U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

MSc

2.º CICLO

FCUP

Extração Automática de Texto em Imagem/Vídeo

Pedro Cunha Travassos

Dissertação de Mestrado apresentada à
Faculdade de Ciências da Universidade do Porto em
Engenharia Matemática
2018

U.PORTO

Extração Automática de Texto em Imagem/Vídeo

Pedro Cunha Travassos

FC

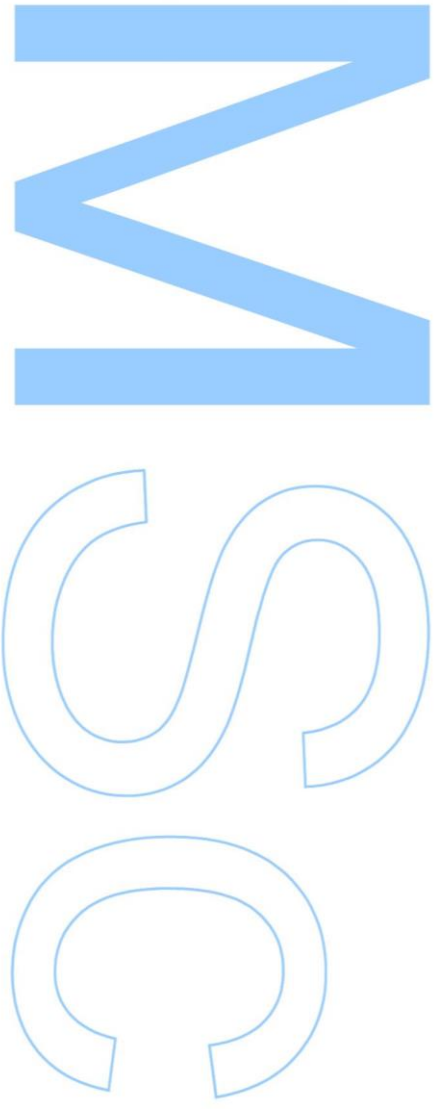
U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Extração Automática de Texto em Imagem/Vídeo

Pedro Cunha Travassos
Mestrado em Engenharia Matemática
Departamento de Matemática
2018

Orientador
André Ribeiro da Silva de Almeida Marçal, Professor, Faculdade de Ciências da Universidade do Porto



U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Agradecimentos

Queria deixar os meus agradecimentos ao Professor André Marçal pela orientação, ajuda e atenção neste projeto. Manifesto também a minha gratidão à Professora Cristina Caridade e à Professora Teresa Mendonça pela disponibilidade, avaliação e recomendações como membros do júri para a prova de mestrado.

Muito obrigado à Amélia e ao José por todo o apoio. À família por toda a força. Aos camaradas, taberneiros e companheiros.

A ti.

Resumo

A revolução informática tem provocado, nas últimas décadas, um grande desenvolvimento no reconhecimento ótico de caracteres. A extração de texto em alfabeto latino impressos é considerado um problema resolvido, com vários sistemas com resultados de eficácia acima do 99%. No entanto, no campo da escrita à mão e de escritas contínuas, como a árabe, ainda há muito trabalho a ser feito para resultados satisfatórios. O OCR em vídeo também representa um desafio com novos obstáculos.

Este trabalho de tese iniciou-se com a construção de um sistema OCR de raiz para imagem em texto, o qual revelou resultados entre os 97% e os 99% de exatidão, para imagens com alta definição, embora a redução desta piore consideravelmente o desempenho do sistema. O programa foi também aplicado a casos de vídeo, com o objetivo de reconhecer texto em movimento em rodapés de notícias e créditos de filmes. Aliado a um processo de extração de imagens consoante o movimento, conseguiu resultados de exatidão à volta dos 97%.

Palavras chave: OCR, imagem, texto, vídeo, caracteres, rodapé, créditos.

Abstract

In the last decades, the revolution in computing science has inflicted a huge development on optical character recognition (OCR). The extraction of text in printed latin alphabet is considered a closed case, with many applications having an accuracy over 99%. However, in handwritten and connected scripts, like in arabic, there is still a lot to be done in order to achieve satisfying results. Video OCR also represents a new challenge with added difficulties.

In this dissertation, we started by building an OCR system from scratch, with the objective of recognizing text in image, with an accuracy varying between 97% and 99%, for high definition images. However, there is a considerable reduction of performance for lower resolutions. We also applied the system to video cases, with the purpose of recognizing moving text in news reels and film credits. Allied to an image extraction process dependent on the characters movement, we were able to get accuracies rates around 97%.

Keywords: OCR, image, text, video, characters, newsreel, credits.

Índice

1 - Introdução 13
2 - Métodos de reconhecimento ótico de texto 17
2.1 - Biblioteca *template* e técnicas de classificação. 17
2.2 - Pré-processamento da imagem input 19
2.3 - Pós-processamento dos resultados 22
2.4 - Caligrafia e outras escritas unidas. 23
2.5 - Reconhecimento de texto em vídeo. 24
2.6 - Métodos de avaliação. 27
3 - Método experimental utilizado. 33
3.1 - Descrição do método base. 33
3.2 - Biblioteca de caracteres. 33
3.3 - Preparação da imagem a analisar. 36
3.4 - Análise e classificação. 36
3.5 - Pós-processamento dos resultados. 37
3.6 - Teste do programa. 42
3.7 - Aplicação em vídeo 42
3.7.1 – Rodapé de canais noticiosos 43
3.7.2 – Créditos finais de filmes 48
3.8 – Fluxogramas 51
4 – Resultados e Discussão. 55
4.1 – Resultados para texto estático 55
4.1.1 – Resultados para Arial. 56
4.1.2 – Resultados para Times New Roman 60
4.2 – Resultados para vídeo 61
4.2.1 – Resultados para rodapé de canais noticiosos 61
4.2.2 – Resultados para créditos finais de filmes 66
5 – Conclusão. 71
6 - Referências. 73
6.1 – Referências de dados. 76

Lista de quadros e de figuras

Figura 1 – Fluxograma de um sistema OCR usual 15
Figura 2 – Exemplo de normalização da letra H numa matriz 5x5. 18
Figura 3 – Exemplos de letras conectadas entre elas em textos degradados. 20
Figura 4 – Exemplo de binarização por etapas 21
Figura 5 – Exemplos de separação de blocos de texto em jornais 22
Figura 6 – Exemplo de segmentação de uma palavra em árabe. 23
Figura 7 – Exemplo da melhor sequência entre as possibilidades de palavras, por modelos de Markov. 24
Figura 8 – Exemplo de texto de cena e texto artificial 25
Figura 9 – Créditos finais em movimento vertical 26
Figura 10 – Rodapé na base da imagem em movimento horizontal 26
Figura 11 – Imagem composta por frames, não interrompendo o rodapé 27
Figura 12 – Exemplo de edições entre duas sequências. 29
Figura 13 – Imagem para formação de biblioteca de tipo Arial 34
Figura 14 – Exemplo de normalização de imagens de caracteres para a biblioteca . . .34
Figura 15 – Exemplos de letras conectadas, presentes no suplemento à biblioteca . . .35
Figura 16 – Exemplo de comparação matricial de um caractere com elementos similares da biblioteca. 37
Figura 17 – Fluxograma de decisões sobre o “bloco branco” 38
Figura 18 – Fluxograma de decisões sobre “O/0/o” 39
Figura 19 – Fluxograma de decisões sobre o “1/l/i 40
Figura 20 – Impressão final após a aplicação do OCR à imagem 41
Figura 21 – Exemplos de rodapés em canais noticiosos. 43
Figura 22 – Imagens de original a binarizadas sobreposta indicando onde há movimento, na faixa branca. 44
Figura 23 – Exemplo do cálculo da diferença de coordenadas entre o mesmo objeto em frames diferentes. 45
Figura 24 – Exemplo de cálculo do frame seguinte. 46
Figura 25 – Exemplo do corte final 47
Figura 26 – Esquematização da leitura do OCR em rodapé. 47
Figura 27 – Exemplo de cálculo do frame seguinte 49
Figura 28 – Exemplos de tamanhos diferentes na mesma linha. 50

Figura 29 – Exemplo de dois blocos desalinhados 50
Figura 30 – Exemplo de uma parcela de créditos finais unidos, para leitura do OCR . 51
Figura 31 – Fluxograma do programa OCR base 52
Figura 32 – Fluxograma da variação aplicada a texto móvel em vídeo. 53
Figura 33 – Amostras de vídeos testados com rodapé 61
Figura 34 – Forma da letra “Q” nos vídeos da Fox News. 64
Figura 35 – Exemplo de leitura de caracteres dos créditos de “Gravity” e a respetiva leitura. 68

Lista de tabelas

Tabela 1 – Exemplo de formação de biblioteca segundo características. 35
Tabela 2 – Frequência de letras no Português.. 41
Tabela 3 – Lista de dados para os testes em texto estático. 56
Tabela 4 – Resultados do OCR (Livros 1 a 3) 56
Tabela 5 – Matriz de Confusão parcial (Livros 1 a 3). 57
Tabela 6 – Resultados do OCR (Livros 4 a 6). 58
Tabela 7 – Matriz de Confusão parcial (Livros 4 a 6). 58
Tabela 8 – Resultados do OCR (Livros 7 a 9). 59
Tabela 9 – Matriz de Confusão parcial (Livros 7 a 9). 59
Tabela 10 – Resultados do OCR (Livros 10 e 11). 60
Tabela 11 – Matriz de Confusão parcial (Livros 10 a 11). 60
Tabela 12 – Resultados para o OCR em vídeos da Fox News. 62
Tabela 13 – Matriz de confusão para o 1º teste da Fox News. 63
Tabela 14 – Resultados após atualização da biblioteca. 64
Tabela 15 – Matriz de confusão do novo teste. 64
Tabela 16 – Resultados para o teste da MSNBC e BBC News. 65
Tabela 17 – Matriz de confusão da para o teste da MSNBC e BBC News.. 65
Tabela 18 – Resultados para o teste dos créditos finais em Arial. 66
Tabela 19 – Resultados para o teste dos créditos finais em Arial, por filme. 67
Tabela 20 – Matriz de confusão para o filme 2. 68
Tabela 21 – Matriz de confusão para o filme 10. 69
Tabela 22 – Matriz de confusão para o filme 4. 69
Tabela 23 – Matriz de confusão para Contact.. 70
Tabela 24 – Exemplo de cálculo do frame seguinte. 70

1 - Introdução

Para a pessoa comum, ler e escrever são capacidades triviais. Consegue reconhecer caracteres individuais e compreendê-los dentro de um texto. Está preparada, por exemplo, para redigir uma pequena coleção de artigos impressos para um documento em computador.

No entanto, se a coleção for de tamanho considerável, o processo pode revelar-se moroso e impraticável em tempo útil. Uma alternativa contemporânea será automatizar a tarefa, ou seja, programar uma máquina para a executar. Porém, fazê-lo constitui um novo desafio. É necessário fornecer-lhe a capacidade de reconhecer caracteres, identificá-los e replicá-los num diferente formato. Em suma, é necessário ensinar a máquina a ler.

O reconhecimento ótico de caracteres - ou na denominação inglesa mais familiar *Optical Character Recognition* (OCR) - surge como resposta a este desafio. Esta ferramenta tornou-se indispensável no contexto da era da informação. Não é por acaso que foi usado o termo “máquina” em vez de “computador”, no parágrafo anterior, dado que a primeira aplicação desta ferramenta precede o último: o optofone foi conceptualizado na década de 1910 para codificar texto impresso e traduzi-lo para uma nota musical correspondente de acordo com a letra impressa. O objetivo era fazer com que os cegos pudessem “ler” através da audição [1].

Contudo, a ferramenta como hoje a conhecemos dá o arranque na década de 1940 com o desenvolvimento do computador digital. Em 1954 o Reader’s Digest revela uma máquina OCR que lia relatórios de vendas e passava para cartões perfurados para serem usados em computadores [2]. O desenvolvimento e otimização deste método continuou para as mais variadas situações, mas só nos anos 1980 é que os sistemas OCR começaram a estar disponíveis para um público maior, com a descida de preços dos computadores [2]. As aplicações possíveis de OCR aumentaram: digitalização de manuscritos históricos [3] e jornais antigos, reconhecimento de matrículas e códigos postais, inteligência artificial, scan de documentos, entre outros. A aplicação da ferramenta ao vídeo veio trazer um maior leque de aplicações, bem como de obstáculos próprios da natureza do formato.

Hoje em dia há centenas de programas OCR no mercado, podendo ser corridos em qualquer computador normal. Um dos mais utilizados e famosos é o Tesseract [4]. Foi desenvolvido pela HP entre 1984 e 1995. Dez anos depois foi lançado em *open*

source e desde então foi apadrinhado pela Google, patrocinando o seu desenvolvimento futuro. [4]

Dentro desta lista de diferentes sistemas OCR, existe uma variação de especificidade. Alguns são especializados em resolver um problema em concreto, por exemplo, leitura de letras em moedas antigas [5]. Sendo de aplicação limitada, é esperado que tenha uma alta taxa de sucesso de leitura. Outros programas são mais generalizados, servindo para uma alta variedade de documentos escritos como jornais, livros e manuscritos, sobretudo para textos à máquina. Dentro destes programas, há alguns de nível muito avançado, com acesso a grandes bases de dados e altas capacidades de reconhecimento. Outros são mais "modestos", servindo como apoio a leituras menos exigentes e compatíveis com scanners. Costumam ter algumas configurações adaptativas opcionais, de maneira a melhorar a eficácia em casos especiais [6].

A ferramenta, no entanto, não envolve apenas imagem estática e letras à máquina. A pesquisa do reconhecimento de letra à mão tem progredido nos últimos anos. No entanto, existe uma enorme variação entre caligrafias de diferentes pessoas, em comparação com a uniformidade de cada caractere dentro de uma fonte tipográfica, sendo ainda um problema em estudo [7]. Por outro lado, no campo multimédia, há cada vez um maior esforço em aplicar os mesmos princípios na área do vídeo [8-14].

Apesar de todas estas variantes, qualquer programa OCR essencialmente baseia-se no reconhecimento de padrões, nomeadamente, de caracteres. Para isso, o programa deve primeiro "conhecer" os glifos que quer posteriormente reconhecer, tal como para aprender a ler necessitamos de conhecer o abecedário. Esse conhecimento pode ser uma simples imagem, um conjunto de caraterísticas morfológicas, informações em termos de forma, entre outras, ou, usualmente, uma combinação de todas, à qual se pode chamar de biblioteca. Segue-se a "observação" da imagem. Esta sofre normalmente um pré-processamento para a análise dos glifos. Cada um destes será separado e será comparado à biblioteca de maneira a encontrar a melhor opção. Dado o reconhecimento e classificação, o programa pode devolver logo o *output* final, ou efetuar adicionalmente mudanças contextuais ao resultado.

Podemos fazer um resumo geral e dividir o processo OCR por cinco etapas, como mostra o fluxograma, na figura 1:

1. Obtenção da imagem a analisar, por exemplo a digitalização de um documento;
2. Pré-processamento da imagem, ou seja, prepará-la para a análise;

3. Comparação com a biblioteca e reconhecimento de cada caractere;
4. Pós-processamento dos resultados;
5. Apresentação final dos resultados.

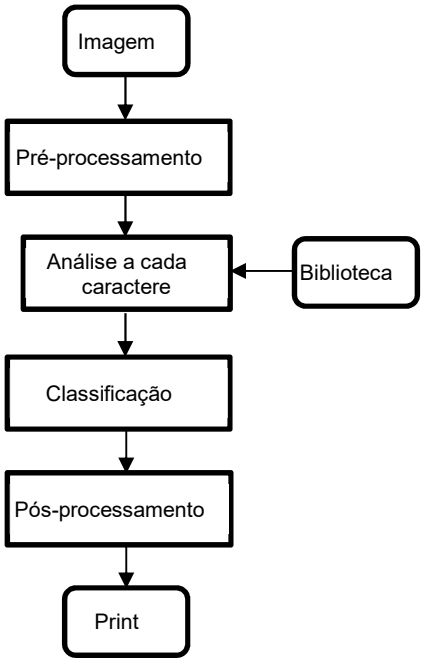


Fig. 1 – Fluxograma de um sistema OCR usual

O tema desta dissertação começa por focar-se na construção de um programa OCR de raiz, tendo como objetivo uma boa eficácia - não só em termos de caracteres alfanuméricos, mas também tendo em conta a pontuação e acentuação. A procura de novas aplicações do programa constitui a fase seguinte.

A primeira situação envolve o rodapé de cabeçalhos de canais de notícias. Hoje em dia é habitual qualquer um destes canais – ou até nos programas noticiosos dos canais mais genéricos – terem um rodapé com texto móvel. A ideia é reconhecer e extrair o texto destes cabeçalhos através do vídeo, algo com uma possível utilidade prática na obtenção de informação sobre, por exemplo, notícias de última hora ou informações sobre economia.

Já com um caminho mais definido no trabalho a fazer, surgiu mais uma ideia, análoga ao caso acima mas com as suas particularidades. Após o final de um filme, usualmente somos presenteados com os créditos finais de todas as pessoas e companhias envolvidas na execução da película. O objetivo, mais uma vez, será passar toda essa informação para texto através da análise do vídeo.

Ambas as abordagens trazem novos desafios. Por definição, um vídeo é uma coleção de imagens sequenciais. Nestes casos, contêm um texto em movimento contínuo uniforme e velocidade constante, ou seja, imagens seguidas do mesmo vídeo vão repetir a mesma porção do texto mas em diferentes posições. O programa OCR deverá saber quando é que deve fazer a leitura e deve saber quando não repetir texto.

Em suma, os objetivos principais desta dissertação são:

1. Construção de um programa OCR de raiz para imagem estática;
2. Teste e melhoramento do mesmo, com especial atenção para acentuação e pontuação;
3. Aplicação para extração de rodapés em vídeos de notícias;
4. Aplicação para extração de créditos finais de filmes.

Segue-se uma revisão bibliográfica mais detalhada do tema, fruto da pesquisa efetuada. Serão abordados diferentes métodos existentes e aplicações gerais e mais específicas.

Posteriormente, é feita a descrição detalhada do método base e algoritmo do programa construído nesta dissertação. Um fluxograma, bem como imagens e outros esquemas apresentam-se para ajudar na compreensão do mesmo. Isto é feito tanto para o método em texto estático bem como para as variações em vídeo.

Para cada caso, são descritos os dados experimentais em termos de formato, resolução e sua origem, mostrando-se alguns exemplos – páginas de documentos, no caso de imagem estática e clips de vídeo, no caso do vídeo. A avaliação do programa é feita comparando os textos output, produzidos pela sua aplicação aos dados, com o texto original, medindo a eficácia consoante o número de caracteres e palavras corretamente identificados. Os erros de identificação serão abordados, dando alguns exemplos de matrizes de confusão, onde se mostra para cada classificação errada, o glifo original. A partir daí fazem-se alguns ajustes ao programa de maneira a melhorar a performance do mesmo e novos testes. Analisa-se o que se pode melhorar e que trabalho futuro pode ser feito neste contexto, sobretudo nas abordagens de identificação de texto em vídeo.

2 - Métodos de reconhecimento ótico de texto

Um sistema de reconhecimento de texto passa por várias etapas, como ilustrado na figura 1. Dada uma imagem *input*, é necessário primeiro prepará-la para a análise. Posteriormente, vem a análise em si com a biblioteca própria do programa, sendo feita a classificação de cada caractere. De seguida, é feito um pós-processamento do resultado e, por fim, a impressão final.

Comece-se por descrever a formação da biblioteca do programa e como é que a análise e decisão do caractere certo será feita, já que é o cerne do programa. O tipo de biblioteca vai depender do tipo de análise comparativa feita ao caractere, pelo que estão interligados.

2.1 - Biblioteca *template* e técnicas de classificação

A formação da biblioteca *template* constitui o processo de aprendizagem do programa. A partir de uma fonte de dados, o sistema obtém um conjunto de informações para cada caractere, informando-se o programa quanto ao glifo correspondente à partida. Quando se der a análise de um caractere de uma imagem *input*, o programa faz a correspondência, comparando com todos os elementos da biblioteca e escolhe a melhor classificação.

Um dos métodos mais usados é a correspondência por imagens matriciais, ou *matrix matching*. A biblioteca, neste caso, é formada por uma imagem template de cada caractere, adaptada a uma matriz com dimensões horizontal e vertical normalizadas. Cada valor da matriz corresponde a um pixel. Normalmente, as imagens matriciais são binarizadas, ou seja, só possuem valores 0 (preto) e 1 (branco). O branco corresponde à letra em si, e o preto ao fundo de imagem da matriz – ou vice-versa.

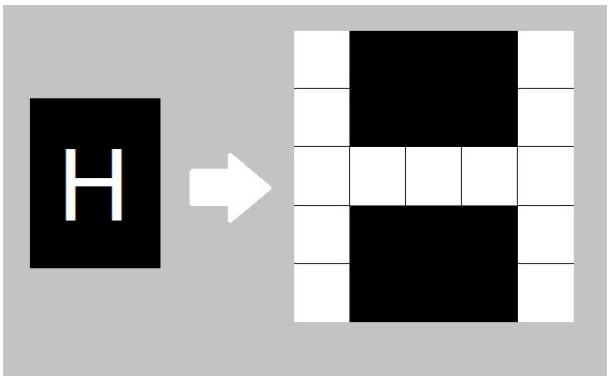


Fig. 2 – Exemplo de normalização da letra H numa matriz 5x5.

Aquando da análise ao caractere a ser lido, este será transformado numa imagem matricial binarizada, com as mesmas dimensões dos caracteres *template*. Após esta normalização (Fig.2), faz-se então a comparação matricial com cada elemento da biblioteca e é escolhido desta o glifo mais semelhante. O critério para esta escolha pode ser uma entre várias medidas de semelhança as quais envolvem medir a distância entre matrizes. Por exemplo, a distância *City-block* (Eq.1) é a soma do módulo da diferença entre os elementos de cada matriz com a mesma coordenada. Como as matrizes são binarizadas, este cálculo representa o número de pixéis diferentes entre as duas matrizes. Há também a distância Euclidiana (Eq.2) que calcula a raiz da soma dos quadrados da diferença entre elementos de cada matriz, e o coeficiente de correlação entre as matrizes (Eq.3) [15], sendo $\mathbf{A}$ e $\mathbf{B}$ duas matrizes com dimensões i x j e com médias de valores $\bar{A}$ e $\bar{B}$, respetivamente.

$$d_{CB}(\mathbf{A},\mathbf{B}) = \sum_{i=1}^{n}\sum_{j=1}^{n}\left|A_{ij} - B_{ij}\right| \tag{1}$$

$$d_{E}(\mathbf{A},\mathbf{B}) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(A_{ij} - B_{ij})^2} \tag{2}$$

$$r = \frac{\sum_i\sum_j\left(A_{ij} - \bar{A}\right)\left(B_{ij} - \bar{B}\right)}{\sqrt{\left(\sum_i\sum_j\left(A_{ij} - \bar{A}\right)^2\right)\left(\sum_i\sum_j\left(B_{ij} - \bar{B}\right)^2\right)}} \tag{3}$$

Esta abordagem é boa para comparar letras de máquina, preferencialmente do mesmo tipo ou próximas. Se o tipo for suficientemente diferente, a diferença de pixéis pode aumentar demasiado para resultados satisfatórios. Para escrita à mão, com demasiadas variantes para a mesma letra, este sistema é menos eficaz.

A biblioteca pode basear-se, em vez de imagens, em informações morfológicas e em características próprias de cada caractere. Estas podem passar pela curvatura, número de buracos interiores, relação entre altura e largura, número de cruzamentos e bifurcações, concavidades, traçados, cantos, etc. A classificação do programa é feita por um sistema de regras de decisão pré-definido ou pela vizinhança mais próxima em termos de características [6,16]. Por vezes, aliam-se as duas abordagens, ou seja, há uma comparação com uma biblioteca de caracteres, mas só com os que partilham uma certa característica morfológica.

Há ferramentas adicionais para minimizar más classificações. A biblioteca pode ser formada via treino menos supervisionado através de *Neural Networks*. Estas tentam replicar o processo das estruturas neurais biológicas através da análise de grandes quantidades de dados. O programa retira as informações de um grande número de variantes de cada caractere e constrói uma grande coleção de informação [17-20]**.**

2.2 - Pré-processamento da imagem input

A imagem *input*, antes de ser analisada, precisa de ser preparada para o efeito. Há um pré-processamento necessário que pode ser feito com supervisão humana ou embutido dentro do próprio programa. Em primeiro lugar, é preciso abordar os defeitos ou a baixa qualidade que a imagem possa ter. O caso mais clássico são documentos antigos e degradados, com falhas na impressão que alteram a forma dos caracteres ou com ruído na imagem. Podem também alterar a individualidade do caractere, isto é, dois ou mais glifos estarem unidos uns aos outros quando deveriam estar separados, ou um glifo estar desconectado entre si. A figura 3 mostra alguns exemplos deste tipo de problemas.

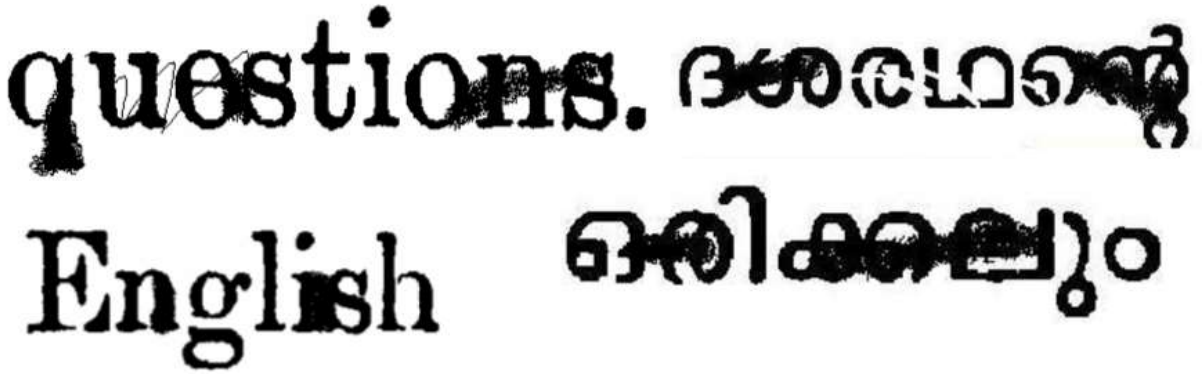


Fig. 3 – Exemplos de letras conectadas e desconectadas entre si em textos degradados. [21]

Os caracteres podem também ser muito finos ou muito grossos, podendo haver a necessidade de alterar o seu tamanho e espessura. Baixas resoluções ampliam ainda mais estes problemas, principalmente abaixo dos 200 *dpi* (*dots per inch*, pontos por polegada) [6].

Um bom programa OCR sabe prever algumas situações que possam afetar a leitura dos documentos a ser analisados. Por exemplo, corrigir deformações e degradações dos documentos, melhorando a qualidade da imagem para uma melhor leitura do texto, bem como a aplicação de filtros especializados em reduzir ou eliminar ruido nas imagens. A correção e melhoramento já pode ser feita automaticamente, com programas a medir que tipo de falhas possui o texto e que filtros e operações aplicar. Alguns exemplos são a separação de linhas que se possam tocar, habitual em documentos de máquinas de escrever antigas; encher buracos não esperados dentro das letras; ou interpolação para aumentar a qualidade dos caracteres. A aplicação deve ser cuidada, pois a solução para um tipo de defeito pode significar o agravamento de outro [22,23].

Ainda dentro do pré-processamento e feita a correção de defeitos, vem então a binarização da imagem. Letras e fundo terão cores diferentes (preto e branco) de maneira a diferenciá-los. Aplica-se um processo de limiarização, isto é, converte-se a imagem para *grayscale* (tons de cinzento) e passam-se os cinzentos mais escuros para preto e os mais claros para branco. Devido à diferença de contrastes e luminosidade que pode haver numa imagem, a limiarização pode ser feito localmente, mas convém sempre haver um claro contraste inicial entre os caracteres e o fundo. A figura 4 mostra um exemplo de binarização de uma imagem a cores, com texto.

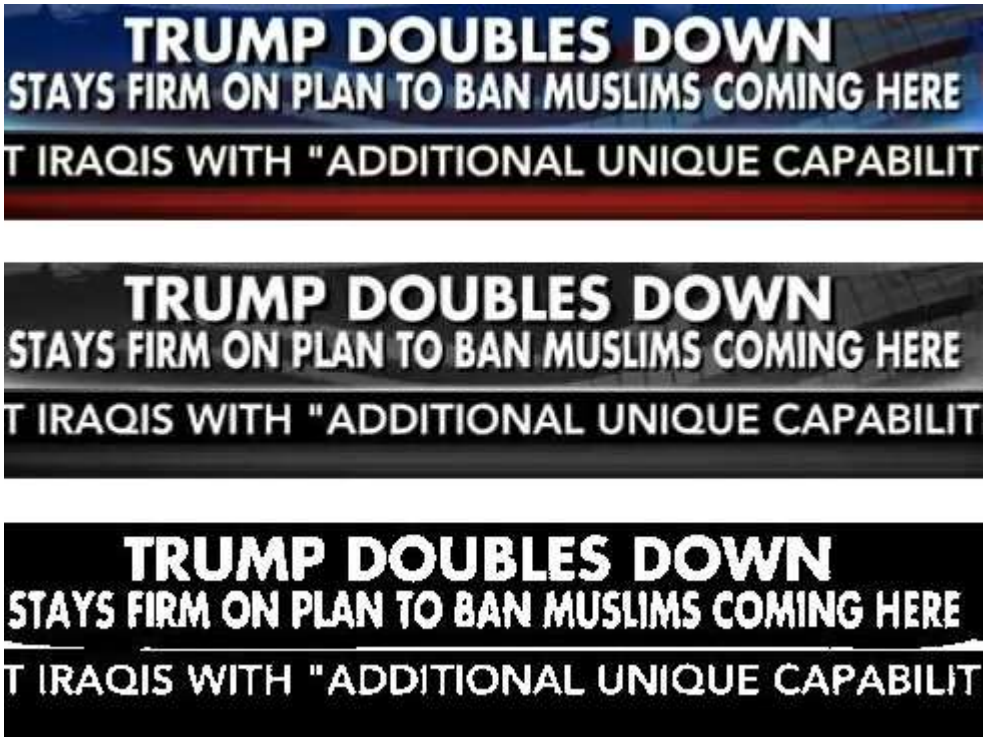


Fig. 4 – Exemplo de binarização por etapas; em cima, a imagem original; no meio a imagem em *grayscale*; em baixo, a imagem binarizada [45]

Segue-se a segmentação da imagem em diferentes secções. O objetivo final desta etapa é ter uma imagem binarizada separada de cada caractere para se poder então fazer a comparação com a biblioteca ou, em alternativa, recolher as informações morfológicas necessárias para a decisão. Porém, costuma ser necessário a segmentação em várias fases, com diferentes ordens de grandeza.

Primeiro, caso necessário, identificar as zonas de texto da imagem. Esta pode conter áreas com figuras, gráficos, símbolos, fotografias, etc… áreas sem texto e, portanto, sem propósito para o uso do OCR. Além disso, a imagem pode possuir diferentes blocos de texto, independentes uns dos outros, sendo imperativa a separação e a colocação por ordem desses blocos. Artigos científicos e jornais são dois exemplos onde esta operação é necessária [6,24,25], como mostra o exemplo da figura 5. É feita também a deteção de linhas (esta até pode ser usada para identificar os próprios blocos), através das coordenadas dos caracteres e agrupamentos destas. Em alternativa, podem ser utilizadas transformadas como a de Hough, sendo esta capaz igualmente de detetar se a linha faz um ângulo com a horizontal [26,27]. A separação

por linhas é importante para o programa impor a ordem dos caracteres, da esquerda para a direita de cada linha, começando pela de cima até à de baixo.[1]

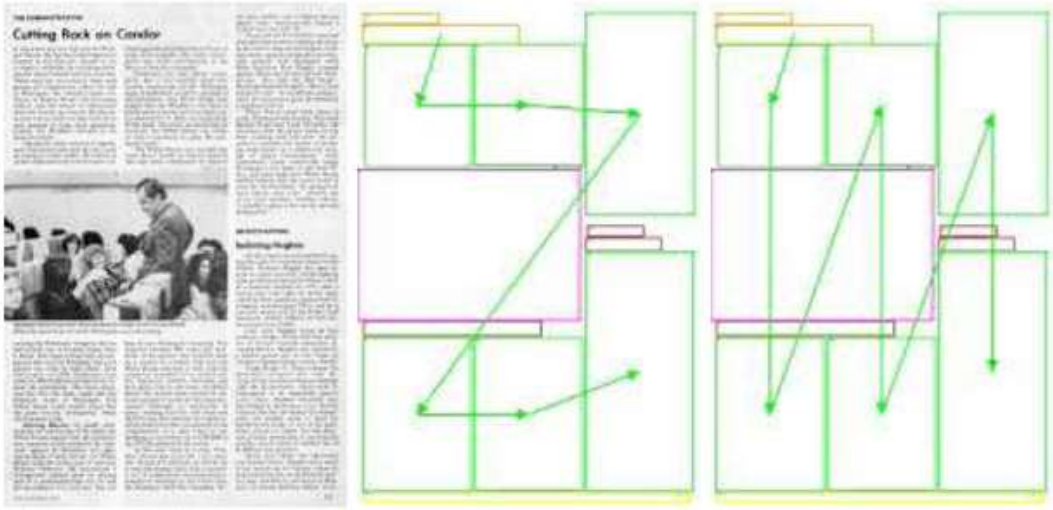


Fig. 5 – Exemplos de separação de blocos de texto em jornais [6].

Finalmente, chega a parte da separação de cada caractere. Para cada um dos objetos da imagem é feita uma imagem da sua *Bounding Box*, isto é, o menor retângulo que a contém. Considera-se que cada grupo de pixéis brancos (portanto, que fazem parte da letra) conectados constituem um objeto da imagem. Idealmente, cada caractere constitui um objeto individual. Caso o algoritmo se baseie em comparação de informações morfológicas, estas são retiradas para cada caractere individualmente.

2.3 - Pós-processamento dos resultados

Feita a segmentação, segue-se a classificação da letra pelos métodos já descritos na secção anterior. Mas o processo não acaba aqui. Obter o caractere correto a partir da leitura do OCR nem sempre passa por apenas escolher o elemento da biblioteca mais próximo. Há glifos quase ou mesmo idênticos morfologicamente, como a letra “o” e o algarismo “0”. Outros são compostos por mais que um elemento conectado, como o símbolo “%” ou a letra minúscula “i”. O próprio ponto desta letra é idêntico ao ponto final, e vírgulas podem ser confundidas com acentos e vice-versa. O próprio redimensionamento normalizado de cada caractere na comparação por imagens matriciais não, tem por si só, a informação do real tamanho relativo ou da posição de

[1] De notar que outras escritas seguem ordens diferentes, como o árabe da direita para a esquerda.

cada caractere, pelo que até um “l” pode ser confundido com um simples ponto final [28]. Há então a necessidade de uma revisão contextual a cada caractere para verificar se a sua presença faz sentido onde está. Por exemplo, se o programa nos devolve “1O92”, é muito provável que o “O” seja o algarismo “0”. Um bom sistema OCR saberá que a letra não faz sentido naquele contexto e alterará para o algarismo de forma idêntica. Outra alternativa é usar verificadores de erros, *spellcheckers*, para a correção de resultados incorretos [29,30].

2.4 - Caligrafia e outras escritas unidas

O OCR focou-se inicialmente na escrita do alfabeto latino à máquina. Neste campo, muitos sistemas já têm grandes níveis de eficácia, acima do 99%, e já se pode considerar, em grande parte, um problema resolvido. Porém, outros sistemas de escrita apresentam novas dificuldades. Alguns, mesmo na sua versão à máquina, possuem os seus caracteres conectados dentro da mesma palavra, exigindo um processo de segmentação mais profundo do que existe no alfabeto latino que, por norma, possui caracteres separados. A escrita em árabe e em bengali são dois exemplos onde tem havido muito desenvolvimento, mas a separação ainda não é perfeita. No caso do árabe, há abordagens onde, em vez separação direta de caracteres, se efetua a deteção de características morfológicas dentro de cada palavra de maneira a identificar dentro dela os glifos. [8,31,32] Outras tentam analisar as projeções horizontal e vertical de cada palavra para localizar as zonas de corte, processo exemplificado na figura 6.

Fig. 6 – Exemplo de segmentação de uma palavra em árabe, com histogramas das projeções horizontal e vertical [33]

De modo semelhante, a caligrafia apresenta-se com os caracteres unidos entre si dentro da mesma palavra. Porém, junta mais um grande obstáculo: a alta variação dentro do mesmo caractere, já que cada pessoa tem a sua escrita própria. A construção de um OCR para caligrafia com alta eficácia continua a ser um problema em estudo.

A aplicação de Cadeias de Markov Ocultas (*Hidden Markov Chains*) tem feito avanços neste ramo [34]. Através delas, é possível montar um modelo de Markov que, face a um leque de possibilidades quando se observa uma sequência de palavras, compara com um dicionário já definido para tentar ajudar a decidir a frase mais óbvia. Em alternativa, pode-se fazer a segmentação por caracteres e escolher a melhor sequência de glifos dentro das possibilidades [7,34]. A figura 7 mostra um exemplo desse processo.

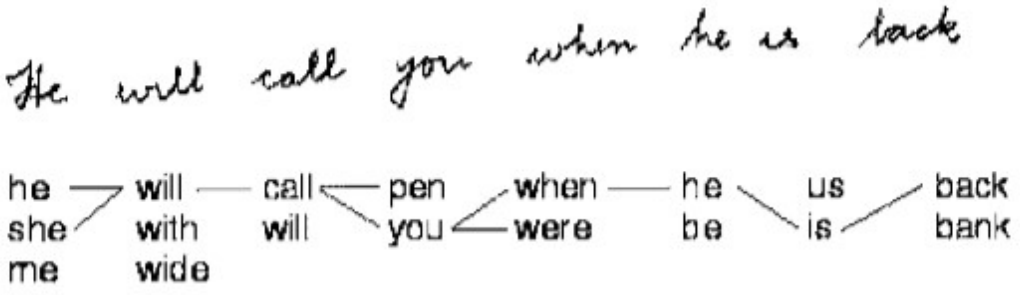


Fig. 7 – Exemplo da melhor sequência entre as possibilidades de palavras, por modelos de Markov [7]

2.5 - Reconhecimento de texto em vídeo

Da imagem estática, o OCR passou a fronteira do multimédia. O vídeo tem sido alvo nos últimos anos de novos desenvolvimentos na tecnologia de extração de texto. As aplicações podem variar, e com elas, variam as características de sistema de Video OCR (VOCR).

Podem-se definir dois grupos diferentes de texto em vídeo: de cena e artificial. Texto de cena é o texto que faz parte do local onde a filmagem é feita - por exemplo, numa filmagem de uma rua, nomes de lojas em toldos ou tabuletas de informação são texto de cena. Texto artificial é texto adicionado ao vídeo - por exemplo, a data e horas em filmes de câmaras de filmar, legendas de séries e filmes, cabeçalhos de notícias num noticiário, entre outros. Este tipo de texto costuma estar alinhado horizontalmente, ter um tipo de letra convencional e ter cores que contrastam com o fundo, normalmente

mais claras ou brancas. Já o texto de cena depende sempre do ângulo de câmara [9,10]. A figura 8 mostra uma imagem com os dois tipos de texto, retirada de um vídeo.

Fig. 8 – Exemplo de texto de cena (a verde) e texto artificial (a vermelho) [11]

O VOCR irá depender do tipo de texto que se quer extrair. Se o objetivo for tirar referências, por exemplo, do local filmado, pode fazer sentido a análise do texto de cena. A sua extração pode ser complicada, já que a disposição da câmara pode dificultar a análise. Um texto desalinhado, por exemplo, pode precisar de uma rotação. Pode também não ter dimensões adequadas, dependendo da distância da câmara. Dependerá também sempre da qualidade do vídeo, a qual também afeta a extração de texto artificial. No entanto, este por estar mais evidente apresenta, em princípio, menos dificuldades. Em ambos os casos, é importante uma boa diferenciação do fundo [12].

Existem várias razões para se recorrer ao VOCR. Como alguns exemplos, temos: programas de informação, com texto passível de ser extraído nos seus cabeçalhos, tanto os principais como os de rodapé; canais de economia, com informação sobre subidas e quedas de ações; filmes e séries, com legendas, títulos e créditos finais, onde há muita informação acerca de quem trabalhou no filme. Há ainda utilidade do VOCR na indexação de vídeos, onde o texto extraído pode revelar tópicos de interesse e outras informações.

O texto artificial aparece sobretudo estático, na mesma posição, durante um certo número de *frames*. Pode dar-se também o caso de o texto se mover com velocidade constante. O sentido do movimento pode ser vertical, ilustrado na figura 9, ou horizontal, como mostra a figura 10.

Fig. 9 – Créditos finais em movimento vertical [46].

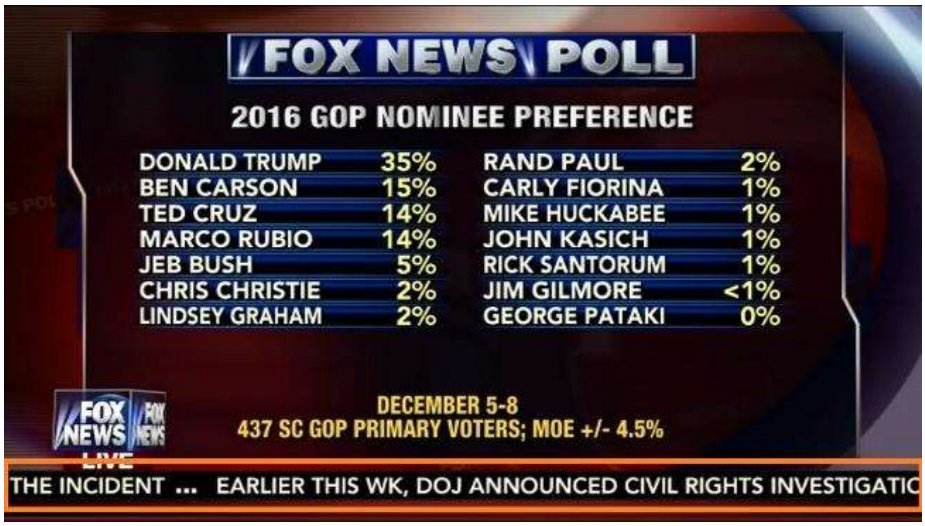


Fig. 10 – Rodapé na base da imagem (a laranja) em movimento horizontal [45].

Há sistemas VOCR já preparados para a leitura, mas onde a escolha da imagem a analisar é supervisionada. Outros analisam todos os *frames* e ignoram resultados repetidos, o que pode ser impraticável e demorado. No entanto, esta análise pode ser usada para aumentar a qualidade do texto a ser extraído, se este não mudar de sítio. Em vez de fazer uma leitura de uma imagem, sobrepõem-se primeiro os vários *frames* onde o texto alvo aparece. Como as letras, em princípio, estão no mesmo sítio mas o resto da imagem move-se, cria-se um melhor contorno à volta delas [13].

Há a possibilidade de se captar *frames* respeitando um intervalo. Quanto maior este, menos imagens a analisar e menos repetições, mas maior o risco de ficar texto por ler.

No caso mais específico do texto em movimento, pode-se estabelecer um intervalo entre captação de *frames* conjugado com a velocidade dos caracteres e fazer

uma leitura de todo o texto, com o menor número de captações possível. O cálculo da velocidade pode-se fazer pela diferença de distância em pixéis entre os caracteres de uma imagem e os correspondentes de outra imagem, um número determinado de *frames* a seguir. A figura 11 exemplifica a possibilidade de se calcular o intervalo entre *frames* certo e manter a integridade do texto.

Fig. 11 – imagem composta por *frames* (separados a amarelo), não havendo interrupção do texto do rodapé (a laranja) [45].

2.6 – Métodos de Avaliação

Para avaliar o desempenho de um sistema OCR, é feita a comparação entre o resultado da leitura da imagem e, se disponível, a versão desta em texto, chamada neste contexto de referência ou *ground truth*.

Existem duas medidas clássicas para avaliar o desempenho de um sistema OCR: precisão e sensibilidade. A precisão é a percentagem de caracteres corretos no *output* do OCR (Eq. 4). A sensibilidade é a percentagem de caracteres do texto original corretamente identificados pelo OCR (Eq. 5). Devido a possíveis inserções e eliminações, o número de caracteres no *ground truth* nem sempre coincide com o número no *output*. Também é habitual apresentar-se a precisão e a sensibilidade em termos de palavras (Eqs. 6 e 7). É costume aparecerem as quatro medidas em estudos de eficácia de OCRs (se só aparecer uma medida, normalmente refere-se à sensibilidade de caracteres) [6].

$$precisão = \frac{\#\ de\ caracteres\ corretamente\ identificados}{\#\ total\ de\ caracteres\ no\ output\ do\ OCR} \quad (4)$$

$$sensibilidade = \frac{\#\ de\ caracteres\ corretamente\ identificados}{\#\ total\ de\ caracteres\ no\ ground\ truth} \quad (5)$$

$$precisão\_palavras = \frac{\#\ de\ palavras\ corretamente\ identificadas}{\#\ total\ de\ palavras\ no\ output\ do\ OCR} \quad (6)$$

$$sensibilidade\_palavras = \frac{\#\ de\ palavras\ corretamente\ identificadas}{\#\ total\ de\ palavras\ no\ ground\ truth} \quad (7)$$

Por exemplo, suponha-se um *ground truth* com 5000 caracteres e 900 palavras e um *output* de OCR do mesmo documento com 4990 caracteres e 895 palavras, sendo que o número de caracteres corretos são 4950 e as palavras corretas 870. Os valores de precisão e sensibilidade, em termos de caracteres, serão de 99,20% e 99,00%, respetivamente. Em termos de palavras, serão de 97,21% e 96,67%.

Dado que o número de palavras é menor que o de caracteres, a sua precisão e sensibilidade associados vão ser mais sensíveis. Considera-se uma palavra errada se tiver pelo menos um caractere errado. Ou seja, um só erro "contamina" toda a palavra.

Outras maneiras de avaliar o OCR, como a taxa de erro, partem do número de edições necessárias para transformar um *output* num *ground truth.* É preciso uma métrica adequada que trate os textos como sequências de caracteres e calcule o número de edições necessárias para uma sequência ser transformada na outra. As edições podem ser substituições, inserções, eliminações ou transposições. A escolha da métrica adequada depende do tipo de edições que possuem relevância para o caso. No contexto da avaliação do OCR, o costume é considerar apenas substituições, inserções e eliminações. Usa-se então a distância de Levenshtein (Eq.8), definida como o número mínimo deste tipo de edições que se aplicam numa sequência para ficar igual à outra [35]. Dadas duas sequências a e b, com tamanhos respetivos de i e j, e sendo $1_{(a_i \neq b_j)}$ a função indicadora igual a 1 se a_i e b_j forem diferentes e 0 se forem iguais :

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{se } \min(i,j) = 0 \\ \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{caso contrário} \end{cases} \quad (8)$$

A figura 12 mostra um exemplo de edições entre duas sequências. Neste caso, para transformar a sequência da esquerda na da direita é preciso aplicar uma inserção, uma eliminação e duas substituições. A distância de Levenshtein é, portanto, 4.

```
V <-> V     (correto)
a <-> a     (correto)
m <-> m     (correto)
o <-> o     (correto)
s <-> s     (correto)
  <->       (correto)
c <-> c     (correto)
   -> o     (inserção)
m <-> m     (correto)
  <->       (correto)
a <-> a     (correto)
  <->       (correto)
R <-> R     (correto)
i <-> u     (substituição)
t <-> t     (correto)
a <-> e     (substituição)
  <->       (correto)
p <-> p     (correto)
a <-> a     (correto)
s  <-       (eliminação)
s <-> s     (correto)
s <-> s     (correto)
e <-> e     (correto)
a <-> a     (correto)
r <-> r     (correto)
. <-> .     (correto)
```

Fig. 12 – Exemplo de edições entre duas sequências.

Algumas avaliações incluem erros de transposição, mas é muito raro um programa OCR trocar a ordem de caracteres. Se o fizer, a distância usada será a de Damerau-Levenshtein, que tem em conta este tipo de erro, além dos supracitados [36].

Sabendo o número de edições, é possível calcular a taxa de erro (Eq. 9) [6,37].

$$Erro = \frac{\#\ substituições + \#eliminações + \#inserções\ de\ itens}{\max(\#\ caracteres\ no\ output\ do\ OCR, \#\ caracteres\ no\ ground\ truth)} \quad (9)$$

Em alternativa, em vez de um máximo, pode-se usar ambos os valores de número de caracteres, ou de palavras, de *ground truth* e de *output* (eq. 10-13) para dois tipos de erro.

$$Erro_{op} = \frac{\#\ substituições + \#eliminações + \#inserções\ de\ caracteres}{\#\ caracteres\ no\ output\ do\ OCR} \quad (10)$$

$$Erro_{gt} = \frac{\#\ substituições + \#eliminações + \#inserções\ de\ caracteres}{\#\ caracteres\ no\ ground\ truth} \quad (11)$$

$$Erro_{op\_palavras} = \frac{\#\ substituições + \#eliminações + \#inserções\ de\ palavras}{\#\ palavras\ no\ output\ do\ OCR} \quad (12)$$

$$Erro_{gt\_palavras} = \frac{\#\ substituições + \#eliminações + \#inserções\ de\ palavras}{\#\ palavras\ no\ ground\ truth} \quad (13)$$

O erro em relação ao *ground truth* é uma boa maneira de comparar vários testes à mesma imagem, dando a quantidade de edições necessárias sobre o número total de caracteres original [38]. O erro em relação ao *output* pode ser usado como termo de comparação: se for maior que o erro do *ground truth*, significa que tem um menor número de caracteres, e vice-versa.

A partir dos erros, pode-se calcular a exatidão, ou *accuracy* [38], tanto em relação ao *ground truth* (*Acc* e *Acc_palavras*) como ao *output* (Acc_{op} e Acc_{op}*_palavras*) (Eqs. 14-17).

$$Acc = 1 - Erro_{gt} = 1 - \frac{\#\ substituições + \#eliminações + \#inserções\ de\ caracteres}{\#\ caracteres\ no\ ground\ truth} \quad (14)$$

$$Acc\_palavras = 1 - Erro_{gt\_palavras} = 1 - \frac{\#\ substituições + \#eliminações + \#inserções\ de\ palavras}{\#\ palavras\ no\ ground\ truth} \quad (15)$$

$$Acc_{op} = 1 - Erro_{op}$$
$$= 1 - \frac{\#\ substituições + \#eliminações + \#inserções\ de\ caracteres}{\#\ caracteres\ no\ output\ do\ OCR} \quad (16)$$

$$Acc_{op_{palavras}} = 1 - Erro_{op_{palavras}}$$
$$= 1 - \frac{\#\ substituições + \#eliminações + \#inserções\ de\ palavras}{\#\ palavras\ no\ output\ do\ OCR} \quad (17)$$

3 - Método experimental utilizado

3.1 - Descrição do método base

De entre todos os métodos pesquisados, a opção para a função de reconhecimento de texto recaiu numa fusão entre a análise de propriedades morfológicas com a comparação de imagens matriciais, recorrendo a uma biblioteca de caracteres.

A função criada recebe como input uma imagem com texto alinhado e, para cada caractere, regista o número de buracos e a altura relativa e separa-o numa sub-imagem individual normalizada. De seguida, esta é compara com as sub-imagens dos elementos da biblioteca com a mesma altura e número de buracos. O glifo com a melhor correspondência será o selecionado. Posteriormente, há uma avaliação final para confirmar a escolha ou uma eventual substituição.

Este é o cerne do programa. Como em qualquer código mais complexo, existem muitos pormenores a descrever, bem como três variações do mesmo adaptados a diferentes situações: a mais convencional, direcionada para imagens estáticas, e duas especializadas para situações de vídeo particulares. Destas, a primeira centra-se em ler texto em rodapé de canais de notícias e a segunda foca-se na extração da lista de créditos finais de filmes.

Além da própria leitura do texto, a preparação prévia dos dados a serem estudados é também de grande importância, sobretudo na componente vídeo. Foram incluídas algumas operações de pré-processamento dentro do programa com o objetivo de prever algumas situações que surgiram com os dados utilizados.

Fluxogramas a ilustrar o procedimento dos programas são apresentados no fim da secção, nas figuras 31 e 32. Todo os programas foram construídos e testados no *software* MATLAB [39].

3.2 - Biblioteca de caracteres

A construção da coleção de glifos partiu de uma imagem contendo os caracteres pretendidos para a análise. Selecionaram-se as 27 letras (cedilha incluída) do alfabeto latino, na forma minúscula e maiúscula, os 10 algarismos árabes e 28 símbolos de pontuação, acentuação e outros. Nesta imagem original, a dimensão de uma letra grande (maiúscula e minúsculas altas) é aproximadamente 50X40 pixéis e a de uma

pequena cerca de 40X30. A biblioteca tem várias versões consoante o tipo de letra. Na figura 13, apresenta-se um exemplo da fonte Arial.

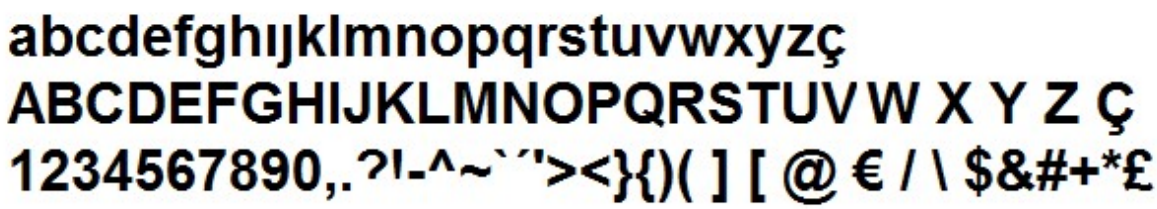


Fig. 13 - Imagem para formação de biblioteca de tipo Arial

A partir de uma imagem destas, são recolhidas as seguintes informações para cada glifo:

- uma imagem matricial, redimensionada para 20x20, da *bounding box* do caractere, o retângulo mais pequeno que contém a letra;
- o número de buracos interiores;
- um parâmetro ligado à altura relativa, formando dois grupos: letras “pequenas” (minúsculas com altura baixa ou menores, ex.: “a”, “c”, x”) e “grandes” (maiúsculas e minúsculas com a mesma altura, algarismos, etc. ex.: “A”, “2”, “€”).

A figura 14 mostra como é feita a normalização dos caracteres a partir da imagem original. Cada glifo é transformado numa matriz 20X20, com o branco a representar o corpo da letra e o preto a representar o fundo. Os caracteres na imagem input vão receber o mesmo tratamento.

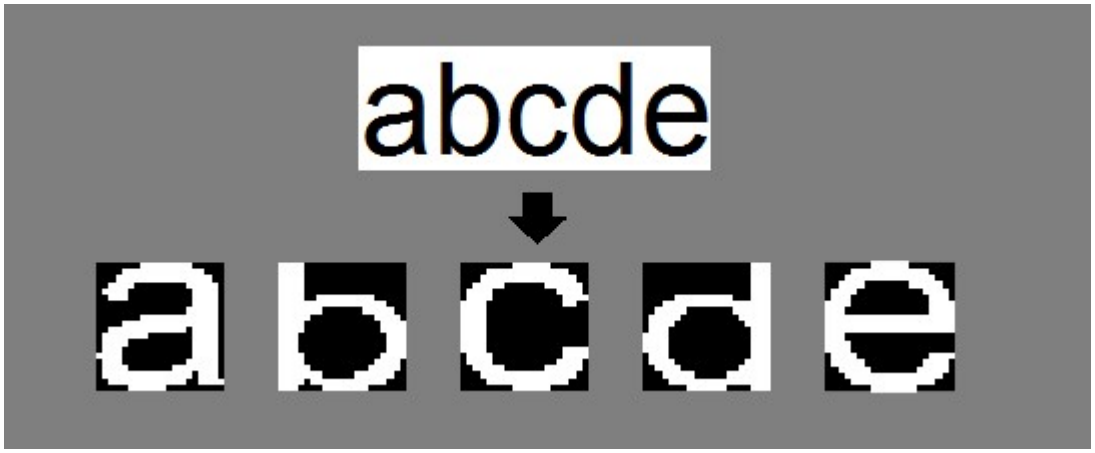


Fig. 14 – Exemplo de normalização de imagens de caracteres para a biblioteca.

As informações sobre altura relativa e o número de buracos servem para dividir a biblioteca em diferentes categorias, como mostra a Tabela 1. Um caractere a ser analisado será apenas comparado com elementos da mesma categoria.

Tabela 1 – Exemplo de formação de biblioteca segundo características

# buracos	Letras pequenas	Letras grandes
0	c i m n r s u v w x z , . ? ! - ^ ~ ` ´ ' > < + *	f h k l t C E F G H I J K L M N S T U V W X Y Z Ç 1 2 3 5 7 } {) ( ] [€ / \ £ j y ç
1	a e o	b d A D O P Q R 4 6 9 0 @ # g p q
2		B 8 $ &

O programa está preparado para receber uma imagem de vários tipos de letra de caracteres latinos, desde que apresentados pela ordem da figura 13, e com resolução semelhante. A escolha do tipo de letra para a leitura final deve ser feita previamente. Se houver dúvida em relação ao estilo de letra presente na imagem alvo, haverá um teste prévio: os primeiros dez caracteres serão comparados a vários estilos de letra e o programa escolherá os melhores resultados gerais. O programa contém *templates* já feitos para Arial e Times New Roman.

Há um suplemento ao *template*, caso seja necessário. Mesmo numa imagem com alta qualidade, algumas letras podem aparecer conectadas entre si. O programa lê essas letras como uma só, distorcendo alguns resultados. Então, além das letras, algarismos e símbolos, há a possibilidade de acrescentar à biblioteca alguns pares de letras ligados, funcionando só como um caractere. Acrescenta-se à imagem *template* inicial as imagens desses pares, e indica-se no programa a que caracteres correspondem. Alguns exemplos de letras conectadas apresentam-se na figura 15.

Fig. 15 – Exemplos de letras conectadas, presentes no suplemento à biblioteca.

O uso desta possibilidade e quais os pares de letras a usar serão mais detalhados nos resultados.

3.3 - Preparar a imagem a analisar

Em primeiro lugar, é necessário a binarização da imagem para a obtenção futura dos caracteres individuais. Esta é feita por um processo de limiarização, semelhante ao descrito na secção 2.2. Se na imagem original o fundo for branco e as letras pretas, o programa inverte as cores na binarização final. Caso contrário, deixa a imagem como está.

São efetuadas igualmente operações de maneira a eliminar objetos na imagem muito pequenos, mas tendo em conta pequenos caracteres válidos, como acentos ou pontos. Para o efeito, é feita uma rápida análise da área de cada objeto e faz-se a média de todos. Valores abaixo de um décimo da média serão ignorados. De semelhante modo, orifícios dentro de letras muito pequenos serão apagados, sempre com o cuidado de não interferir com o parâmetro de número de buracos.

3.4 - Análise e classificação

O programa aplica o mesmo método que usou para a formação da biblioteca. A partir dos caracteres da imagem em análise, obtém uma imagem redimensionada normalizada para cada um, bem como o número de buracos e o parâmetro de altura. Além disso, as coordenadas e dimensões da sua Bounding Box serão anotadas para correções subsequentes e para determinação de espaçamento entre letras.

O parâmetro da altura relativa tem mais uma particularidade: caso o texto seja só de maiúsculas, a altura pouco variará entre caracteres. Há então, um rápido teste, em cada linha, à variação da altura dos caracteres (excluindo pontuação e acentuação). Se a variação for muito pequena, quer dizer que temos uma linha só de maiúsculas. Então, a comparação com a biblioteca exclui as minúsculas.

A ordem de leitura é feita por linha, de cima para baixo, e em cada linha, da esquerda para a direita. A cada caractere será verificado o seu número de buracos e altura relativa, sendo então comparado com cada elemento da sub-biblioteca correspondente. Ou seja, um glifo com um buraco e altura baixa, será comparado apenas com as letras, algarismos e símbolos do alfabeto com um buraco[2] e altura baixa.

A comparação é feita com a distância *City-block* (Eq.1) que dá a quantidade de pixéis diferentes entre a matriz do caractere e cada matriz da biblioteca. A diferença

[2] Saliente-se que o número de buracos é um parâmetro opcional. Para imagens mais defeituosas, pode ser contraproducente usar esta característica dado que falhas na imagem podem atribuir mais ou menos buracos a um caractere do que ele realmente tem.

menor será selecionada como o caractere correspondente. Apresenta-se o exemplo de um caractere a ser testado na figura 16.

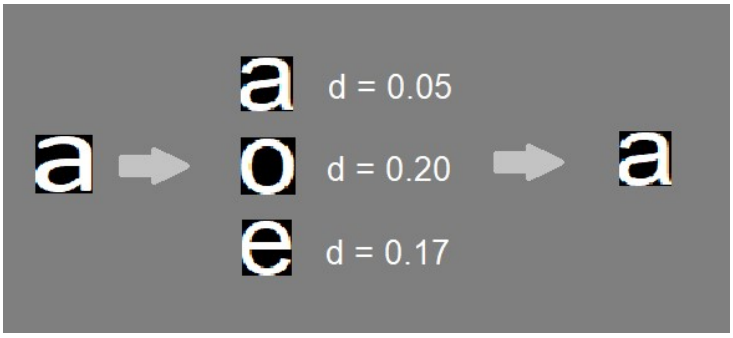


Fig. 16 – Exemplo de comparação matricial de um caractere com elementos similares da biblioteca.

3.5 - Pós-processamento dos resultados

Em certos casos não chega apenas o teste matricial e morfológico. Alguns caracteres dependem de uma revisão contextual para serem bem identificados. Por exemplo, no tipo de letra Arial, face a uma imagem matricial de um caractere, redimensionada toda ou quase toda branca, as hipóteses são várias: pode ser a vogal "i", a consoante "l", um "-" ou até um ponto "." (chame-se a esta matriz "bloco branco"). O programa precisa de mais informação para saber a escolha certa. Por isso, são registadas as coordenadas e dimensões do glifo. Após os resultados, analisam-se estes parâmetros para reconsiderar a escolha. Por vezes, é feita também uma análise da palavra onde o caractere se encontra. Por exemplo, no caso do resultado "bloco branco", são tomadas as seguintes decisões, esquematizadas na figura 17[3]:

[3] Verifica-se que este fluxograma deixa o caso da vogal maiúscula "I" vulnerável no caso de nomes próprios a meio de uma frase. Porém, são raras as ocorrências quando comparadas com a da consoante minúscula "l", pelo que desta maneira se minimizam o número de erros. Uma alternativa será passar um *spellchecker*.

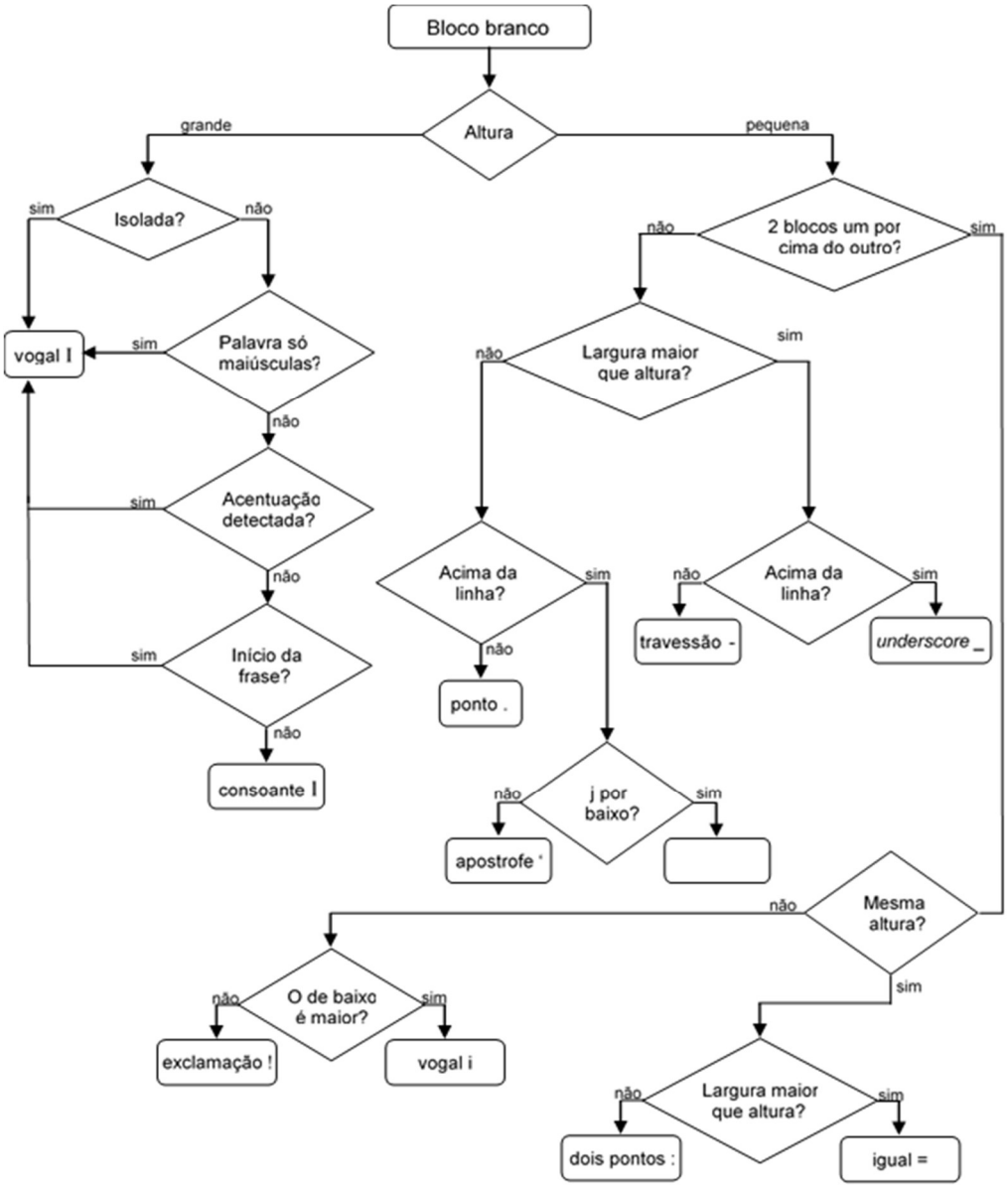


Fig. 17 – Fluxograma de decisões sobre o “bloco branco”.

Outro caso, mais geral para qualquer dos tipos de letra, é o do algarismo “0” e da letra “O”, na figura 18:

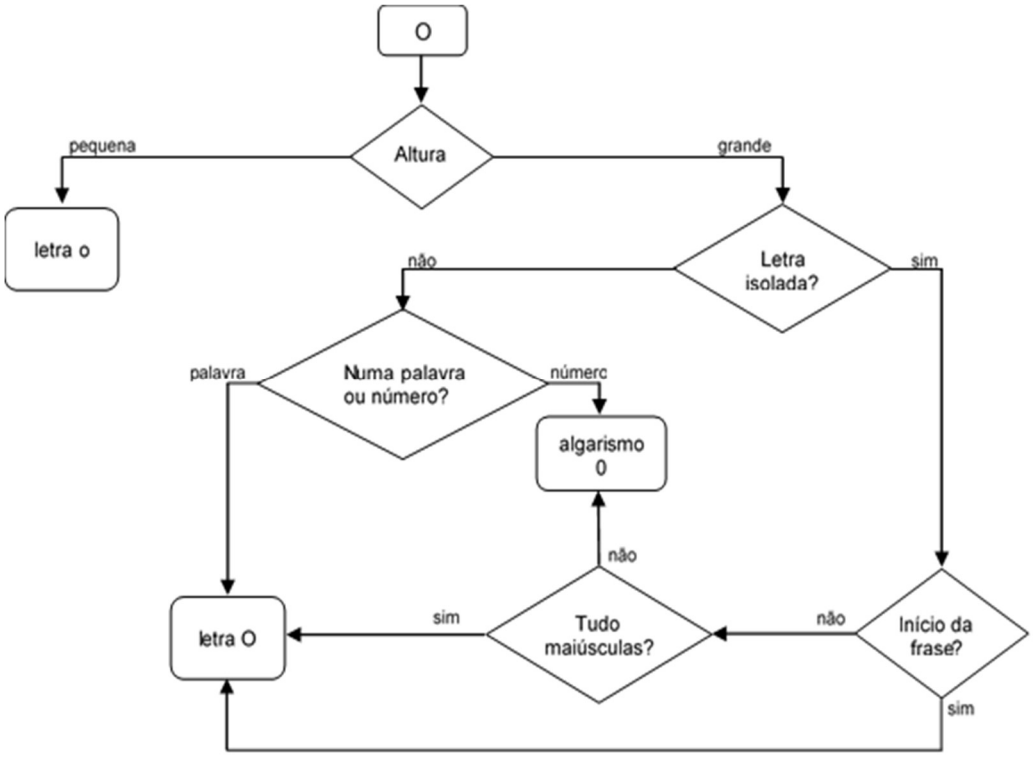


Fig. 18 – Fluxograma de decisões sobre “O/0/o”.

Temos igualmente o caso da acentuação, importante na língua portuguesa. Consideram-se acentos os seguintes: “´”, “`”, “~”, “^”, “¨”. O programa regista se estão numa posição superior da linha e se estão sobre uma vogal. Caso suceda, o caractere será mudado para a sua versão acentuada.

No tipo de letra Times New Roman, a questão do “bloco branco” pouco se verifica porque os caracteres em questão já se confundem menos, devido ao design mais clássico da letra. No entanto, surge similaridade entre a vogal “i”, a consoante “l” e o algarismo “1”. A figura 19 mostra as decisões a fazer num caso destes.

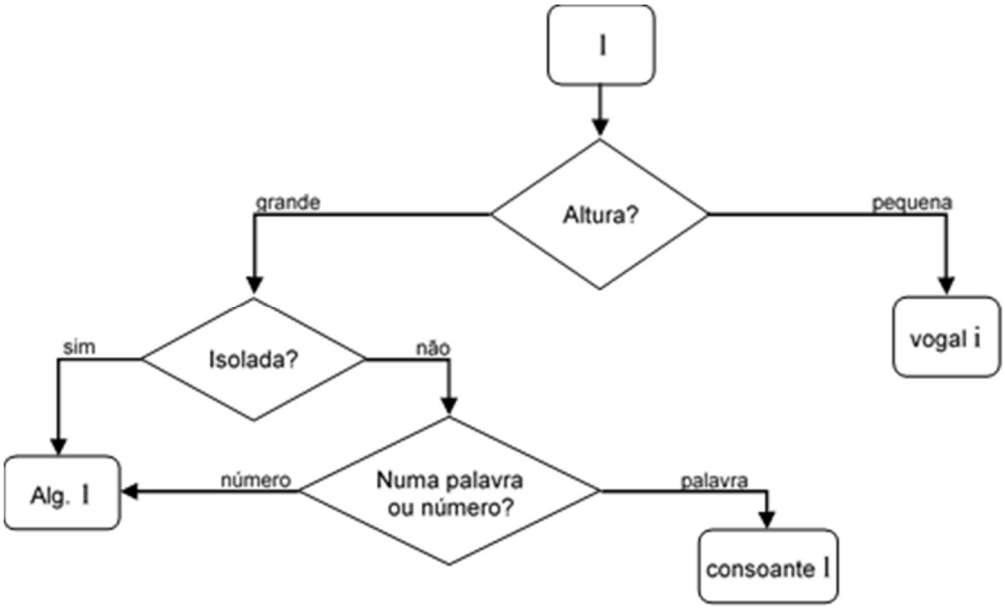


Fig. 19 – Fluxograma de decisões sobre o “1/l/i.

Também haverá uma correção para alguns algarismos em lugares inesperados. Um “5” no meio de uma palavra sem outros algarismos pode-se assumir com segurança que é um muito provável “S”. O mesmo acontece para o “8”, confundido por vezes com o “B”. Caso se deem estas situações, os algarismos serão mudados para as respetivas letras.

Finalmente, dá-se a impressão final num ficheiro em formato txt, ilustrado com um exemplo na figura 20. Cada linha da imagem corresponde a uma linha de texto, acompanhando o alinhamento do texto original. O espaçamento é determinado pela distância entre cada letra. Supõe-se que, numa linha, há mais espaçamentos pequenos – entre letras da mesma palavra – e menos espaçamentos grandes – entre palavras. É calculada a mediana e para valores muito maiores que esta (cerca de 3 vezes maiores pelo menos) corresponde-se um espaço em branco em termos de texto final.

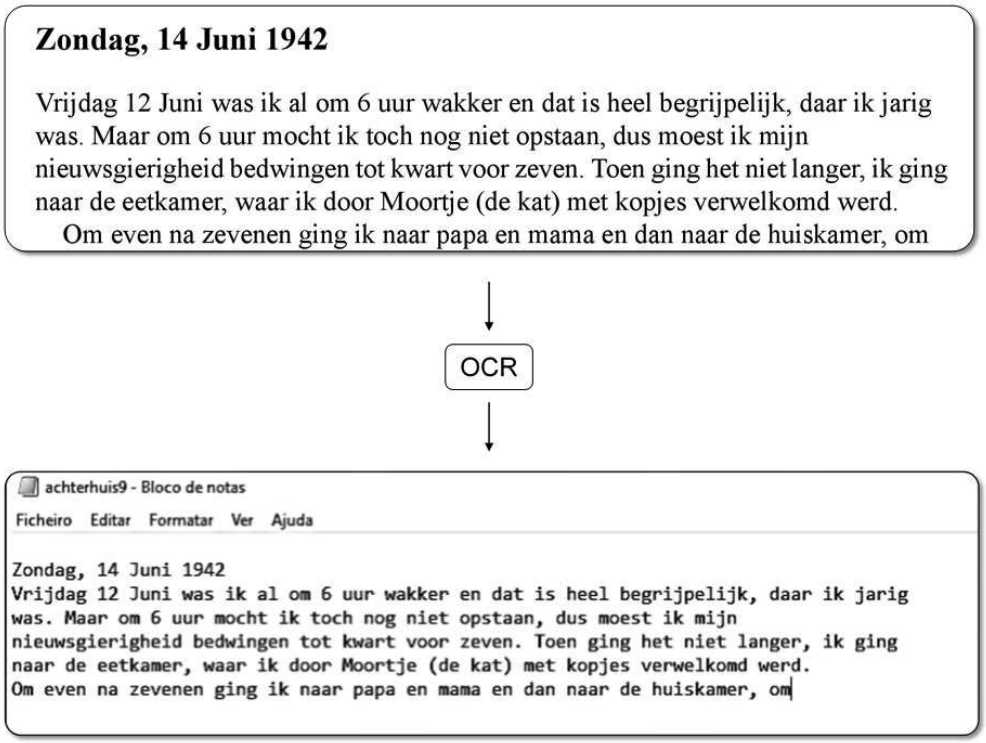


Fig. 20 – Impressão final (baixo) após a aplicação do OCR à imagem (cima).

A aplicação tem uma funcionalidade extra. Além de extrair texto, também faz uma verificação ao idioma. O programa calcula a frequência em percentagem de cada letra do texto e compara com valores de frequência estabelecidos de uma série de idiomas, inseridos no código, devolvendo o resultado mais próximo. Os idiomas que podem ser testados são: Português, Inglês, Castelhano, Alemão, Francês, Italiano, Turco, Sueco, Polaco, Holandês, Dinamarquês, Islandês, Finlandês e Checo [40]. A Tabela 2 apresenta as frequências de letras na língua Portuguesa, como exemplo.

Tabela 2 – frequência de letras no Português

Freq. de letras no Português(%)		**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**
		12,21	1,01	3,35	4,21	13,19	1,07	1,08	1,22
I	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**
5,49	0,30	0,13	3,00	5,07	5,02	10,22	3,01	1,10	6,73
S	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	**À**	**Á**
7,35	5,07	4,46	1,72	0,05	0,28	0,04	0,45	0,04	0,41
Â	**Ã**	**Ç**	**É**	**Ê**	**Í**	**Ó**	**Ô**	**Õ**	**Ú**
0,03	0,83	0,40	0,52	0,36	0,18	0,17	0,01	0,04	0,11

3.6 - Teste do programa

Além do ficheiro em texto, há mais dois outputs: uma string contendo todos os caracteres, incluindo espaços em branco, pela ordem do texto, e uma célula de várias strings, cada uma contendo uma palavra inteira extraída pelo programa. Estes outputs serão usados para testar a eficácia do programa com o *ground truth*.

A versão do texto do *ground truth* será transformada igualmente neste tipo de strings. A comparação é feita através da distância de Levenshtein, isto é, o número mínimo de alterações necessárias – eliminações, inserções e substituições - para uma string ficar igual à outra. Translações serão ignoradas, já que há bastante confiança que o programa não trocará ordens de letras.

A partir da distância de Levenshtein, serão calculadas as taxas de erros, e a partir daí, as exatidões (*accuracies*).

É feita também uma matriz de confusão para explicitar que erros o programa está a fazer, que caracteres ele confunde com outros, de maneira a facilitar possíveis alterações específicas.

A matriz em si é construída através do algoritmo de Smith-Waterman [41]. Partiu-se de uma versão deste algoritmo já construída [42] que escolhe o menor alinhamento entre duas sequências, bem com as alterações – eliminações, inserções e substituições – a ser feitas entre as duas. Neste caso, esperamos que o algoritmo indique um alinhamento perfeito entre a versão de texto extraída e a versão original, dado que o programa de leitura, mesmo que ainda confunda alguns caracteres, consegue ter sucesso na maioria dos casos, na maioria das vezes separados entre eles, logo o desfasamento será muito improvável. O que será útil na versão do algoritmo é precisamente que alterações ele indica, que glifo está a ser confundido com qual. A partir daqui, será mais fácil verificar as falhas do programa OCR.

3.7 - Aplicação em vídeo

São exploradas duas situações de extração de caracteres em vídeo, ambas com texto móvel. No primeiro caso, a partir de um vídeo de um canal de notícia, é feita a leitura do texto em rodapé que aparece, normalmente, na base da imagem (por vezes, aparece no topo). No segundo caso, faz-se a leitura de créditos finais de filme.

Um vídeo, como coleção de imagens que formam uma sequência coerente, requere uma nova abordagem. O programa poderia ler imagens uma a uma. Porém, dado que dois *frames* seguidos pouco mudam entre si, haveria leitura repetida da

mesma porção de texto. Além disso, a aplicação iria tornar-se morosa. Surge então a necessidade de estabelecer um intervalo de tempo entre captação de *frames*. Dado que os casos em discussão possuem um texto móvel a uma velocidade constante, o movimento é usado para determinar quais os *frames* a captar e a usar.

3.7.1- Rodapé de canais noticiosos

Hoje em dia é comum um canal de notícias ter um rodapé na base (ou no topo) da imagem com descrições curtas de notícias de última hora ou de menor interesse, ou informações sobre bolsa, no caso de canais mais especializados em economia. A figura 21 mostra um exemplo do canal BBC News. Geralmente, direcionam-se da direita para a esquerda (ou da esquerda para a direita, no caso por exemplo da escrita arábica) a uma velocidade constante, adequada para permitir uma leitura do telespectador. As letras são, quase sempre, maiúsculas para uma leitura melhor, e com fonte Arial, ou próxima desta.

Fig. 21 – Exemplo de rodapé em canais noticiosos (a laranja) [47].

O programa especializado nestes vídeos alia a aplicação usada para a leitura de texto estático a uma função que determine quais os *frames* do vídeo a serem lidos, baseada esta na velocidade do rodapé. Dado que as letras são maiúsculas, a biblioteca *template* da aplicação não tem as letras minúsculas, desnecessárias, o que traz a vantagem adicional de eliminar muitas possíveis confusões. São eliminados também alguns símbolos desnecessários e será usada a fonte Arial.

Além disso, há um localizador inicial do rodapé na imagem, de maneira a restringir a leitura do texto à parte da imagem que interessa analisar. O localizador analisa o oitavo inferior de uma imagem (este estudo focou-se nos rodapés na base da

imagem, os do topo são raros). Em cada parte, aplica-se uma versão simplificada do programa OCR inicial. A cada objeto da imagem, faz-se a comparação com a biblioteca de caracteres, mas elimina qualquer um cujo melhor resultado seja maior que 25% de pixéis diferentes. O propósito é determinar dentro da imagem onde se encontram linhas com texto e eliminar outro tipo de objetos. Pode haver alguma confusão com cabeçalhos de notícias e, por regra, escolhe-se a linha mais abaixo.

O rodapé por vezes não vai de uma ponta à outra de toda a imagem. Escolhido um *frame* aleatório, é sobreposto aos 10 *frames* seguintes e é analisada a zona do rodapé. A "mancha" indica a zona de movimento real do rodapé.

A figura 22 mostra todo o processo de localização, desde a imagem original, em cima, passando pela localização de linhas de texto e seleção da linha mais abaixo (meio), terminando na determinação dos limites do rodapé (baixo). Neste caso, nem toda a faixa horizontal é preenchida pelo rodapé.

Fig. 22 – Imagem original (cima); linhas de texto localizadas (meio), com a linha de baixo selecionada a verde como o rodapé; imagens binarizadas sobrepostas indicando onde há movimento, na faixa branca (baixo) [47].

Localizado o texto a ler, segue-se o cálculo da sua velocidade (Eq. 18). O princípio será pegar num dos *frames*, fazer a leitura OCR ao seu rodapé, e repetir o processo com o *frame* n-imagens a seguir. Feita a leitura em ambas as imagens, comparam-se os mesmos caracteres em ambas as imagens e calcula-se, através das coordenadas horizontais deles, o quanto avançaram no vídeo em termos de número de pixéis, sendo este intervalo o Δx da equação. A figura 23 mostra um exemplo de cálculo

da velocidade: dadas duas imagens do rodapé separadas por 10 *frames*, é determinada a diferença entre as coordenadas horizontais da letra “H”.

$$v = \frac{\Delta x}{\Delta f} = \frac{\left|x_{f+n} - x_f\right|}{\left|(f+n) - f\right|} = \frac{\left|x_{f+n} - x_f\right|}{|n|}; n \in \mathbb{Z} \tag{18}$$

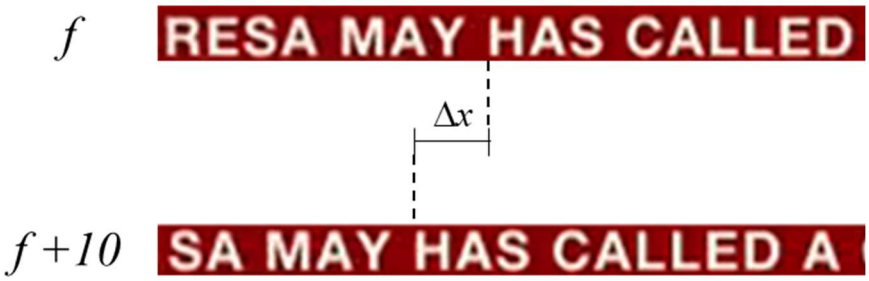


Fig. 23 – Exemplo do cálculo da diferença de coordenadas entre o mesmo objeto em *frames* diferentes. Δx é distância percorrida pela letra “H” entre um *frame* e o décimo a seguir.

Poderá haver alguma discrepância na ordem dos caracteres se, no intervalo escolhido, alguns caracteres detetados na primeira imagem já tiverem desaparecido na segunda. Para evitar esta confusão, usa-se o algoritmo de Smith Waterman para alinhar as duas sequências de caracteres lidos, determinar o desfasamento entre elas, e associar cada glifo da primeira imagem com o seu correspondente na segunda. No caso deste estudo, escolhe-se uma diferença entre 10 *frames*, calcula-se a distância percorrida em pixéis por cada um dos caracteres e escolhe-se a mediana (a variação entre valores de diferença, quanto muito, será muito pequena).

Calculada a velocidade, segue-se a determinação do intervalo entre captações de *frames*. De uma maneira simples, poder-se-ia, tendo o rodapé *l* de comprimento e *v* de velocidade, captar um *frame* a cada *f* = *l*/*v*. Sucede que há o risco de apanhar letras cortadas na margem da área do rodapé. O *frame* seguinte apanha a outra porção do caractere, mas o programa ignora os objetos nas margens. Corta-se então esta porção do rodapé até ao espaço entre palavras mais próximo, calcula-se o comprimento do corte em termos de tamanho em pixéis e, sabendo já a velocidade, altera-se a altura de captação do próximo *frame*. Seja f_n o n-ésimo *frame* a captar. É calculado através da equação 19, em função da captação anterior, f_{n-1}, do respectivo corte de segurança k_{n-1}, do comprimento l do rodapé e da velocidade do texto v.

$$f_1 = 1$$
$$f_2 = f_1 + \frac{l}{v} - \frac{k_1}{v} \tag{19}$$
$$\dots$$
$$f_n = f_{n-1} + \frac{l}{v} - \frac{k_{n-1}}{v}$$

A figura 24 mostra como será calculado a ordem do próximo *frame* a captar. O frame f_i tem uma letra cortada, aplicando-se então um corte k_i, até ao espaço antes da palavra da qual faz parte. Sabendo o comprimento do rodapé l e a velocidade v, é possível calcular a altura do *frame* seguinte.

Suponha-se o exemplo: uma captação do vídeo com comprimento de 1280p e velocidade de 10 pixéis por *frame*. Em princípio, a próxima captação seria o 128º *frame* seguinte (1280p/(10p/*frame*) = 128 *frame*), mas a primeira captação tem uma parte de uma letra na última margem com 30 pixéis de largura. Cortando essa porção, a próxima captação será então o 125º *frame* seguinte ((1280p-30p)/(10p/*frame*) = 125 *frame*).

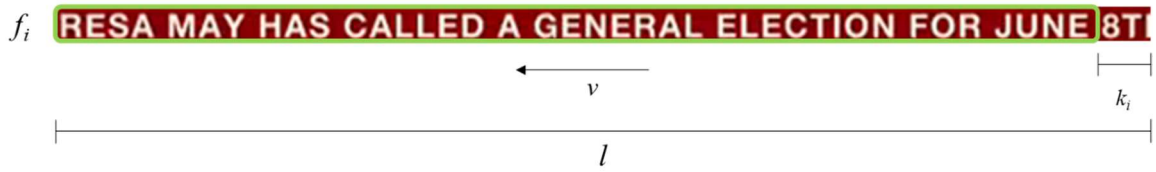


Fig. 24 – O *frame* f_i tem a última palavra cortada (“8T…”) com um comprimento k_i, num rodapé com comprimento *l* e velocidade *v*. Para o cálculo da seguinte captação, aplica-se a eq.15. Neste *frame* o OCR só fará a leitura à zona a verde.

Por fim, é feita a captação do último *frame* do vídeo. O mais provável é não coincidir com o previsto calculado pela captação anterior. Pela equação 20, faz-se a diferença de número de *frames* entre o previsto (f_{fp}) e o real (f_f) e, multiplicando com a velocidade v, sabemos o tamanho em termos de pixéis da porção horizontal que devemos retirar a essa última captação para alinhar bem com o resto do texto. Suponha-se que, com os dados do último exemplo (x = 1280p, v = 10p/f), a última captação é prevista para o *frame* 1000 mas o vídeo só tem 940 *frames*: o corte na última captação será de 10.(1000 – 940) = 600 pixéis. Ou seja, só se considera o texto na última imagem a partir do 600-ésimo pixél horizontal.

$$x_{cf} = v\left(f_{fp} - f_f\right) \tag{20}$$

.

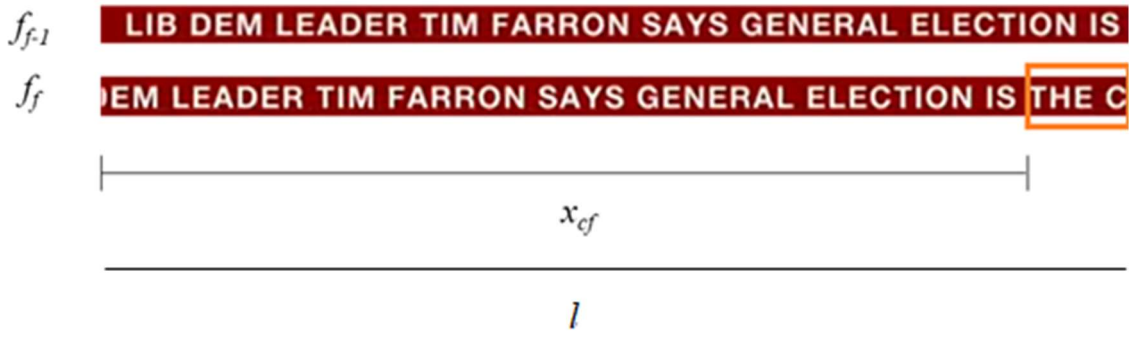


Fig. 25 – Exemplo do corte final x_{cf}; apenas a parte laranja será contada da última imagem. f_{f-1} e f_f são o antepenúltimo e o último *frame* e *l* é o comprimento do rodapé.

A figura 25 ilustra este cálculo. Neste caso, os caracteres finais "THE C" são as letras novas que apareceram no último *frame* em relação ao penúltimo. Pode dizer-se que o comprimento da parte que é lida pelo OCR no último frame, de valor l-x_{cf}, é equivalente à distância percorrida pelas letras do rodapé entre os dois *frames*

Para cada captação, é feita a leitura e no final todas as diferentes porções de texto são unidas num só ficheiro, com o devido espaçamento. A figura 26 mostra como cada captação corresponde a uma linha no texto final.

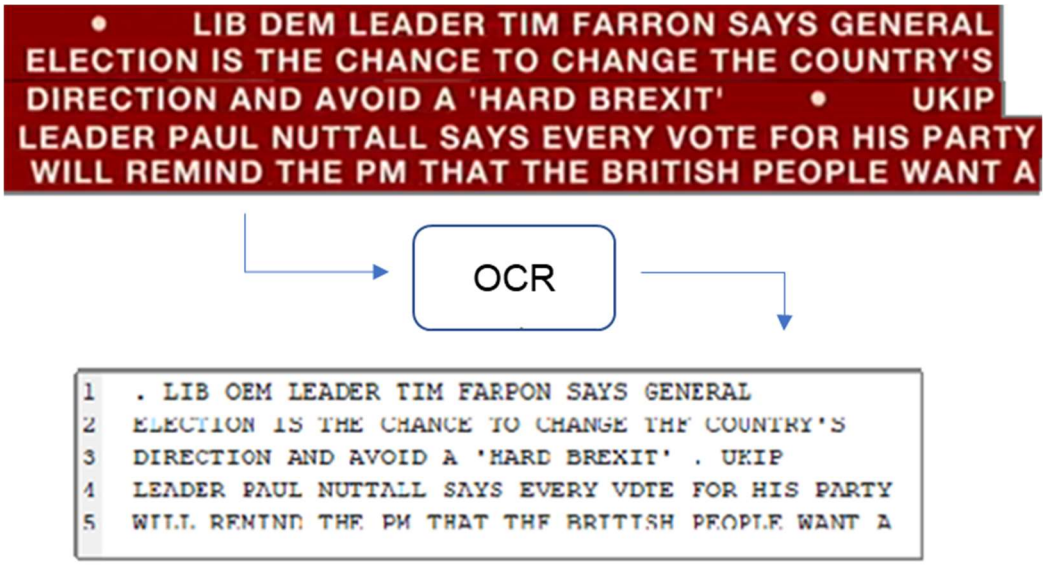


Fig. 26 – Esquematização da leitura do OCR em rodapé.

3.7.2 - Créditos finais de filmes

A lista de créditos finais de filmes é a lista de pessoas que trabalharam na película, bem como a sua respetiva função. Além disso, possuem informações adicionais como agradecimentos e lista de banda sonora.

Este caso é análogo ao do rodapé, mas com o movimento do texto com sentido de baixo para cima ao invés da direita para a esquerda. Enquanto que no caso acima a linha e o movimento têm o mesmo sentido, aqui serão perpendiculares. O texto é normalmente alinhado ao centro e costuma estar dividido em linhas perfeitas, excetuando alguns casos que se irão descrever.

A função está essencialmente pensada para créditos de cor branca ou clara com fundo preto, de maneira a haver um bom contraste. Enquanto que no caso anterior, o rodapé é acessório à imagem, agora o texto é a parte central do vídeo. A localização da área de texto fica bem mais simplificada, mas ainda é necessária, já que alguns vídeos possuem margens negras na parte superior e inferior, fazendo com o texto não percorra a totalidade da imagem Serão escolhidas 5 imagens aleatórias do vídeo e, no conjunto delas, registam-se a coordenadas mais acima e mais baixo onde se encontra porções de letras. Estes valores serão definidos como o limite superior e inferior da zona de texto em movimento, sendo sempre impostos a cada nova captação de *frame*.

A velocidade dos créditos é calculada de maneira semelhante ao caso anterior (Eq. 18). A única diferença é que agora, como o movimento é vertical, vamos ter em conta a diferença em termos de coordenadas verticais. Calculada a velocidade, é possível calcular o incremento a usar em número de *frame*. Mais uma vez, tem-se em conta a possibilidade de haver letras cortadas no limite da imagem. Neste caso, não será só uma letra como no rodapé, mas sim uma linha inteira de letras, dado o movimento vertical. Se assim acontecer, corta-se essa porção da captação e ajusta-se o salto para o próximo *frame*, como mostra a figura 27. Usa-se para o efeito a equação 19. Analogamente, haverá o ajuste entre o penúltimo e o último *frame*, através da equação 20, tendo em conta que agora o movimento é vertical.

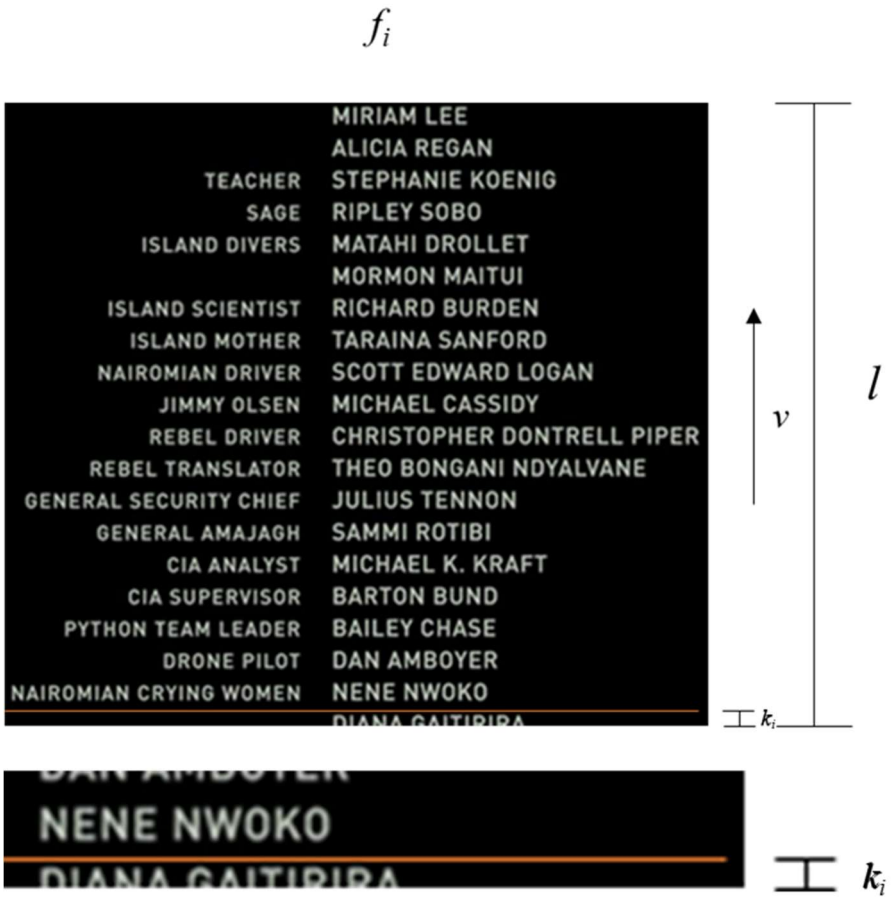


Fig. 27 – O frame f_i tem a última linha cortada com um comprimento k_i, num rodapé com comprimento l e velocidade v. Para o cálculo da seguinte captação, aplica-se a eq.15. Em baixo uma aproximação da zona de corte.

Todas as captações vão ser unidas numa só imagem. Montada toda a linha de créditos, é feita a leitura final. Há, porém, dificuldades novas neste caso. Os nomes das pessoas envolvidas no filme costumam estar apenas em maiúsculas, enquanto que o mesmo só às vezes acontece com as suas respetivas funções, como mostra a figura 28. Além disso, o nome pessoal muitas vezes tem uma fonte maior. Relembre-se que o programa OCR original pega numa linha e separa os carateres por escalas de altura (grande ou pequena). Como a função e o nome costumam estar na mesma linha, diferentes tamanhos de letra podem deturpar a leitura e confundir carateres. Usualmente, estão separados por um espaço, centralizado na imagem, de tamanho maior que espaços entre letras e palavras. Quando detetado pelo programa, em vez de ler toda a linha como uma só, separa-o em dois bocados pelo centro e lê um de cada vez, independentemente. Assim, caso numa porção detete só maiúsculas, não afeta toda a linha.

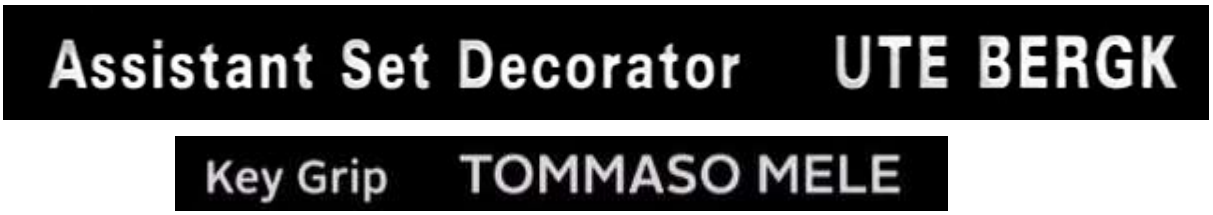


Fig. 28 – Exemplos de tamanhos diferentes na mesma linha [48,49]

Outro caso comum, normalmente mais para o fim da sequência de créditos, é a existência de dois ou mais blocos de nomes, juntos lado a lado. Isto acontece sobretudo com listas de músicas presentes no filme, e respetivos compositores e intérpretes. Por vezes, a linhagem de um é independente dos outros. O programa está preparado para alguns casos, caso sejam apenas dois blocos e a separação seja maior que 6 vezes a largura média dos caracteres. Mostra-se um exemplo deste caso na figura 29.

Fig. 29 – Exemplo de dois blocos desalinhados entre si, lado a lado [50]

Feita a captação de *frames* e a sua união numa só imagem, como mostra a figura 30, o programa OCR faz a leitura final de toda a imagem, imprimindo os resultados num ficheiro de texto respeitando as linhas.

Fig. 30 – Exemplo de uma parcela de créditos finais unidos, prontos para leitura OCR [49]

3.8 - Fluxogramas

Como complemento auxiliar, apresenta-se na figura 31 o fluxograma geral do programa OCR construído. Há dois ramos iniciais paralelos, um com a formação da biblioteca, e outro com a preparação da imagem input. Ambos encontram-se na classificação de cada caractere, a componente principal do programa. Feita a análise total, o programa devolve o output final.

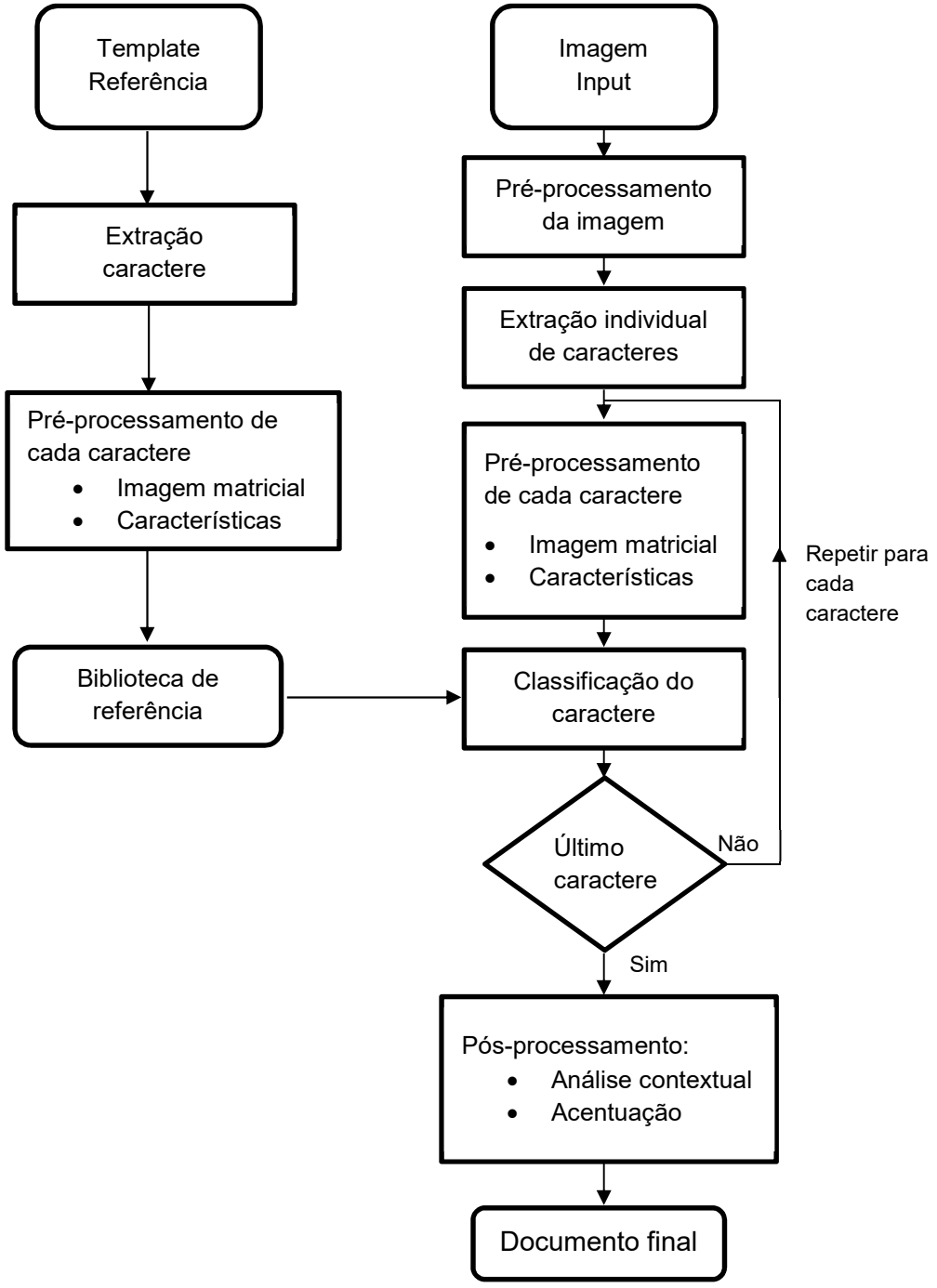


Fig. 31 – Fluxograma do programa OCR base

O segundo fluxograma, na figura 32, é uma generalização dos casos em vídeo, evidenciando melhor o processo de captação de cada *frame*. Aqui o programa OCR é resumido a apenas um passo, já que se comporta de maneira similar ao da figura 31.

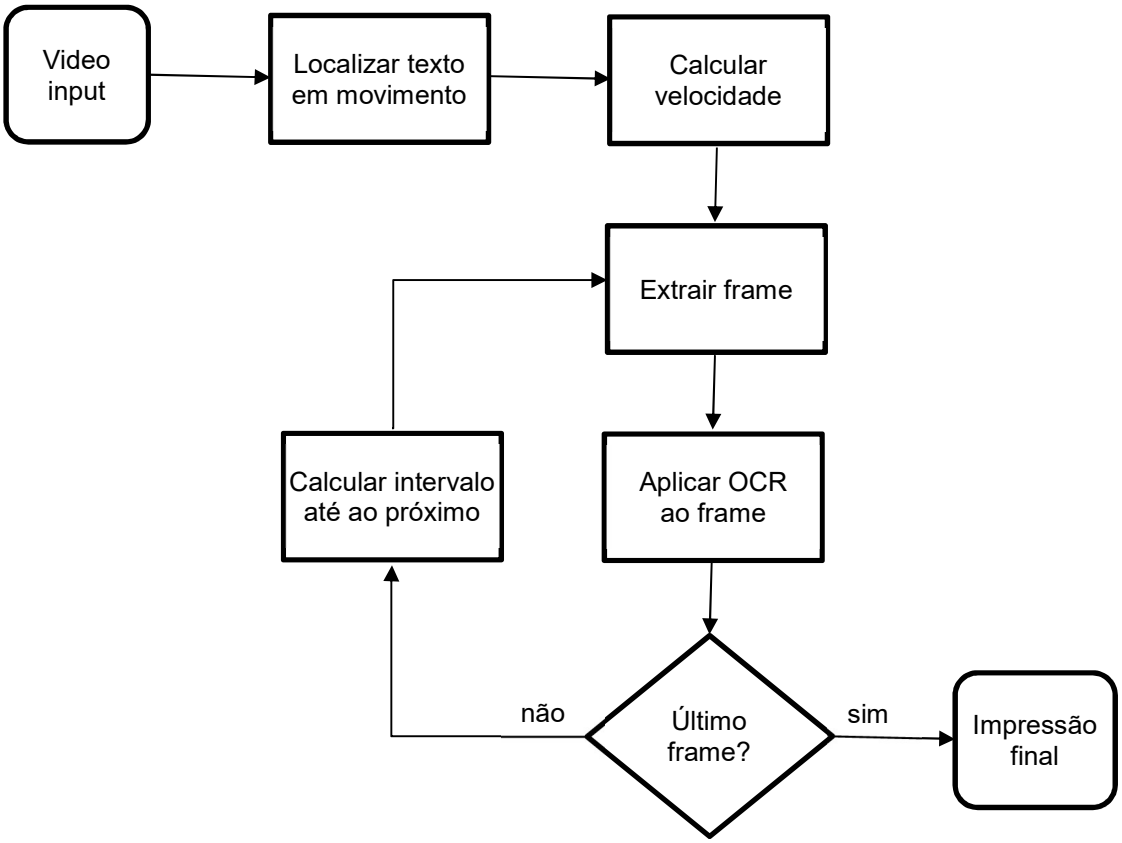


Fig. 32 – fluxograma da variação aplicada a texto móvel em vídeo

4 - Resultados e Discussão

4.1 – Resultados para texto estático

Os dados para os testes de texto estático são páginas de texto em pdf, com letras negras sobre fundo branco. Foram formadas a partir de texto de obras literárias selecionadas, com um tipo de letra e uma resolução pré-definidas. A partir de cada página, há duas conversões: um scan da página em formato jpeg, e uma cópia do texto em formato txt. O scan serve para o programa OCR fazer a leitura em si. O texto serve como *ground truth* de comparação com ficheiro de texto criado pela leitura, de maneira a avaliar a eficácia do programa em termos de exatidão, tanto de caracteres como de palavras. As exatidões relativas ao *ground truth* (Eqs. 14 e 15) são usadas como a avaliação principal do OCR. As exatidões relativas ao *output* (Eqs 16 e 17) são usadas como termo de comparação com as anteriores. Se forem maiores, significa que possuem mais caracteres/palavras que o *ground truth*, e vice-versa.

Foram selecionadas amostras de 5 obras para testar o programa, com 10 páginas cada. O tipo de letra varia entre Arial e Times New Roman. Há diferentes níveis de qualidade, em termos de resolução e *dpi*:

- **resolução A**: 1920x2845 pixéis, 225 *dpi*;
- **resolução B:** 1440x2037 pixéis, 232 *dpi*;
- **resolução C:** 930x1358 pixéis, 232 *dpi*.

As diferentes resoluções servem o propósito de testar a eficácia do programa relativamente à qualidade da imagem.

A Tabela 3 descreve os dados usados, com o número aproximado de palavras e caracteres, bem com a resolução, idioma e referência.

Tabela 3 - Lista de dados para os testes em texto estático

Obra	#caracteres	#palavras	Tipo de letra	Resolução	Idioma	Referência
Livro 1	~30000	~7000	Arial	A	Castelhano	[51]
Livro 2	~40000	~9000	Arial	A	Português	[52]
Livro 3	~30000	~6000	Arial	A	Alemão	[53]
Livro 4	~30000	~7000	Arial	B	Castelhano	[51]
Livro 5	~40000	~9000	Arial	B	Português	[52]
Livro 6	~30000	~6000	Arial	B	Alemão	[53]
Livro 7	~30000	~7000	Arial	C	Castelhano	[51]
Livro 8	~40000	~9000	Arial	C	Português	[52]
Livro 9	~30000	~6000	Arial	C	Alemão	[53]
Livro 10	~16000	~3500	Times New Roman	A	Holandês	[54]
Livro 11	~20000	~4000	Times New Roman	A	Inglês	[55]

De salientar que, no caso do Alemão, foi acrescentado mais um caractere à biblioteca *template*, a letra “ß”.

4.1.1 - Resultados para Arial

Na Tabela 4, mostram-se os resultados dos livros 1 a 3, de tipo Arial com a resolução mais alta (A).

Tabela 4 – Resultados do OCR (Livros 1 a 3)

	Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} Palavras (%)
Média	99,30	99,30	97,96	97,96
Mediana	99,34	99,34	97,97	97,97
Máximo	99,76	99,76	98,96	98,96
Minimo	98,88	98,88	96,55	96,55
Desvio padrão	0,27	0,27	0,77	0,77

Tabela 5 – Matriz de Confusão parcial (Livros 1 a 3)

	l (con)	n-	´,	P	T	i	!’	c.
l(vog)	57							
ñ		91						
;			86					
R				34				
f					73			
¿								10
¡						7		
.”							15	

Os resultados são bastante satisfatórios, acima dos 99% de exatidão, ou seja, a percentagem de edições é menos de 1% do número de caracteres no *ground truth*. Para as palavras, os valores de exatidão estão acima dos 97%.

Ainda assim, há erros que podem ser corrigidos, com alguns exemplos na tabela 5. “R” é várias vezes confundido com “P”. Isto pode dever-se à perna direita do “R”, que pode variar de sítio. Se na comparação matricial a perna do “R” *template* e do “R” a testar se desencontram, a quantidade de pixéis diferentes vai ser aproximadamente o dobro da diferença em relação a “P”. Uma solução pode passar por impor uma condição morfológica que explicite que só se selecionará “P” caso a zona inferior direita esteja vazia ou quase vazia.

Os casos do “ñ”, “¡” e do “¿ ” também não foram previstos, dado serem caracteres característicos da língua castelhana. A solução passará por incorporá-los na biblioteca.

Quanto ao ponto e vírgula, o programa deteta o ponto como um acento agudo. É necessário explicitar neste caso que, se está sobre uma vírgula, então deve ser transformado em “;”:

O já esperado caso da vogal “l” com a consoante “l” está bem presente. A melhor maneira de contornar será mesmo uma verificação ortográfica. O mesmo resolverá os “f” confundidos com “T”, embora este último caso possa ser resolvido com uma condição morfológica, semelhante ao caso dos “R” e “P”.

Na Tabela 6, mostram-se os resultados dos livros 4 a 6, de tipo Arial com a resolução média (B). A tabela 7 mostra alguns erros predominantes.

Tabela 6 – Resultados do OCR (Livros 4 a 6)

	Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} Palavras (%)
Média	95,26	95,29	85,82	85,81
Mediana	95,38	95,42	88,11	88,12
Máximo	98,31	98,32	93,77	93,76
Minimo	92,05	92,10	72,99	72,99
Desvio padrão	2,11	2,09	7,08	7,07

Tabela 7 – Matriz de Confusão parcial (Livros 4 a 6)

	O	.	J	o	l(consoante)	;
Q	79					
,		274				
l (consoante)			498			
e				874		
i					285	74

A redução de resolução diminuiu a exatidão e aumentou a variedade de casos de confusão, como por exemplo, “e” e “o”. A primeira vogal aproxima-se de uma forma mais redonda à medida que a imagem perde qualidade. Outras caracteres próximos vão ser confundidos cada vez mais, como o “Q” e o “O”. Vírgulas começam a parecer pontos, e até o ponto do “i” começa a ligar-se ao corpo da letra, aparentando ser a consoante “l”.

Na Tabela 8, mostram-se os resultados dos livros 7 a 9, de tipo Arial com a resolução mais baixa (C). A tabela 9 mostra alguns erros predominantes.

Tabela 8 – Resultados do OCR (Livros 7 a 9)

	Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} Palavras (%)
Média	90,84	91,01	74,74	74,68
Mediana	90,55	90,78	74,05	73,92
Máximo	94,62	94,67	89,11	89,11
Minimo	88,68	88,93	68,70	68,70
Desvio padrão	1,79	1,73	5,47	5,48

Tabela 9 – Matriz de Confusão parcial (Livros 7 a 9)

	8	i	:	´l	\
g	369				
;		240	159		
,			153		
i				234	
í		132			
t					402

O “g” ao perder definição começa a fechar-se e pode aparentar ter dois buracos, sendo então comparado na sub-biblioteca errada (dois buracos em vez de um). Torna-se mais difícil determinar a relação com a linhagem e “;” podem passar por “i. Vírgulas e acentos começam a ser confundidos com pontos.

A exatidão do OCR começa a diminuir sendo abaixo de 90%, claramente sendo frágil a níveis mais baixos de resolução.

4.1.2 - Resultados para Times New Roman

A tabela 10 apresenta os resultados em Times New Roman, para os livros 10 e 11, na resolução mais alta.

Tabela 10 – Resultados do OCR (Livros 10 e 11)

	Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} Palavras (%)
Média	97,40	97,38	88,19	88,06
Mediana	97,41	97,39	88,21	88,09
Máximo	98,10	98,09	90,30	90,20
Mínimo	96,69	96,67	86,07	85,93
Desvio padrão	0,36	0.36	1.07	1.08

Tabela 11 – Matriz de Confusão parcial (Livros 10 e 11)

	A	t	v	m	w	!’
ak	87					
l(consoante		87				
r			84			
rn				20		
ru				17		
u					34	
.”						4

O Times New Roman apresenta uma exatidão de 97,4% (Tabela 8). Teria, em princípio, menos hipóteses de confusão, já que o estilo das letras faz com que sejam mais distintas. No entanto, as condições de análise contextual no caso do Arial fazem ultrapassar a eficácia do estilo mais clássico. Este também é mais sensível a ligações entre caracteres não esperadas.

Nos mesmos documentos e como modo de comparação, o programa Tesseract [4] teve uma média de exatidão de 99,85 % em caracteres e 99,54 % em termos de palavras. Fora, eliminar alguma pontuação, as únicas confusões que fez foram “1” pela consoante “l” e o algarismo 0 pela letra “o”. Neste campo, o OCR aqui construído tem

consoante “l” e o algarismo 0 pela letra “o”. Neste campo, o OCR aqui construído tem ligeiramente melhores resultados. No entanto, o Tesseract é bem melhor nos outros casos, como caracteres ligados. Tem a capacidade de detetar possíveis pontos de união e verifica se a separação aumenta um melhor resultado de classificação. [4]

Voltando ao OCR construído, em todos os casos, o programa detetou sem erros o tipo de letra e o idioma presentes no texto.

4.2 – Resultados para vídeo

4.2.1– Resultados para rodapé de canais noticiosos

Para testar a leitura do rodapé, foram retiradas amostras de vídeos de 1280x720 pixéis de canais noticiosos:

- Fox News (20 clips) [45,56, 57,58,59,60];
- MSNBC (20 clips) [61;62];
- BBC News (5 clips) [47].

A duração dos clips é de aproximadamente 2 minutos. O formato de todos é mp4. O rodapé é contínuo, sem interrupções por parte da transmissão em qualquer uma das amostras. Exemplos de *frames* de dois dos vídeos apresentam-se na figura 33.

O *ground truth* de cada amostra foi obtido através da visualização de cada uma das amostras, redigindo o texto manualmente.

Fig. 33 – Amostras de vídeos testados com rodapé [47,56].

Foram testados em primeiro lugar os 20 vídeos da Fox News. Foi usado um *template* só de maiúsculas, algarismos e alguns símbolos, sem recorrer a um suplemento de caracteres ligados entre si. Possuem cerca de 7700 caracteres e 1500 palavras no total. A Tabela 12 apresenta os resultados de exatidão e Tabela 13 mostra a matriz de confusão da análise.

Tabela 12 – Resultados para o OCR em vídeos da Fox News

	Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} Palavras (%)
Média	92,70	92,51	77,67	77,67
Mediana	92,67	92,43	77,39	77,39
Máximo	93,57	93.31	80,52	80,52
Mínimo	91,99	91,72	74,50	74,50
Desvio padrão	0,40	0,40	1,74	1,74

Tabela 13 – Matriz de confusão para o 1º teste da Fox News

	B	D	E	F	J	N	O	Q	R	T	U	V	W	4	£	)
AS														1		
CA								47								
CH								2								
D							109									
E															1	
EA		9							12							
F			7													
FT				3												
H				33												
KA		2														
LA		28							9							
LASS		1														
LS											9					
Q							8									
RA	41															
TT										2						
TU						5										
TW																1
TY												6				
U					18											
VT													10			
VY													3			
ZA		1														

Em função dos resultados da matriz de confusão, foi construído um suplemento com os pares de letras ligados encontrados. Só se recorre aos caracteres desta subsecção se os caracteres tiveram uma altura grande e uma largura com 1,3 vezes a sua altura. Uma largura maior que a altura indica a presença de, muito provavelmente, duas letras unidas. Repetiu-se o teste, com esta ajuda adicional. As Tabelas 14 e 15 apresentam os novos resultados e matriz de confusão.

Tabela 14 – Resultados após atualização da biblioteca

	Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} Palavras (%)
Média	97,72	97,72	88,76	88,76
Mediana	97,69	97,69	88,37	88,37
Máximo	98,47	98,47	92,00	92,00
Mínimo	96,92	96,92	85,14	85,14
Desvio padrão	0,42	0,42	1,99	1,99

Tabela 15 – Matriz de confusão do novo teste

	E	F	J	O	£
D				109	
E					1
F	7				
H		33			
Q				8	
U			18		

Os resultados são bem melhores e os casos de pares ligados são eliminados, devido ao suplemento adicionado à biblioteca. Duas letras maiúsculas ligadas têm uma largura claramente fora do normal. Como há um critério bem definido para a sua recorrência – só será usado o suplemento caso a largura seja fora do normal -, não há perigo dos novos elementos interferirem com os casos normais. Isto é mais difícil de concretizar com minúsculas, as quais variam mais de tamanho entre elas.

No entanto, ainda há vários erros críticos. A fonte Arial é próxima, mas não é exatamente a letra do rodapé usado pela Fox News. Uma ligeira grossura adicional pode explicar algumas confusões, bem como alguns estilos de letra pontuais – como o "Q", com uma saliência mais horizontal que o normal (Fig.34).

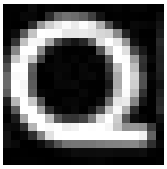


Fig. 34 – Forma da letra "Q" nos vídeos da Fox News.[45]

Em terceiro lugar, segue-se o teste aos vídeos da MSNBC e da BBC News, já com o suplemento. O total de caracteres e palavras é de 9000 e de 2000. Os resultados estão expostos na Tabela 16, mostrando a tabela 17 a respetiva matriz de confusão.

Tabela 16 – Resultados para o teste da MSNBC e BBC News

	Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} Palavras (%)
Média	97,69	97,69	88,86	88,86
Mediana	97,75	97,75	89,23	89,23
Máximo	98,10	98.10	90,77	90,77
Mínimo	97,09	97,09	85,00	85,00
Desvio padrão	0,31	0,31	1,50	1,50

Tabela 17 – Matriz de confusão da MSNBC e Fox News

	E	F	I	M	O	P	£
D					13		
E		8					2
F	62						
H				28			
N			17				
Q					7		
R						3	

As confusões são da mesma natureza das dos vídeos anteriores. Uma solução pode passar por treinar o programa com o estilo próprio do canal do qual se quer extrair o rodapé. Dado que todos os dias há texto novo com nova informação, um treino adequado pode permitir extrair muito texto no futuro (pelo menos, até ao canal mudar de estilo de *news ticker*, como por vezes sucede).

Não se verificaram casos de pares ligados. Tal pode ser devido ao suplemento, ou então a uma possível melhor separação entre letras nestas amostras. De notar que nos últimos dois testes apenas houve edições de substituição, não havendo eliminações nem inserções entre *ground truth* e *output*.

4.2.2 – Resultados para créditos finais de filmes

Os dados são compostos por clips de vídeo dos créditos finais de filmes. O formato é mp4 e a resolução é de 1920x1080p. Todos os elementos das amostras são constituídos por texto branco ou claro sobre fundo preto, tendo um contraste evidente.

Há 12 amostras em Arial, com um total de 140 200 caracteres e 23 600 palavras, aproximadamente. Há uma amostra para Times New Roman, com cerca de 16400 caracteres e 2500 palavras. As Tabela 18 e 19 mostram os resultados finais para Arial. A Tabela 20, 21 e 22 são matrizes de confusão para alguns casos.

O *ground truth* de cada amostra foi obtido através da visualização de cada uma das amostras, redigindo o texto manualmente.

Tabela 18 – Resultados para os créditos finais de filmes

	Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} Palavras (%)
Média	94,20	94,20	79,94	79,87
Média pesada*	97,73	97,73	88,44	88,36
Mediana	98,18	98,18	87.94	88,02
Máximo	99,73	99,73	98.19	97,18
Mínimo	58,08	58,08	16.25	15,94
Desvio padrão	11,53	11,57	22.21	22,21

*Média pesada consoante a quantidade de caracteres de cada vídeo.

Tabela 19 – Resultados para o teste dos créditos finais em Arial

	Tempo (m:ss)	Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} palavras(%)	Referência
Filme 1	3:45	99.17	99.17	95.12	95.13	[63]
Filme 2	6:48	98.46	98.46	90.96	90.96	[64]
Filme 3	3:59	98.18	98.18	89.27	89.27	[65]
Filme 4	3:05	58.08	58.08	16.25	15.94	[66]
Filme 5	2:00	95.37	95.37	75.44	75.44	[67]
Filme 6	4:28	99.73	99.73	98.19	97.18	[68]
Filme 7	2:00	98.48	98.48	91.02	91.02	[69]
Filme 8	4:35	97.80	97.80	87.63	87.63	[70]
Filme 9	1:23	95.30	95.30	75.60	76.26	[46]
Filme 10	4:39	98.17	98.17	88.24	88.41	[48]
Filme 11	2:28	98.52	98.52	87.50	87.23	[49]
Filme 12	1:00	93.08	93.08	64.04	64.04	[71]

A maioria dos créditos teve altas taxas de exatidão na extração, a maioria acima dos 97%. O programa essencialmente mostra-se eficaz a extrair o texto dos créditos finais.

As Tabelas 17 e 18 dão dois exemplos de matrizes de confusão dos filmes 2 e 10, respetivamente. Nestes e nos outros casos, apenas houve substituições de caracteres, nunca eliminações ou inserções. Dado a alta definição das amostras, com os caracteres bem separados entre si, não há casos de caracteres conectados ou degradados, pelo que o programa devolveu um *output* com tamanho similar ao *ground truth*, daí os valores iguais entre *Acc* e Acc_{op} de caracteres. O mesmo não se verifica no caso das palavras. Sucedeu que houve certas discrepâncias em termos de espaçamento. Dado que cada lista de créditos finais pode ter as suas particularidades em termos de formatação, o espaçamento entre palavras por vezes pode não ser bem detetado, pelo que algumas palavras podem na impressão *output* final aparecer coladas entre si, ou uma palavra separada em duas.

Repara-se que no filme 2 os casos de confusão são essencialmente algo esperados, como “D” a ser devolvido como “O”, “F” como “E”, ou “H” como “M”. Uma grossura adicional das letras em relação ao *template* pode explicar o caso do “S” ser

extraído como “I” e o “U” como “E”. No filme 10, sucede o mesmo, predominando as confusões “A” com “R”, “E” e “F” mutuamente ou “D” por “O”.

O filme 4 é, sem dúvida” o caso que mais se destaca, pela negativa. O tipo de letra é mais distinto, afastando-se mais do Arial disponível no programa. A matriz mostra os casos mais comuns. O “C” aproxima-se do parêntesis esquerdo e muitas letras aparecem como “I”. Talvez a grossura adicional das letras face à biblioteca *template* faça como que sejam detetadas como um “bloco branco e, consequentemente, um “I”. A figura 35 ilustra a leitura do programa para este caso.

Uma possível solução para melhorar o programa será ter mais tipos de letra disponível, inclusive permitir o programa fazer a comparação em mais que um ao mesmo tempo. Assim poderia haver uma melhor extração face à variedade de estilos que as fontes dos créditos podem possuir.

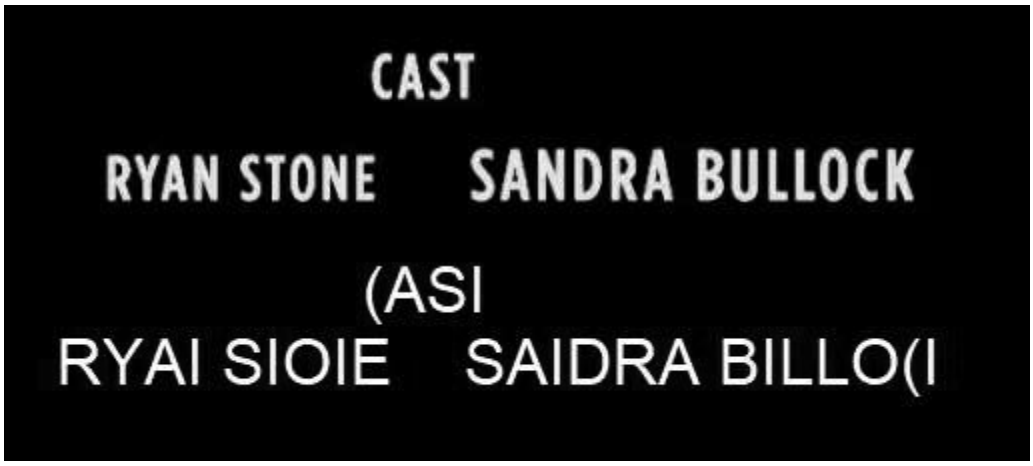


Fig 35. – Exemplo de leitura de caracteres dos créditos do filme 4 e a respetiva leitura. [66]

Tabela 20 – Matriz de confusão para o filme 2.

	E	I	J	M	O
D					197
F	75				
H	21			57	
N					
S		31			
U	32		3		

Tabela 21 – Matriz de confusão para o filme 10

	E	F	O	R
A				44
D			130	
E		79		
F	37			
P				32
Q			15	

Tabela 22 – Matriz de confusão parcial para o filme 4

	E	I	(
c			
C			108
E		4	
F	28		
G	11	32	2
H	3	35	
I			
K		32	
L	65		
M		44	
N		111	
O			
Q			
T		115	
U	3	79	
V			
X		12	
Z			

Tabela 23 – Resultados para os créditos do filme Contact, em Times New Roman

Acc (%)	Acc_{op} (%)	Acc Palavras (%)	Acc_{op} Palavras (%)
96,48	96,48	80,72	80,76

Tabela 24 – Matriz de confusão para Contact

	E	F	H	I	O
D					20
E		15			
F	27				
N			464		
T				2	
U			39		

O caso de Times New Roman do filme “Contact” [50] teve uma satisfatória leitura, com exatidão a rondar os 96,5%. As confusões apresentadas são esperadas. De salientar que o programa detetou um caso de dois blocos, mostrado na figura 29, com desalinhamentos entre eles e leu-os, independentemente, com sucesso. No entanto, os resultados são menos bons que a média dos resultados em Arial.

5 - Conclusão

O trabalho apresentado nesta tese inclui a construção de um programa OCR de raiz, sendo realizada a sua aplicação a texto estático e duas situações de vídeo.

O programa revelou ter um bom desempenho para texto em imagem estática, com aproximadamente 99% de exatidão para tipos de letra Arial e 97% para Times New Roman. Porém, o desempenho sofre uma quebra com a descida de resolução da imagem, ficando longe da qualidade apresentada por outros programas disponíveis no mercado. Além disso, o programa ainda faz determinadas confusões entre alguns caracteres, pelo que necessita de alguns reparos a nível de um pós-processamento contextual.

A aplicação para texto móvel em vídeo teve sucesso, em primeiro lugar, na segmentação do texto a partir do vídeo, formando-se a partir de um texto móvel uma imagem íntegra, com todos os caracteres na sua disposição original, pronta para ser analisada pelo programa. A extração do texto em si tem resultados de exatidão de cerca de 97%.

É necessário, porém, testar o programa para uma maior quantidade e uma maior variedade de casos, tanto para o rodapé como para os créditos finais

Além disso, o programa beneficiaria de algumas melhorias e atualizações, tais como:

- uma maior opção de tipos de letra, de maneira a prever a variedade de fontes que as situações em imagem e vídeo podem apresentar;
- uma operação que detete e separe letras conectadas, em vez de apenas adicionar alguns casos à biblioteca;
- aumentar a capacidade de extração de texto de imagens e vídeos com menor qualidade;
- recorrer a um verificador de ortografia, de maneira a refinar o resultado final.

Apesar dos grandes obstáculos, tem sido notória a evolução do OCR, acompanhando a explosão tecnológica das últimas décadas. Enquanto dantes eram necessárias grandes maquinarias próprias, hoje em dia qualquer computador ou até um smartphone pode suportar uma aplicação de reconhecimento de texto. [43]

O desenvolvimento de maiores bases de dados fomenta a possibilidade de maiores bibliotecas de classificação, bem como de possibilidades de pré e pós processamento. A escrita à mão ainda é um desafio enorme mas tem dado frutos, aliada

a uma crescente investigação em escritas contínuas como a arábica. [32] Uma das chaves de um ainda melhor reconhecimento pode estar nas *neural networks*, bem como na sua aplicação de modelos de Markov, onde tem havido progresso, por exemplo, na leitura de documentos degradados [44].

Sendo um campo em evolução constante, as novas abordagens à aplicação do OCR darão mais e melhores soluções aos obstáculos ainda presentes no desenvolvimento da tecnologia.

6 – Referências

[1] E. E. Fournier D'Albe (1914). On a Type-Reading Optophone. *Proceedings of the Royal Society of London, Series A, Containing Papers of a Mathematical and Physical Character*. Vol. 90, Edição 619, pp. 373-375

[2] Eikvil, L. (1993). Optical Character Recognition.

[3] Zaghden, N., Mullot, R., Alimi, M. A. (2013). Categorizing Ancient Documents. *IJCSI International Journal of Computer Science Issues*. Vol. 10, Edição 2, No 2, Março de 2013

[4] Smith, R. (2007). An Overview of the Tesseract OCR Engine. *Proc. International Conference on Document Analysis and Recognition*.

[5] Kavelar, A., Zambanini, S., Kampel, M. (2013). Reading Ancient Coin Legends: Object Recognition vs. OCR. *OAGM/AAPR Workshop proceedings.*

[6] Borovikov, E. (2004). A Survey of Modern Optical Character Recognition Techniques.

[7] Plamondon, R., Srihari, S. N. (2000). On-line and off-line Handwriting Recognition: A Comprehensive Survey. *1999 IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 22, Edição 1, pp 63-84, Janeiro de 2000.

[8] Bhowmick, S. (2012). Bangla Text Recognition from Video Sequence: A New Focus. *National Conference on Computing and Systems (NaCCS)*, pp. 62-67, 2012.

[9] Halima, M. B, karray, H., Alimi, A.M., et al (2012). NF-SAVO: Neuro-Fuzzy system for Arabic Video OCR. *International Journal of Advanced Computer Science and Applications*. Vol. 3, Nº 10, pp 128-136, 2012.

[10] Sankirti, S., Kamade, P. M. (2011). Video OCR for Video Indexing. *IACSIT International Journal of Engineering and Technology*. Vol.3, Nº 3, Junho de 2011.

[11] ONLY in JAPAN, 2017*, What makes art good or bad?*, gravação de vídeo, YouTube, 17/07/2017, <https://www.youtube.com/watch?v=fQRt56aZnnk>

[12] Gaikwad, B.P., Manza, R.R., Manza, G.R. (2014). Automatic Video Scene Segmentation to Separate Script and Recognition. *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA)*, pp 225-235, 2014.

[13] Sato, T., Kanade, T., Hughes, E.K., et al (1998). Video OCR for Digital News Archives. *Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database.* 3 de Janeiro de 1998.

[14] Sato, T., Kanade, T., Hughes, E., K., et al (1999), Video OCR: Indexing Digital News Libraries by Recognition of Superimposed Captions. *Multimedia Systems*, Vol. 7, Edição 5, pp 385–395, Setembro de 1999.

[15] Muda, N., Kamariah, N., Ismail, N., Bakar, S.A.A., et al. (2007). Optical Character Recognition By Using Template Matching (Alphabet). *National Conference on Software Engineering & Computer Systems 2007 (NACES 2007),* 20-21 Agosto de 2007.

[16] Mamatha, H.R., Karthik, S., Srikanta, M. K. (2013). Classifier Fusion Method to Recognize Handwritten Kannada Numerals, *ICECT 2012 conference*.

[17] Samadiani, N., Hassanpour, H. (2015). A neural-network base approach for recognizing multi-font printed English characters.

[18] Wang, T., Wu, D.J., Coates, A., et al. (2012). End-to-End Text Recognition with Convolutional Neural Networks, *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, 11-15 Novembro de 2012.

[19] Barve, S. (2012). Artificial Neural Network Based On Optical Character Recognition. *International Journal of Engineering Research & Technology (IJERT)*. Vol. 1, Edição 4, Junho de 2012.

[20] Bartz, C., Yang, H., Meinel, C. (2017). STN-OCR: A single Neural Network for Text Detection and Text Recognition.

[21] Dutta, S., Sankaran, N., Sankar, K.P., Jawahar, C.V. (2012). Robust Recognition of Degraded Documents Using Character N-Grams. *2012 10th IAPR International Workshop on Document Analysis Systems*, 27-29 Março de 2012.

[22] Cannon, M., Hochberg, J., Kelly, P. (1999). Quality Assessment and Restoration of Typewritten Document Images. *International Journal on Document Analysis and Recognition*. Vol. 2, Edição 2–3, pp 80–89, Dezembro de 1999

[23] Bansal, K., Kumar, R. (2013). K-Algorithm: A Modified Technique For Noise Removal In Handwritten Documents*. International Journal of Information Sciences and Techniques*. Vol. 3, Nº 3, Maio de 2013.

[24] Palfray, T., Tranouez, P., Nicolas, S., et al (2012). Logical segmentation for article extraction in digitized old newspapers.

[25] F. Cesarini, M. Gori, S. Marinai, et al (1999). Structured Document Segmentation and Representation by the Modified X-Y tree. *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99*. 22 de Setembro de 1999.

[26] Louloudis, G., Gatos, B., Pratikakis, I., et al (2006). A Block-Based Hough Transform Mapping for Text Line Detection in Handwritten Documents. *Tenth International Workshop on Frontiers in Handwriting Recognition*. Outubro de 2006

[27] Javed, M., Nagabhushan, P., Chaudhuri, B.B. (2013). Extraction of Line Word Character Segments Directly from Run Length Compressed Printed Text Documents. *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, 18-21 de Dezembro de 2013.

[28] Nagy, G., Nartkerb, T.A., Ricec, S.V., (1999). Optical Character Recognition: An illustrated guide to the frontier. *Proceedings of SPIE - The International Society for Optical Engineering*. Dezembro de 1999.

[29] Bassil, Y., Alwani, M. (2012). OCR Post-Processing Error Correction Algorithm Using Google's Online Spelling Suggestion. *Journal of Emerging Trends in Computing and Information Sciences*. Vol. 3, Nº 1, Janeiro de 2012.

[30] Agarwal, S. (2013). Utilizing Big Data in Identification and Correction of OCR Errors.

[31] Alginahi, Y. M. (2013). A Survey on Arabic Character Segmentation. *International Journal on Document Analysis and Recognition (IJDAR)*. Vol. 16, Edição 2, pp 105–126, Junho de 2013.

[32] Rashwan, M.A.A., Fakhr, M.W.T., Attia, M., et al (2007). Arabic OCR System Analogous to Hmm-based Asr Systems; Implementation and Evaluation.

[33] Al-Muhtaseb, H.A., Mahmoud, S.A. (2008). Recognition of Off-line Printed Arabic Text Using Hidden Markov Models.

[34] Steinherz, T., Rivlin, E., Intrator, N. (1999). Off-Line Cursive Script Word Recognition – A Survey. *Signal Processing.* Vol. 88, Edição 12, pp 2902-2912 Dezembro de 2008. *International Journal on Document Analysis and Recognition*. Vol. 2, Edição 2–3, pp 90–110, Dezembro de 1999.

[35] V. Levenshtein (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*. Vol. 10, p.707, Fevereiro de 1966.

[36] Mei, J., Islam, A., Wu, Y., (2016). *Statistical Learning for OCR Text Correction*.

[37] MacKenzie, I.S., Soukoreff, R.W. (2002). A Character-Level Error Analysis Technique for Evaluating Text Entry Methods. *Proceedings of the Second Nordic Conference on Human-Computer Interaction -- NordiCHI 2002*, pp. 241-244.

[38] Nartker, T., Taghva, K., Young, R., et al (2003). OCR correction based on document level knowledge. *Document Recognition and Retrieval X*. 22-23 de Janeiro de 2003.

[39] MATLAB R2017, The MathWorks, Inc., Natick, Massachusetts, United States.

[40] Stefan Trost <https://www.sttmedia.com/characterfrequencies>

[41] Smite, T.F., Waterman, M.S. (1981). Identification of Common Molecular Subsequences. *Journal of Molecular Biology*. Vol. 147, Edição 1, pp 195-197, 25 de Março de1981.

[42] smithWaterman < https://www.mathworks.com/matlabcentral/fileexchange/45831-matlab-audio-analysis-library?focused=3812680&tab=function>

[43] Ch, S., Mahna, S., Kashyap, N. (2015). Optical Character Recognition on Handheld Devices, *International Journal of Computer Applications*. Vol, 115, Nº 22, 2015.

[44] Rashid, S.F. (2014). *Optical Character Recognition - A Combined ANN/HMM Approach*. Dissertação de Doutoramento em Engenharia. Departamento de Ciências Informáticas, Universidade Técnica de Kaiserslautern, Kaiserslautern.

6.1 – Referências de dados

[45] *Fox & Friends*, 2015, programa de TV, Fox News, Nova Iorque, 12 de Outubro de 2015.

[46] Giler, D. (Produtor), Hill, W. (Produtor), Scott, R. (Produtor), & Scott, R. (Realizador). (2012). *Prometheus*. Estados Unidos, Reino Unido: Brandwyne Productions, Dune Entertainment, Scott Free Productions.

[47] *BBC News Special: General Election Announcement*, 2017, BBC News, Londres, 18 de Abril de 2017.

[48] Nolan, C. (Produtor), Roven, C. (Produtor), Thomas, E. (Produtor), & Nolan, C. (Realizador). (2008). *The Dark Knight*. Estados Unidos, Reino Unido: Legendary Pictures, Syncopy.

[49] Huffam, M. (Produtor), Kinberg, S. (Produtor), Schaefer, M. (Produtor), Scott, R. (Produtor), Sood, A. (Produtor), & Scott, R. (Realizador). (2015). *The Martian*. Estados Unidos, Reino Unido: Kinberg Genre, Scott Free Prodcutions, TSG Entertainment.

[50] Starkey, S. (Produtor), Zemeckis, R. (Produtor), & Zemeckis, R. (Realizador). (1997). *Contact*. Estados Unidos: South Side Amusement Company

[51] Cervantes, M. (1605). *Don Quijote de la Mancha.* Retirado de http://www.daemcopiapo.cl.

[52] Queirós, E. (1888). *Os Maias.* Retirado de http:// http://figaro.fis.uc.pt/

[53] Kafka, F. (1915). *Die Verwandlung.* Retirado de http://DigBib.Org.

[54] Frank, A. (1947). *Het Achterhuis.* Uitgeverij Contact. Amsterdão.

[55] Taleb, N. (2007). *The Black Swan: The Impact of the Highly Improbable.* Penguin Books. Londres.

[56] *Special Report with Bret Baier*, 2010, programa de TV, Fox News, Nova Iorque, 28 de Julho de 2010.

[57] *The O'Reilly Factor*, 2010, programa de TV, Fox News, Nova Iorque, 14 de Março de 2010.

[58] *The O'Reilly Factor*, 2010, programa de TV, Fox News, Nova Iorque, 17 de Abril de 2010.

[59] *Justice with Judge Jeanine*, 2013, programa de TV, Fox News, Nova Iorque, 6 de Setembro de 2013.

[60] *The Greg Gutfeld Show*, 2017, programa de TV, Fox News, Nova Iorque, 25 de Fevereiro de 2017.

[61] *Morning Joe*, 2016, programa de TV, MSNBC, Nova Iorque, 11 de Novembro de 2016.

[62] *Morning Joe*, 2017, programa de TV, MSNBC, Nova Iorque, 31 de Julho de 2017.

[63] Crockett, D. (Produtor), Milchan, A. (Produtor), Verbinski, G. (Produtor), & Verbinski, G. (Realizador). (2016). *A Cure For Wellness*. Alemanha, Estados Unidos, Luxemburgo: Blind Wink Productions, New Regency Productions, Regency Enterprises.

[64] Roven, C. (Produtor), Snyder, D. (Produtor) & Snyder, Z. (Realizador). (2016). *Batman V Superman*. Estados Unidos: Atlas Entertainment, Cruel and Unusual Films, DC Entertainment, RatPac Entertainment.

[65] Mason, A. (Produtor), Proyas, A. (Produtor), & Proyas, A. (Realizador). (1998). *Dark City*. Australia, Estados Unidos: Mystery Clock Cinema.

[66] Cuarón, A. (Produtor), Heyman, D. (Produtor), & Cuarón, A. (Realizador). (2013). *Gravity*. Estados Unidos, Reino Unido: Esperanto Filmoj, Heyday Films.

[67] Blum, J. (Produtor), Macy, T. (Produtor), & Flanagan, M. (Realizador). (2016). *Hush*. Estados Unidos: Blumhouse Productions, Intrepid Pictures.

[68] Nolan, C. (Produtor), Thomas, E. (Produtor), & Nolan, C. (Realizador). (2010). *Inception*. Estados Unidos, Reino Unido: Legendary Pictures, Syncopy.

[69] Grey, L. (Produtor), Wan, J. (Produtor), & Heisserer, E. (Realizador). (2016). *Lights Out*. Estados Unidos: Atomic Monster Productions, Grey Matter Productions, New Line Cinema, RatPac-Dune Entertainment.

[70] Donner, L. S. (Produtor), Kinber, S. (Produtor), Parker, H. (Produtor), & Mangold, J. (Realizador). (2017). *Logan*. Estados Unidos: Hutch Parkerk Productions, Kinberg Genre, Marvel Entertainment, The Donners' Company, TSG Entertainment.

[71] Blum, J. (Produtor), Estabrook, H. (Produtor), Lancaster, D. (Produtor), Michel, L. (Produtor), & Chazelle, D. (Realizador). (2014). *Whiplash*. Estados Unidos: Bold Films, Blumhouse Productions, Right of Way Films.