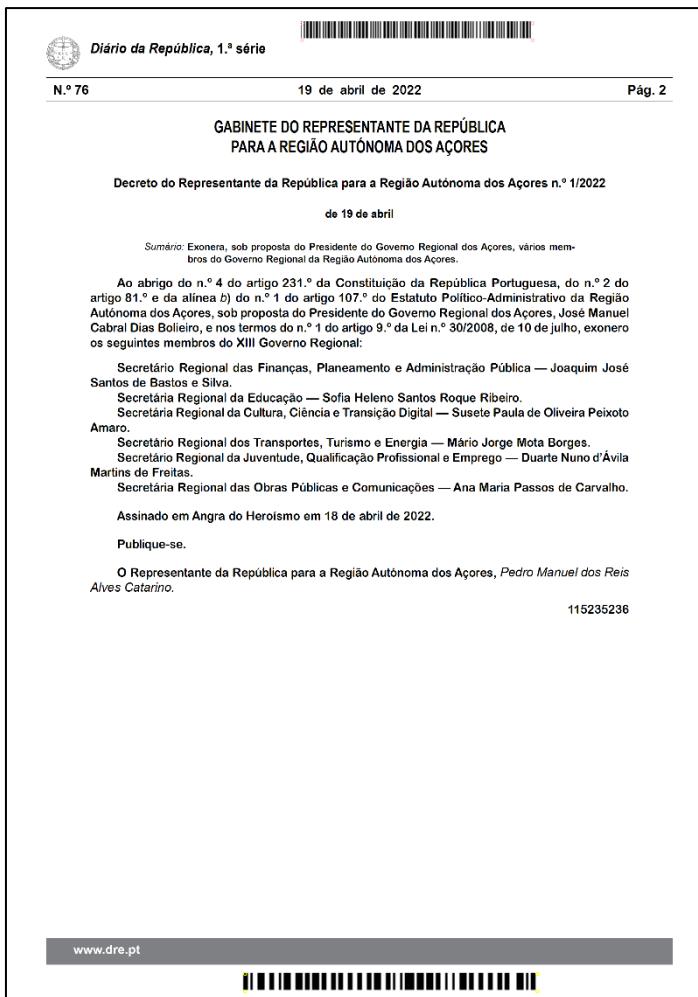


Figura 64 - Representação dos pontos do código de barras

Posições aleatórias para marca de água

Este anexo tem como intuito demonstrar as posições que irão dar origem aos segmentos de reta. A figura x apresenta um pedaço de código que irá escrever as posições dos pontos numa imagem auxiliar (figura x). Através do output da figura x consegue-se perceber que cada ponto tem subPontos aleatórios cujos podem ser definidos na aplicação, neste caso os pontos possíveis é de 10 a 70.

```
Graphics g = Graphics.FromImage(bmp);
Font drawFont = new Font("Arial", 8);
SolidBrush drawBrush = new SolidBrush(Color.Red);

Dictionary<string, Point> circle_points = new Dictionary<string, Point>();
for (int i = 0; i < numberPoints; i++)
{
    string[] pos_circles = position.Split(',');
    string[] circles = pos_circles[i].Split(',');
    int x_circle = int.Parse(circles[0]) + bmp.Width / n;
    int y_circle = int.Parse(circles[1]) + bmp.Height / n;

    // escolher posição adicional aleatoriamente para dificultar a vida aos hackers
    Random random = new Random();
    int randomX = random.Next(min_random, max_random);
    int randomY = random.Next(min_random, max_random);

    Point origin_point = new Point(x_circle, y_circle);
    Point circles_l = new Point(x_circle + randomX, y_circle - randomY);
    Point circles_r = new Point(x_circle - randomX, y_circle - randomY);
    Point circles_b = new Point(x_circle - randomX, y_circle);

    g.DrawString("p", drawFont, drawBrush, origin_point);
    g.DrawString("l", drawFont, drawBrush, circles_l);
    g.DrawString("r", drawFont, drawBrush, circles_r);
    g.DrawString("b", drawFont, drawBrush, circles_b);

    circle_points.Add("point" + (i + 1) + "_l", circles_l);
    circle_points.Add("point" + (i + 1) + "_r", circles_r);
    circle_points.Add("point" + (i + 1) + "_b", circles_b);
}

string path = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + @"\Ficheiros\specifications.png";
bmp.Save(path);

return circle_points;
```

Figura 65 - Código para escrever posições numa imagem

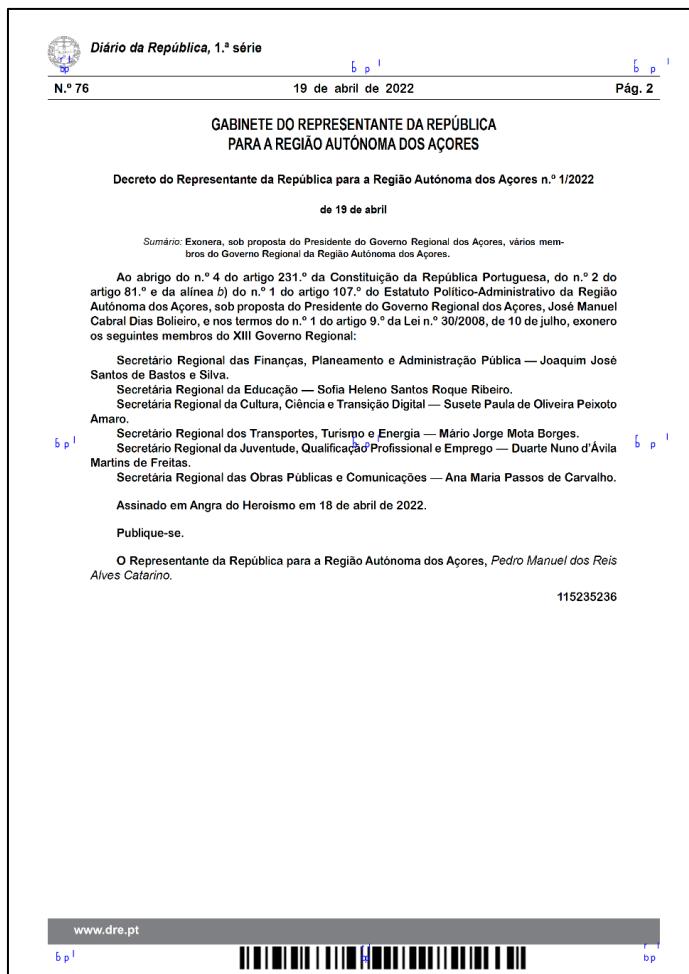


Figura 66 - Resultado

Ponto 1 com desfazamento no x de 15 e no y de 34
Ponto 2 com desfazamento no x de 48 e no y de 21
Ponto 3 com desfazamento no x de 60 e no y de 27
Ponto 4 com desfazamento no x de 33 e no y de 15
Ponto 5 com desfazamento no x de 45 e no y de 20
Ponto 6 com desfazamento no x de 56 e no y de 26
Ponto 7 com desfazamento no x de 38 e no y de 14
Ponto 8 com desfazamento no x de 13 e no y de 37
Ponto 9 com desfazamento no x de 24 e no y de 43

Figura 67 - Desfasamento pontos

Inserção de Metadados

A inserção de metadados é feita manualmente no algoritmo, sendo que posteriormente a empresa vai adaptar para receber os dados dos metadados da base de dados e inserir automaticamente. A figura 65 apresenta um exemplo de metadados de um documento.

```
private static Dictionary<Metadata, string> PreencherMetadadosParaFicheiros()
{
    Dictionary<Metadata, string> conteudos = new Dictionary<Metadata, string>();
    #region Preencher Metadados Aleatorios

    string partialPath = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);

    string ficheiroRC = Path.Combine(partialPath, @"Ficheiros\NACIONAL_1_2022_01000.pdf");
    string ficheiroGNSSR = Path.Combine(partialPath, @"Ficheiros\NACIONAL_19208_2022_29000.pdf");
    string ficheiroMNSR = Path.Combine(partialPath, @"Ficheiros\NACIONAL_23_2022_05000.pdf");
    string ficheiroDELNATOSR = Path.Combine(partialPath, @"Ficheiros\NATO_190_2021_13000.pdf");

    Metadata metadataRegistroRC = new Metadata();
    metadataRegistroRC.NumeroRegisto = @"1/2022/01000";
    metadataRegistroRC.NumeroExemplar = 1;
    metadataRegistroRC.NumeroCopia = 0;
    metadataRegistroRC.ClassificacaoSeguranca = "S";
    metadataRegistroRC.EstadoExemplar = Estado.Arquivado;
    metadataRegistroRC.FormatoExemplar = Formato.Elettronico;
    metadataRegistroRC.Utilizador = "João Francisco";
    metadataRegistroRC.DataOperacao = new DateTime(2022, 01, 31, 15, 01, 35);
    metadataRegistroRC.SiglaPrincipal = @"Decreto do Representante da República para os Açores";
    metadataRegistroRC.PostoAtual = "Registo Central";
    metadataRegistroRC.Dominio = "NACIONAL";
```

Figura 68 - Exemplo metadados

Mudança de escala do documento

A escala do documento é mudada através do uso do package ITextSharp que permite alterações em ficheiros PDF. O código apresentado na figura 69, acede ao ficheiro que se quer alterar, permitindo a mudança de escala em ficheiros seguros através da definição unethicalreading. De seguida cria um ficheiro novo com a extensão da escola desejada. O novo ficheiro com a escala criada vai ser colocada na margem esquerda em cima do ficheiro com as dimensões do ficheiro original, ou seja, o ficheiro vai ter as dimensões do original, mas não vai preencher a página toda. No capítulo 9 apresento um caso exemplo para 80%.

O cálculo da escala do ficheiro é calculado com base na posição do ponto y do código de barras 128 sobre a dimensão do ficheiro original que é sempre 842, somando 0.03 para ajustar a escala. Devido aos comprimentos e pontos sofrerem proporção da escala é necessário calcular o comprimento e os novos pontos do novo ficheiro (figura 70). Para finalizar adapta-se as posições de interseção com base na percentagem de escala como demonstra a figura 71.

```
public void Scale(decimal scalef)
{
    int s = Convert.ToInt16(scalef * 100);
    PdfReader reader = new PdfReader(name);
    PdfReader.unethicalreading = true; // aceder a documentos confidenciais
    Document doc = new Document();
    PdfWriter writer = PdfWriter.GetInstance(doc, new FileStream(name_without_ex + "_scale_" + s + ".pdf", FileMode.Create));
    doc.Open();
    PdfImportedPage page = writer.GetImportedPage(reader, 1); //page #1
    PdfDictionary pageDict = reader.GetPageN(1);
    pageDict.Put(PdfName.PRINTSCALING, new PdfNumber((float)scalef));

    float yPos = reader.GetPageSize(1).Height - (reader.GetPageSize(1).Height * (float)scalef);
    writer.DirectContent.AddTemplate(page, (float)scalef, 0, 0, (float)scalef, 0, yPos);

    doc.NewPage();

    doc.Close();
    reader.Close();
    writer.Close();
}
```

Figura 69 - Alterar escala do documento

```
scale_doc = Math.Round(Convert.ToDecimal((double)p1_dig.Y / 842), 2);
scale_doc += 0.03m;

if (scale_doc == 0.99m)
    scale_doc = 1.00m;

Console.WriteLine($"calculate scale {scale_doc}");

if (p1_39_dig.Y == 10)
    p1_39_dig.Y = 11;

if (scale_doc != 1.00m)
{
    int x_scale = Convert.ToInt16(x_diff_or * scale_doc);
    int x_39_scale = Convert.ToInt16(x_39_diff_or * scale_doc);
    int y_scale = Convert.ToInt16(y_diff_or * scale_doc);

    p1_dig.X += 2;
    p2_dig.X = Convert.ToInt16(p1_dig.X + x_scale);
    p2_dig.Y = Convert.ToInt16(p2_dig.Y + y_scale - y_diff_or);
    p2_39_dig.X = Convert.ToInt16(p1_39_dig.X + x_39_scale);
    p2_39_dig.Y = Convert.ToInt16(p2_39_dig.Y + y_scale - y_diff_or + 1);
}
```

Figura 70 - Cálculo escala do documento

```
if (scale_doc >= 1.00f)
    intersection = new Point(res_x, res_y);
else
{
    int n_x = Convert.ToInt16(res_x * (double)w / bmp.Width);
    int n_y = Convert.ToInt16(res_y * (double)h / bmp.Height);
    int s_x = Convert.ToInt16(n_x * scale_doc);
    int s_y = Convert.ToInt16(n_y * scale_doc);

    int new_x = Convert.ToInt16(s_x * (double)bmp.Width/w);
    int new_y = Convert.ToInt16(s_y * (double)bmp.Height/h);

    intersection = new Point(new_x, new_y);
}
```

Figura 71 - Adaptar cálculo posições