

# Detecção de Textos em Imagens Através da Detecção de Bordas Canny Aplicadas ao Algoritmo Stroke Width Transform

Iverson Luís Pereira

Universidade Federal Rural de Pernambuco

Departamento de Informática e Estatística

Recife, Brasil

iversonpereira@outlook.com

**Resumo**—Este artigo relata a utilização de técnicas de Processamento de Imagens com o objetivo de detectar textos em imagens. Neste trabalho, foi aplicada a abordagem de detecção de textos através da extração de bordas utilizando o algoritmo Canny somada com a técnica Sobel a fim de obter um melhor resultado. Posteriormente, foi aplicado o algoritmo *Stroke Width Transform (SWT)* para obter a largura dos traçados das bordas a fim de realizar a extração dos componentes conectados, com o objetivo de diferenciar o que é texto e o que é background. Os resultados se mostraram bastante promissores, bastando apenas escolher os melhores parâmetros para cada tipo de imagem para que o algoritmo obtenha o melhor resultado.

**Palavras-chave**—detecção de textos; canny; sobel; swt.

## I. INTRODUÇÃO

Com o grande avanço da Visão Computacional nos últimos anos, principalmente em dispositivos móveis, se fez necessário a construção de técnicas e abordagens cada vez mais otimizadas e robustas para apresentar os melhores resultados para os usuários. Dentre essas abordagens para desenvolvimento de aplicações, está as que capturam imagens reais e a transformam em dados para que o usuário tenha acesso à informações que estão além do campo de visão dele. Ou seja, o usuário tem acesso a detalhes e informações que estão diante de seus olhos que sem a Visão Computacional não seria possível. A Detecção de Textos é um exemplo de projetos com essa finalidade, como um exemplo de aplicação temos o *Google Translator* [1] para dispositivos móveis, nela é possível capturar imagens de textos de diferentes idiomas e fontes e reconhecê-los para transformá-las em imagens com textos do idioma de preferência do usuário. Isso é só um exemplo dentre diferentes aplicações.

Por isso, esse trabalho apresenta uma abordagem de como detectar textos em imagens através de um algoritmo

matemático chamado *Stroke Width Transform (SWT)* somada a técnicas de Processamento de Imagens já bem consolidadas.

A estrutura deste artigo está dividido em Seções. Na Seção 2 mostra os principais trabalhos relacionados que serviram como fonte de inspiração para o desenvolvimento desse projeto. Na Seção 3, toda a metodologia aplicada neste projeto mostrando todas as etapas do processo de detecção de textos. Na Seção 4, os resultados iniciais e finais obtidos durante a evolução do algoritmo e na melhor definição dos técnicas e parâmetros utilizados. Por fim, na Seção 5 a conclusão à respeito do projeto e quais serão os passos futuros para melhorá-lo.

## II. TRABALHOS RELACIONADOS

Um dos principais trabalhos é o [2] que serviu de inspiração para o desenvolvimento desse projeto. Nele, é relatado como foi aplicado o algoritmo *SWT* em imagens naturais tiradas de diferentes posições. O algoritmo encontra os textos com uma ótima precisão e é invariante a rotação, além de utilizar textos com idiomas diversos. O trabalho foi dividido em 4 etapas: (1) extração dos componentes, responsável por detectar bordas com *Sobel* e *Canny*, aplicar o *SWT* e associar os resultados; (2) análise dos componentes, onde ocorre a filtragem dos resultados e a verificação dos componentes resultantes; (3) linkar candidatos, agregando cada correspondência de letra; (4) análise de cadeias, é realizada a análise das letras para a interpretá-las como palavras e por fim, gerar uma máscara com a posição do texto.

Outro trabalho muito importante é o [3], desenvolvido por pesquisadores da Microsoft, no qual é mostrado como foi proposto o algoritmo *SWT* e seu pseudocódigo. O trabalho tem os passos bastante similares aos que serão apresentados neste artigo, exceto pela ausência da etapa de pré-processamento. O trabalho funciona da seguinte forma, primeiro, ocorre a detecção de bordas *Canny* para a etapa *SWT*, em seguida encontram-se as letras candidatas, onde as mesmas são

filtradas para a etapa posterior de detecção de palavras e por fim, é gerada uma máscara para ser aplicada na imagem original.

Por fim, o trabalho [4] que realiza a detecção de textos através de imagens capturadas por um smartphone. Nesse trabalho foi implementado o algoritmo *SWT* na linguagem C++ usando *OpenCV*, e além de detectar o texto da forma como mostrado nos trabalhos anteriores, é também realizado o reconhecimento dos caracteres da imagem. Outro ponto importante, é a utilização de um parâmetro para a identificar se o texto é mais escuro ou mais claro que o background, algo que nos trabalhos anteriores não é preciso informar.

Portanto, este artigo irá apresentar uma metodologia muito similar às apresentadas nesses artigos, no entanto, com uma contribuição que é uma melhoraria na imagem, através do pré-processamento antes de ir para a etapa de detecção de textos. Um bom resultado irá depender diretamente de uma boa escolha dos parâmetros, seja de técnicas de detecção de bordas ou operações morfológicas, para que o resultado seja o melhor possível.

### III. METODOLOGIA

O processo de detecção de textos envolve uma série de passos essenciais para a obtenção de um melhor resultado. Esse projeto foi inspirado no trabalho [2] no qual faz uso de um algoritmo que calcula a largura entre bordas paralelas. O algoritmo é *Stroke Width Transform (SWT)*, cujo objetivo é calcular as distâncias entre as bordas paralelas de uma imagem. Pois, segundo [3] as letras possuem um certo padrão entre as distâncias de um ponto a outro, o que torna mais fácil a identificação de uma letra no restante da imagem. Assim, descartando regiões que não estão dentro de um limiar gerado em tempo de execução. A metodologia, segue de acordo com a Figura 1.

Este trabalho utiliza a base de dados *MSRA Text Detection 500 Database (MSRA-TD500)* [5] contendo mais de 500 imagens de textos com diferentes fontes, diferentes orientações e diferentes idiomas. A Figura 1 mostra a metodologia utilizada neste trabalho, e foi aplicada a um conjunto de imagens selecionadas a partir dessa base descrita anteriormente. Cada etapa desse processo é essencial para o objetivo deste trabalho: detectar textos em imagens.

Para a codificação desses algoritmos foi utilizada a linguagem de programação *Python* utilizando a biblioteca *OpenCV* [6], uma das mais utilizadas em projetos de visão computacional. O código-fonte desse projeto está disponível no *GitHub* (<https://github.com/ilp1995/text-detection>) para consulta e possíveis contribuições. O *OpenCV* foi optado por oferecer maior liberdade para criação de aplicações, além de possuir uma documentação mais adequada para a finalidade deste projeto.

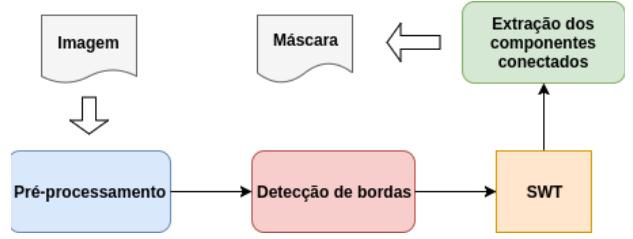


Figura 1. Etapas do processo de detecção de texto.

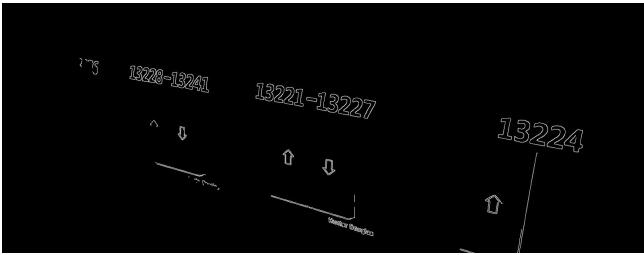
Na Figura 1, é mostrado um esquema de como funciona o processo de detecção de bordas, no qual temos uma imagem como entrada, ou seja, a imagem original que se deseja encontrar o texto. Em seguida, passa para a etapa de pré-processamento que tem o objetivo de melhorar a imagem de entrada para a etapa de detecção de bordas, no qual são aplicados alguns algoritmos de Processamento de Imagens, como *Sobel* e *Canny*. Após isso, temos o *SWT*, que consiste no algoritmo que realiza alguns cálculos sobre a imagem a fim de obter a largura das bordas. Esse resultado será aplicado na etapa posterior, a extração dos componentes conectados da imagem, ou seja, diferencia o que faz parte de um texto e de um objeto qualquer da imagem.

A primeira etapa do processo de detecção de textos é o pré-processamento, que consiste na melhoria da imagem para um melhor resultado nas etapas seguintes. Nesta etapa, a imagem pode (dependendo do parâmetro usado) ter seu histograma equalizado a fim de tornar a imagem com um contraste maior, o que torna as bordas mais evidentes e os textos borrados mais nítidos. No fim, a imagem resultante é convertida em tons de cinza, como mostra a Figura 1.



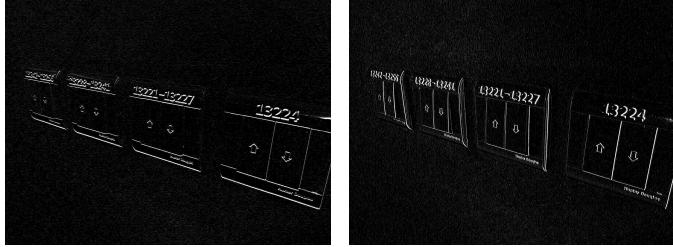
Figura 2. À esquerda a imagem original, à direita a imagem equalizada.

A segunda etapa é a detecção de bordas, o algoritmo utilizado é o *Canny*, por ser um algoritmo que faz a extração de bordas ótimas de uma imagem. O limiar mínimo e máximo é variável no algoritmo *Canny*, podendo ser definido para uma imagem específica ao ser analisada, no entanto, o padrão foi definido entre 250 e 400 para obter apenas as bordas mais definidas, no caso as bordas de textos. Além disso é possível aplicar algum algoritmo morfológico, sendo uma operação de fechamento ou uma de abertura, dependendo do parâmetro escolhido para a imagem. Na Figura 2, mostra o resultado após a aplicação do algoritmo *Canny*.



**Figura 3.** Resultado da aplicação do algoritmo Canny.

Nessa mesma etapa, ainda ocorre a extração dos gradientes vertical e horizontal da imagem, mostrado na Figura 3, a fim de obter as direções de todas as bordas, tarefa fundamental no algoritmo *SWT*. A terceira etapa, mais matemática, consiste na aplicação do algoritmo *SWT* nas imagens resultantes do algoritmo *Canny* e *Sobel*. O algoritmo *SWT* funciona primeiro identificando as bordas de alto contraste do resultado do algoritmo *Canny*, depois gerar traços nas direções das bordas obtidos no algoritmo *Sobel*.



**Figura 4.** Imagens resultantes após a aplicação do algoritmo Sobel na horizontal e vertical.

Para um maior entendimento de como funciona o cálculo da largura dos traçados basta olhar para a Figura 5. Nela é possível notar linhas de um canto da borda para o outro. Esse traçado corresponde ao tamanho da largura da borda de uma letra. No geral, as letras possuem um padrão, com variações pequenas da largura, por isso o algoritmo *SWT* é tão importante para a detecção de bordas, já que com o resultado é possível descartar partes da imagem que não possuem o padrão da fonte do texto. Assim, obtendo a imagem resultante do *SWT*, durante a etapa de extração dos componentes conectados o cálculo será mais prático, tornando a exclusão do que não faz parte do texto simplificada.



**Figura 5.** À esquerda a imagem original, à direita após aplicação do algoritmo *Stroke Width Transform (SWT)*.

A última etapa consiste em analisar a imagem resultante do algoritmo *SWT* com o objetivo de verificar as larguras das bordas para descartar o que não é texto. A etapa de extração dos componentes conectados, verifica os resultados comparando com um limiar mínimo e máximo, verificando as larguras que mais aparecem. Se uma determinada largura estiver abaixo ou acima do limiar, então o algoritmo considera aquela parte da imagem como background e é descartado da imagem. Esse passo é executado para todas as bordas. No final, teremos as áreas que o algoritmo considera como palavra ou texto. Desse modo, podemos aplicar uma máscara na imagem original, criando um retângulo nas posições obtidas no processo de extração dos componentes conectados. A Figura 6 mostra o resultado final. As áreas destacadas em verde claro, são as posições obtidas após a execução do algoritmo.



**Figura 6.** Resultado final na localização de textos em imagens.

#### IV. RESULTADOS

Os resultados foram obtidos através de testes realizados em imagens da base de dados *MSRA-TD500*. Foram selecionadas 10 imagens aleatórias. Cada imagem contém um arquivo que representa as posições reais dos textos, mostrando a posição inicial ( $x, y$ ), a largura e altura do retângulo desenhado na posição do texto na imagem. Para fins de medição da precisão

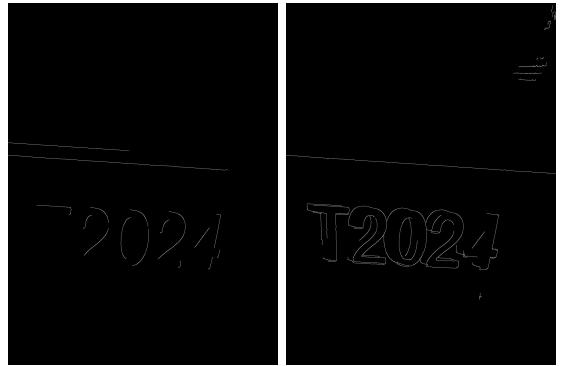
do algoritmo, ao término da execução, foi escrito em um arquivo a posição inicial ( $x, y$ ), a largura e altura do texto detectado na imagem.

Na Figura 7, foi utilizado o limiar manual no Canny, com valor mínimo em 200 e valor máximo em 300. Não foi utilizado equalização de histograma e nem operação morfológica durante o pré-processamento. O resultado se mostrou muito próximo aos valores do arquivo da base de dados. A primeira imagem corresponde ao resultado da detecção de bordas. A segunda o resultado do algoritmo SWT. A terceira o resultado das posições das letras (azul) e posições das palavras (verde). O valores de  $x$  e  $y$ , largura e altura foram, respectivamente, 154, 450, 1018 e 634 enquanto os valores da base de dados foram: 171, 596, 1083 e 355. Os valores resultam em posições muito próximas, isto é, o algoritmo obteve sucesso em encontrar o texto.



**Figura 7. Resultado aplicado com o limiar manual no Canny.**

Outra possibilidade é utilizar os parâmetros para escolher outras técnicas de processamento de imagens. Como por exemplo na Figura 8, é mostrado à direita o resultado utilizando equalização do histograma no pré-processamento das bordas e à esquerda o resultado sem a equalização. É notório que a detecção de bordas foi muito mais eficiente com a equalização, algo que é fundamental para a etapa do algoritmo SWT que tem a necessidade das bordas do texto sejam detectadas. Na figura posterior (Figura 9), o resultado final após a aplicação do algoritmo com equalização de histograma no pré-processamento.

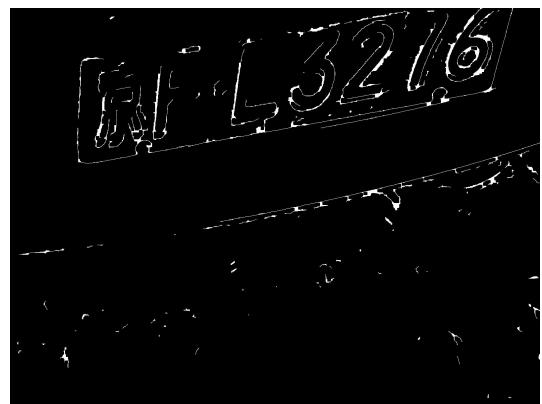


**Figura 8. À esquerda sem equalização de histograma, à direita com o histograma equalizado.**



**Figura 9. Resultado final (à esquerda a imagem *SWT* e à direita a imagem original com a máscara aplicada em verde).**

Um exemplo de imagem utilizando operação de morfologia, foi o resultado mostrado na Figura 10. Nela, foi preciso equalizar o histograma, utilizar o limiar automático no *Canny* e aplicar uma operação de fechamento (morfologia) na imagem resultante. Antes sem a definição desses parâmetros, a imagem não tinha bordas detectadas e consequentemente o resultado não foi obtido com sucesso. No entanto, o resultado final está mostrado na Figura 11. Os valores dos ponto inicial  $x$  e  $y$ , largura e altura foram, respectivamente: 225, 54, 1283 e 266. Os mesmos valores no arquivo da base de dados são: 225, 54, 1283 e 266. Ou seja, os valores possuem proximidades, o que reflete na corretude da detecção de texto.



**Figura 10. Imagem resultante após o pré-processamento.**



**Figura 11. Resultado da detecção de texto usando morfologia no pré-processamento.**

Como resultado final, alguns resultados dos testes realizados em 12 imagens da base dados e qual foi a taxa de precisão obtida desses resultados levando em consideração às posições x e y, larguras e alturas dos textos em cada imagem. Das 12 imagens testadas, 6 tiveram seus textos completamente detectados, 4 tiveram os textos parcialmente detectados e 2 tiveram a detecção incorretas ou nenhuma detecção. Ou seja, com isso podemos afirmar que o algoritmo tem uma precisão de 50% em relação a detecção de todos os textos e de 83% para detecção parcial de textos em imagens. No entanto, esse resultado pode variar dependendo das escolhas dos parâmetros para que o projeto considere outras técnicas de Processamento de Imagens durante a execução do algoritmo.

#### V. CONCLUSÕES E TRABALHOS FUTUROS

O trabalho apresentou resultados bastante satisfatórios desde que as escolhas dos parâmetros fossem bem definidas para cada tipo de imagem a ser analisada. O uso de algoritmos como os de morfologia foi fundamental para imagens em que as bordas não ficaram muito claras fossem dilatadas ou erodidas para um melhor resultado. E o uso de equalização de imagem, foi importante para imagens que tinham distorções e cores não muito simples de diferenciá-las em relação as diferentes partes da imagem, o que facilitou o algoritmo Canny identificar as bordas.

A detecção de textos utilizando o algoritmo *SWT* é um diferencial, pois o texto é detectado independentemente de sua orientação e idioma, basta está dentro de um padrão que seja

fácil o algoritmo diferenciar o que é background ou letra. Existem outras possibilidades como a detecção de textos através da detecção de regiões usando segmentação de imagem. Mas em imagens com muitos detalhes o algoritmo será enganado, uma vez que haverá diferentes regiões, por isso, o algoritmo *SWT* é muito bom em imagem com muitos detalhes.

Conclui-se que se uma imagem tiver um bom pré-processamento, os resultados obtidos serão cada vez melhores. Os algoritmos *Canny* e *Sobel* foram ótimas escolhas para a detecção de bordas, uma vez que o primeiro traz as bordas ótimas da imagem, e o segundo a direção do gradiente de diferentes posições, o que facilita o algoritmo *SWT* gerar os traçados com as larguras das bordas. No entanto, deixar que o programa decida os parâmetros por si próprio ou parâmetros padrão é bastante complexo, uma vez que as imagens são muito distintas, o que pode gerar falsos positivos ou no pior caso não detectar o texto. Por isso se fez necessário que o próprio usuário escolhesse os parâmetros baseado no tipo de imagem para prover um melhor resultado.

Como trabalho futuro está a aplicação de algoritmo de Inteligência Artificial para avaliar os resultados, ou seja, o quão efetivo está sendo o algoritmo de detecção de textos. Além disso, também espera-se automatizar a escolha de parâmetros e testar outros algoritmos de Processamento de Imagens para melhorar os resultados.

#### REFERÊNCIAS

- [1] *Google Translator*. Disponível em <<https://support.google.com/translate/answer/6142483?hl=pt-BR>>. Acessado em 01 de fevereiro de 2018.
- [2] *Detecting Texts of Arbitrary Orientations in Natural Images*. Yao, C.; Bai, X.; Liu, W.; Ma, Y.; Tu, Z. Huazhong University of Science and Technology.
- [3] *Detecting Text in Natural Scenes with Stroke Width Transform*. Epshtain, B.; Ofek, E.; Wexler, Y. Microsoft Corporation.
- [4] *Text Detection on Nokia N900 Using Stroke Width Transform*. Kumar, S.; Perrault, A. CS4670 - Computer Vision. 2010.
- [5] *MSRA Text Detection 500 Database (MSRA-TD500)*. Disponível em <[http://www.iapr-tc11.org/mediawiki/index.php/MSRA\\_Text\\_Detection\\_500\\_Database\\_\(MSRA-TD500\)](http://www.iapr-tc11.org/mediawiki/index.php/MSRA_Text_Detection_500_Database_(MSRA-TD500))> Acessado em 01 de fevereiro de 2018.
- [6] *OpenCV Python*. Disponível em <[https://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_tutorials.html](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html)>. Acessado em 01 de fevereiro de 2018.

Abaixo alguns resultados obtidos pelo algoritmo proposto neste projeto:

