Bruno Legora Souza da Silva

Estudo e Comparação de Técnicas de Segmentação de Textos em Imagens

Brasil

Vitória, 2016

Bruno Legora Souza da Silva

Estudo e Comparação de Técnicas de Segmentação de Textos em Imagens

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Universidade Federal do Espírito Santo – UFES

Programa de Pós Graduação em Engenharia Elétrica – PPGEE

Laboratório de Computadores e Sistemas Neurais – CISNE

Orientador: Patrick Marques Ciarelli

Brasil

Vitória, 2016

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

S586e

Silva, Bruno Légora Souza da, 1991-
Estudo e comparação de técnicas de segmentação de texto em imagens / Bruno Légora Souza da Silva. – 2016.
66 f. : il.

Orientador: Patrick Marques Ciarelli.
Dissertação (Mestrado em Engenharia Elétrica) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Processamento de imagens. 2. Indexação automática. 3. Visão por computador . 4. Segmentação de texto. 5. Localização de texto em imagens. 6. Análise de componentes conectados (CCA). I. Ciarelli, Patrick Marques. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 621.3

Bruno Legora Souza da Silva

Estudo e Comparação de Técnicas de Segmentação de Textos em Imagens

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Trabalho aprovado. Brasil, 8 de dezembro de 2016:

Prof. Dr. Patrick Marques Ciarelli
Orientador

Prof. Dr. Klaus Fabian Côco - UFES

Profa. Dra. Kelly Assis De Souza Gazolli - IFES

Brasil
Vitória, 2016

Agradecimentos

A todos os meus familiares, especialmente meus pais, que sempre me apoiaram e incentivaram, mesmo nos momentos mais difíceis.

A todos os meus professores, especialmente meu orientador, pois sem eles este trabalho não seria possível.

A todos os meus colegas de laboratório, que sempre ajudaram com dicas e informações importantes durante o curso de mestrado. Aos demais colegas do Programa de Pós-Graduação em Engenharia Elétrica que me ajudaram nas disciplinas.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pela bolsa concedida.

Aos professores da banca, que aceitaram avaliar este trabalho.

Resumo

Localização de texto em imagens do mundo real é um problema muito abordado na atualidade, já que pode ser empregado em diversas aplicações, como geolocalização, indexação de imagens, identificação de produtos através de seus rótulos e auxílio aos deficientes visuais, melhorando a qualidade de vida destes. Atualmente, existem diversas pesquisas nesta área, além de uma competição realizada na *International Conference of Document Analysis and Recognition* para acompanhar os avanços destas, o que mostra relevância da área.

Este trabalho apresenta técnicas de segmentação de texto em imagens do mundo real, onde texto ocorre de diversas formas, orientações e tamanhos. Tais técnicas são baseadas em análise de componentes conectados, utilizando detecção de bordas ou técnicas como as *Maximally Stable Extremal Regions* ou a técnica FASText, que encontra *keypoints* baseado no detector FAST, baseadas em janelas deslizantes ou a combinação destas duas abordagens.

Uma comparação experimental de três diferentes técnicas de segmentação de texto é feita. Estas possuem um custo computacional reduzido, de modo que não causem um grande impacto no custo computacional de um sistema que as usem. Para esta comparação, três bases de dados são utilizadas: as duas divisões da base de dados da competição ICDAR 2013 e a base KAIST Scene Text, analisando métricas como área de texto extraída, tempo de processamento, número de caracteres sem segmentação e a imprecisão do detector, definida como a razão entre o número de segmentações e a quantidade de caracteres presentes na base de dados.

Os resultados obtidos nas bases mostraram que todas as técnicas analisadas obtiveram uma boa segmentação nas bases de dados, quando as imagens apresentavam caracteres bem definidos, sem problemas de iluminação, oclusão ou caracteres muito pequenos. Em imagens que apresentam estes últimos, as técnicas analisadas obtiveram seu desempenho prejudicado.

Palavras-chave: Localização de Texto, Segmentação de Texto, Componentes Conectados, Detecção de Bordas, Método de Canny, Mapa de Confiança, *Local Binary Pattern*, MSER, FASText.

Abstract

Text localization in real world images is a problem which has received significant attention in the present days, since it can be used in various applications, such as geolocation, indexing of images, identification of products through their labels and assistance to the visually impaired, improving their quality of life. Currently, there are several researches in this area, where a competition held in the International Conference of Document Analysis and Recognition follows the progress of these researches, and this highlights the relevance of this area.

This work presents some techniques of text segmentation in real world images, where text occurs in several forms, orientations and sizes. Such techniques are based on connected component analysis, using edge detection, techniques such as Maximally Stable Extremal Regions, the FASText technique that finds keypoints based on the FAST detector, techniques based on sliding windows or the combination of these two approaches.

An experimental comparison of three different text segmentation techniques is done. These have a reduced computational cost, so that they do not have a major impact on the computational cost of a system that uses them. For this comparison, three databases are used: the two divisions of the ICDAR 2013 competition database and the KAIST Scene Text database, analyzing metrics such as extracted text area, processing time, number of characters without segmentation and the imprecision of the detector, defined as the ratio of the number of targets to the number of characters in the database.

The results obtained in the databases have shown that all the analyzed techniques obtained a good segmentation ratio in the databases, when the images presented well defined characters, without problems of illumination, occlusion or very small characters. In the images that present the latter, the analyzed techniques obtained impared performances.

Keywords: Text Localization, Text Segmentation, Connected Components, Edge Detection, Canny Method, Confidence Map, Local Binary Pattern, MSER, FASText.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AdaBoost	*Adaptive Boosting*
ER	*Extremal Regions*
FAST	*Features from Accelerated Segment Test*
HOG	*Histogram of Oriented Gradients*
ICDAR	*International Conference of Document Analysis and Recognition*
LBP	*Local Binary Pattern*
MSER	*Maximally Stable Extremal Regions*
SBK	*Stroke Bending Keypoint*
SEK	*Stroke Ending Keypoint*
SVM	*Support Vector Machine*

Sumário

1 Introdução

1.1 Motivação

Texto é uma das informações mais importantes presentes em imagens, já que pode ser utilizado em inúmeras aplicações, como tradução automática de documentos ou rótulos, assistência a pessoas com deficiência visual, reconhecimento de placas ou indexação de imagens (LU et al., 2015).

No contexto de texto com origem digital, presentes em imagens escaneadas ou geradas diretamente em computadores, a localização e detecção de textos podem ser considerados problemas resolvidos, já que existem diversos sistemas comerciais e livres, como o sistema Tesseract (ITSEEZ, 2015), capazes de atingir um desempenho de até 99% (OPITZ, 2013).

Porém, a tarefa de detecção (ou localização) de textos em imagens do mundo real ainda é uma tarefa desafiadora na área de visão computacional, já que estes ocorrem com uma grande variação de fontes, cores, tamanhos e orientações. Além disso, excesso ou falta de iluminação, sombras, reflexões, ruídos e distorções da câmera também afetam estes componentes (ZHU; ZANIBBI, 2016).

Um sistema robusto de localização de texto deve levar em consideração todas estas possíveis variações de ocorrência de caracteres em imagens, o que o torna computacionalmente custoso. Porém, com o avanço tecnológico, dispositivos digitais como computadores, câmeras e celulares estão se tornando cada vez mais populares, com alta capacidade de processamento e baixo custo, o que incentiva pesquisas nesta área. Para acompanhar o desenvolvimento de tais pesquisas, a *International Conference of Document Analysis and Recognition* (ICDAR) organiza competições (LUCAS, 2005; KARATZAS et al., 2013; KARATZAS et al., 2015) em que uma base de dados pública é disponibilizada em cada edição e utilizada para testar os sistemas dos participantes.

Em diversas aplicações, é necessário que esses sistemas obtenham um desempenho próximo ao de um humano (OPITZ, 2013). Não só a qualidade de localização deve ser boa, como também é esperado que o tempo gasto para fazer essa tarefa seja reduzido. Para isto, é ideal que cada uma de suas etapas também apresente estas características.

Entre suas etapas usuais, está a extração de candidatos a texto de imagens, sendo uma das mais importantes etapas do sistema (OPITZ, 2013), pois sua função é detectar regiões onde existem possíveis caracteres, fornecendo uma hipótese inicial para um sistema de localização de texto.

1.2 Objetivos

Este trabalho tem como objetivo a implementação e comparação de métodos computacionais capazes de segmentar textos (caracteres) em imagens do mundo real, capturadas em diversas situações. A ocorrência de textos nestas imagens é restringida de modo que, na maioria dos casos, sigam a orientação horizontal e que o dispositivo utilizado na geração das imagens esteja focalizado nos elementos textuais.

As técnicas analisadas têm custo computacional reduzido, segmentando caracteres de forma rápida. Os métodos computacionais foram testados em bases de dados públicas, como a utilizada na competição ICDAR *Robust Reading Competition*, e comparados entre si, usando métricas de avaliação como área de texto extraído, tempo de processamento, caracteres não segmentados e a razão entre o número de segmentações e o número de caracteres da base de dados.

1.3 Caracterização do Problema

O problema de segmentação de texto é bastante similar a um problema de detecção comum de objetos em imagens, ao considerarmos caracteres isolados como tais objetos. Segundo (AMIT; FELZENSZWALB, 2014), um detector de objetos de uma mesma classe analisa uma imagem e retorna um conjunto de informações sobre a existência ou não de tais objetos. Usualmente, isto é indicado através da posição das detecções, representadas por suas *bounding-boxes*.

Porém, este detector deve construir um modelo através de um conjunto de exemplos que devem ser altamente variados, para considerar diversas variações de iluminação, posição ou rotação, entre outros. No caso de caracteres, por exemplo, o número 7 pode ser escrito com ou sem uma linha em seu centro. Portanto, é necessária a utilização de funções e características robustas a tais variações (AMIT; FELZENSZWALB, 2014). A Figura 1 apresenta alguns exemplos de ocorrência de textos com diferentes características em imagens.

Mesmo com técnicas robustas de detecção de objetos, existem ainda os problemas de múltiplas detecções do mesmo elemento, que podem ser eliminadas através de técnicas como *mean-shift*, como feito por Cosmo, Salles e Ciarelli (2015), e falsos positivos, que podem ser eliminados através de uma etapa de classificação binária, utilizando características que também devem ser invariantes a diversos tipos de variações, como escala e rotação (AMIT; FELZENSZWALB, 2014), caso haja a necessidade de um sistema muito preciso.

Figura 1 – Exemplos de problemas encontrados nas tarefas de segmentação e localização de texto em imagens

Fonte: Karatzas et al. (2013) e (NEUMANN; MATAS, 2015)

1.4 Breve Revisão de Literatura

Um sistema com o objetivo de localizar texto em imagens, usualmente segue a seguinte combinação de etapas: detecção de candidatos a caracteres, eliminação de possíveis falsos positivos, e agrupamento dos candidatos em palavras.

A etapa de extração ou segmentação de caracteres é a etapa mais importante de sistemas que tem o objetivo de localizar texto, pois sua função é encontrar hipóteses iniciais de caracteres (OPITZ, 2013). As técnicas que extraem tais componentes de imagens são divididas em duas principais abordagens (NEUMANN; MATAS, 2015): sistemas que deslizam janelas pela imagem, calculando descritores e os classificando em janela que contém caractere ou não, e sistemas que utilizam análise de componentes conectados.

Sistemas da primeira categoria, usualmente, possuem custo computacional superior aos da segunda, já que é necessário analisar diversas escalas e rotações da imagem para detectar tais variações em caracteres. Além disso, usualmente são utilizados descritores de alta dimensão, como o *Histogram of Oriented Gradients*, cuja extração e classificação pode ser demorada.

Já os sistemas da segunda categoria consideram que caracteres possuem uma característica constante, como a cor (MOSLEH; BOUGUILA; HAMZA, 2012) ou largura do traçado (EPSHTEIN; OFEK; WEXLER, 2010), e encontram candidatos agrupando *pixels* conectados. Entre as técnicas mais utilizadas nesta categoria, se destacam as *Maximally Stable Extremal Regions* (MSER) (MATAS et al., 2004; NEUMANN; MATAS, 2015), a segmentação através da extração de bordas (LU et al., 2015) e *keypoints* encontrados na imagem (BUŠTA; NEUMANN; MATAS, 2015).

1.5 Estrutura da Dissertação

Este trabalho está dividido da seguinte forma: No Capítulo 2 são detalhados alguns descritores e classificadores utilizados por sistemas que possuem o objetivo de segmentar caracteres de imagens. No Capítulo 3 são apresentadas algumas técnicas de segmentação de texto em imagens. No Capítulo 4 são descritas as bases de dados utilizadas neste trabalho e as métricas usadas para avaliar e comparar as técnicas de segmentação de texto em imagens. No Capítulo 5 são apresentados os experimentos realizados e os resultados obtidos. Finalmente, no Capítulo 6 são apresentadas as conclusões do trabalho e propostas para trabalhos futuros.

2 Classificadores e Descritores

Este capítulo descreve alguns descritores e classificadores utilizados por técnicas que têm o objetivo de segmentar caracteres de imagens do mundo real. Entre os classificadores utilizados para segmentação de texto, estão o *Adaptive Boosting* (AdaBoost) e o *Support Vector Machine* (SVM), descritos nas seções 2.1 e 2.2, respectivamente. Estes classificadores são utilizados geralmente por técnicas que utilizam o conceito de janela deslizante, onde a combinação de um descritor com um classificador é utilizada para identificar regiões de interesse na imagem. Já entre os descritores utilizados, estão o *Histogram of Oriented Gradients* (HOG) e o *Local Binary Pattern* (LBP), descritos nas seções 2.3 e 2.4, respectivamente.

2.1 Classificador *Adaptive Boosting*

Em aprendizado de máquina, o princípio de *Boosting* diz que a combinação de múltiplos classificadores "fracos" é capaz de produzir um classificador robusto (ZHOU, 2012). Considerando um conjunto de amostras de treino X, composto por três classes $X_1,\ X_2$ e X_3 equilibradas, e a disponibilidade de apenas um classificador fraco h_1, que ao ser treinado classifica corretamente as classes X_1 e X_2, é possível treinar um classificador h_2 que evidencia as amostras classificadas incorretamente de X_3. Caso ainda necessário, é possível inserir um novo classificador h_3, que será treinado com amostras incorretas de h_1 e h_2. Ao combinarmos os três classificadores, é possível formar um classificador capaz de separar as três classes corretamente (ZHOU, 2012).

Em (FREUND; SCHAPIRE, 1995) foi proposta a utilização de pesos nas amostras de treino, que são atualizados a cada iteração da etapa de treinamento. Esta técnica é chamada de Adaptive Boosting (AdaBoost) e considera um problema de classificação binário, com amostras de treinamento $(\mathbf{x}_i, y_i)$, onde $\mathbf{x}_i$ é o vetor de características e $y_i \in \{-1,\ 1\}$ é a classe da amostra. Inicialmente, todas as amostras terão pesos iguais, que são atualizados através de uma função exponencial baseada no erro de treinamento (FRIEDMAN et al., 2000).

A saída deste algoritmo combina o resultado dos T classificadores fracos através de uma soma ponderada pelos pesos α_t, como mostra a Equação 2.1. A classe retornada pelo algoritmo é dada pela função $sign(F(\mathbf{x}))$, que assume valor 1 para $F(\mathbf{x})$ positivo, zero para $F(\mathbf{x})$ igual a zero, e -1 caso contrário.

$$F(\mathbf{x}) = \sum_{i=1}^{T} \alpha_t h_t(\mathbf{x}) \tag{2.1}$$

A etapa de treinamento desta técnica consiste em treinar um novo classificador fraco $h_t(\mathbf{x})$ em cada uma de suas $t = \{1, 2, \ldots, T\}$ iterações, utilizando versões ponderadas do conjunto de treinamento através de uma distribuição de probabilidade $D_t(i)$ associada a cada uma das M amostras de treinamento, que inicialmente é uniforme. Para cada classificador, seu peso α_t é calculado com base no erro obtido através da soma de todos os $D_t(i)$ associados às amostras incorretamente classificadas. Finalmente, os pesos são ajustados com base no coeficiente α_t, diminuindo a importância das amostras corretamente classificadas e aumentando das demais. Este processo é apresentado no Algoritmo 1.

Outras versões deste algoritmo também foram propostas, como o *Real AdaBoost* (FRIEDMAN et al., 2000) o *Gentle AdaBoost* (FRIEDMAN et al., 2000). A primeira considera classificadores fracos capazes de retornar um valor real ao invés de um valor discreto, onde o sinal $sign(h_t(\mathbf{x}_n))$ é a classe e $|h_t(\mathbf{x}_n)|$ é uma medida de confiança da classificação. O fator de ponderação α_t é definido pela relação da Equação 2.2,

$$\alpha_{t_{real}} = \frac{1}{2} \ln \left(\frac{w_{+1}}{w_{-1}} \right), \tag{2.2}$$

onde w_{+1} e w_{-1} são dados pelas Equações 2.3 e 2.4, respectivamente.

$$w_{+1} = \sum_{m|y_m h_t(\mathbf{x}_m)=+1} D_t(m) \tag{2.3}$$

$$w_{-1} = \sum_{m|y_m h_t(\mathbf{x}_m)=-1} D_t(m) \tag{2.4}$$

onde o termo $m|y_m h_t(\mathbf{x}_m) = +1$ indica os valores de m que representam as amostras que foram classificadas corretamente, enquanto $m|y_m h_t(\mathbf{x}_m) = -1$ as classificadas incorretamente.

Já o classificador *Gentle AdaBoost* é uma versão mais robusta e estável do *Real AdaBoost*, que utiliza o fator de ponderação da Equação 2.5 (SILVA JÚNIOR, 2010). Além disso, esta técnica associa pesos menores a amostras consideradas *outliers*.

$$\alpha_{t_{gentle}} = \frac{1}{2} \ln \left(\frac{w_{+1} - w_{-1}}{w_{-1} + w_{-1}} \right) \tag{2.5}$$

2.2 Classificador *Support Vector Machine*

O *Support Vector Machine* (SVM) é uma técnica de aprendizado de máquina que pode ser utilizado para classificação binária, isto é, suas amostras pertencem a uma das duas classes $\{c_+, c_-\}$. Seu objetivo é encontrar um hiperplano

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 = 0, \tag{2.6}$$

através das N amostras de treino $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ que é capaz de separar as classes corretamente, onde $\mathbf{x}_i$ é o vetor de características e $y_i \in \{c_+, c_-\}$ indica

Algoritmo 1: Treinamento do *AdaBoost* (SILVA JÚNIOR, 2010)

Entrada: Conjunto de dados de treinamento $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$ e suas classes $\mathbf{y} = \{y_1, \ldots, y_M\}$, com M amostras.
Classificador fraco $f(x)$;
Número de etapas de treinamento M

Saída: Classificador **H** formado por T classificadores fracos h_t e seus pesos α_t

início

Inicia pesos das M amostras $D_1(m) = 1/M$, $m = 1, 2, \ldots, M$

para $t = 1, 2, \ldots, T$ **faça**

1. Obtenha o classificador $h_t \in \{-1, 1\}$ usando os pesos D_t;
2. Calcule o erro e_t do classificador h_t, igual a:

$$e_t = \sum_{m|h_t(x_m) \neq y_m} D_t(m)$$

3. Calcule o escalar α_t:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - e_t}{e_t} \right)$$

4. Atualize os pesos D_t, calculando:

$$D_{t+1} = \begin{cases} \frac{D_t(m) \exp(-\alpha_t)}{z_t}, & \text{se } h_t(\mathbf{x}_m) = y_m. \\ \frac{D_t(m) \exp(\alpha_t)}{z_t}, & \text{se } h_t(\mathbf{x}_m) \neq y_m. \end{cases}$$

onde z_t é um termo de normalização que garante que D_{t+1} seja uma distribuição de probabilidade, ou seja:

$$\sum_m D_{t+1} = 1.$$

fim

Construa o classificador final $\mathbf{H}(\mathbf{x})$:

$$\mathbf{H}(\mathbf{x}) = sign \left[\sum_{i=1}^{T} \alpha_i h_i(\mathbf{x}) \right]$$

Onde $sign(\cdot)$ é a função sinal.

fim

a classe que a amostra pertence (THEODORIDIS; KOUTROUMBAS, 2008). Idealmente, a separação é feita de modo que todas as amostras atendam as condições

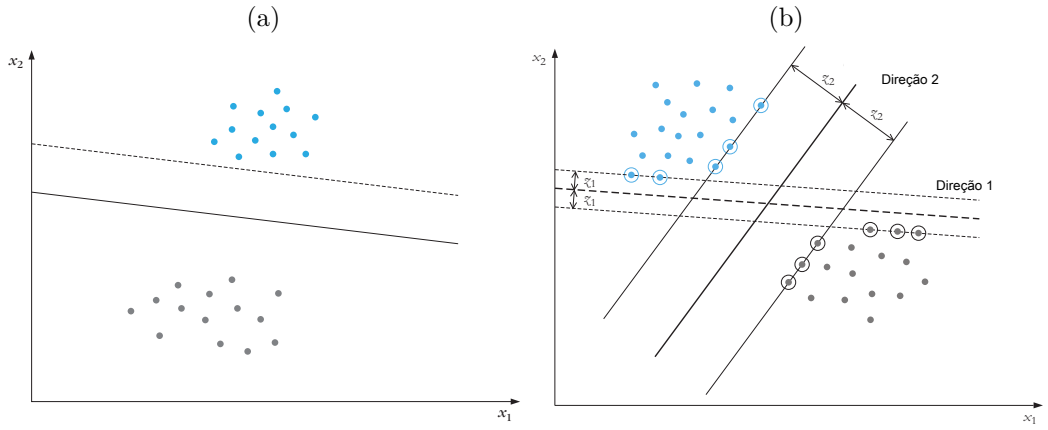
$$\mathbf{w} \cdot \mathbf{x} - w_0 > 0, \forall \mathbf{x} \in c_+ \tag{2.7}$$

$$\mathbf{w} \cdot \mathbf{x} - w_0 < 0, \forall \mathbf{x} \in c_-. \tag{2.8}$$

Encontrados os parâmetros $\mathbf{w}$ e w_0 do hiperplano, dado pela Equação 2.6, capaz de separar tais classes, é possível associar a amostra analisada a uma das duas classes $\{c_+, c_-\}$ utilizando apenas a função sinal, como mostra a Equação 2.9. Porém, existem diversos hiperplanos que atendem as condições das Equações 2.7 e 2.8, ou seja, que são capazes de separar tais amostras, como pode ser visto na Figura 2.

$$f(\mathbf{x}, \mathbf{x}, w_0) = sign(\mathbf{w}^T\mathbf{x} + w_0), \tag{2.9}$$

Figura 2 – Exemplos de problemas de duas classes linearmente separáveis, onde hiperplanos distintos (a) e (b) são capazes de separá-las.



Fonte: Theodoridis e Koutroumbas (2008).

Com o objetivo de aumentar a generalização do classificador, isto é, sua capacidade de identificar corretamente dados desconhecidos, é necessário encontrar o melhor hiperplano g_i dentro de um conjunto de todas as superfícies $g_1, g_2, \ldots, g_t$ no espaço das características $\mathbf{x}_i$, de modo que g_i maximize a distância entre as superfícies que limitam as duas classes. Na Figura 2b, o hiperplano que possui a maior margem é o da direção 2 e, portanto, deve ser o encontrado pela técnica SVM.

Para não favorecer nenhuma das classes, é desejável que o hiperplano escolhido tenha a mesma distância em relação aos pontos mais próximos das duas classes. Ou seja, se a distância entre tais pontos e o hiperplano for z, a margem do classificador deve ser $2z$. Normalizando o hiperplano, é possível impor a condição de que as amostras mais próximas

deste devem satisfazer as condições

$$\mathbf{w}^T\mathbf{x_p} + w_0 = +1 \text{ e} \tag{2.10}$$

$$\mathbf{w}^T\mathbf{x_n} + w_0 = -1, \tag{2.11}$$

onde a condição da Equação 2.10 se aplica para as amostras da classe positiva x_p e a Equação 2.11 para a classe negativa x_n.

A distância entre qualquer ponto $\mathbf{x}$ e o hiperplano $g(\mathbf{x})$ é dada pela Equação 2.12 (THEODORIDIS; KOUTROUMBAS, 2008):

$$z = \frac{|g(\mathbf{x})|}{||\mathbf{w}||}, \tag{2.12}$$

e portanto, a margem M é dada pela Equação 2.13:

$$M = 2 \cdot z = \frac{2|g(\mathbf{x})|}{||\mathbf{w}||} = \frac{2 \cdot 1}{||\mathbf{w}||} \tag{2.13}$$

Como o objetivo do classificador SVM é encontrar o hiperplano $\mathbf{w}$ que maximiza a margem M, seu objetivo pode ser representado pelo problema de otimização quadrática[1] apresentado nas Equações 2.14 e 2.15 (THEODORIDIS; KOUTROUMBAS, 2008):

$$\text{minimizar} \qquad \frac{1}{2}||\mathbf{w}||^2 \tag{2.14}$$

$$\text{sujeito a} \quad y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1, \ i = 1, 2, \ldots, N \tag{2.15}$$

Como o problema da Equação 2.15 é uma otimização não-linear, ela deve atender as condições de Karush-Kuhn-Tucker (KKT) para ter solução ótima. As condições para este problema são dadas pelas Equações 2.16, 2.17, 2.18 e 2.19:

$$\frac{\partial}{\partial \mathbf{w}} \pounds(\mathbf{w}, w_0, \lambda) = \mathbf{0}; \tag{2.16}$$

$$\frac{\partial}{\partial w_0} \pounds(\mathbf{w}, w_0, \lambda) = 0; \tag{2.17}$$

$$\lambda_i \geq 0, \ \ i = 1, 2, \ldots, N; \tag{2.18}$$

$$\lambda_i[y_i(\mathbf{w}^T\mathbf{x_i} + w_0) - 1] = 0, \ \ i = 1, 2, \ldots, N \tag{2.19}$$

onde λ_i é o multiplicador de Lagrange associado a amostra i, e $\pounds(\mathbf{w}, w_0, \lambda)$ é a função de Lagrange, definida pela Equação 2.20:

$$\pounds(\mathbf{w}, w_0, \lambda) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i-1}^{N} \lambda_i[y_i(\mathbf{w}^T\mathbf{x_i} + w_0) - 1]. \tag{2.20}$$

[1] Maximizar uma fração do tipo $\frac{a}{b}$ e minimizar uma fração do tipo $\frac{b}{a}$ são problema equivalentes

A solução do problema é, então, dada pelas Equações 2.21 e 2.22 (THEODORIDIS; KOUTROUMBAS, 2008):

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \tag{2.21}$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0, \tag{2.22}$$

Com a Equação 2.21, é possível calcular o vetor $\mathbf{w}$. A componente de translação do hiperplano w_0 pode ser calculado através das igualdades da Equação 2.19 para os valores de λ_i não-nulos. O vetor de multiplicadores de Lagrange λ assumirá valores diferentes de zero apenas nas amostras mais próximas do hiperplano, que satisfazem as condições 2.10 e 2.11. Estas amostras são chamadas de *Support Vectors*, e são representadas pelas amostras circuladas na Figura 2b.

Para calcular o vetor de multiplicadores de Lagrange λ, a dualidade de Lagrange é utilizada. A representação dual do problema de otimização da Equação 2.15 é dada pelas Equações 2.23, 2.24, 2.25 e 2.26:

$$\text{maximizar} \quad \pounds(\mathbf{w}, w_0, \lambda), \tag{2.23}$$

$$\text{sujeito a} \quad \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i, \tag{2.24}$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0, \tag{2.25}$$

$$\lambda \geq \mathbf{0}. \tag{2.26}$$

Substituindo as Equações 2.24 e 2.25 na Equação 2.23, obtêm-se as Equações 2.27, 2.28 e 2.29:

$$\text{maximizar} \quad \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{ij} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{2.27}$$

$$\text{sujeito a} \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{2.28}$$

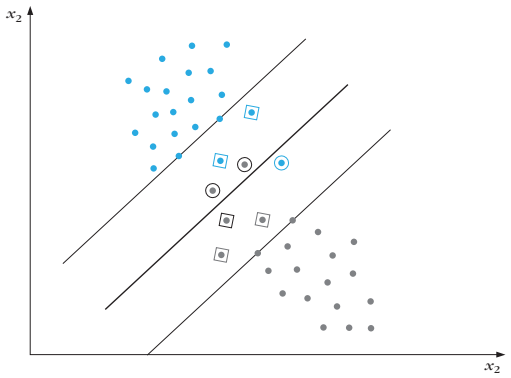
$$\lambda \geq \mathbf{0} \tag{2.29}$$

Este problema é de maximização quadrática com restrições lineares e pode ser resolvido por diversas técnicas, tais como: Gradiente Descendente, *Simulated Annealing*, *Expectation Maximization* (EM) ou Otimização Mínima Sequencial (KINTO, 2011). Assim, é possível encontrar um hiperplano ótimo capaz de separar duas classes linearmente separáveis.

Porém, existem casos em que as classes podem apresentar distribuições como a apresentada na Figura 3. Neste caso, o hiperplano não é capaz de separá-las perfeitamente,

restando amostras dentro da margem, que podem ser corretamente classificadas ou não. Para resolver este problema, são introduzidas um conjunto de variáveis de folga ξ na restrição da Equação 2.15, como mostra a Equação 2.30.

Figura 3 – Classes que não são linearmente separáveis.



Fonte: Theodoridis e Koutroumbas (2008).

$$y_i(\mathbf{w}^T\mathbf{x_i} + w_0) \geq 1 - \xi_i. \tag{2.30}$$

Amostras corretamente classificadas obedecem a restrição original, e portanto, têm a variável ξ_i igual a zero. Variáveis corretamente classificadas, mas dentro da margem possuem ξ_i maior que zero e menor ou igual a 1. Variáveis dentro da margem que são classificadas incorretamente possuem ξ_i maior que 1. É desejável que o classificador obtenha a maior margem com o maior número possível de amostras associadas a $\xi_i = 0$. Matematicamente, isto significa minimizar a Equação 2.31 (THEODORIDIS; KOUTROUMBAS, 2008):

$$J(\mathbf{w}, w_0, \xi) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} I(\xi_i), \tag{2.31}$$

onde ξ é o vetor dos parâmetros ξ_i, o parâmetro $\mathbf{C}$ é uma constante positiva que controla a influência das amostras incorretamente classificadas, e a função I indica se ξ_i é diferente de zero. Esta última, por ser descontínua, inviabiliza a utilização da Equação 2.31 no problema de otimização. Então, uma forma aproximada é utilizada, dada pela Equações 2.32, 2.33 e 2.34.

$$\text{minimizar} \quad \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} \xi_i \tag{2.32}$$

$$\text{sujeito a} \quad y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1 - \xi_i, \;\; i = 1, 2, \ldots, N \tag{2.33}$$

$$\xi_i \geq 0, \;\; i = 1, 2, \ldots, N \tag{2.34}$$

De forma similar a adotada para o caso do problema de otimização 2.14, através das condições de Karush-Kuhn-Tucker e da função Lagrangiana para este novo problema,

é possível encontrar os multiplicadores lagrangianos através do problema de otimização das Equações 2.35, 2.36 e 2.37.

$$\text{maximizar} \quad \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{ij} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \tag{2.35}$$

$$\text{sujeito a} \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{2.36}$$

$$0 \leq \lambda_i \leq C, \quad i = 1, 2, \ldots, N \tag{2.37}$$

Com o vetor λ encontrado neste problema, é possível encontrar o plano através da Equação 2.38

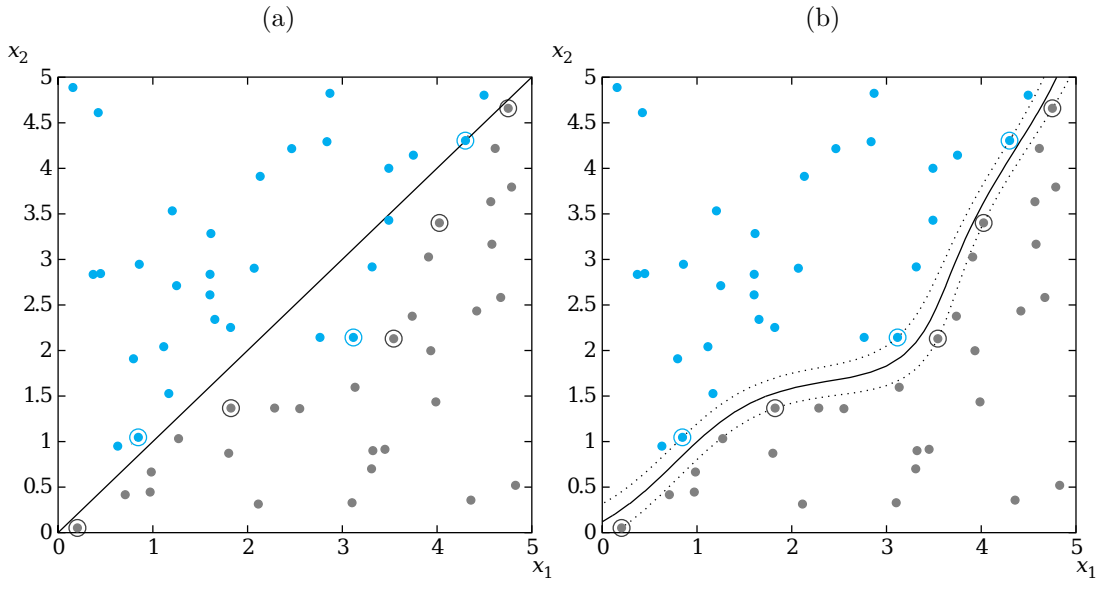
$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i, \tag{2.38}$$

e w_0 através da Equação 2.39 (THEODORIDIS; KOUTROUMBAS, 2008)

$$\lambda_i [y_i (\mathbf{w}^T \mathbf{x_i} + w_0) - 1 + \xi_i] = 0, \quad i = 1, 2, \ldots, N. \tag{2.39}$$

Mesmo com essa abordagem que lida com *outliers*, existem situações em que as duas classes não possuem uma separação razoável através de um plano e, portanto, o problema de otimização do SVM não será capaz de encontrar um bom plano de separação entre as amostras, como mostrado na Figura 4a.

Figura 4 – Exemplo de classes não linearmente separáveis. Superfície de separação do SVM usando kernel linear (a) e kernel RBF (b).



Fonte: Adaptado de Theodoridis e Koutroumbas (2008).

Para lidar com este tipo de problema, Boser, Guyon e Vapnik (1992) propuseram a utilização de uma função *kernel*, que é uma espécie de produto interno, onde uma amostra é mapeada para um novo espaço, onde seria possível realizar uma separação linear entre as classes (KINTO, 2011). Alguns exemplos de funções *kernel* são apresentados na Tabela 1, onde p é o grau do *kernel* polinomial e γ é a largura da função RBF.

Tabela 1 – Funções *Kernel* utilizadas no classificador SVM (KINTO, 2011)

Funções Kernel	
Kernel Linear	$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \cdot \mathbf{y}$
Kernel Polinomial	$K(\mathbf{x}, \mathbf{y}) = (1 + (\mathbf{x}^T \cdot \mathbf{y}))^p$
Kernel Radial Basis Function (RBF)	$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma}\|\|\mathbf{x} - \mathbf{y}\|\|^2\right)$

Substituindo a Equação 2.21 na Equação 2.6, a Equação 2.40 é obtida:

$$g(x) = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + w_0, \tag{2.40}$$

onde existe um produto interno $\mathbf{x}_i^T \mathbf{x}$, que é a função *kernel* linear $K(\mathbf{x}_i, \mathbf{x})$. Ao substituir esta função pelo *kernel* RBF, por exemplo, é possível separar as classes da Figura 4a, como apresentado na Figura 4b (THEODORIDIS; KOUTROUMBAS, 2008).

2.3 Descritor *Histogram of Oriented Gradients*

Proposto por Dalal e Triggs (2005), o descritor HOG é um operador que calcula um histograma de gradientes em uma densa grade de células, sobrepostas e uniformemente espaçadas na imagem. Originalmente proposto para utilização em aplicações que envolvam detecção de pedestres, o descritor HOG pode ser utilizado para detecção de objetos em geral (VONDRICK et al., 2013).

Este operador, inicialmente, calcula os gradientes de primeira ordem da imagem, capturando contornos, silhuetas e texturas da imagem. Estes gradientes são obtidos através da convolução da imagem original com as máscaras $[-1\ 0\ 1]^T$ e $[-1\ 0\ 1]$ para as direções vertical e horizontal, respectivamente. Finalmente, estes são combinados, onde o gradiente resultante de cada pixel é dado pelo máximo entre os valores nas duas direções.

Em um segundo momento, o histograma de orientações do gradiente é calculado. A janela da imagem analisada é dividida em pequenas regiões espaciais de $n_{pc} \times n_{pc}$, denominadas células. O histograma das orientações do gradiente de cada célula é calculado dividindo o intervalo de orientações $[0^o, 180^o]$ em 9 divisões igualmente espaçadas. A contribuição de cada pixel para o histograma é ponderada pelo valor do seu gradiente.

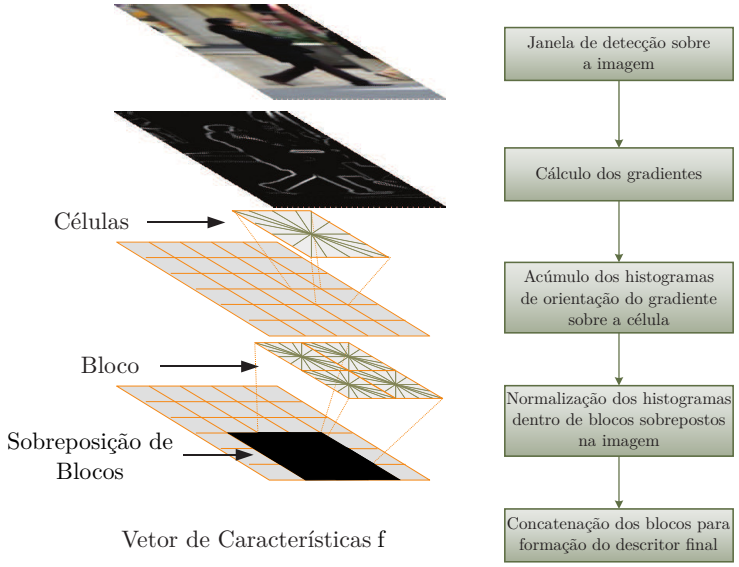
Em um terceiro estágio, uma normalização local dos histogramas em regiões espaciais da imagem, denominadas blocos, é feita. Cada bloco é formado por um conjunto de 2×2 células, e "deslizam" por elas com passo de 1 célula. Assim, cada célula fará parte de 4 blocos diferentes, com exceção daquelas nas fronteiras das imagens. Deste modo, cada célula contribui para a normalização do histograma de diferentes blocos.

Supondo que $\mathbf{v}$ seja o descritor de cada bloco, formado pela concatenação dos histogramas de suas células, não normalizado, $||\mathbf{v}||_2$ sua norma Euclidiana, e ϵ uma constante para evitar divisão por zero, a normalização é feita através da Equação 2.41

$$\mathbf{v}^* = \frac{\mathbf{v}}{\sqrt{||\mathbf{v}||_2^2 + \epsilon^2}}. \tag{2.41}$$

O passo final do descritor HOG consiste em concatenar os descritores de todos os blocos presentes na janela de detecção, criando assim o descritor final. Uma imagem demonstrando o processo de obtenção do descritor HOG para uma janela contendo um pedestre é apresentada na Figura 5.

Figura 5 – Exemplo de obtenção do descritor HOG para uma janela contendo um pedestre.



Fonte: Adaptado de Cosmo, Salles e Ciarelli (2015).

2.4 Descritor *Local Binary Pattern*

Proposto por Ojala, Pietikainen e Harwood (1994), o descritor LBP é um operador que transforma uma imagem em uma matriz de números inteiros que descrevem a aparência local da imagem. Estes números ou suas estatísticas, usualmente os histogramas, são usados para analisar imagens em diversas aplicações (PIETIKÄINEN et al., 2011).

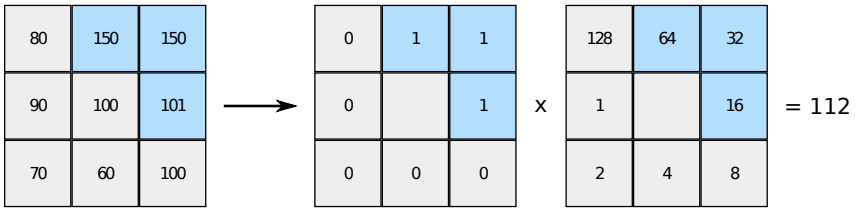
Originalmente, o operador LBP analisa um bloco de tamanho 3×3 centralizado no pixel analisado. Os pixels deste bloco são limiarizados pelo valor central, assumindo valor 1 caso a diferença x entre as intensidades seja maior que zero (isto é, a intensidade do pixel seja superior a central do bloco) e zero caso contrário, como mostra a Equação 2.42.

$$s(x) = \begin{cases} 1, & x > 0 \\ 0, & \text{caso contrário} \end{cases} \tag{2.42}$$

O descritor LBP associa um valor binário para cada um dos oito pixels vizinhos, que posteriormente formam um número binário através da concatenação de seus valores em uma ordem pré-determinada. Este processo pode ser definido através da Equação 2.43 e um exemplo é ilustrado na Figura 6.

$$LBP(p_c) = \sum_{n=0}^{7} 2^n s(i_n - i_c) \tag{2.43}$$

Figura 6 – Exemplo do descritor LBP de um pixel.

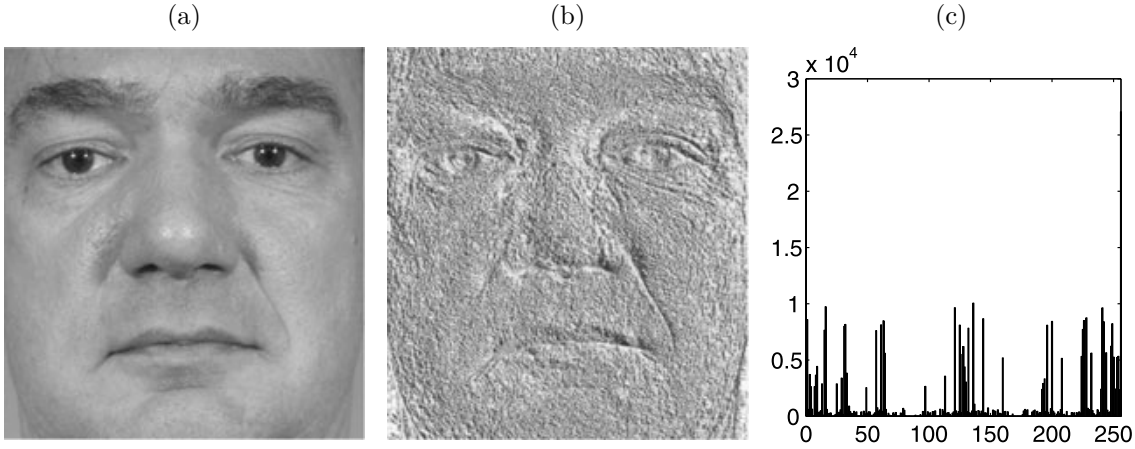


Fonte: Adaptado de Opitz (2013).

Um exemplo do processo de extração do descritor LBP de uma imagem pode ser visto na Figura 7, onde dada uma imagem em tons de cinza, como a da Figura 7a, a Equação 2.43 é aplicada a todos os pixels, resultando na Figura 7b. Finalmente, o histograma do descritor LBP da imagem é obtido, como mostra a Figura 7c. Este histograma possui dimensão 256, devido ao fato de que cada pixel assume um valor de 8 bits, obtido através de sua vizinhança.

O histograma do descritor LBP pode ser calculado tanto para a imagem inteira, como mostra a Figura 7, quanto para janelas da imagem, de forma similar ao cálculo do descritor HOG.

Figura 7 – Imagem Original (a). Imagem dos descritores LBP (b). Histograma geral da imagem LBP (c).



Fonte: Pietikäinen et al. (2011).

3 Técnicas de Segmentação de Texto em Imagens

Este capítulo descreve algumas técnicas propostas na literatura que utilizam os descritores e classificadores apresentados no Capítulo 2 com o objetivo de segmentar caracteres de imagens. Entre os diversos trabalhos publicados nesta área, se destacam duas principais categorias de sistemas (NEUMANN; MATAS, 2015): os que encontram caracteres deslizando janelas pela imagem e os que utilizam análise de componentes conectados para isto.

Com base em diversos outros sistemas que possuem o objetivo de detectar um tipo específico de característica ou objeto, como faces (VIOLA; JONES, 2004) ou presença de pedestres em imagens (COSMO; SALLES; CIARELLI, 2015), sistemas da primeira categoria buscam caracteres em um subconjunto de janelas da imagem original. Este tipo de sistema possui a vantagem de maior robustez a ruído e distorções, como as causadas por movimento ou falta de foco do dispositivo responsável por adquirir a imagem, mas tendem a possuir um custo computacional maior, pois necessitam analisar um grande número de janelas para conseguirem encontrar diversas variações de características, como tamanho, inclinações, razão de aspecto, entre outros (NEUMANN; MATAS, 2015).

Já aqueles da segunda categoria consideram que caracteres possuem pelo menos uma característica, como a cor (MOSLEH; BOUGUILA; HAMZA, 2012), cujo valor varia pouco em todos os seus pixels, sendo possível extrair candidatos através de análise de componentes conectados. Isto torna o sistema menos custoso, já que a maioria dos caracteres pode ser extraída em apenas um passo, porém com uma maior sensibilidade a componentes unidos, onde duas ou mais letras compõem um único componente conectado, distorcidos ou afetados por ruído (NEUMANN; MATAS, 2015; CHEN et al., 2011).

Trabalhos nesta categoria usam, principalmente, bordas, o conceito de *Extremal Regions* (ER) ou ainda as *Maximally Stable Extremal Regions* (MSER). Em 2015, um detector dessa categoria, chamado de FASText e baseado em *keypoints*, foi proposto com as vantagens de maior velocidade e melhor precisão que o detector MSER (BUŠTA; NEUMANN; MATAS, 2015).

Na Seção 3.2 são apresentados métodos que utilizam janelas deslizantes para segmentar candidatos a texto de imagens reais. Na Seção 3.2 são apresentadas considerações feitas para detectar caracteres de imagens reais a partir de bordas, bem como um método que as utilizam. Na Seção 3.3 são apresentadas as ER e as MSER. Na Seção 3.4 é apresentada a técnica FASText, baseada em *keypoints*. Na Seção 3.4.1 é apresentado o

método proposto por Silva e Ciarelli (2016) para detectar caracteres.

3.1 Métodos Baseados em Janelas Deslizantes

Também chamados de métodos baseados em textura, este tipo de método na maioria das vezes se resume a classificação binária de descritores extraídos de janelas deslizantes sobre a imagem. Usualmente, diversas escalas e rotações da imagem são utilizadas para detectar candidatos de fontes e tamanhos diferentes.

No trabalho de Wang, Babenko e Belongie (2011), é proposta a utilização de descritores como o *Histogram of Oriented Gradients* (HOG) (DALAL; TRIGGS, 2005) associados a um classificador do tipo *Random Ferns* (OZUYSAL; FUA; LEPETIT, 2007) para encontrar caracteres em janelas de várias escalas de uma imagem, que é treinado utilizando janelas contendo texto de imagens sintéticas e reais, como as apresentadas na Figura 8. De forma similar, Mishra, Alahari e Jawahar (2012) propõem a utilização do mesmo descritor HOG com o classificador *Suppor Vector Machine* (SVM) para detectar e classificar janelas da imagem analisada.

Figura 8 – Exemplos de janelas retiradas de imagens sintéticas (a) e de imagens reais (b) contendo caracteres utilizadas na etapa de treinamento de um classificador de caracteres.

(a) (b)

Fonte: Wang, Babenko e Belongie (2011).

Em trabalhos como de Coates et al. (2011) são utilizados métodos de aprendizado não supervisionados de *features*, treinados com pedaços da imagem de tamanho 8×8 pixels que passaram por um processo de branqueamento ZCA[1] (KESSY; LEWIN; STRIMMER, 2015). Estes pedaços são extraídos aleatoriamente de amostras de textos, e agrupados com uma variante do método *k-means*. Dado um conjunto $X \in \mathbb{R}^{n \times m}$ de m amostras, esta técnica considera distância angular entre amostras e gera um dicionário $D \in \mathbb{R}^{n \times k}$, minimizando a Equação 3.1.

$$\sum_i ||Ds_i - x_i||^2 \tag{3.1}$$

[1] Também chamada de Transformação ou Branqueamento de Mahalanobis (KESSY; LEWIN; STRIMMER, 2015)

Cada coluna de D é o centro de um *cluster* e representa um vetor de uma base normalizada; x_i é uma amostra de X e s_i é um vetor com apenas um valor diferente de zero na posição referente ao *cluster* que ela pertence, cujo valor é igual ao produto interno entre x_i e a coluna de D mais próxima a esta amostra. O dicionário é utilizado no treinamento de uma *Convolutional Neural Network* (CNN), que irá detectar regiões de texto na imagem (COATES et al., 2011).

Baseado no trabalho de Coates et al. (2011) e Wang, Babenko e Belongie (2011), Zhu e Zanibbi (2016) também utilizam CNN em seu sistema, porém combinando um detector grosseiro com um para ajuste fino, onde cada um destes detectores utiliza 1000 centros de *clusters*. O primeiro desliza janelas da imagem de tamanho 32×32, com passo de 16 pixels, e as analisa com a CNN. Para melhorar a resposta desta etapa, o sistema utiliza informação contextual de um grid de 9×9 janelas. O resultado desta análise é, então, acumulado nos pixels referentes a janela, que se localiza no centro do *grid* analisado.

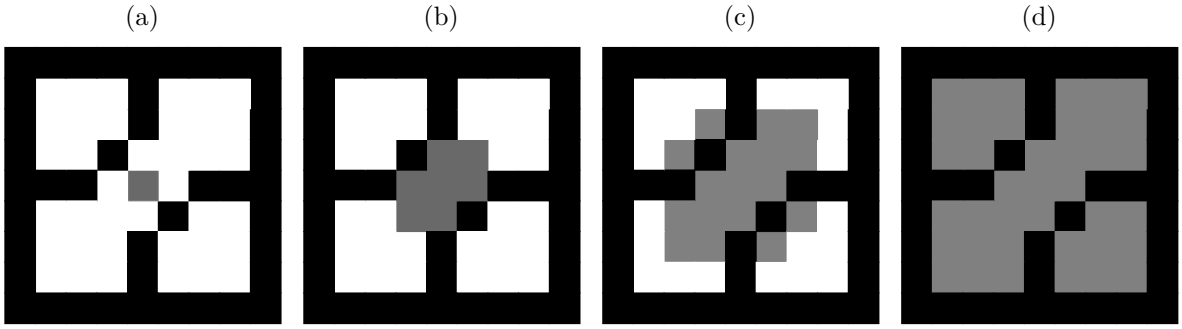
Assim, este detector gera um mapa de confiança, onde cada pixel possui valor proporcional à probabilidade de pertencer a uma região de texto. Um limiar é aplicado neste mapa para eliminar regiões com menor probabilidade. Nas restantes, o detector para ajuste fino é aplicado, onde o passo é reduzido para 1 pixel e não são utilizadas informações contextuais. Para detectar os caracteres, é feita uma busca em um *grid* de diferentes razões de aspecto e rotações. Além disso, o sistema repete todo o processo em uma pirâmide da imagem, onde cada nível é 10% menor que o anterior.

O resultado do detector de ajuste fino também é um mapa de confiança, porém mais preciso que o anterior. Novamente, um limiar é aplicado para eliminar pontos com baixa probabilidade de pertencerem a um caractere, enquanto os restantes são utilizados como sementes para uma técnica de *flood-fill* (TREUENFELS, 1994) para extrair componentes conectados. Partindo de um ponto escolhido, esta técnica preenche os pixels de intensidade similar da vizinhança, até que não seja possível preencher nenhum pixel sem ultrapassar o limiar estabelecido. Um exemplo deste processo é apresentado na Figura 9.

3.2 Métodos Baseados em Detecção de Bordas

Bordas são detectadas nas regiões onde existem variações bruscas de intensidade entre pixels vizinhos. A detecção destas é uma ferramenta fundamental em processamento de imagens e visão computacional. Com elas, é possível extrair características e detectar objetos de forma simples, rápida e, muitas vezes, com resultados suficientes para diversas aplicações (GONZALEZ; WOODS, 2000). Idealmente, a detecção de bordas resulta apenas em um conjunto de curvas que limitam objetos ou cores distintas. Nas imagens geradas em computadores, por exemplo, a utilização de algoritmos simples pode ser suficiente para detecção de tais curvas.

Figura 9 – Um exemplo da técnica *flood-fill*. Partindo de um ponto (a), a vizinhança é preenchida (b). O processo é repetido (c) até que não seja possível preencher nenhuma região com intensidade similar a inicial (d).



Fonte: Produção do próprio autor.

Porém, imagens reais contêm imperfeições, sejam causadas por ruídos ou problemas de amostragem e compressão, que prejudicam a detecção de bordas importantes e causam um aumento no número de detecções desnecessárias. Considerando este problema, é necessária a utilização de técnicas robustas para extração de bordas. Entre as mais conhecidas está o Método de Canny (CANNY, 1986), utilizado por diversos pesquisadores em diversas aplicações de processamento de imagem e visão computacional (GONZALEZ; WOODS, 2000).

Para a extração de caracteres através das bordas de uma imagem, as seguintes considerações podem ser feitas: os pixels de um caractere e do fundo em que ele está inserido têm intensidades pouco variantes, e portanto, o gradiente dos pixels na fronteira entre estas regiões possui pouca variação. A partir destas considerações, Lu et al. (2015) calculam três características: F_1, F_2 e F_3.

A primeira, F_1, é a razão entre a média ($\mu(G_e)$) e o desvio padrão ($\sigma(G_e)$) do gradiente G_e dos pixels de uma borda conectada, como mostra a Equação 3.2. Já as demais características, F_2 e F_3 levam em consideração as *bounding-boxes*[2] associadas as bordas conectadas encontradas pelo método Canny.

A característica F_2 considera que caracteres possuem bordas aproximadamente fechadas e, portanto, suas *bounding-boxes* possuirão um número maior ou igual a dois de pixels de borda em todas as suas linhas e colunas. Assim, a razão entre o número de linhas e colunas que apresentam esta característica e o número total destas é calculada e utilizada como expoente de uma constante R, como mostra a Equação 3.3, onde W e H representam a largura e altura da imagem, respectivamente, cn_i e cn_j representam o

[2] Assumindo que uma borda é formada por um conjunto de pixels conectados com coordenadas contidas em $B = \{(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})\}$, a *bounding-box* que a contém tem canto superior esquerdo localizado em $Tl = (min(\mathbf{x}), min(\mathbf{y}))$ e canto inferior direito localizado em $Br = (max(\mathbf{x}), max(\mathbf{y}))$, onde $\mathbf{x} = \{x_0, ..., x_{n-1}\}$ e $\mathbf{y} = \{y_0, ..., y_{n-1}\}$

número de pixels de borda em cada linha i e em cada coluna j da *bounding-box* da borda, e a função $f(x)$ possui valor um se seu argumento é maior ou igual a dois e zero em caso contrário, como mostra a Equação 3.5. Segundo Lu et al. (2015), R pode possuir valores entre 10 e 30, sendo 20 o utilizado.

Já a característica F_3 também considera que caracteres possuem bordas aproximadamente fechadas, e portanto o número de pixels em cada linha e coluna é múltiplo de 2. Assim, a razão entre o número de linhas e colunas que apresentam esta característica e o número total de linhas e colunas é calculado, como mostra a Equação 3.4, onde a função $g(x)$ retorna um quando x é par e zero caso contrário, como mostra a Equação 3.6.

$$F_1 = \frac{\mu(G_e)}{\sigma(G_e) + \xi} \tag{3.2}$$

$$F_2 = R^{\frac{\sum_{i=1}^{H} f(cn_i) + \sum_{j=1}^{W} f(cn_j)}{W+H}} \tag{3.3}$$

$$F_3 = \frac{\sum_{i=1}^{H} g(cn_i) + \sum_{j=1}^{W} g(cn_j)}{W + H} \tag{3.4}$$

$$f(x) = \begin{cases} 1, & x \geq 2 \\ 0, & \text{caso contrário} \end{cases} \tag{3.5}$$

$$g(x) = \begin{cases} 1, & \text{x é par} \\ 0, & \text{caso contrário} \end{cases} \tag{3.6}$$

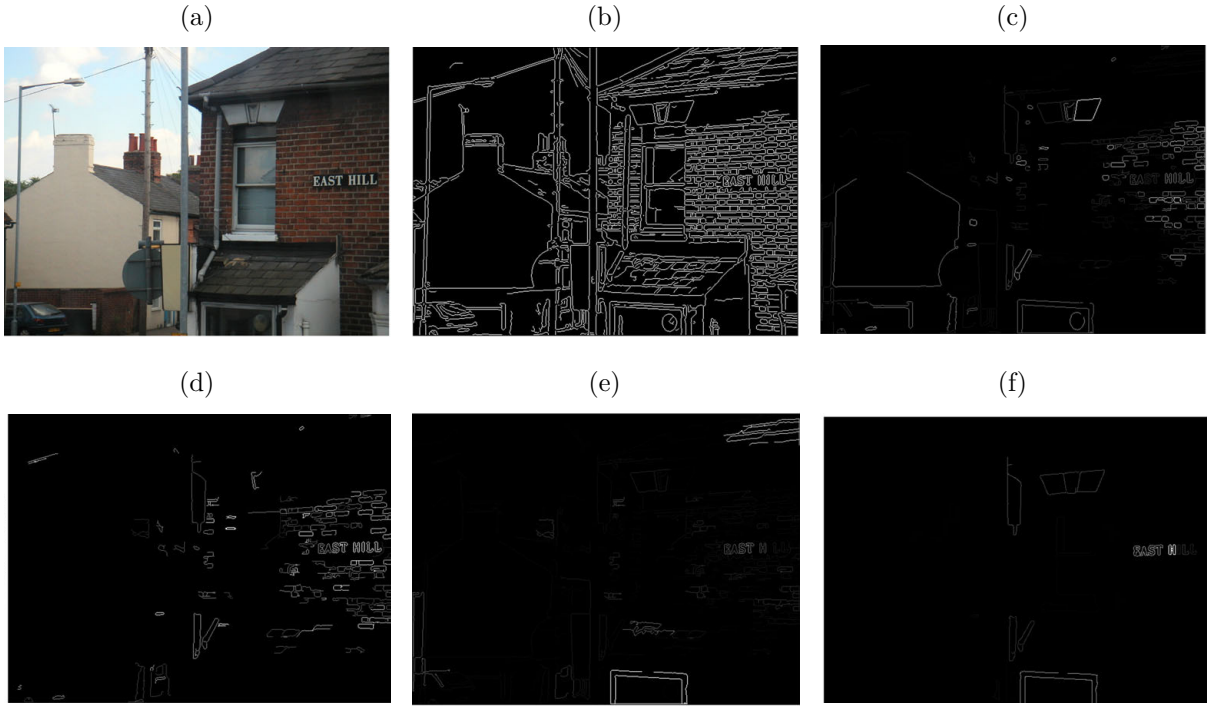
Segundo Lu et al. (2015), as três características tendem a destacar bordas de caracteres em relação a outros objetos. Para evidenciar esse destaque, o processo é repetido para I canais de cor e J escalas da imagem [3], e a multiplicação elemento a elemento [4] de F_1, F_2 e F_3 para cada combinação é obtida. Finalmente, a soma destas multiplicações é obtida, como apresenta a Equação 3.7. Deste modo, é possível eliminar a maior parte de bordas não relacionadas a caracteres eliminando pixels com valor inferior a média da matriz M. Na Figura 10 é apresentado um exemplo do processo feito por (LU et al., 2015) para encontrar bordas de caracteres.

$$M = \frac{1}{I \times J} \sum_{i=1}^{I} \sum_{j=1}^{J} F_{1_{i,j}} \circ F_{2_{i,j}} \circ F_{3_{i,j}} \tag{3.7}$$

[3] O autor originalmente utiliza os três canais de cor YUV e as escalas $\{0,2,0,4,0,6,0,8,1,0,1,2,1,4\}$ para destacar caracteres.

[4] A operação $\mathbf{A} \circ \mathbf{B}$ representa o produto de Hadamard das matrizes $\mathbf{A}$ e $\mathbf{B}$, ou a multiplicação elemento a elemento destas.

Figura 10 – Processo de extração de candidatos a caracteres através de bordas. Imagem original (a), suas bordas (b), suas características F1 (c), F2 (d) e F3 (e), e a multiplicação destas (f).



Fonte: Adaptado de Lu et al. (2015).

Porém, não é possível extrair caracteres apenas com as bordas de uma imagem. É necessária a utilização de uma técnica capaz de segmentar os caracteres a partir das bordas detectadas. Lu et al. (2015) utilizam uma limiarização adaptativa na região de cada borda para extrair os componentes conectados da imagem.

Muitas vezes, as técnicas empregadas para a detecção de bordas e segmentação dos caracteres são simples, o que torna a utilização deste tipo de técnica comum para detecção de caracteres em imagens naturais. Porém, assim como a técnica MSER, técnicas baseadas em extração de bordas também são sensíveis a caracteres escondidos, distorcidos ou agrupados, além de apresentar dificuldades nos caracteres inseridos em fundo não homogêneo.

3.3 Métodos Baseados em *Extremal Regions* (ER)

Proposto por Matas et al. (2004) como um método para encontrar correspondências de regiões em duas imagens do mesmo local com diferentes pontos de vista, as ER se mostraram úteis em diversos outros problemas, como classificação de células (ARTETA et al., 2012) ou detecção de caracteres em imagens (NEUMANN; MATAS, 2015; CHEN et al., 2011; YIN et al., 2014). Para uma definição formal de ER, três definições auxiliares

são necessárias (MATAS et al., 2004):

Definição 3.1. *Imagem I é um mapeamento $I\colon \mathbb{D} \subset \mathbb{Z}^2 \to \mathbb{S}$. Onde $\mathbb{S}$ é um conjunto totalmente ordenado, onde uma relação binária $\leq$ é bem definida. Para imagens em tons de cinza, por exemplo, o conjunto pode ser $\mathbb{S} = \{0, 1, \ldots, 255\}$.*

Definição 3.2. *Uma região $\mathbb{Q}$ é um subconjunto contíguo de $\mathbb{D}$, isto é: $\forall p, q \in \mathbb{Q}$ existe uma sequência $p, a_1, a_2, \ldots, a_n, q$ onde todas as relações $p\mathbb{A}a_1, a_i\mathbb{A}a_{i+1}, a_n\mathbb{A}q$ são verdadeiras. A relação $\mathbb{A} \subset \mathbb{D} \times \mathbb{D}$ é de vizinhança (ou conectividade). Por exemplo, a relação de conectividade-4 torna dois pixels $p, q \in \mathbb{D}$ vizinhos ($p\mathbb{A}q$) se e somente se $\sum_{i=0}^{d} |p_i - q_i| \leq 1$.*

Definição 3.3. *Borda externa de uma região $\partial\mathbb{Q} = \{q \in \mathbb{D} \setminus \mathbb{Q}\colon \exists p \in \mathbb{Q}\colon qAp\}$, isto é, $\partial\mathbb{Q}$ é o conjunto de pixels não pertencentes a $\mathbb{Q}$ mas que estão conectados a algum pixel de $\mathbb{Q}$ através da relação $\mathbb{A}$.*

As ER então são definidas como (MATAS et al., 2004):

Definição 3.4. *As Extremal Regions $ER \subset \mathbb{D}$ de uma imagem é o conjunto de regiões onde uma das seguintes propriedades é válida: $\forall p \in \mathbb{Q}, q \in \partial\mathbb{Q} : I(p) > I(q)$ para regiões de intensidade máxima ou $\forall p \in \mathbb{Q}, q \in \partial\mathbb{Q} : I(p) < I(q)$ para regiões de intensidade mínima.*

Em outras palavras, ER são regiões de uma imagem em que todos os seus pixels têm intensidade maior ou menor que todos os pixels da borda externa desta região. Matas et al. (2004) também propuseram as *Maximally Stable Extremal Regions* (MSER), definidas como:

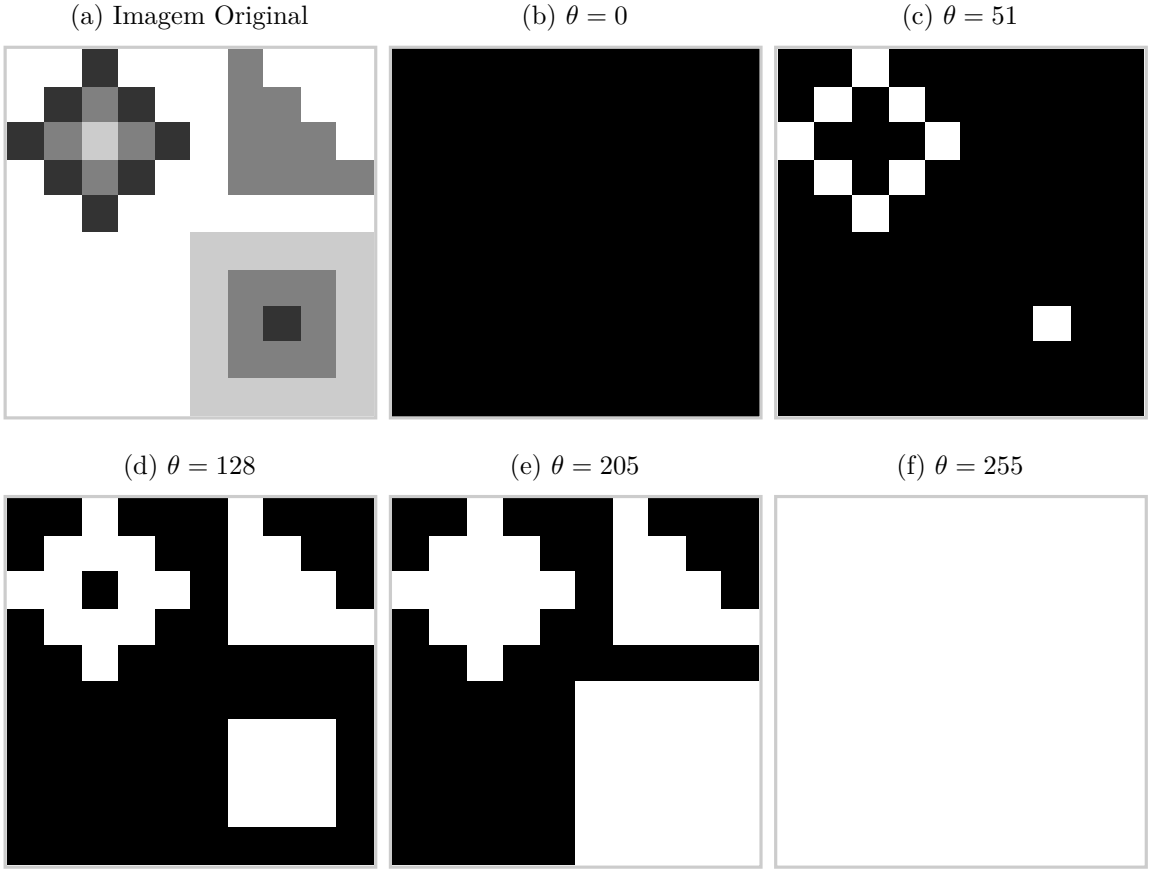
Definição 3.5. *Considerando uma sequência de Regiões $\mathbb{Q}_1, \mathbb{Q}_2, \ldots, \mathbb{Q}_n$, todas extremas e aninhadas, onde $\mathbb{Q}_i \subset \mathbb{Q}_{i+1}$, uma ER $\mathbb{Q}_i^*$ é maximamente estável se e somente se a função $q(i) = |\mathbb{Q}_{i+\Delta} \setminus \mathbb{Q}_{i-\Delta}|/|\mathbb{Q}_i|$ possui um mínimo local em i^*, onde $|\cdot|$ é a cardinalidade, o símbolo $\setminus$ indica a diferença de dois conjuntos, e $\Delta \in \mathbb{S}$ é uma constante.*

Ou seja, se uma ER $\mathbb{Q}_i^*$ conter uma região $\mathbb{Q}_{i-\Delta}$ e estiver contida em uma região $\mathbb{Q}_{i+\Delta}$, de modo que a diferença de área entre estas (normalizada pela área de $\mathbb{Q}_i^*$) seja mínima dentre seu conjunto de regiões, ela é considerada maximamente estável (MSER). Uma definição menos formal destes tipos de regiões pode ser obtida através de um processo de limiarização, utilizando uma função como a apresentada na Equação 3.8, onde um pixel p da imagem terá sua intensidade $I(p)$ comparada a um limiar θ, resultando no valor $I_b(p)$.

$$I_b(p, \theta) = \begin{cases} 1, & I(p) \leq \theta \\ 0, & \text{caso contrário} \end{cases} \tag{3.8}$$

Dada uma imagem, é possível obter um conjunto de imagens aplicando a Equação 3.8 com todos os possíveis valores de $\theta = \{0, 1, \ldots, 255\}$, onde a primeira imagem deste conjunto possui pixels não nulos apenas nos locais com intensidade zero da imagem original. Ao longo dos limiares, componentes conectados aparecem e aumentam sua área, até que se juntem e formem uma imagem com todos os pixels iguais a 1. Este processo é ilustrado na Figura 11.

Figura 11 – Processo de obtenção das Extremal Regions. Imagem original (a), com pixels de intensidade 51, 128, 205 e 255. O processo de limiarização de (a), utilizando a Equação 3.8, para vários valores de θ (b)-(f). Os componentes conectados destas imagens são as ER.



Fonte: Produção do próprio autor.

Neste conjunto de imagens binárias, é possível perceber que componentes conectados na imagem referente ao limiar θ_{i+1} são os mesmos pixels da imagem referente ao limiar θ_i adicionados aos pixels de intensidade θ_{i+1}. Isto significa que todos os pixels vizinhos a cada componente conectado têm intensidade maior que todos os seus internos. Todos os componentes conectados desse conjunto de imagens podem ser considerados ER.

Ao utilizar a função de limiarização da Equação 3.8, o processo anterior irá obter todas as ER de intensidade mínima, pois todos os pixels tem valor menor que os externos.

Para obtermos as ER de intensidade máxima, é necessário considerar como valor 1 apenas os pixels cuja intensidade é maior que o limiar da Equação 3.8.

Entre todos os componentes conectados extraídos, existirão aqueles cujo tamanho praticamente não irá se alterar em um determinado número de limiares, que é o parâmetro Δ do algoritmo. Estas regiões são as MSER (MATAS et al., 2004).

Fazendo as mesmas considerações da Seção 3.2, ou seja, que usualmente caracteres possuem uma cor com pouca variação e estão inseridos em regiões de cor distinta do caractere, é possível assumir que eles podem ser detectados como ER ou até MSERs. Estas técnicas conseguem detectar regiões, caso não seja aplicado um limite de área, de tamanho 1 até N (número de pixels da imagem), e portanto conseguem extrair elementos de diversos tamanhos utilizando apenas uma escala. Além disso, o conjunto de MSER de uma imagem pode ser encontrado de forma eficiente, com complexidade linear (NISTÉR; STEWÉNIUS, 2008). Estas vantagens tornam o uso deste tipo de técnica comum para detecção de caracteres em imagens naturais.

Normalmente são extraídos caracteres de diversas escalas de uma imagem utilizando a técnica MSER, devido a uma limitação de área do algoritmo de extração destas regiões. Um método de redimensionamento de imagens utilizado é através de uma pirâmide gaussiana, para compensar caracteres quebrados ou formados por múltiplos objetos (NEUMANN; MATAS, 2015).

Para compensar a sensibilidade da técnica ER na detecção de caracteres sobrepostos, distorcidos ou agrupados, alguns autores a utilizam combinando com outras técnicas, melhorando a detecção com a desvantagem de um maior custo computacional.

Em Chen et al. (2011), é utilizada uma combinação de detecção de bordas Canny com a técnica MSER para extrair candidatos a caracteres. O método Canny é robusto a distorções de caracteres, e pode ser utilizado como um limitador dos componentes extraídos pela técnica MSER, ajudando a lidar com o problema de caracteres distorcidos. Este processo é chamado de *Edge-Enhanced Maximally Stable Extremal Regions*, e um exemplo pode ser visualizado na Figura 12.

Figura 12 – Candidatos extraídos com a técnica MSER e as bordas Canny em vermelho (a), e candidatos extraídos com a técnica *Edge-Enhanced* MSER (b).

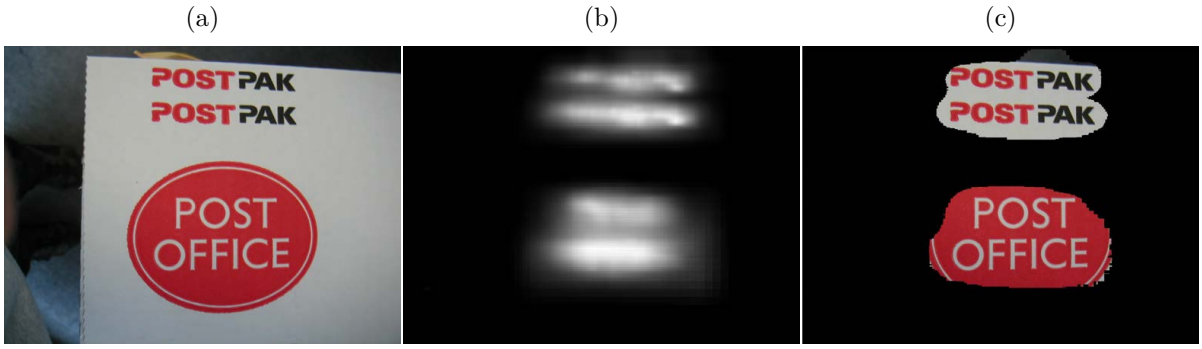
(a) MSER (b) Edge-Enhanced MSER

Fonte: Chen et al. (2011).

Já Opitz (2013) também utiliza a técnica MSER para extrair candidatos, porém limita sua busca em regiões da imagem com maior probabilidade de existência de elementos textuais, como pode ser visto na Figura 13c. Para achar tais regiões, o autor encontra um Mapa de Confiança baseado na imagem analisada, utilizando janelas deslizantes em várias escalas da imagem, de forma similar aos métodos apresentados na Seção 3.1. Este tipo de técnica, como vista na Seção 3.1, pode ser interpretada como o cálculo de descritores em uma janela deslizante, utilizados em um classificador que irá dizer se a janela é uma região de interesse. Porém, o uso deste tipo de técnica implica em um aumento do custo computacional do sistema. Para evitar grandes atrasos, é desejável a utilização de descritores e classificadores rápidos nesta etapa (OPITZ, 2013).

Com o objetivo de reduzir o custo computacional, Opitz (2013) utiliza uma combinação de descritores simples baseados nas técnicas *Local Binary Pattern* (OJALA; PIETIKAINEN; HARWOOD, 1994) e *Local Ternary Pattern* (TAN; TRIGGS, 2007) e um classificador AdaBoost (FRIEDMAN et al., 2000). Com esta combinação, é possível eliminar um grande número de falsos positivos com pouco aumento no custo computacional. Um exemplo do mapa de confiança obtido para a imagem apresentada na Figura 13a é apresentado na Figura 13b.

Figura 13 – Um imagem (a) e seu mapa de confiança (b). MSER serão detectados a partir da imagem (c).



Fonte: Opitz (2013).

3.4 Métodos Baseados em *Keypoints*

Em muitas aplicações de processamento de imagens e visão computacional, existem pontos capazes de representar informações sobre objetos ou formas, podendo ser utilizados em tarefas como reconhecimento, casamento de imagens, rastreamento, entre outras (ROSTEN; PORTER; DRUMMOND, 2010). Estes são chamados de pontos de interesse ou pontos-chave (*keypoints*). Entre os tipos de *keypoints* estão os cantos (ou *corners*), que podem ser detectados por diversas técnicas, como o *Harris Corner Detector* e a técnica

Features from Accelerated Segment Test (FAST). Esta última se mostrou capaz de detectar tais pontos de interesse em alguns caracteres de imagens naturais, com exceção daqueles que não possuem um canto (como a letra "O" ou o dígito "8") (BUŠTA; NEUMANN; MATAS, 2015). Outra desvantagem deste tipo de detector é o grande número de falsos positivos e múltiplas detecções no mesmo caractere, causando um aumento desnecessário no custo computacional da etapa de extração de caracteres do sistema.

Baseado na técnica FAST, Bušta, Neumann e Matas (2015) propuseram um tipo de detector de *keypoints*, denominado FASText. Esta técnica foca na detecção de caracteres, e busca dois tipos de *keypoints*: os *Stroke Ending Keypoints* (SEK) que tendem a ocorrer em finais de caracteres, e os *Stroke Bending Keypoints* (SBK), que tendem a ocorrer em segmentos curvos de caracteres.

Para detectar se um pixel p é um SEK ou SBK, é necessário analisar as intensidades relativas de 12 pixels em um círculo centrado em p. Cada um é associado a um dos três seguintes conjuntos: pixels mais escuros (P_d), pixels de intensidade similar (P_s) e pixels mais claros (P_b) seguindo a regra apresentada na Equação 3.9, onde m é um parâmetro do detector e I_k é a intensidade do pixel k.

$$L(p,x) = \begin{cases} P_d, & I_x \leq I_p - m \text{ (escuros)} \\ P_s, & I_p - m < I_x < I_p + m \text{ (similares)} \\ P_b, & I_x \geq I_p + m \text{ (claros)} \end{cases} \tag{3.9}$$

Após a análise dos pixels vizinhos a p, cada um estará associado a apenas um dos três conjuntos P_b, P_s ou P_d, que devem ser analisados. Caso P_s possua de 1 a 3 elementos vizinhos utilizando conectividade-8, o pixel pode ser considerado um SEK. O pixel p então, será um SEK negativo caso P_d contenha os pixels restantes, e um SEK positivo em caso contrário.

Em outras palavras, para um pixel p ser um SEK, é necessário que exista um segmento de círculo com 9 a 11 pixels com intensidade superior ou inferior a p, enquanto todos os restantes possuem intensidade similar a p. Um exemplo deste tipo de *keypoint* é apresentado na Figura 14a.

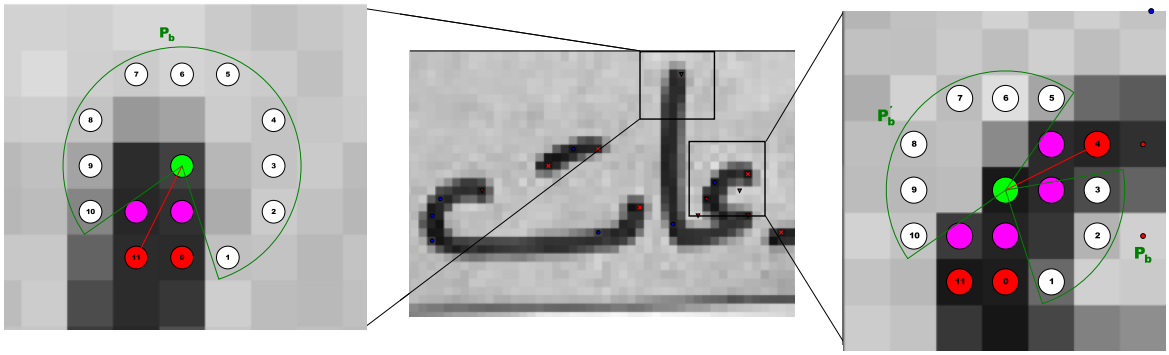
Caso o conjunto P_d (ou P_b) seja vazio, e que os conjuntos restantes não sejam totalmente conectados (utilizando conectividade-8), mas divisíveis em dois conjuntos conectados $P_s = P_{s_1} \bigcup P_{s_2}$ e $P_b = P_{b_1} \bigcup P_{b_2}$ (ou $P_d = P_{d_1} \bigcup P_{d_2}$). Além disso, caso P_{s_1} e P_{s_2} tenham no máximo 3 elementos, e uma das divisões de P_b (ou P_d) tenha 6 ou mais elementos, o pixel p é considerado um SBK.

Em outras palavras, para um pixel p ser um SBK, devem existir 4 segmentos de círculo em torno de p, onde dois possuem no máximo 3 elementos de intensidade similar a p, um dos outros possua 6 ou mais elementos com intensidade superior ou inferior a p,

e o segmento restante tenha elementos com intensidade similar ao maior segmento. Um exemplo deste tipo de *keypoint* é apresentado na Figura 14b.

Figura 14 – Exemplos de *Stroke Ending Keypoints* (a) e *Stroke Bending Keypoints* (b) de um caractere. Pixels marcados em verde representam os pixels analisados, enquanto os marcados na cor branca e vermelha representam os vizinhos associados ao conjunto P_b e P_d, respectivamente.

(a) SEK (b) SBK



Fonte: Bušta, Neumann e Matas (2015).

Segundo Bušta, Neumann e Matas (2015), a implementação deste algoritmo é rápida, mas a inserção de uma regra simples reduz o tempo de processamento pela metade. Esta regra analisa se todos os pixels opostos no círculo de comprimento 12 são mais claros ou escuros que p. Dois pixels são opostos se a reta entre eles divide o círculo em duas partes iguais, como os pixels 1 e 7 da Figura 14a. Caso isso aconteça, o pixel não pode ser SEK nem SBK, e é rejeitado instantaneamente.

Além disso, o autor utiliza mais um teste de conectividade e uma supressão de não-máximos em uma vizinhança 3x3 para eliminar detecções repetidas. O teste de conectividade checa se os pixels entre p e aqueles do grupo P_s têm valores entre a margem $I_p - m < I_x < I_p + m$. Tais pixels estão marcados com a cor rosa nas Figuras 14a e 14b.

O algoritmo FASText procura *keypoints* analisando um círculo de tamanho fixo, que está diretamente associado com a largura de uma letra. Para detectar caracteres de diversos tamanhos, a técnica é executada em diversas escalas, utilizando o conceito de pirâmide de uma imagem, onde cada nível é um redimensionamento do anterior por um fator de escala.

A técnica FASText possui três parâmetros além do tamanho do círculo: margem m, fator de escala da pirâmide e número máximo de *keypoints* detectados. Através de validação cruzada na base de dados ICDAR 2013 (KARATZAS et al., 2013), os valores escolhidos foram 13, 1.6 e 4000, respectivamente (BUŠTA; NEUMANN; MATAS, 2015).

Do mesmo modo que nas técnicas que utilizam detecção de bordas, a técnica FASText não extrai candidatos a caracteres, apenas pontos importantes da imagem.

Portanto, é necessária a utilização de uma técnica capaz de segmentar caracteres a partir dos pontos detectados.

Baseado em outros trabalhos que segmentam objetos através de um ponto e um limiar, Bušta, Neumann e Matas (2015) analisam se o *keypoint* em questão é do tipo positivo ou negativo (mais claro ou escuro que o fundo), e utiliza um limiar um nível acima da intensidade do pixel mais claro do conjunto P_d associado, para o primeiro caso, e um nível abaixo da intensidade do pixel mais escuro do conjunto P_b associado, no segundo caso.

Com tais limiares e a posição de cada *keypoint* p, um algoritmo do tipo *flood-fill* (TREUENFELS, 1994) é utilizado. Assim, a técnica FASText extrai mais caracteres reais, com aproximadamente metade do número de candidatos a caracteres e gastando 25% do tempo da técnica MSER, que é bastante utilizada para extrair caracteres (BUŠTA; NEUMANN; MATAS, 2015).

3.4.1 Método Proposto por Silva e Ciarelli (2016)

O método proposto por Silva e Ciarelli (2016) para extrair candidatos de imagens do mundo real combina uma abordagem baseada em janelas deslizantes, como as apresentadas na Seção 3.1, com uma abordagem baseada em detecção de bordas como as apresentadas na Seção 3.2. Em uma primeira etapa, um mapa de confiança é construído com a utilização de janelas deslizantes que calculam um descritor, utilizando um classificador para encontrar uma "confiança" da presença de texto na imagem.

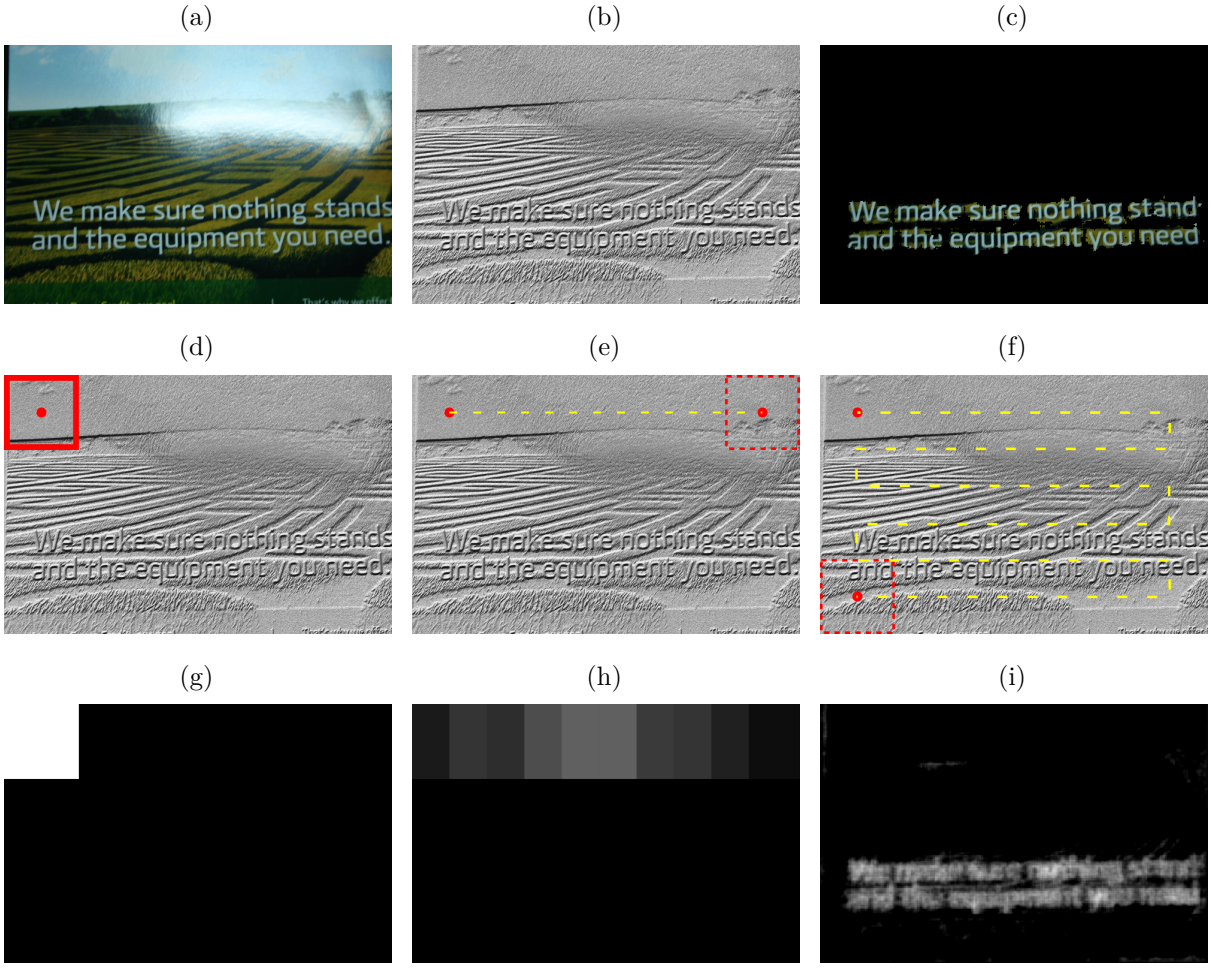
As combinações utilizando descritores HOG e LBP e os classificadores AdaBoost e SVM foram testadas. Por apresentarem resultados similares, a combinação que apresentou o menor tempo de processamento foi escolhida, sendo esta a que utiliza o descritor LBP e o classificador Gentle AdaBoost (SILVA; CIARELLI, 2016).

Para construir o mapa de confiança a partir de uma imagem, o sistema calcula o descritor LBP para todos os seus pixels, armazenando o resultado em uma "imagem LBP", onde uma janela de tamanho 32×32 pixels desliza com passo de 16 pixels. O histograma de cada janela é utilizado como entrada do classificador binário Gentle AdaBoost (FRIEDMAN et al., 2000).

Este tipo de classificador é formado por uma combinação de classificadores "fracos" com pesos, que analisam a amostra de entrada. Neste trabalho, cada janela analisada tem a soma ponderada destes componentes "fracos" acumulada nos respectivos pixels desta janela no mapa de confiança. Este processo é repetido para todas as janelas da imagem em uma pirâmide, onde cada nível é 1,6 vezes menos que o anterior, enquanto a imagem analisada possuir mais de 64 linhas e colunas. Para treinar este classificador foi utilizado a base de dados ICDAR 2013 Training.

Após o término de todo este processo, o sistema calcula a média de todos os valores dos pixels do mapa, e elimina aqueles com valor inferior a esta média. Este processo é ilustrado na Figura 15. As Figuras 15g, 15h e 15i são normalizadas para o intervalo $\{0, \ldots, 255\}$.

Figura 15 – Exemplo de geração de um mapa de confiança. Uma imagem (a), sua imagem LBP (b) e o mapa de confiança gerado (c). Em (d-f), uma janela desliza sobre a imagem (b), onde seu histograma é calculado e utilizado num classificador, cujo resultado é acumulado nos pixels referentes a esta janela no mapa (g-i). Estes mapas estão normalizadas para o intervalo [0,255]



Fonte: Produção do próprio autor.

Na segunda parte deste método, o sistema extrai as bordas da imagem a partir do método de Canny, que possui três principais etapas (CANNY, 1986): uma suavização Gaussiana, o cálculo do gradiente da imagem e a aplicação de dois limiares para identificar os pixels de borda.

Assim, são necessários três parâmetros para este método: dois limiares e o tamanho da máscara gaussiana utilizada para suavização. Esta última, caso tenha valor fixo e alto (ou baixo), pode apresentar bom desempenho em imagens de alta (baixa) resolução e

eliminar (deixar) um grande número de bordas importantes (irrelevantes) em imagens de baixa (alta) resolução.

Para evitar estes problemas, o tamanho da máscara gaussiana utilizado neste trabalho é aproximadamente 0,5% da diagonal da imagem analisada, de modo que ela tenha dimensão de no mínimo 3×3 e no máximo 15×15 pixels. Estes valores foram encontrados empiricamente.

Após a etapa de suavização da imagem, o gradiente é calculado pelo operador de Sobel, e um processo de limiarização é feito, onde os dois limiares são aplicados para extração das bordas. Pixels com gradiente acima do limiar superior são marcados como bordas "fortes", que automaticamente são considerados como bordas. Aqueles com valores abaixo do limiar inferior são eliminados, e os restantes são marcados como bordas "fracas". Estes últimos só são considerados pixels de borda se estiverem conectados a algum pixel de borda "forte" (por exemplo, utilizando conectividade-8) (GONZALEZ; WOODS, 2000).

Novamente, a utilização de um valor fixo para tais limiares não garante um bom funcionamento da técnica para todas as imagens. Como proposto por Fang, Yue e Yu (2009), neste trabalho são usados limiares proporcionais ao limiar de separação ótimo calculado através do método de Otsu (GONZALEZ; WOODS, 2000) do gradiente da imagem analisada. O limiar superior utilizado é o próprio valor obtido através deste processo, enquanto o limiar inferior é igual a metade deste valor.

Extraídas as bordas da imagem, é necessária a utilização uma técnica para segmentar os caracteres. Do mesmo modo que foi proposto por Lu et al. (2015), esta técnica considera que caracteres possuem borda aproximadamente fechada e que a menor *bounding-box* que contém sua borda possui pelo menos dois pixels não nulos em cada linha e coluna. A *bounding-box* é definida da mesma forma que na Seção 3.2.

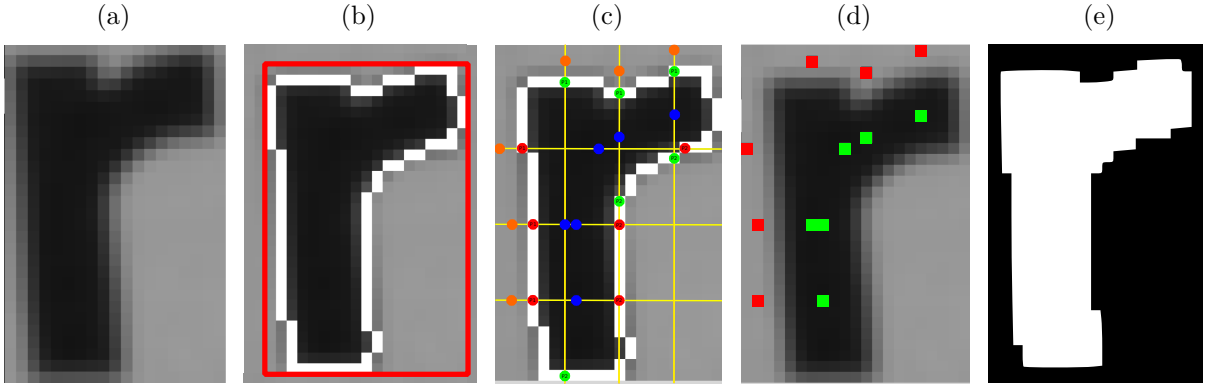
Com esta consideração, é possível segmentar caracteres a partir de bordas da seguinte forma: três linhas e três colunas da *bounding-box* de uma borda conectada são analisadas de modo a descobrir a relação entre as intensidades interior e exterior à borda. O sistema analisa linhas e colunas localizadas em aproximadamente 25%, 50% e 75% da altura e largura da *bounding-box*, respectivamente.

Para cada linha (coluna) analisada, o sistema acha o primeiro pixel não nulo p_1[5] e continua a busca nas colunas (linhas) até achar o próximo pixel não-nulo, p_2, desde que este não esteja conectado a p_1. Se os dois pixels forem encontrados, é considerado pixel interno à borda aquele com coordenada igual a média entre p_1 e p_2. Caso contrário, é considerado pixel interno aquele posicionado 2 pixels a direita (abaixo) de p_1. O pixel externo é aquele posicionado duas posições à esquerda (acima) de p_1. Um exemplo deste

[5] Como uma borda é um segmento conectado, a menor *bounding-box* que a contém terá pelo menos um pixel em cada linha e coluna, independentemente do objeto que ela representa. Assim, p_1 sempre será encontrado.

procedimento é apresentado na Figura 16.

Figura 16 – Um caractere (a), sua borda e a menor *bounding-box* que a contém (b). Em (c), pixels de borda marcados em vermelho e verde são encontrados analisando cada linha e coluna do componente, enquanto os marcados em azul são considerados internos, e na cor laranja os externos. Em (d), os pixels verdes são marcados como internos, enquanto os de cor vermelha são marcados como externos. Em (e), o caractere segmentado utilizando a técnica proposta.



Fonte: Produção do próprio autor.

Após a análise das 3 linhas e das 3 colunas, o sistema terá uma lista de 6 pixels externos e 6 internos a borda analisada. Deste modo, é possível estimar se o interior da borda é mais claro ou escuro que o exterior analisando apenas a média destes valores. Uma limiarização simples é aplicada a região da *bounding-box* na imagem original, utilizando a média dos 12 pixels armazenados como limiar. Caso a média das intensidades interiores seja maior que a média das exteriores, os pixels acima do limiar são segmentados. Caso contrário, os pixels com valor inferior ao limiar são segmentados. A Figura 16e mostra um caractere segmentado com este procedimento, que é feito com todas as bordas conectadas de uma imagem.

Este procedimento é repetido em uma pirâmide gaussiana da imagem original de N_p níveis, aumentando, assim, o número de candidatos detectados. Cada nível da pirâmide tem área $\frac{1}{4}$ menor que o nível anterior, o que não implica em um grande aumento do custo computacional do sistema. O valor de N_p utilizado foi 5, encontrado empiricamente (SILVA; CIARELLI, 2016).

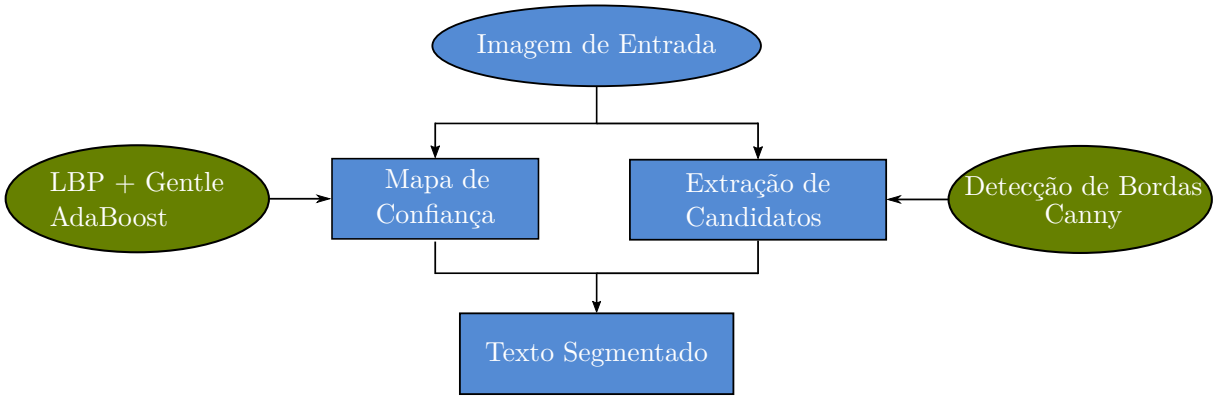
Após a extração de candidatos a caracteres, o sistema confere se cada candidato está, em sua maior parte, incluído nas regiões de interesse encontradas pela etapa do mapa de confiança. Caso um componente tenha pelo menos 20% do número de pixels segmentados fora da região de interesse do mapa, ele é excluído das segmentações.

É possível extrair bordas, e posteriormente segmentar caracteres, apenas nas regiões de interesse do mapa de confiança construído a partir das imagens. Porém, em muitos casos este mapa interfere nas bordas de caracteres, que podem perder sua característica de

caminho “fechado”, consequentemente prejudicando o processo de segmentação proposto (SILVA; CIARELLI, 2016).

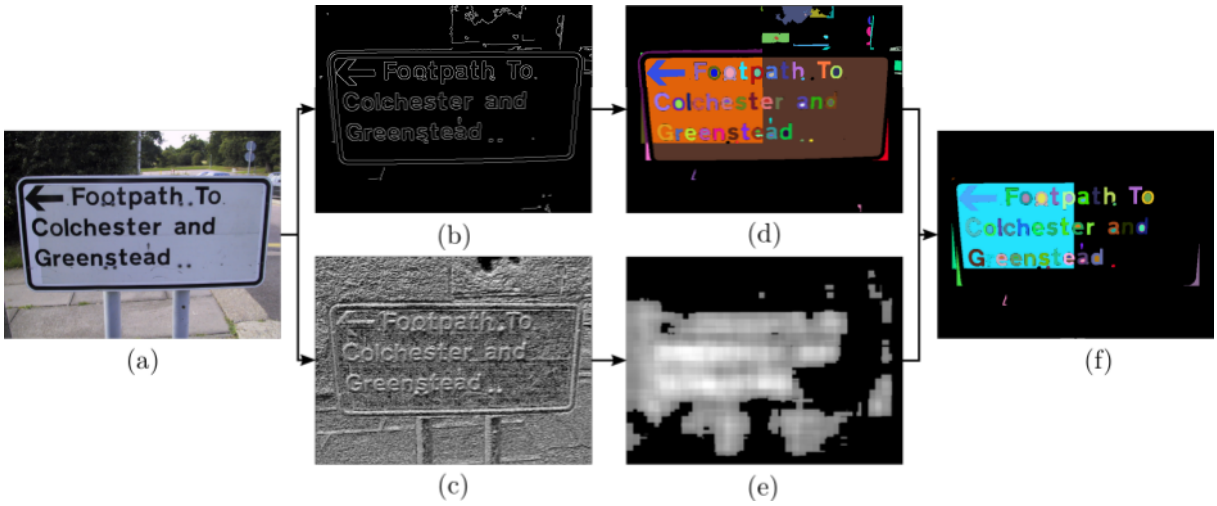
A Figura 17 apresenta um fluxograma do sistema proposto por Silva e Ciarelli (2016). Um mapa de confiança é gerado através da combinação do descritor LBP com o classificador AdaBoost. Candidatos a caracteres são extraídos e eliminados com a utilização do mapa.

Figura 17 – Fluxograma do funcionamento do sistema proposto por Silva e Ciarelli (2016), onde as etapas de extração de candidatos e mapa de confiança podem ser executadas paralelamente



Fonte: Produção do próprio autor.

Figura 18 – Funcionamento do sistema proposto por Silva e Ciarelli (2016) para uma imagem (a), que tem suas bordas (b) e descritor LBP (c) calculados. A partir destes, candidatos a caracteres (d) e o mapa de confiança (e) são encontrados. Estes são combinados, eliminando falsos positivos (f).



Fonte: Produção do próprio autor.

4 Metodologia

Neste capítulo são apresentadas as bases de dados e as métricas utilizadas para comparação das técnicas de segmentação de texto.

4.1 Bases de Dados

Neste trabalho, duas bases de dados foram utilizadas. A primeira, disponibilizada por Lee et al. (2010), se chama *KAIST Scene Text* (LEE; KIM, 2010) e é composta por mais de 3000 imagens capturadas em diferentes ambientes, incluindo internos e externos, em diferentes condições de iluminação. As imagens foram capturadas utilizando câmeras digitais de alta resolução ou câmeras de dispositivos móveis.

As imagens desta base de dados, que apresentam principalmente *posters*, fachadas, placas ou anúncios, são divididas em três partes, classificadas de acordo com o idioma dos caracteres contidos em cada imagem: Inglês (ou números), Coreano e a mistura destes. Acompanhado de cada imagem da base de dados, existe uma segunda imagem contendo as segmentações de seus caracteres que serve como um parâmetro de comparação para avaliar as técnicas de extração de caracteres. Nos experimentos realizados neste trabalho, as três divisões da base de dados KAIST foram agrupadas e tratadas como uma base única. A Figura 19 apresenta exemplos de imagens desta base de dados e suas respectivas segmentações.

A segunda base de dados utilizada neste trabalho é a mesma da *ICDAR Robust Reading Competition* 2013 (KARATZAS et al., 2013). Esta competição é realizada na *International Conference on Document Analysis and Recognition* com o objetivo de quantificar e acompanhar o progresso nas tarefas de localização e reconhecimento de texto em imagens reais. Esta base de dados inclui imagens de diversas resoluções, contendo texto de diferentes fontes, tamanhos, orientações e cores. Não há restrições quanto ao fundo em que os caracteres estão inseridos.

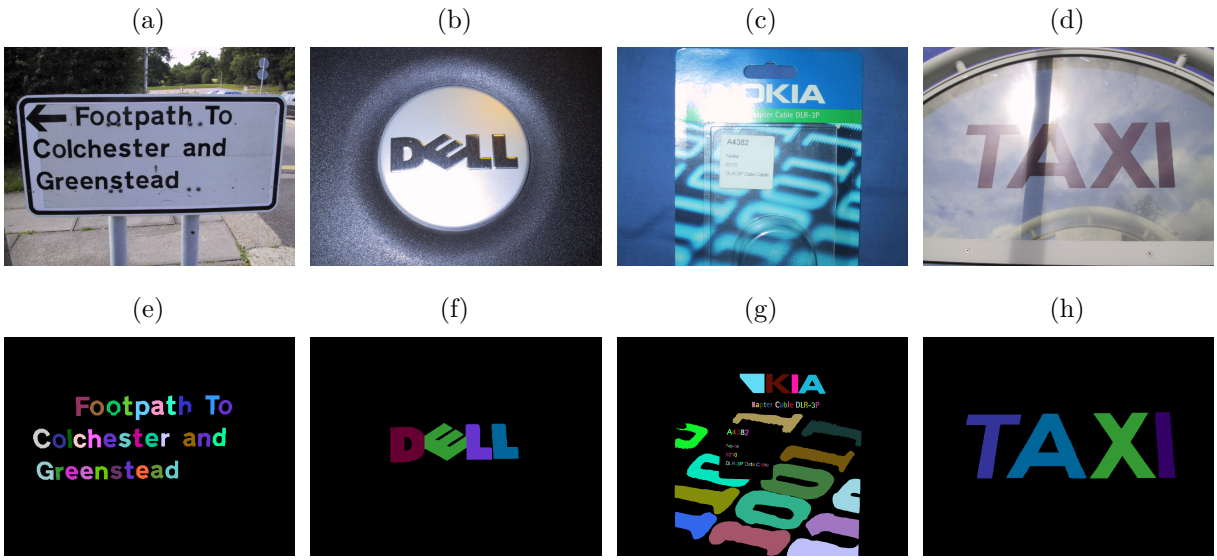
Esta base de dados é dividida em dois conjuntos, onde o primeiro é utilizado para treino dos participantes da competição, contendo 229 imagens. Associadas a cada imagem, existe uma segunda imagem com o texto segmentado, além de marcações indicando os caracteres, as palavras e as localizações de ambos. Da mesma forma que a base de dados KAIST, esta base pode ser utilizada para comparar outras técnicas com a etapa proposta de extração de candidatos. A Figura 20 apresenta exemplos de imagens desta base de dados e suas respectivas segmentações.

O segundo conjunto da base de dados ICDAR 2013 é o utilizado para testar os

Figura 19 – Exemplos de imagens da base de dados KAIST (a-d) e as respectivas segmentações (e-h).

Fonte: Lee et al. (2010).

Figura 20 – Exemplos de imagens da base de dados ICDAR 2013 Treino (a-d) e as respectivas segmentações (e-h).



Fonte: Karatzas et al. (2013).

métodos de seus participantes, contendo 233 imagens. Associada a cada imagem desta base, existem marcações das palavras e um arquivo contendo todas os caracteres segmentados da imagem original. Nas competições da ICDAR, esta base de dados é utilizada para testar as tarefas de localização e segmentação de texto, bem como a tarefa de reconhecimento de palavras dos sistemas submetidos. A Figura 21 apresenta exemplos de imagens desta base

de dados.

Figura 21 – Exemplos de imagens da base de dados ICDAR 2013 Teste.

Fonte: (KARATZAS et al., 2013).

4.2 Métricas Utilizadas

Para comparação da etapa de extração de candidatos a caracteres com outras técnicas, as seguintes métricas foram calculadas: tempo de processamento, número de caracteres reais não segmentados CP, a imprecisão do detector, definida como a razão entre o número de candidatos extraídos da base de dados $|D|$ e o número de caracteres reais presentes $|GT|$ nesta (BUŠTA; NEUMANN; MATAS, 2015), e as métricas R e R_m, calculadas através da razão entre o número de pixels pertencentes aos textos extraídos das imagens $Npx_{ext}(i)$ e o número de pixels pertencentes aos textos da marcação da base de dados $Npx_{base}(i)$, onde a primeira considera a base de dados inteira, e o segundo considera a média dos valores calculados por imagem, como mostram as Equações 4.1 e 4.2, respectivamente.

$$R = \frac{\sum_{i=1}^{N} Npx_{ext}(i)}{\sum_{i=1}^{N} Npx_{base}(i)} \tag{4.1}$$

$$R_m = \frac{\sum_{i=1}^{N} \frac{Npx_{ext}(i)}{Npx_{base}(i)}}{N} \tag{4.2}$$

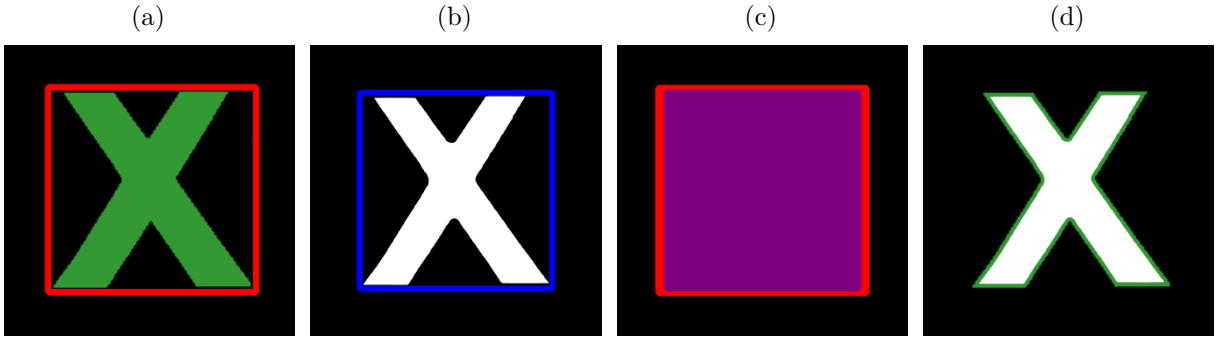
Dada uma imagem que contém uma lista de caracteres $\mathbf{c} = \{c_1, \ldots, c_n\}$, e a lista de segmentações obtidas pela técnica de extração de candidatos $\mathbf{d} = \{d_1, \ldots, d_m\}$, todos os c_i e d_j são comparados. Para um caractere c_i ter uma segmentação válida d_j, é

necessário que suas *bounding-boxes* $BB(c_i)$ e $BB(d_j)$ atendam a restrição da medida de Pascal (EVERINGHAM et al., 2010) (ou *overlap*), dada pela Equação 4.3, superior a 50%. Para atender tal restrição, a *bounding-box* de uma segmentação d_j deve possuir uma alta sobreposição e tamanho similar a *bounding-box* de um caractere real c_i.

$$overlap_{ij} = \frac{\text{área}(BB(c_i) \cap BB(d_j))}{\text{área}(BB(c_i) \cup BB(d_j))} \tag{4.3}$$

Caso as *bounding-boxes* de c_i e d_j atendam a condição da medida de Pascal, os pixels das segmentações destes são comparados utilizando uma operação lógica de conjunção. Esta comparação pode ser interpretada como a sobreposição das duas segmentações, onde apenas os pixels presentes em ambas permanecem como resultado. Assim, a razão R de um caractere é a razão entre o número de pixels restantes após esta operação e o número de pixels do caractere c_i. Assim, é possível descobrir quais caracteres foram segmentados e a área extraída de cada um. Um exemplo desta operação é apresentado na Figura 22.

Figura 22 – Um caractere (a) e uma segmentação (b). Em (c), as *bounding-boxes* são comparadas, onde a área de interseção é apresentada na cor roxa, enquanto a área da união são os pixels com cor diferente da cor preta. Finalmente, as segmentações são sobrepostas (d).



Fonte: Produção do próprio autor.

5 Experimentos e Resultados

Esta seção descreve os experimentos feitos para comparação das técnicas apresentadas na tarefa de segmentação de texto em imagens reais. As imagens são processadas em escala de cinza, sendo feita a conversão para este espaço de cor caso necessário. Os experimentos foram realizados em um computador com processador Intel® Core™ i5-2500 com *clock* de 3.30GHz e 8GB de memória RAM. O sistema operacional utilizado foi o Xubuntu 16.04.1 LTS. Todas as técnicas foram implementadas na linguagem de programação C++ com a utilização da biblioteca OpenCV 2.4.9 (ITSEEZ, 2015), exceto quando informado o contrário.

As métricas R, dada pela razão entre o número de pixels extraídos de textos e o número real de pixels pertencentes aos textos de todas as imagens da base de dados, R_m, que é a média das razões R considerando cada imagem da base de dados separadamente, o número de caracteres reais não segmentados CP e a imprecisão do detector, $\frac{|D|}{|GT|}$, apresentadas na Seção 4.2, foram calculadas através dos candidatos extraídos utilizando as técnicas MSER (MATAS et al., 2004), FASText (BUŠTA; NEUMANN; MATAS, 2015) e o método proposto em (SILVA; CIARELLI, 2016). Os experimento foram repetido 100 vezes para cálculo da média e do desvio padrão do tempo de processamento em milissegundos.

As métricas obtidas para as bases de dados ICDAR 2013 Treino e Teste são apresentados nas Tabelas 2 e 3.

Tabela 2 – Resultados dos experimentos da etapa de extração de caracteres na base de dados ICDAR 2013 - Treino

Método	R	R_m	$\frac{\lvert D\rvert}{\lvert GT\rvert}$	CP	$t_m(ms)$
MSER (MATAS et al., 2004)	0,8898	**0,8874**	26,13	206	307,68±2,35
FASText (BUŠTA, 2016)	**0,9053**	0,8815	41,68	**135**	97,81±0,63
Método de Silva e Ciarelli (2016)	0,8960	0,8785	**8,70**	293	310,50±0,36
Método de Silva e Ciarelli (2016)[1]	0,8975	0,8801	15,24	291	**78,75±0,22**

Os resultados da Tabela 2 mostram que, em geral, todos os métodos analisados extraem uma proporção de pixels pertencentes a texto próxima de 90% da base de dados ICDAR 2013 Treino, tanto na métrica R, que considera a base de dados inteira, quanto na métrica R_m, que considera cada imagem individualmente.

Quanto a métrica que analisa a imprecisão do detector, a técnica que apresentou melhores resultados foi a proposta em (SILVA; CIARELLI, 2016). Sem a utilização do mapa de confiança deste método, o detector já encontrava um número menor de candidatos (e consequentemente falsos positivos) quando comparado as técnicas MSER e FASText,

[1] O mapa de confiança não foi usado nesse experimento.

porém, a utilização do mapa reduziu o número de detecções em aproximadamente 45%, com a desvantagem de um maior custo computacional, aumentando o tempo de processamento em cerca de 2,3 vezes. Esta técnica, porém, apresentou o pior resultado na métrica que leva em consideração caracteres sem segmentação, em que a técnica FASText apresentou melhores resultados.

Já em relação ao tempo de processamento, os métodos que se destacaram foram o FASText e o método proposto em (SILVA; CIARELLI, 2016) quando o mapa de confiança não é utilizado. Uma comparação entre os dois métodos mostra que o segundo leva em média 75% do tempo do método FASText para extrair candidatos, extraindo cerca de 36,5% do número de candidatos, e recuperando quase a mesma proporção de pixels pertencentes a texto da base de dados. Porém, o segundo método perde mais do que o dobro de caracteres, o que pode indicar uma sensibilidade a caracteres pequenos. Além disso, o método é sensível a imagens com presença de borramento, por usar bordas para detectar candidatos.

Na Figura 23, são apresentadas duas imagens onde caracteres pequenos não são encontrados pelo sistema proposto em (SILVA; CIARELLI, 2016). Na Figura 23a, existem caracteres grandes, médios e pequenos. Estes últimos não são extraídos, como pode ser visto na Figura 23c. Já na Figura 23b, existem apenas caracteres pequenos, que também não são encontrados.

O sistema proposto em (SILVA; CIARELLI, 2016) também apresentou dificuldades na extração de candidatos em imagens com caracteres de diferentes cores, principalmente quando a representação da cor destes em escala de cinza seja distante (por exemplo, em uma imagem com caracteres nas cores preta e branca). A Figura 24 mostra imagens que apresentaram este problema. Os caracteres mais escuros da Figura 24a (ou com cor mais distante do fundo branco), foram detectados, já os mais claros não. Este problema ocorre pois as bordas destes caracteres foram eliminadas pelo limiar utilizado, baseado no método de Otsu, na etapa de extração de bordas, e consequentemente das etapas posteriores do sistema. A Figura 24c mostra um exemplo onde caracteres de diferentes cores são encontrados.

O uso do mapa de confiança elimina quase metade dos candidatos extraídos das imagens. Já que a área extraída é praticamente mantida, isso significa que grande parte destes candidatos eliminados são falsos positivos. Isto implica numa maior eficiência das fases posteriores do método.

A técnica MSER, quando testada utilizando o conjunto padrão de parâmetros da implementação desta técnica na biblioteca OpenCV, também apresentou alguns problemas com falta de contraste entre caractere e fundo e com caracteres pequenos, além do grande número de falsos positivos detectados. Exemplos de segmentações obtidas através da técnica MSER a partir de imagens da base de dados ICDAR 2013 Treino são apresentados na Figura 25.

Figura 23 – Exemplos de imagens onde caracteres pequenos não são encontrados

Fonte: Karatzas et al. (2013) (a) e (b). Produção do próprio autor (c) e (d).

A técnica FASText, quando testada utilizando o conjunto padrão de parâmetros apresentado em (BUŠTA; NEUMANN; MATAS, 2015), também apresentou alguns problemas com falta de contraste entre caractere e fundo, além do grande número de falsos positivos detectados. Esta técnica não apresentou grandes dificuldades em detectar caracteres pequenos. Exemplos de segmentações obtidas através da técnica FASText a partir de imagens da base de dados ICDAR 2013 Treino são apresentados na Figura 26.

Os experimentos foram realizados com a mesma configuração para a base de dados ICDAR 2013 Teste, e as mesmas métricas foram calculadas. Os resultados obtidos são apresentados na Tabela 3.

Resultados semelhantes aos obtidos na base de dados ICDAR 2013 Treino foram obtidos na base de dados ICDAR 2013 Teste, onde os mesmos tipos de problemas foram encontrados. Embora todas as técnicas tenham recuperado uma razão de texto inferior e perderem um número bastante superior de caracteres quando comparados aos números encontrados na Tabela 2, referentes aos testes na base ICDAR 2013 Treino. As técnicas

Figura 24 – Exemplos de imagens onde caracteres com bordas fortes são detectados, enquanto os com gradientes fracos não

Fonte: Karatzas et al. (2013) (a), (b) e (c). Produção do próprio autor (c), (d) e (e)

Tabela 3 – Resultados dos experimentos da etapa de extração de caracteres na base de dados ICDAR 2013 - Teste

Método	R	R_m	$\frac{\|D\|}{\|GT\|}$	CP	$t_m(ms)$
MSER (MATAS et al., 2004)	**0,8452**	**0,8594**	21,94	923	221,20±0,95
FASText (BUŠTA, 2016)	0,8185	0,8451	33,18	**577**	98,51±0,41
Método de Silva e Ciarelli (2016)	0,8014	0,8004	**10,77**	1554	236,31±3,70
Método de Silva e Ciarelli (2016)[2]	0,8287	0,8230	14,65	1237	**67,75±0,25**

que apresentaram melhores resultados nas métricas se repetiram, com exceção da técnica MSER encontrando uma razão R superior as demais.

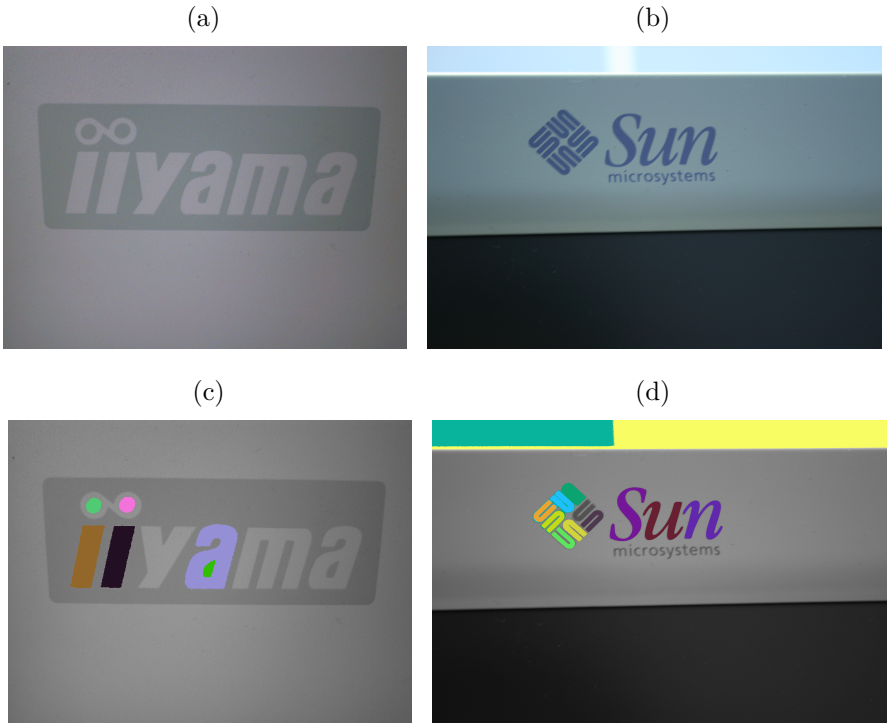
Exemplos de imagens desta base de dados obtidos com a técnica proposta em (SILVA; CIARELLI, 2016), e as técnicas MSER e FASText a partir de imagens da base de dados ICDAR 2013 Teste são apresentados nas Figuras 27, 28 e 29, respectivamente.

Os experimentos realizados foram repetidos para a base de dados Kaist Scene Text (LEE; KIM, 2010), calculando as mesmas métricas. Os resultados obtidos são apresentados na Tabela 4.

Novamente, os resultados mostram que o método proposto por Silva e Ciarelli (2016) (sem a utilização do mapa de confiança), extrai uma razão de área pertencente ao texto parecida ao de outras técnicas, gastando um menor tempo de processamento para

[2] O mapa de confiança não foi usado nesse experimento.

Figura 25 – Exemplos de imagens da base de dados ICDAR 2013 Treino (a) e (b) e suas respectivas segmentações obtidas com a técnica MSER



Fonte: Karatzas et al. (2013) (a) e (b). Produção do próprio autor (c) e (d).

Tabela 4 – Resultados dos experimentos da etapa de extração de caracteres na base de dados Kaist Scene Text

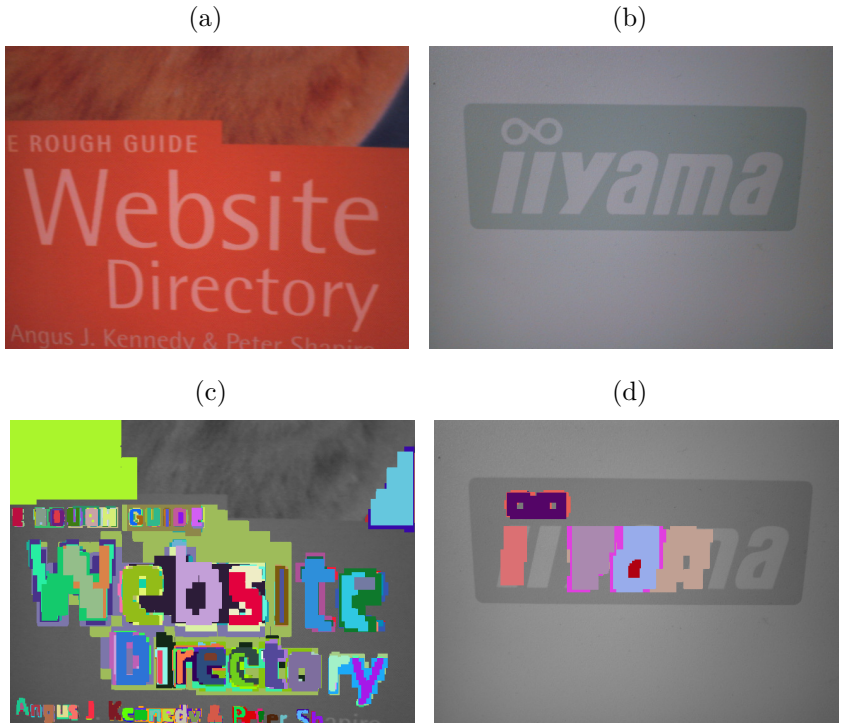
Método	R	R_m	$\frac{\lvert D\rvert}{\lvert GT\rvert}$	CP	$t_m(ms)$
MSER (MATAS et al., 2004)	**0,9633**	**0,9652**	13,92	4460	69,77±0,53
FASText (BUŠTA, 2016)	0,9085	0,9183	18,45	**3506**	39,14±0,49
Método de Silva e Ciarelli (2016)	0,9010	0,9038	**7,75**	8792	67,71±0,54
Método de Silva e Ciarelli (2016)[3]	0,9383	0,9353	9,42	5370	**28,72±0,32**

isto. Nesta base, este método supera as métricas obtidas pelo método FASText, extraindo menos candidatos. Quando comparado a técnica MSER, o método proposto por Silva e Ciarelli (2016) extrai menos candidatos em aproximadamente 40% do tempo, perdendo um pouco de área extraída.

Do mesmo modo que na base de dados ICDAR 2013 Treino, a técnica FASText extraiu um número maior de caracteres reais quando comparada às outras. Porém, ela obteve a menor área extraída, o que indica uma maior sensibilidade a caracteres grandes. Já o método proposto por Silva e Ciarelli (2016), extraiu uma maior proporção da área de candidatos, mesmo perdendo um grande número de caracteres. Este resultado também indica que este método é sensível a elementos pequenos.

[3] O mapa de confiança não foi usado nesse experimento.

Figura 26 – Exemplos de imagens da base de dados ICDAR 2013 Treino (a) e (b) e suas respectivas segmentações obtidas com a técnica FASText



Fonte: Karatzas et al. (2013) (a) e (b). Produção do próprio autor (c) e (d).

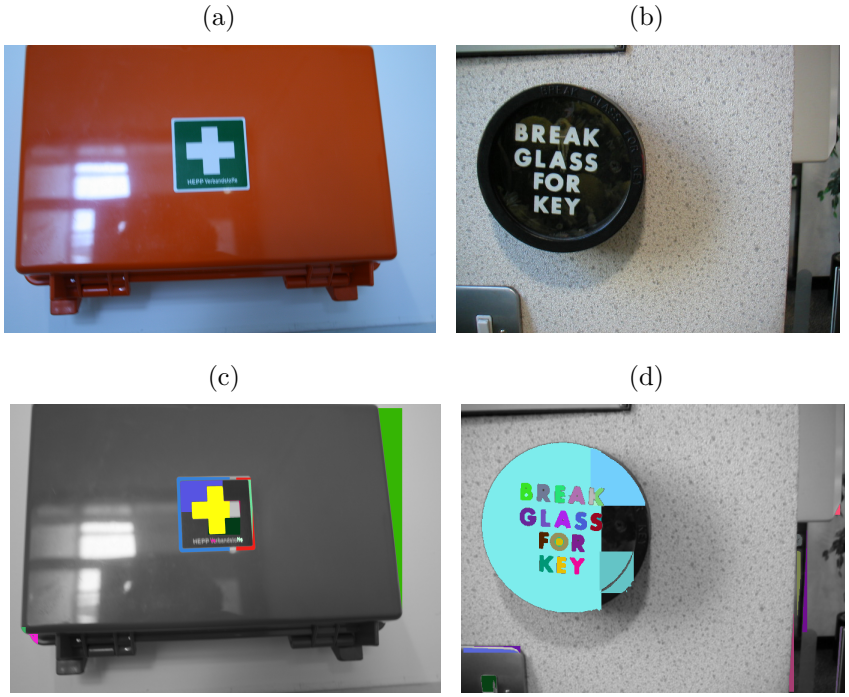
A utilização do mapa de confiança, nesta base de dados, eliminou cerca de 18% dos candidatos a caracteres. Porém, um grande número de caracteres reais foi eliminado, juntamente com cerca de 3,8% de área pertencente ao texto extraído.

Exemplos de imagens desta base de dados obtidos com a técnica proposta em (SILVA; CIARELLI, 2016) e as técnicas MSER e FASText a partir de imagens da base de dados KAIST são apresentadas nas Figuras 30, 31 e 32, respectivamente.

A Figura 33 mostra três imagens das bases de dados e suas respectivas segmentações obtidas com a técnica proposta em (SILVA; CIARELLI, 2016), e as técnicas MSER e FASText. É possível ver que a técnica proposta em (SILVA; CIARELLI, 2016) segmenta menos candidatos, enquanto as demais encontram um grande número de falsos positivos. As técnicas foram capazes de segmentar todos os caracteres das imagens 33a e 33c, mas apresentaram dificuldades em alguns caracteres da imagem 33b.

De modo geral, todas as técnicas funcionaram de forma similar em imagens sem a presença distorções ou problemas com iluminação em todas as bases de dados analisadas. As técnicas apresentaram as mesmas dificuldades nestas bases, como detectar caracteres pequenos, distorcidos, ou com cor muito similar ao fundo em que estão inseridos. Em imagens com distorções ou reflexões, a técnica proposta em (SILVA; CIARELLI, 2016) apresentou maiores dificuldades, pois as bordas dos caracteres não foram corretamente

Figura 27 – Exemplos de imagens da base de dados ICDAR 2013 Teste (a) e (b) e suas respectivas segmentações obtidas com a técnica proposta em (SILVA; CIARELLI, 2016)

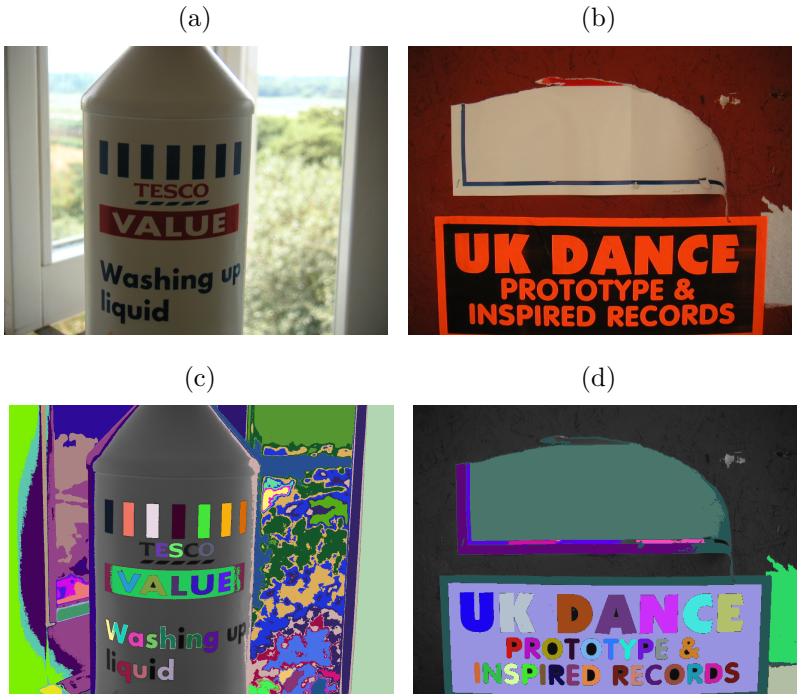


Fonte: Karatzas et al. (2013) (a) e (b). Produção do próprio autor (c) e (d).

detectadas.

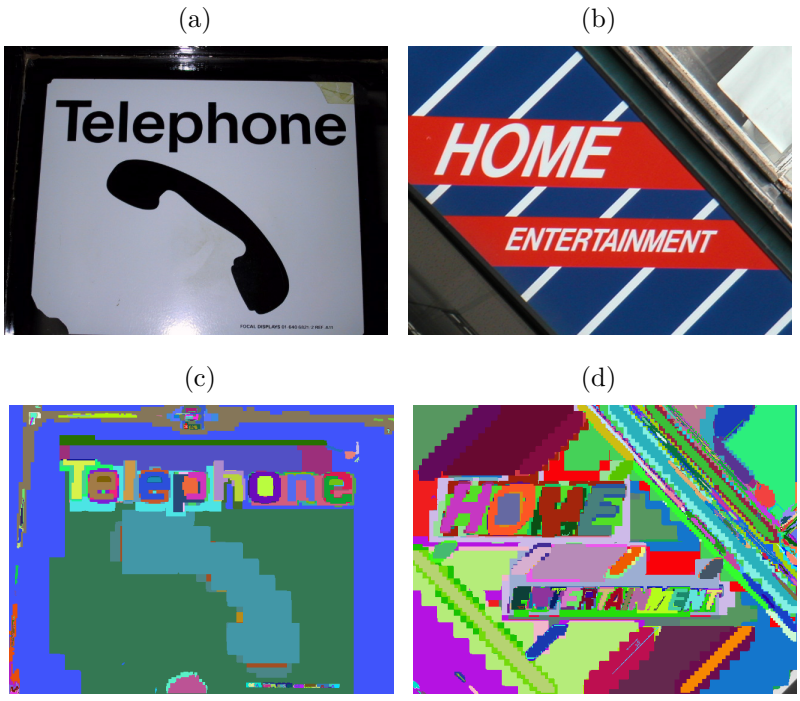
A técnica proposta em (SILVA; CIARELLI, 2016) apresentou os melhores resultados entre as técnicas analisadas nas métricas imprecisão ($\frac{|D|}{|GT|}$), utilizando o mapa de confiança proposto, e tempo de processamento, sem a utilização do mapa de confiança, em todas as bases de dados analisadas. Já a técnica FASText obteve o melhor resultado na métrica que analisa o número de caracteres sem segmentação em todas as bases de dados. Já a técnica MSER, embora tenha recuperado uma maior área de texto na base de dados KAIST e ICDAR Teste, apresentou, na média, o maior tempo de processamento e o maior número de segmentações.

Figura 28 – Exemplos de imagens da base de dados ICDAR 2013 Teste (a) e (b) e suas respectivas segmentações obtidas com a técnica MSER



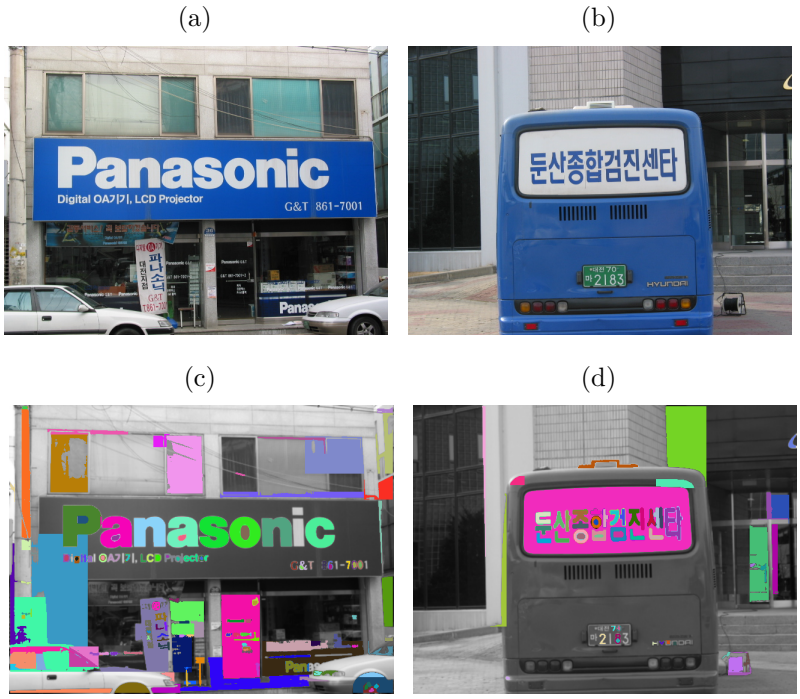
Fonte: Karatzas et al. (2013) (a) e (b). Produção do próprio autor (c) e (d).

Figura 29 – Exemplos de imagens da base de dados ICDAR 2013 Teste (a) e (b) e suas respectivas segmentações obtidas com a técnica FASText



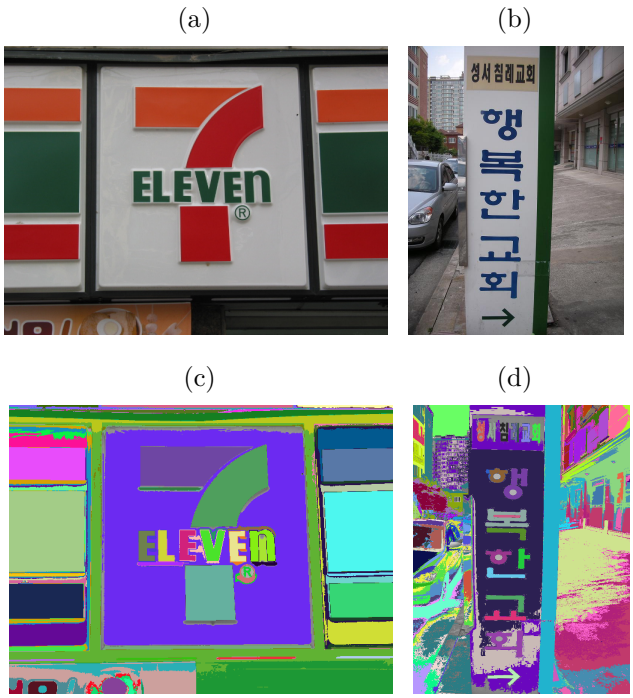
Fonte: Karatzas et al. (2013) (a) e (b). Produção do próprio autor (c) e (d).

Figura 30 – Exemplos de imagens da base de dados KAIST (a) e (b) e suas respectivas segmentações obtidas com a técnica proposta em (SILVA; CIARELLI, 2016)



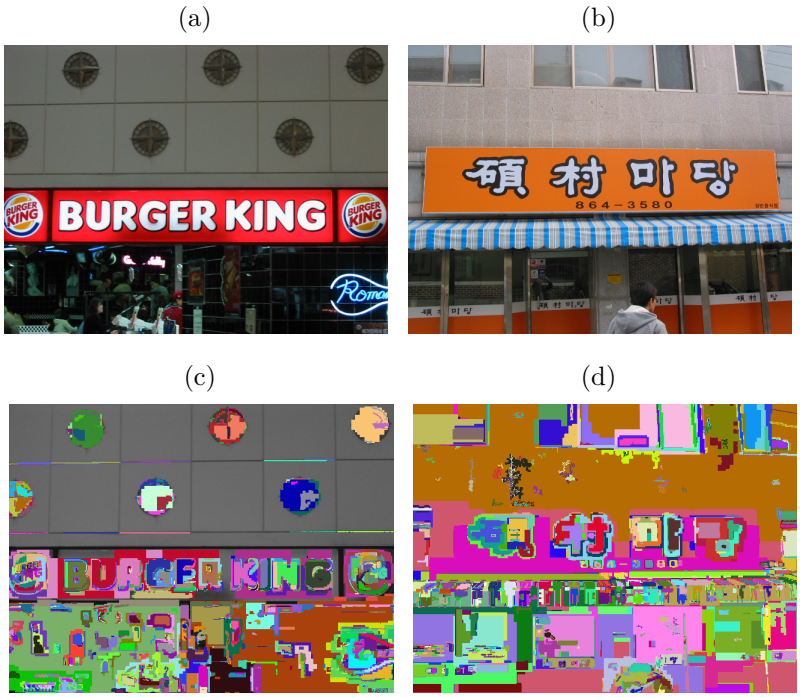
Fonte: Lee e Kim (2010) (a) e (b). Produção do próprio autor (c) e (d).

Figura 31 – Exemplos de imagens da base de dados ICDAR 2013 Teste (a) e (b) e suas respectivas segmentações obtidas com a técnica MSER



Fonte: Lee e Kim (2010) (a) e (b). Produção do próprio autor (c) e (d).

Figura 32 – Exemplos de imagens da base de dados ICDAR 2013 Teste (a) e (b) e suas respectivas segmentações obtidas com a técnica FASText



Fonte: Lee e Kim (2010) (a) e (b). Produção do próprio autor (c) e (d).

Figura 33 – Exemplos de imagens das bases de dados ICDAR 2013 Treino (a), ICDAR 2013 Teste (b) e KAIST (c). Segmentações obtidas com a técnica proposta em (SILVA; CIARELLI, 2016) (d)-(f), MSER (g)-(i) e FASText (j)-(l)

Fonte: (KARATZAS et al., 2013) (a) e (b). Lee e Kim (2010) (c). Produção do próprio autor (d)-(l).

6 Conclusões e Trabalhos Futuros

Texto corresponde a uma das informações mais importantes que podem ser extraídas de imagens. Para isso, métodos cada vez mais sofisticados têm sido desenvolvidos na literatura com o objetivo de detectar com boa precisão e recall a ocorrência de textos em imagens. Neste contexto, este trabalho propôs a realização de um estudo comparativo de algumas técnicas de segmentação de texto em imagens do mundo real utilizando para isso bases de dados de domínio público.

Para a comparação foram utilizadas técnicas que apresentam baixo custo computacional, tal que podem ser utilizadas em sistemas que necessitam processar um grande número de imagens em um tempo limitado.

Entre as técnicas de segmentação de texto comparadas, estão as MSER, bastante utilizadas em sistemas que necessitam segmentar quaisquer tipos de regiões em imagens, não sendo uma técnica específica para segmentar caracteres de imagens.

Já as outras técnicas analisadas fazem considerações específicas para caracteres, sendo estas baseadas em keypoints (BUŠTA; NEUMANN; MATAS, 2015) e extração de bordas (SILVA; CIARELLI, 2016). Devido ao fato de serem específicas para segmentar texto de imagens, estas técnicas apresentaram desempenho similar ao método MSER, porém com tempo de processamento bastante reduzido.

As técnicas MSER e o método proposto em (SILVA; CIARELLI, 2016) apresentaram dificuldades em extrair caracteres com área pequena, quando comparada com a área da imagem. O mesmo problema ocorreu na técnica FASText, porém de forma atenuada.

De um ponto de vista geral, todas as três técnicas realizaram seu objetivo de forma satisfatória. Todas apresentaram um grande número de detecções repetidas e falsos positivos, principalmente pela detecção de caracteres em vários níveis da imagem. A técnica proposta por Silva e Ciarelli (2016) (sem a utilização do mapa de confiança) extraiu um número menor de candidatos em menor tempo, porém perdendo muitos caracteres. Já a FASText extraiu um grande número de candidatos, perdendo menos caracteres reais, com um tempo de processamento um pouco superior.

Em imagens sem a presença de distorções, problemas de iluminação, tamanho de caracteres reduzido, ou mudança de cor em um caractere ou entre caracteres distintos, todas as técnicas analisadas apresentaram bons resultados, segmentando os caracteres com eficiência. Em imagens onde a diferença da cor de um caractere e o fundo em que ele está inserido é pequena, todas as técnicas não apresentaram bom desempenho, pois a diferença de intensidades não foi suficiente para os caracteres serem detectados.

Em imagens que apresentaram problemas de distorção, como movimentação da câmera, as técnicas FASText e MSER apresentaram melhores resultados, mesmo tendo a desvantagem de detectarem caracteres agrupados como um elemento só, já que a técnica proposta por Silva e Ciarelli (2016) depende de bordas bem definidas para extrair corretamente os caracteres. Estas técnicas também apresentaram vantagens em imagens com a presença de caracteres de cor muito diferente ou aqueles com área muito pequena quando comparada a área da imagem.

Para trabalhos futuros é esperado a comparação dos métodos analisados com outras técnicas, como técnicas baseadas em *deep learning*, bem como a utilização de novas bases de dados direcionadas a problemas específicos de detecção de texto em imagens, como páginas de jornais escaneados, placas de trânsito, de automóveis, etc.

Referências

AMIT, Y.; FELZENSZWALB, P. Object detection. *Computer Vision: A Reference Guide*, Springer, p. 537–542, 2014.

ARTETA, C. et al. Learning to detect cells using non-overlapping extremal regions. In: SPRINGER. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. [S.l.], 2012. p. 348–356.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: ACM. *Proceedings of the fifth annual workshop on Computational learning theory*. [S.l.], 1992. p. 144–152.

BUŠTA, M. *FASText: Efficient Unconstrained Scene Text Detector*. 2016. Disponível em: <https://github.com/Michal/FASText>. Acesso em 15 de Novembro de 2016.

BUŠTA, M.; NEUMANN, L.; MATAS, J. Fastext: Efficient unconstrained scene text detector. In: *2015 IEEE International Conference on Computer Vision (ICCV 2015)*. California, US: IEEE, 2015. p. 1206–1214.

CANNY, J. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, n. 6, p. 679–698, 1986.

CHEN, H. et al. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In: IEEE. *18th IEEE International Conference on Image Processing (ICIP)*. [S.l.], 2011. p. 2609–2612.

COATES, A. et al. Text detection and character recognition in scene images with unsupervised feature learning. In: IEEE. *International Conference on Document Analysis and Recognition (ICDAR)*. [S.l.], 2011. p. 440–445.

COSMO, D. L.; SALLES, E. O. T.; CIARELLI, P. M. Pedestrian detection utilizing gradient orientation histograms and color self similarities descriptors. *IEEE Latin America Transactions*, v. 13, n. 7, p. 2416–2422, July 2015. ISSN 1548-0992.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. [S.l.], 2005. v. 1, p. 886–893.

EPSHTEIN, B.; OFEK, E.; WEXLER, Y. Detecting text in natural scenes with stroke width transform. In: IEEE. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.], 2010. p. 2963–2970.

EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, v. 88, n. 2, p. 303–338, 2010.

FANG, M.; YUE, G.; YU, Q. The study on an application of otsu method in canny operator. In: CITESEER. *International Symposium on Information Processing (ISIP)*. [S.l.], 2009. p. 109–112.

FREUND, Y.; SCHAPIRE, R. E. *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*. 1995.

FRIEDMAN, J. et al. Additive logistic regression: a statistical view of boosting. *The annals of statistics*, Institute of Mathematical Statistics, v. 28, n. 2, p. 337–407, 2000.

GONZALEZ, R.; WOODS, R. *Processamento de imagens digitais*. Edgard Blucher, 2000. ISBN 9788521202646. Disponível em: <https://books.google.com.br/books?id=d3MnAgAACAAJ>.

ITSEEZ. *Open Source Computer Vision Library*. 2015. Disponível em: <https://github.com/itseez/opencv>. Acesso em 15 de Novembro de 2016.

KARATZAS, D. et al. Icdar 2015 competition on robust reading. In: IEEE. *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. [S.l.], 2015. p. 1156–1160.

KARATZAS, D. et al. Icdar 2013 robust reading competition. In: IEEE. *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. [S.l.], 2013. p. 1484–1493.

KESSY, A.; LEWIN, A.; STRIMMER, K. Optimal whitening and decorrelation. *ArXiv e-prints*, dez. 2015.

KINTO, E. A. *Otimização e análise das máquinas de vetores de suporte aplicadas à classificação de documentos*. Tese (Doutorado) — Universidade de São Paulo, 2011.

LEE, S. et al. Scene text extraction with edge constraint and text collinearity. In: IEEE. *Pattern Recognition (ICPR), 2010 20th International Conference on*. [S.l.], 2010. p. 3983–3986.

LEE, S.; KIM, J. H. *KAIST Scene Text Database*. 2010. Disponível em: <http://www.iapr-tc11.org/mediawiki/index.php/KAIST_Scene_Text_Database>. Acesso em 15 de Novembro de 2016.

LU, S. et al. Scene text extraction based on edges and support vector regression. *International Journal on Document Analysis and Recognition (IJDAR)*, v. 18, n. 2, p. 125–135, 2015. ISSN 1433-2825.

LUCAS, S. M. Icdar 2005 text locating competition results. In: IEEE. *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. [S.l.], 2005. p. 80–84.

MATAS, J. et al. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, Elsevier, v. 22, n. 10, p. 761–767, 2004.

MISHRA, A.; ALAHARI, K.; JAWAHAR, C. Top-down and bottom-up cues for scene text recognition. In: IEEE. *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. [S.l.], 2012. p. 2687–2694.

MOSLEH, A.; BOUGUILA, N.; HAMZA, A. B. Image text detection using a bandlet-based edge detector and stroke width transform. In: *Proceedings of the British Machine Vision Conference*. [S.l.: s.n.], 2012. p. 63.1–63.12.

NEUMANN, L.; MATAS, J. Efficient scene text localization and recognition with local character refinement. In: *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. [S.l.]: IEEE, 2015. p. 746–750. ISBN 1520-5363.

NISTÉR, D.; STEWÉNIUS, H. Computer vision – eccv 2008: 10th european conference on computer vision. In: _____. [S.l.: s.n.], 2008. cap. Linear Time Maximally Stable Extremal Regions.

OJALA, T.; PIETIKAINEN, M.; HARWOOD, D. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In: *Proceedings of the 12th IAPR International Conference on Pattern Recognition*. [S.l.: s.n.], 1994. v. 1, p. 582–585.

OPITZ, M. *Text Detection and Recognition in Natural Scene Images*. [S.l.]: na, 2013.

OZUYSAL, M.; FUA, P.; LEPETIT, V. Fast keypoint recognition in ten lines of code. In: IEEE. *2007 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.], 2007. p. 1–8.

PIETIKÄINEN, M. et al. Local binary patterns for still images. In: _____. *Computer Vision Using Local Binary Patterns*. London: Springer London, 2011. p. 13–47. ISBN 978-0-85729-748-8.

ROSTEN, E.; PORTER, R.; DRUMMOND, T. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 32, n. 1, p. 105–119, 2010.

SILVA, B. L. S.; CIARELLI, P. M. Edge detection and confidence map applied to identify textual elements in images. In: *Proceedings of XII Workshop de Visão Computacional*. [S.l.: s.n.], 2016.

SILVA JÚNIOR, W. S. *Reconhecimento de Padrões Utilizando Filtros de Correlação com Análise de Componentes Principais*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2010.

TAN, X.; TRIGGS, B. Enhanced local texture feature sets for face recognition under difficult lighting conditions. In: SPRINGER. *International Workshop on Analysis and Modeling of Faces and Gestures*. [S.l.], 2007. p. 168–182.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition*. [S.l.]: Elsevier Science, 2008. ISBN 9780080949123.

TREUENFELS, A. An efficient flood visit algorithm. *C/C++ Users J.*, CMP Media, Inc., USA, v. 12, n. 8, p. 39–62, aug 1994. ISSN 1075-2838.

VIOLA, P.; JONES, M. J. Robust real-time face detection. *International journal of computer vision*, Springer, v. 57, n. 2, p. 137–154, 2004.

VONDRICK, C. et al. Hoggles: Visualizing object detection features. In: *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.: s.n.], 2013. p. 1–8.

WANG, K.; BABENKO, B.; BELONGIE, S. End-to-end scene text recognition. In: IEEE. *Proceedings of the IEEE International Conference on Computer Vision*. [S.l.], 2011. p. 1457–1464.

YIN, X.-C. et al. Robust text detection in natural scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 36, n. 5, p. 970–983, 2014.

ZHOU, Z.-H. *Ensemble methods: foundations and algorithms*. [S.l.]: CRC press, 2012.

ZHU, S.; ZANIBBI, R. A text detection system for natural scenes with convolutional feature learning and cascaded classification. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2016. p. 625–632.