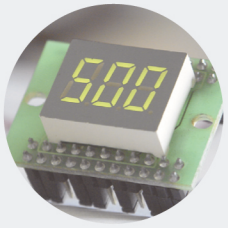
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Método visual de deteção de linhas elétricas para veículos aéreos não tripulados

TIAGO ANDRÉ MIRANDA DOS SANTOS
abril de 2017

POLITÉCNICO
DO PORTO

Método visual de deteção de linhas elétricas para veículos aéreos não tripulados

Tiago André Miranda dos Santos
Nº 1131368

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Sistemas Autónomos
Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

2017

Dissertação, para satisfação parcial dos requisitos do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Tiago André Miranda dos Santos

Nº 1131368

Orientador: José Miguel Soares De Almeida

Co-Orientador: André Miguel Pinheiro Dias

Mestrado em Engenharia Eletrotécnica e de Computadores

Área de Especialização de Sistemas Autónomos

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

3 de Abril de 2017

Agradecimentos

A realização desta dissertação não seria possível sem o apoio e contributo de várias pessoas. A todos eles deixo o meu agradecimento profundo.

Em primeiro lugar, queria agradecer ao meu orientador, Engº José Almeida por me ter proporcionado esta oportunidade e pelo suporte prestado ao longo deste percurso.

Gostaria de agradecer ao Engº André Dias pelo apoio e acompanhamento disponibilizado ao longo do mestrado.

A todos os membros do laboratório de sistemas autónomos (LSA), por todo o conhecimento transmitido.

Aos meus colegas de trabalho, Alexandre Oliveira, André Ferreira, Fábio Azevedo e Miguel Moreira pela ajuda e disponibilidade oferecida.

À EDP Labelec, em especial, ao João Formiga e Jorge Dinis, pela colaboração na recolha de dados utilizados nesta dissertação.

A todos os meus amigos de longa data pela amizade e companheirismo demonstrado ao longo destes anos.

À minha namorada, Joana Santos, pelo apoio, carinho, paciência e incentivo durante a elaboração desta dissertação.

Por fim, um agradecimento à minha família, em especial aos meus pais, pela educação que me proporcionaram e pelo apoio, incentivo e esforços efetuados ao longo da minha formação académica.

Esta página foi intencionalmente deixada em branco.

Resumo

A qualidade do serviço das empresas de eletricidade é assegurada pela inspeção regular de linhas elétricas e pelos procedimentos de manutenção preventiva. Métodos tradicionais como um helicóptero têm muitas desvantagens, tais como, altos custos e riscos. Os veículos aéreos não tripulados (UAV) estão agora a ser utilizados para efetuar inspeções, uma vez que podem ser equipados com sensores semelhantes aos do helicóptero.

Nesta dissertação propõe-se endereçar o problema de aumentar as capacidades de perceção em tempo real dos UAVs para dotá-los de capacidades para operações autónomas e semi-autónomas seguras e robustas. Nesse sentido o trabalho irá endereçar um dos componentes do sistema de perceção, nomeadamente a deteção de linhas elétricas por visão.

Um novo método para deteção de linhas elétricas em imagens, designado de PLineD, foi desenvolvido, e é capaz de melhorar a robustez da deteção mesmo na presença de imagens com ruído de fundo. Como base, usa o algoritmo Edge Drawing (ED), selecionado a partir da comparação efetuada do desempenho de três algoritmos, identificados no estado da arte.

O algoritmo foi testado num conjunto de várias imagens reais de linhas elétricas, com múltiplos fundos e condições climatéricas. Os resultados experimentais demonstram que o PLineD é eficaz e capaz de ser implementado numa *pipeline* de processamento de imagem em tempo real.

Palavras-Chave: Linhas Elétricas, Edges, Deteção, PLineD

Esta página foi intencionalmente deixada em branco.

Abstract

The electrical companies service quality is ensured by regular power line inspection and by preventive maintenance procedures. Traditional methods like an helicopter have many disadvantages such as, high costs and risks. Unmanned Aerial Vehicles (UAV) are now being used to carry out inspections given that it can be equipped with similar sensors to helicopter.

This paper is part of the efforts to address the problem of improving the real-time perception capabilities of UAVs for endowing them with capabilities for safe and robust autonomous and semi autonomous operations. And focuses in one of the components of the perception system, namely the vision based power line detection.

A new method of vision based power line detection algorithm denoted by PLineD, was developed, and it's able to improve the detection robustness even in the presence of images with background noise. As a base, it uses the Edge Drawing (ED) algorithm, selected from the performance comparison of three algorithms, identified in the state of the art.

The algorithm was tested in real outdoor images of a dataset with multiple backgrounds and weather conditions. The experimental results demonstrates that the PLineD is effective and able to implemented in real-time image processing pipeline.

Palavras-Chave: Power lines, Edges, Detection, PLineD

Esta página foi intencionalmente deixada em branco.

Conteúdo

Lista de Figuras

Lista de Tabelas

Esta página foi intencionalmente deixada em branco.

Lista de Siglas e Acrónimos

CRAS Centro de Robótica e Sistemas Autónomos

ED Edge Drawing

EDP Energias de Portugal

INESC TEC Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

LSA Laboratório de Sistemas Autónomos

LiDAR Light Detection and Ranging

LSD Line Segment Detector

NFA Número de Falsos Alarmes

UAV Unmanned aerial vehicle

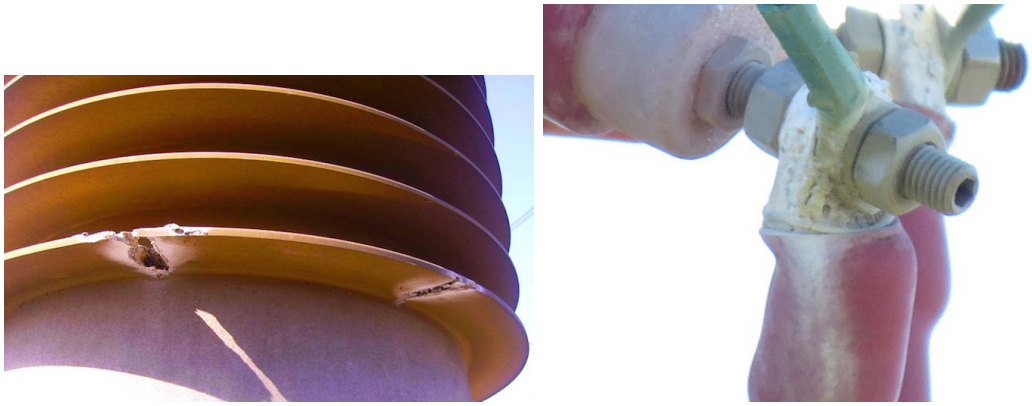
Esta página foi intencionalmente deixada em branco.

Capítulo 1

Introdução

A qualidade de serviço prestada pelos fornecedores de eletricidade está bastante dependente das possíveis falhas que possam ocorrer nos vários elementos constituintes das linhas elétricas. Uma das possíveis medidas de manutenção preditiva, de forma a prevenir falhas, e garantir ações preventivas é a inspeção visual dos ativos elétricos. Estas ações contribuirão não só para criar condições que permitam evitar interrupções inesperadas na rede elétrica, mas também para prevenir possíveis acidentes. Isto, resulta numa poupança a nível monetário, pois reduz-se os gastos em possíveis remodelações nos troços das linhas elétricas ou em indemnizações a clientes.

Alguns dos problemas expectáveis que possam ser detetados por uma inspeção visual são quebras nos isoladores ou sinos, perda de galvanização, ferrugem e também a invasão da vegetação perto dos ativos [1] [2], como demonstrado pela figura 1.1.



(a) Quebra nos isoladores [3] (b) Corrosão galvânica [4]

Figura 1.1: Exemplo de ativos elétricos com problemas.

Os métodos de inspeção de ativos elétricos podem ser divididos em três categorias: inspeção terrestre por uma equipa humana, veículos aéreos tripulados [1] e mais recentemente por veículos aéreos não-tripulados (*Unmanned Aerial Vehicles* - UAV's) operados remotamente [1].

Dos métodos anteriormente enumerados, a inspeção visual pelo solo é aquela que apresenta menor risco, contudo o processo é lento, menos preciso e necessita de boa acessibilidade aos terrenos. Os veículos aéreos tripulados, como os helicópteros, fornece um processo de inspeção rápido, no entanto com custos operacionais muito elevados (o que o faz ser economicamente viável apenas para inspeção de áreas grandes) e apresenta algum risco para os operadores devido ao voo próximo das linhas elétricas e apoios que requerem uma inspeção de proximidade. Mais recentemente, a inspeção de ativos com UAVs, maioritariamente tele-operados, tem crescido a um ritmo elevado, uma vez que reduzem o risco humano e os custos da operação (mais significativo em cenários de inspeção pontuais), e simultaneamente permitem obter dados a partir de múltiplas posições, ângulos e distâncias para os ativos, contribuindo para uma melhor qualidade dos dados recolhidos.

Apesar das vantagens dos UAVs, existem também riscos, como a existência de redes complexas de linhas elétricas, que devem ser detetadas e evitadas durante o voo do UAV. Isto, juntamente com o facto de que a perceção visual do ambiente pode ser perturbada por vários ruídos de fundo (por exemplo relva, árvores, cercas e estradas), como ilustrado na figura 1.2,a deteção das linhas elétricas pelo operador pode se tornar uma tarefa impossível. Para tal, a utilização de um módulo de deteção de linhas elétricas, que permita evitar a colisão, revela-se essencial para auxilio do operador. A perceção de linhas elétricas contribui também para aprimorar as capacidades autónomas de um UAV, não só como um módulo anti-colisões, mas como um guia que permite que o voo seja efetuado ao longo da linha.

Nesta dissertação propõe-se então endereçar o problema de aumentar as capacidades de perceção em tempo real dos UAVs para dotá-los de capacidades para operações autónomas e semi-autónomas seguras e robustas. Nesse sentido o trabalho irá endereçar um dos componentes do sistema de perceção, nomeadamente a deteção de linhas elétricas por visão, mesmo na presença de imagens com ruído de fundo e linhas elétricas curvas.

Figura 1.2: Exemplo do ambiente que rodeia as linhas elétricas.

1.1 Motivação

O Centro de Robótica e Sistemas Autónomos (CRAS) do INESC TEC, ao longo dos últimos anos, tem participado em vários projetos de robótica aplicada, no meio aquático, aéreo e terrestre, tal como demonstrado por alguns dos robôs presentes na figura 1.3.

Esta dissertação insere-se nos projetos do CRAS em particular no projeto "Drones para inspeção de ativos elétricos"em parceira com a EDP Labelec e consiste no desenvolvimento de um UAV adaptado ao processo de inspeção aérea de ativos elétricos, como por exemplo, torres eólicas, subestações, barragens, apoios e linhas elétricas. O primeiro protótipo, designado de BYRD, presente na figura 1.4, já efetuou vários voos autónomos nas várias zonas de interesse para inspeção, permitindo assim a recolha de dados georreferenciados de LiDAR e imagens de espetro visível e termográfico. A informação adquirida permite que em *backoffice* os técnicos da EDP Labelec possam avaliar e gerar relatórios do estado dos ativos EDP.

Figura 1.3: Alguns dos robôs desenvolvidos pelo CRAS

Figura 1.4: Byrd I em processo de inspeção autónoma de um apoio de Média Tensão.

1.2 Objetivos

A dissertação endereça o problema de reconhecimento de linhas elétricas com recurso a uma câmara de espetro visível. O trabalho desenvolvido tem como objetivo dotar um UAV da capacidade de navegar em modo autónomo e semi-autónomo na presença de linhas elétricas.

Deste modo, o desenvolvimento do projeto implica a concretização dos seguintes objetivos:

- Identificação do problema/limitações do processo de deteção de linhas elétricas;
- Análise do estado de arte do processo de reconhecimento de linhas elétricas;
- Análise comparativa de métodos que permitam endereçar o reconhecimento de linhas elétricas;
- Desenvolvimento de um método robusto que permita fazer o reconhecimento de linhas elétricas com base nos requisitos definidos para a solução;
- Validação do método desenvolvido para os diferentes cenários anteriormente identificados e avaliação do desempenho relativamente aos métodos discutidos no estado de arte.

1.3 Estrutura

Esta dissertação encontra-se organizada em oito capítulos. No segundo capítulo é apresentado um estudo acerca de diferentes métodos já desenvolvidos pela comunidade científica sobre reconhecimento de linhas elétricas com câmaras visíveis.

De seguida, no capítulo 3 serão abordados conceitos e fundamentos necessários para a compreensão desta dissertação, tendo como base alguns conceitos de processamento de imagem e alguns métodos analisados no capítulo do estado da arte.

O capítulo 4 contém uma comparação entre três algoritmos para deteção de linhas elétricas em imagens, apresentados no estado da arte, e do qual será escolhido um como base para o método desenvolvido.

No capítulo seguinte, Capítulo 5, é descrito o método desenvolvido para deteção de linhas elétricas, bem como uma explicação de cada um dos passos constituintes.

Os resultados obtidos são apresentados no capítulo 6, onde o algoritmo é testado em várias imagens, recolhidas por um UAV, de linhas elétricas e é comparado com os algoritmos selecionados no estado da arte.

Por último, no capítulo 7 são apresentadas algumas conclusões sobre o trabalho desenvolvido, bem como o trabalho futuro a realizar.

Capítulo 2

Estado da Arte

Neste capítulo são endereçados os métodos que se enquadrem no processo de deteção de linhas elétricas com recurso a imagens de câmaras visíveis. O estudo apresentado neste capitulo irá permitir uma melhor compreensão das métodos existentes e dos desafios associados aos processo de deteção de linhas elétricas.

2.1 Métodos de deteção de linhas elétricas

No estado da arte, os algoritmos de visão para deteção de linhas elétricas são geralmente divididos em dois passos principais, transformação da imagem a tons de cinza numa imagem binária, obtendo segmentos, e extração dos segmentos que pertencem às linhas elétricas [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15].

A transformação numa imagem binária é feita na maioria por detetores de edges, obtendo segmentos que correspondem intuitivamente aos limites dos objetos, tais como, aplicar um *threshold* na imagem a tons de cinza [5] [6] [7], Line Detector Mask ou Ratio Line Detector [8],CVK method [9], canny edge detector [10] [11] e Marr-Hildreth detector [12]. No entanto, existem também aqueles que usam extração de pontos ridge [11] [14] [15], em vez da deteção de edges, ou filtros, como Pulse-Coupled Neural Network [13].

A maioria dos autores usa a transformada de Hough como base dos seus algoritmos de extração dos segmentos que pertencem às linhas elétricas [5] [7] [9] [10] [12] [13]. Outros, optam pela utilização de métodos diferentes, tais como, Heuristics based line detection [6], Radon Transform [8], Circle Based Search [11] e Line Fitting [15] [14].

Zhang em [5] usa o método de *threshold* de Otsu [16] seguido da deteção de linhas retas usando a transformada de Hough [17]. As linhas elétricas são filtradas usando um

K-means no espaço de Hough, isto é, agrupa segmentos de linhas consoante o ângulo e a distância. Para cada grupo faz a estimação de uma linha reta pelo método *least squares*. Aquando do uso de uma sequência de imagens, tal como vídeo continuo, o filtro de Kalman [18] pode ser aplicado no espaço de Hough para efetuar *tracking* às linhas elétricas.

Yang em [7] recorre a um método de *threshold* adaptado, que divide a imagem em secções NxM e calcula um valor de *threshold* para cada uma das secções. Os segmentos de linha são então obtidos pela transformada de Hough. A discriminação das linhas elétricas é feita pelo algoritmo de *cluster* Fuzzy C-means [19] [20] que agrupa as linhas em vários grupos, sendo um desses grupos, o grupo das linhas elétricas.

Gerke em [10], propõe a utilização de um filtro de *Range* [21] que realça as linhas na imagem. Os objetos que não sejam longos nem finos são eliminados por operações morfológicas. Neste instante é aplicado o detetor de Canny edge [22], permitindo remover os chamados *outer* edges. A transformada de Hough é então utilizada, onde partindo do espaço de Hough é removido todos os picos que não tenham um pico correspondente na proximidade e com uma orientação semelhante.

Em [12], Tong recorre ao detetor de Marr-Hildreth edge [23] e à transformada de Hough para obter segmentos de linhas. A estes segmentos é aplicado um passo de análise morfológica, que preenche as falhas em segmentos de linhas e remove todos os segmentos que sejam menores em comprimento e largura que um *threshold*.

Zhang em [13] propõe a utilização de um modelo modificado de Pulse-Coupled Neural Network [24] como filtro para reduzir ruído, seguido da transformada de Hough com o procedimento de votação melhorado [25]. As linhas que não são paralelas são removidos por um *cluster*.

Sharma em [6], recorre a um *kernel* Gaussiano 3x3 seguido de um operador *majority based erosion*, que reduz a largura das linhas para um único pixel, como método de *threshold*. A deteção de linhas é alcançadas por *Heuristics based line detection*, isto é, como a deteção de linhas elétricas apenas é feita por vista vertical, assume que estas vão desde a parte inferior da imagem até à parte superior. Na parte inferior, identifica o inicio das linhas e acumula o tamanho da linha até que encontre uma quebra.

Yan em [8] recorre ao uso do Line Detector Mask [26] ou Ratio Line Detector [27] restringido pela refletância do material da linha elétrica, como detetor de edges da imagem, tendo cada um as suas vantagens. Recorre à transformada de Radon [28] para detetar linhas na imagem, seguido de um agrupamento de segmentos de linha consoante o ângulo e a distância entre os pontos da extremidade. Com base nos segmentos obtidos, o método propõe efetuar a estimação da linha elétrica assumindo a aproximação a uma

linha reta. No sentido a melhorar a estimação da posição das linhas, o filtro de Kalman é usado para preencher as falhas entre linhas, resultante do passo anterior.

Céron em [11] utiliza o método de Canny ou a identificação dos pontos de ridge [29] a partir do filtro *steerable* [21] como métodos para obter segmentos. Para deteção das linhas usa o método de Circle Based Search, que a partir da simetria do círculo procura por pontos válidos que pertencem a uma linha. Baseado nos princípios de Gestalt [30], faz a junção das linhas, recorrendo às caracteristicas como similaridade, comprimento e proximidade das linhas. As linhas elétricas são aquelas que apresentam um comprimento maior, na direção com maior número de linhas e aquelas com linhas paralelas.

Liu apresenta dois métodos de deteção de linhas elétricas em [14] e [15] em que, o segundo método acrescenta um passo final ao primeiro. O primeiro método alcança a extração de segmentos baseados na identificação dos pontos ridge a partir da função de energia do filtro *steerable* e usa a técnica de análise aos elementos agregados para ser efetuado a estimação de uma linha pelo método dos *least squares*. O segundo método acrescenta um passo de agrupamento de linhas baseado nos princípios de Gestalt, usando as caracteristicas como proximidade, continuidade e similaridade.

Os métodos de deteção de retas também podem ser utilizados para extração das linhas elétricas [11] [14] e [15] onde, a performance dos algoritmos desenvolvidos é avaliada face a dois métodos de deteção de retas, chamados de LSD [31] e EDLines [32].

Yetgin em [33] apresenta um estudo comparativo destas duas técnicas, juntamente com a transformada de Hough, para deteção de linhas elétricas por aviões que voem a baixa altitude.

Rafael Gioi apresenta o método *Linear-time Line Segment Detector* (LSD) [31] que agrupa pixeis em regiões que partilham o mesmo ângulo de gradiente e efetua uma aproximação dos mesmos a um retângulo, devido à sua aproximação a uma reta. A validação do retângulo como reta é calculado pelo número de falsos positivos usando o método de Desolneux [34].

Akinlar em [32] apresenta o algoritmo EDLines que pode ser dividido em 3 passos: Extração de segmentos de edges usando o algoritmo Edge Drawing [35] [36]. Geração de retas de segmentos pelo método *least squares line fitting*. Tal como no LSD, é efetuada a validação da reta pelo método de Desolneux.

Lu em [37] apresenta um método para deteção de retas chamado de CannyLines. O operador de Canny, usado em conjunto com o método desenvolvido que calcula os *thresholds* automaticamente para cada imagem, faz a extração de edges, que efetuando um passo de ligação e divisão de edges, alimenta o método *least squares line fitting*. Tal como no EDLines e LSD, é efetuada a validação da reta por uma versão mais robusta

do mesmo método.

2.2 Discussão do Estado da Arte

Na análise efetuada do estado de arte, foram identificados uma grande variedade de métodos de processamento de imagem, tendo todos em comum o procurar tirar proveito de algumas caracteristicas físicas da linhas elétricas como é o caso do paralelismo, o fato da linha puder ser aproximada a uma reta e a distância entre linhas.

O passo da extração dos segmentos que pertencem às linhas elétricas, os autores partem do pressuposto que as linhas elétricas em imagens podem ser aproximadas a uma reta. Esta característica é verdadeira, apenas quando se utiliza imagens retiradas com uma vista vertical face às linhas elétricas, não indo de encontro aos objetivos definidos nesta dissertação.

Os resultados apresentados pelos métodos de deteção de linhas elétricas consistem em ilustrações das linhas elétricas extraídas a partir de imagens de baixa resolução e em cenários não muito complexos. Casos mais complexos de cenários com linhas elétricas não são abordados, deixando na dúvida a verdadeira valia dos métodos apresentados. Em alguns casos, também é apresentado o tempo de execução de cada algoritmo.

Os algoritmos de deteção de retas apresentam caracteristicas interessantes para deteção de linhas elétricas, mas como não se encontram focados em linhas elétricas, apresentam um maior número de falsos positivos.

Nesse sentido, identificaram-se 3 métodos promissores, LSD[31], EDLines[32] e Canny-Lines [37], para qual pretendemos analisar a sua performance através de uma análise comparativa e identificar uma linha de trabalho que vai ser objeto de contribuição da dissertação.

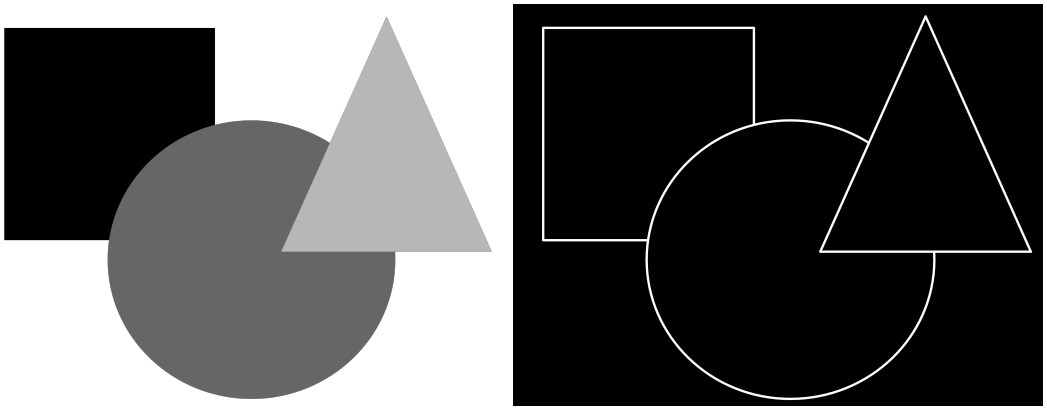
Capítulo 3

Fundamentos Teóricos

Neste capítulo será efetuado um estudo teórico de algumas técnicas e conceitos usados em processamento de imagem, que servirão de apoio à compreensão das temáticas abordadas na presente dissertação. Uma outra linha de trabalho que irá ser analisado é o funcionamento detalhado dos algoritmos selecionados no Estado da Arte.

3.1 Edges

O termo edge é associado a uma alteração local, significativa, na intensidade da imagem, que ocorre tipicamente nos limites entre duas regiões diferentes. Normalmente, um edge é associado a uma descontinuidade na intensidade da imagem ou na primeira derivada da intensidade da imagem. A figura 3.1 ilustra um exemplo de deteção de edges, onde todos os limites das formas geométricas são edges.



(a) Imagem Original (b) Deteção de edges

Figura 3.1: Ilustração de deteção de edges. (a) Imagem original com algumas figuras geométricas, (b) Deteção de edges

3.2 Magnitude e Direção do Gradiente

O gradiente de uma imagem é uma medida da alteração da direção da intensidade. Do ponto de vista matemático, o gradiente de uma função de duas variáveis (neste caso, a função da intensidade da imagem), em cada pixel da imagem, é um vetor 2D em que as suas componentes são dadas pelas derivadas da função nas direções verticais e horizontais (fórmula 3.1) . Em cada pixel da imagem (x,y), o vetor do gradiente aponta na direção onde ocorre a alteração mais rápida na intensidade e o tamanho do vetor do gradiente corresponde à taxa de variação nessa direção.

$$G[f(x,y)] = \begin{bmatrix} Gx \\ \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.1}$$

A magnitude do gradiente é dado por:

$$G[f(x,y)] = \sqrt{G_x^2 + G_y^2} \tag{3.2}$$

A partir da análise dos vetores, a direção do gradiente é definida como:

$$\alpha(x,y) = \arctan\left(\frac{G_x}{G_y}\right) \tag{3.3}$$

3.3 Operador de Sobel

O operador de Sobel é uma técnica que calcula uma aproximação do gradiente da função da intensidade da imagem. Considerando os pixeis vizinhos em torno do pixel [i,j]:

$$\begin{bmatrix} a0 & a1 & a2 \\ a7 & [i,j] & a3 \\ a6 & a5 & a4 \end{bmatrix}$$

e as equações da derivada parcial:

$$s_x = (a_2 + 2a_3 + a_4) - (a_0 + 2a_7 + a_6) \tag{3.4}$$

$$s_y = (a_0 + 2a_1 + a_2) - (a_6 + 2a_5 + a_4) \tag{3.5}$$

o operador de Sobel é a magnitude do gradiente calculado por:

$$M(i,j) = \sqrt{s_x^2 + s_y^2} \tag{3.6}$$

.

As variáveis s_x e s_y podem ser calculados usando matrizes de convolução (kernel):

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} s_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.7}$$

A figura 3.2 mostra um exemplo de convolução entre um kernel e um pedaço da imagem, resultando na magnitude do pixel central.

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \right) [2,2] = (i*1) + (h*2) + (g*3) + (f*4) + (e*5) + (d*6) + (c*7) + (b*8) + (a*9).$$

Figura 3.2: Convolução de duas matrizes.

3.4 Kernel Gaussiano

O kernel Gaussiano é usado para suavizar imagens e remover ruído.

Os coeficientes do kernel gaussiano são amostrados a partir da função Gaussiana 2D:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.8}$$

onde, σ é o desvio padrão da distribuição. Assume-se que a distribuição tem uma média de (0,0).

A figura 3.3 mostra um exemplo do kernel 5x5 que aproxima a função gaussiana com um σ de 1.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figura 3.3: Aproximação discreta da função Gaussiana com σ=1.

3.5 Least Squares line fit

O least Squares line fit é um procedimento matemático que tenta encontrar a melhor reta que representa um conjunto de dados, minimizando a soma dos erros quadráticos.

Dado um conjunto de dados ($x1$,$y1$),...,(x_N,y_N), o erro associado à reta $y = ax + b$ é

$$E(a,b) = \sum_{n=1}^{N} (y_n - (ax_n + b))^2 \tag{3.9}$$

O objetivo deste processo é descobrir os valores de a e b que minimiza o erro. Isto requer que se encontre os valores (a,b) tais que

$$\frac{\partial E}{\partial a} = 0, \frac{\partial E}{\partial b} = 0 \tag{3.10}$$

resultando na matriz final

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} 1 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{n=1}^{N} x_n y_n \\ \sum_{n=1}^{N} y_n \end{bmatrix} \tag{3.11}$$

3.6 Métodos identificados no Estado da Arte

No Estado da Arte os algoritmos LSD, EDLines e CannyLines, foram selecionados como métodos promissores para a deteção de linhas elétricas. Nesta secção, é explicado detalhadamente o funcionamento de cada um dos métodos.

3.6.1 LSD

Rafael Gioi em conjunto com Jéremie Jakubowicz, Jean-Michel Morel and Gregory Randall desenvolveram o *Line Segment Detector* (LSD) [31], que tem como intuito de retorna uma lista de segmentos de retas detetadas a partir de uma imagem em tons de cinza. O LSD foi desenvolvido com o objetivo de ser aplicado sem a existência de qualquer tipo de calibração de parâmetros, mas depende de alguns números que determina o seu comportamento. O algoritmo 1 apresenta a descrição completa em pseudocódigo.

Algorithm 1 : LSD: Line Segment Detector

input: An image I
output: A list out of rectangles
$I_s \leftarrow \text{ScaleImage}(I, S, \sigma = \frac{\sum}{S})$
$(LLA, |\bigtriangledown I_s|, OrderedListPixels) \leftarrow Gradient(I_s)$
$Status \leftarrow= \begin{cases} USED, & pixels\ with\ |\bigtriangledown I_s| \leq \rho \\ NOT\ USED, & otherwise \end{cases}$
for *pixel* $P \in OrderedListPixels$ **do**
  if $Status(P) = NOT\ USED$ **then**
    $region \leftarrow RegionGrow(P, \tau)$
    $rect \leftarrow Rectange(region)$
    while $AlignedPixelDensity(rect, \tau) < D$ **do**
      $region \leftarrow CutRegion(region)$
      $rect \leftarrow Rectange(region)$
    end while
    $rect \leftarrow ImproveRectangle(rect)$
    $nfa \leftarrow NFA(rect)$
    if $nfa \leq \varepsilon$ **then**
      $rect \rightarrow out$
    end if
  end if
end for

O LSD encontra-se dividido em 7 passos, listados seguidamente:

- Escalar a imagem;

- Cálculo do gradiente;
- Pseudo-Ordenação do gradiente;
- *Threshold* do gradiente;
- Região crescente;
- Aproximação a um retângulo;
- Cálculo do número de falsos alarmes.

3.6.1.1 Reescalar a imagem

O primeiro passo do LSD consiste em reescalar a imagem de entrada para 80% (S=0.8) do seu tamanho. Este passo ajuda a lidar com o efeito de *aliasing* (especialmente o efeito de escada) presente em muitas imagens.

A figura 3.4 mostra dois casos de deteção de edges em ângulos diferentes, acompanhados do resultado do LSD sem a aplicação deste passo inicial. No primeiro caso o edge é detetado como quatro retas horizontais e no segundo caso não é detetada qualquer reta. A figura 3.5 mostra o resultado do LSD, usando uma escala de 80%. Ambos os edges são detetados e com a orientação certa (mesmo com o primeiro a estar igualmente fragmentado).

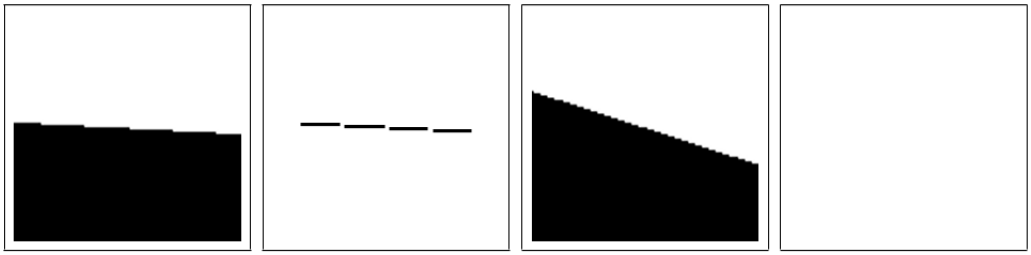


Figura 3.4: Deteção de dois edges com ângulos diferentes sem recurso à etapa de reescalar a imagem. [31]

Esta reescala é feita por uma sub-amostragem gaussiana. A imagem é filtrada por kernel gaussiano e depois é sub-amostrado. O desvio padrão do kernel gaussiano é determinado por $\sigma = \sum /S$, onde S é o fator de escala. O valor de $\sum$ é igualado a 0.6.

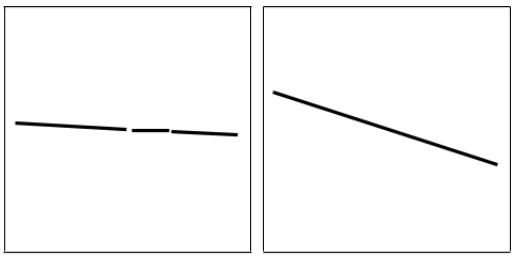


Figura 3.5: Deteção para as imagens na figura 3.4 recorrendo ao passo de reescalar a imagem. [31]

3.6.1.2 Cálculo do gradiente

O gradiente da imagem é calculado em cada pixel usando uma máscara 2x2, dado por:

$$\begin{bmatrix} i(x,y) & i(x+1,y) \\ i(x,y+1) & i(x+1,y+1) \end{bmatrix}$$

onde i(x,y) é o valor em tons de cinza do pixel (x,y). O gradiente da imagem é calculado por:

$$g_x(x,y) = \frac{i(x+1,y) + i(x+1,y+1) - i(x,y) - i(x,y+1)}{2} \tag{3.12}$$

$$g_x(x,y) = \frac{i(x,y+1) + i(x+1,y+1) - i(x,y) - i(x+1,y)}{2} \tag{3.13}$$

A direção do edge é calculado por:

$$\alpha(x,y) = \arctan\left(\frac{g_x(x,y)}{-g_y(x,y)}\right) \tag{3.14}$$

E a magnitude do gradiente é calculado pela equação 3.2.

3.6.1.3 Pseudo-Ordenação do gradiente

A ordem em que os pixeis são processados tem impacto no resultado final. Num edge, o pixel central normalmente tem um valor maior de gradiente, sendo por este que se começa a procurar segmentos de retas.

Para tal, são criados 1024 compartimentos correspondentes a intervalos iguais de magnitude entre zero e o maior valor observado na imagem. Os pixeis são classificados nestes compartimentos de acordo com o valor da magnitude do gradiente.

3.6.1.4 *Threshold* do gradiente

Os pixeis com gradiente inferior a ρ são rejeitados e não são usados nos passos seguintes. O *threshold* ρ é dado pela equação:

$$\rho = \frac{q}{\sin \tau} \tag{3.15}$$

onde, q é um limite no erro possível no valor do gradiente devido aos efeitos de quantização, e τ é o ângulo de tolerância usado no algoritmo de região crescente.

Pelas experiências efetuadas valor de q é usado como 2.

3.6.1.5 *Region Growing*

Começando por um pixel na lista ordenada de pixeis não usados, o algoritmo de *region growing* é aplicado para formar um conjunto de pixeis agregados, chamado de *line-support region*. Recursivamente, os vizinhos não usados dos pixeis já na região são testados, e aqueles com ângulo igual ao ângulo da região θ_{region}, até a uma tolerância τ, são adicionados. O ângulo inicial da região θ_{region} é o ângulo do primeiro pixel, e cada vez que um pixel é adicionado à região, o valor do ângulo da região é atualizado por:

$$\theta_{region} = \arctan\left(\frac{\sum_j \sin(anguloPixel_j)}{\sum_j \cos(anguloPixel_j)}\right) \tag{3.16}$$

onde, o índice j passa por todos os pixeis da região.

O algoritmo 2 descreve o que é efetuado neste passo.

No paper [31], a vizinhança dos 8 pixeis mais próximos é usada e a tolerância de τ é usada como 22.5 graus.

A figura 3.6 detalha o processo apresentado no algoritmo 2. À esquerda encontra-se um edge ruidoso e é acompanhado pelo resultado do algoritmo de região crescente para um τ de 11.25, 22.5 e 45 graus, respetivamente.

3.6.1.6 Aproximação a um retângulo

Um segmento de reta pode ser aproximado pela forma geométrica de um retângulo. Antes de avaliar a *line-support region*, o retângulo associado deve ser encontrado. A região de pixeis é interpretada como um objeto sólido e a magnitude do gradiente de cada pixel é usada como a "massa"desse ponto. O comprimento e largura do retângulo é definida para os valores mínimos que coloque o retângulo a cobrir toda a *line-support region*.

Algorithm 2 : Region Grow

input: A pixel angle LLA, a start pixel P, an angle tolerante τ and a Status variable for each pixel
output: A set of pixels: region
Add $P \rightarrow region$
$\theta_{region} \leftarrow LLA(P)$
$S_x \leftarrow \cos(\theta_{region})$
$S_y \leftarrow \sin(\theta_{region})$
for each *pixel* $P \in region$ **do**
  for each *pixel* $Q \in Neighborhood(P)$ **and** $Status(Q) \neq USED$ **do**
    if $AngleDiff(\theta_{region}, LLA(Q)) \leq \tau$ **then**
      Add $Q \rightarrow region$
      $STATUS(Q) \leftarrow USED$
      $S_x \leftarrow S_x + \cos(LLA(Q))$
      $S_y \leftarrow S_y + \sin(LLA(Q))$
      $\theta_{region} \leftarrow \arctan(S_y/S_x))$
    end if
  end for
end for

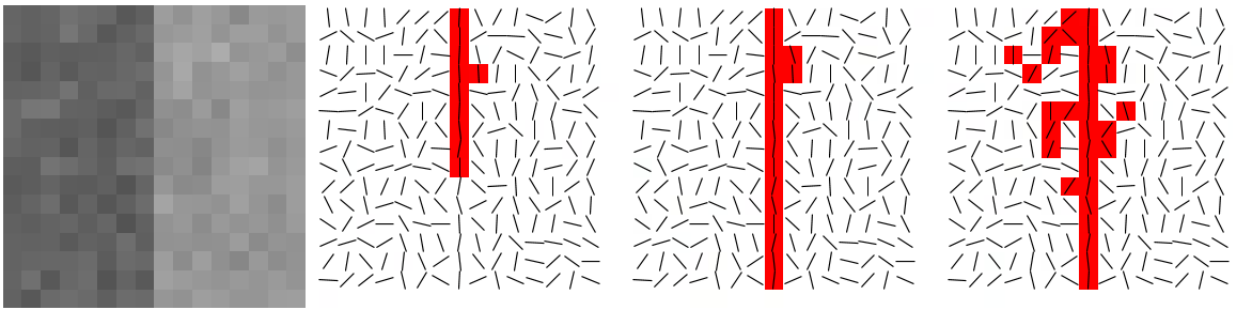


Figura 3.6: Exemplos de *region growing* começando pelo pixel central do topo para três valores de tolerância de ângulo Da esquerda para a direita: $\tau = 11.25$, $\tau = 22.5$ e $\tau = 45$. [31]

O centro do retângulo (cx,cy) é dado por

$$c_x = \frac{\sum_{j \in Region} G(j) * x(j)}{\sum_{j \in Region} G(j)} \tag{3.17}$$

$$c_y = \frac{\sum_{j \in Region} G(j) * y(j)}{\sum_{j \in Region} G(j)} \tag{3.18}$$

onde G(j) é a magnitude do gradiente do pixel j. O ângulo do retângulo é igualado ao ângulo do vetor próprio associado ao valor inferior dos valores próprios da matriz

$$\begin{bmatrix} m^{xx} & m^{xy} \\ m^{xy} & m^{yy} \end{bmatrix}$$

com

$$m^{xx} = \frac{\sum_{j \in Region} G(j) * (x(j) - c_x)^2}{\sum_{j \in Region} G(j)} \tag{3.19}$$

$$m^{yy} = \frac{\sum_{j \in Region} G(j) * (y(j) - c_y)^2}{\sum_{j \in Region} G(j)} \tag{3.20}$$

$$m^{xy} = \frac{\sum_{j \in Region} G(j) * (x(j) - c_x)(y(j) - c_y)}{\sum_{j \in Region} G(j)}. \tag{3.21}$$

Em alguns casos, o ângulo τ produz uma interpretação errada. Isto pode acontecer quando dois edges retos formam um ângulo inferior à da tolerância τ. A figura 3.7 mostra um exemplo de uma *line-support region* encontrada (a cinzento) e o retângulo correspondente.

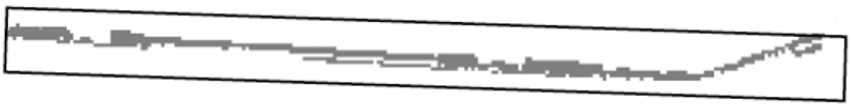


Figura 3.7: Um problema que pode ocorrer no processo de *region growing* e a aproximação a um retângulo. [31]

No LSD este problema é tratado encontrando as regiões de *line-support* problemáticas e separando-as em duas regiões mais pequenas, na esperança que se corte a região no sítio certo para resolver o problema.

A deteção é feita pelo cálculo da densidade de pontos alinhados num retângulo pelo rácio entre o número de pontos alinhados k e a área do retângulo:

$$d = \frac{k}{length(r) * width(r)} \tag{3.22}$$

Um *threshold* D é definido como mínimo da densidade de pontos alinhados de um retângulo, que pelos autores em [31], foi definido como 0.7 (70%).

Dois métodos de corte da região são tentados: Reduzir o ângulo de tolerância e reduzir o raio da região.

3.6.1.7 Reduzir o ângulo de tolerância

O primeiro método, reduzir o ângulo de tolerância, tenta estimar um novo valor de tolerância τ' que se adapte bem à região. Assim, utiliza todos os pixeis que tenham uma distância, ao pixel inicial, inferior à largura do retângulo inicialmente calculado, gerando a nova tolerância τ' para o dobro do desvio padrão dos ângulos desses pixeis. Com este novo valor, o mesmo algoritmo de crescimento de região é aplicado, começando pelo mesmo pixel inicial.

3.6.1.8 Reduzir o tamanho da região

O método anterior, reduzir o ângulo de tolerância, é apenas tentado uma vez, e se o resultado da região *line-support* falhar em satisfazer o critério da densidade, o segundo método é tentado repetidamente. O objetivo desta etapa do método é de remover gradualmente os pixeis mais afastados do pixel inicial, até que o critério seja satisfeito ou a região seja pequena e rejeitada.

A distância entre o pixel inicial e o pixel mais afastado na região representa o tamanho da região. Cada iteração deste método remove os pixeis mais afastados para reduzir o tamanho região para 75% do seu valor.

3.6.1.9 Cálculo do número de falsos alarmes

A validação do retângulo é dada pelo numero de pixeis alinhados, isto é, os pixeis que respeitam a orientação do retângulo, até uma tolerância de $p\pi$. A tolerância p é inicializado com o valor τ/π. O número total de pixeis no retângulo é denotado por n e o número de pixeis alinhados é denotado por k.

O número de falsos alarmes (NFA) associado ao retângulo r é

$$NFA(r) = (NM)^{5/2}\gamma * \sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{n-j} \tag{3.23}$$

onde, N e M é o número de colunas e linhas da imagem (depois do reescalonamento), γ a quantidade de valores de p testados. Os retângulos com NFA(r)$\leq \varepsilon$ são validados como deteções. O valor de ε é igual a 1.

O binomial é usado com a seguinte relação à função *Gamma*:

$$\binom{n}{k} = \frac{\Gamma(n+1)}{\Gamma(k+1) * \Gamma(n-k+1)} \tag{3.24}$$

Otimização do retângulo

Antes de rejeitar a região de *line-support* por não ser significativo (NFA $< \varepsilon$), o LSD otimiza os parâmetros de configuração do retângulo de forma a ter um válido.

A rotina de otimização do retângulo do LSD consiste nos seguintes passos:

1. Variar o valor (p) para precisões mais apertadas;
2. Reduzir a largura do retângulo;
3. Reduzir um lado do retângulo;
4. Reduzir o outro lado do retângulo;
5. Variar o valor (p) para precisões ainda mais apertadas;

Se um retângulo significativo for encontrado (NFA $< \varepsilon$) a rotina de melhoria será parada depois do passo que o encontrou.

O passo 1 testa os seguintes valores de precisão: $p/2$, $p/4$, $p/8$, $p/16$ e $p/32$, onde p é o valor inicial de precisão.

O passo 2 reduz a largura do retângulo por 0.5 pixeis, até 5 vezes.

O passo 3 reduz um lado do retângulo por 0.5 pixeis, até 5 vezes, Isto implica reduzir o tamanho do retângulo por 0.5 pixeis e mover o centro do retângulo 0.25 pixeis.

O passo 4 reproduz o passo 3 para o outro lado do retângulo.

O passo 5 testa os seguintes valores de precisão $\hat{p}/2$, $\hat{p}/4$, $\hat{p}/8$, $\hat{p}/16$ e $\hat{p}/32$, onde $\hat{p}$ é a precisão do início deste passo.

A figura 3.8 mostra um exemplo do resultado final do algoritmo do LSD.

3.6.2 EDLines

No seguimento dos algoritmos selecionados no estado de arte, efetuou-se uma análise mais cuidada do algoritmo EDLines, criado por Cuneyt Akinlar e Cihan Topal [32]. O algoritmo EDLines pode ser dividido em 3 passos: extração de edges usando o método Edge Drawing, extração de segmentos de retas usando o método *least squares line fitting* e por último validação das retas.

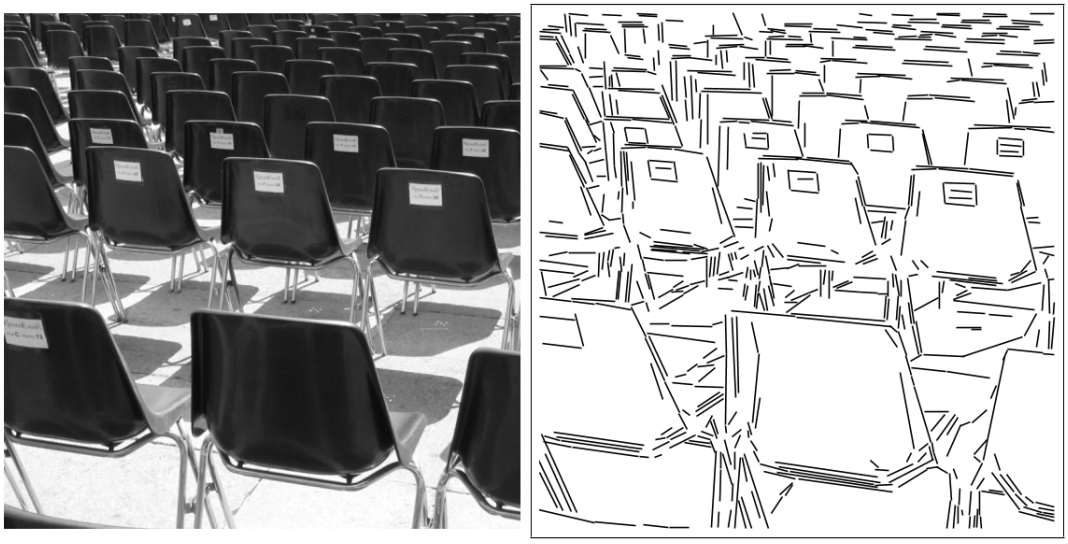


Figura 3.8: Exemplo de implementação do algoritmo LSD. [31]

3.6.2.1 Edge Drawing

O algoritmo de Edge Drawing requer como entrada imagens em tons de cinza e está dividido em três etapas:

- Cálculo da magnitude do gradiente e da direção dos edges;
- Extração de *anchors*;
- Conexão entre *anchors* por um caminho inteligente.

O resultado é um mapa de edges que consiste em segmentos com largura de 1 pixel, contíguos e bem localizados.

Magnitude e Direção

A imagem em escala cinza é filtrada por um kernel Gaussiano 5x5 e com σ de 1 (Secção 3.4) para eliminar ruído e suavizar a imagem. De seguida, é calculado os gradientes verticais e horizontais, Gx e Gy, respetivamente, usando o método de Sobel, recorrendo à equação 3.6.

Simultaneamente com o mapa da magnitude do gradiente, o mapa da direção é também calculado comparando o gradiente vertical e horizontal em cada pixel. Se o

gradiente horizontal for maior,isto é, $|Gx| \geq |Gy|$, assume-se que um edge vertical passa através do pixel.

Para eliminar localizações da imagem onde acontecem mudanças suaves na intensidade é utilizado um *threshold* no mapa do gradiente e elimina-se os chamados de pixeis fracos.

Extração de *anchors*

As *anchors* correspondem aos pixeis onde os edges são mais significativos. O algoritmo 3 apresenta o algoritmo de teste para o pixel (x,y), onde este é considerado uma *anchor*, quando é um pico local do mapa de gradiente. Isto é, apenas é feita uma comparação do valor de gradiente do pixel com os pixeis vizinhos. Para um edge horizontal, o vizinho superior e inferior são comparados. Se o valor de gradiente do pixel for maior que os dos vizinhos acima de um valor de *threshold*, o pixel é marcado como *anchor*.

Algorithm 3 : Algorithm to test for an anchor at (x,y)

Symbols used in the algorithm:
(x,y): Pixel being processed
G: Gradient map
D: Direction map

IsAnchor(x,y,G,D,AnchorThresh)
$G[x, y] \leftarrow$ Not Anchor
if $D[x, y] = HORIZONTAL$ **then**
  if $G[x, y] - G[x, y - 1] \geq AnchorThresh$ **and**
  $G[x, y] - G[x, y + 1] \geq AnchorThresh$ **then**
    $G[x, y] \leftarrow$ Anchor
  end if
else
  if $G[x, y] - G[x - 1, y] \geq AnchorThresh$ **and**
  $G[x, y] - G[x + 1, y] \geq AnchorThresh$ **then**
    $G[x, y] \leftarrow$ Anchor
  end if
end if

Conexão entre *anchors*

A conexão entre *anchors* consecutivas é feita através da ida de uma *anchor* para a outra seguindo os picos do mapa do gradiente: Começando por uma *anchor*, verifica-se a direção do edge. Se a direção do edge for horizontal, começa-se a conectar seguindo

para os edges à esquerda e para os edges à direita,figura 3.9(b). No caso de a direção do edge ser vertical, o processo é idêntico mas usa-se os pixeis vizinhos que estão acima e abaixo da *anchor*, figura 3.9(c). Durante um movimento, apenas o vizinho com maior gradiente dos três é selecionado para conetar à *anchor*. Este processo de conexão para devido a duas condições:

1. Os vizinhos do pixel que se está a analisar não é considerado um edge, pois o gradiente com *threshold* é 0;

2. Encontra-se um edge que previamente já foi conectado a uma *anchor*.

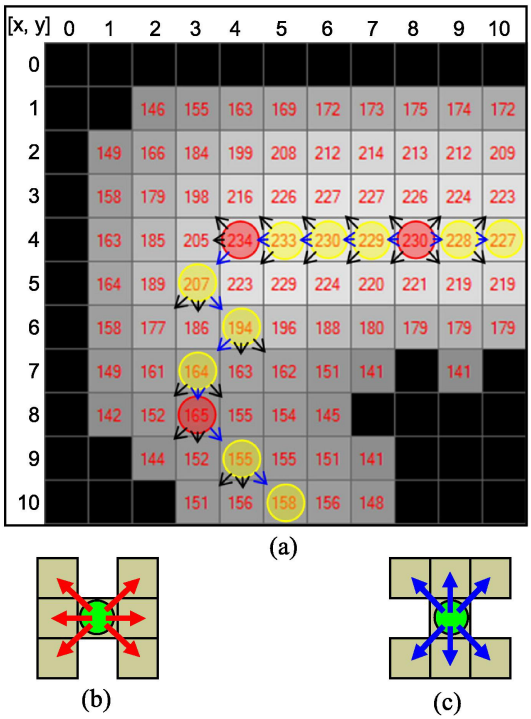


Figura 3.9: (a) Ilustração do processo de conexão entre *anchors*, (b) Processo horizontal, (c) Processo vertical. [36]

O algoritmo 4 descreve o processo de conexão começando na *anchor* (x,y). Assume-se que neste pixel existe um edge horizontal, começando o processo de conexão para a esquerda (efetuar o processo de conexão para as outras direções é bastante semelhante, e não é elaborado). A cada pixel selecionado olha-se para os 3 pixeis vizinhos e seleciona-se o pixel com o valor de gradiente maior, até que ocorra uma das condições de paragem.

A figura 3.9(a) demonstra detalhadamente o processo de conexão de *anchors*. Os números dentro dos pixeis representa a magnitude do gradiente e os pixeis a pretos representam os pixeis com gradiente 0 devido ao *threshold*. As *anchors* estão marcadas com os círculos vermelhos e os pixeis selecionados pelo processo estão marcadas com os círculos amarelos. O pixel representado a vermelho na posição (8,4) da figura 3.9(a) representa a *anchor* inicial.

Algorithm 4 : Algorithm to proceed left of an anchor at (x,y)

Symbols used in the algorithm:
(x,y): Pixel being processed
G: Gradient map
D: Direction map
E: Edge map

GoLeft(x,y,G,D,E)
while $G[x,y] > 0$ **and** $E[x,y] \neq EDGE$ **and** $D[x,y] = HORIZONTAL$ **do**
  $E[x,y] = EDGE$; // Mark this pixel as an edge
  //Look at 3 neighbors to the left & pick the one with the max. gradient value
  if $G[x-1,y-1] > G[x-1,y]$ **and** $G[x-1,y-1] > G[x-1,y+1]$ **then**
    $x = x-1; y = y-1$; // Up-Left
  else if $G[x-1,y+1] > G[x-1,y]$ **and** $G[x-1,y+1] > G[x-1,y-1]$ **then**
    $x = x-1; y = y+1$; // Down-Left
  else
    $x = x-1$; // Straight-Left
  end if
end while

A figura 3.10 apresenta o resultado da implementação do algoritmo ED. A figura 3.10 (b) detalha o mapa da magnitude, a figura 3.10 (c) apresenta um exemplo de um conjunto de *anchors* e o mapa de edges final, apresentado na figura 3.10 (d), é obtido usando a ligação entre *anchors*.

3.6.2.2 Extração de Segmentos de retas

O objetivo deste passo é de dividir uma cadeia contígua de segmentos de edges em um ou mais segmentos de retas. A ideia básica é de percorrer os os pixeis sequencialmente, e efetuar a estimação das retas com base nos edges pelo método *least squares line fitting* enquanto o erro for inferior ao valor de *threshold* definido.

A figura 3.11 ilustra o procedimento. Selecionando-se o primeiro pixel do segmento da cadeia (p. ex. canto superior esquerdo), utiliza-se um certo número de pixeis (p. ex.

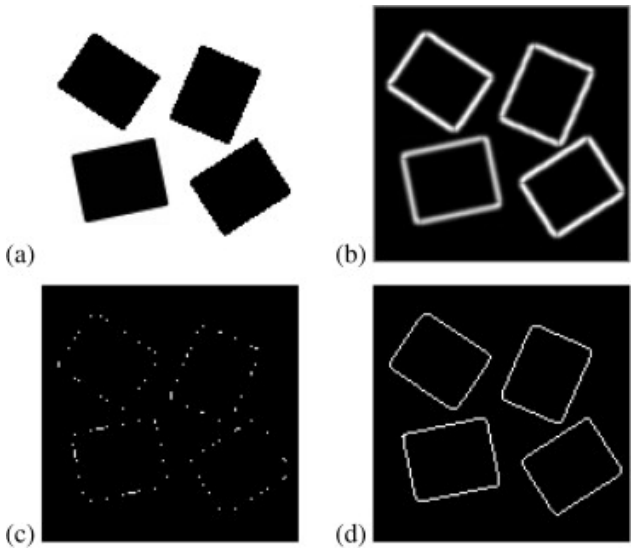


Figura 3.10: (a) Uma imagem em tons de cinza contendo 4 retângulos, (b) Mapa do gradiente, (c) *Anchors*, (d) Mapa final de edges. [32]

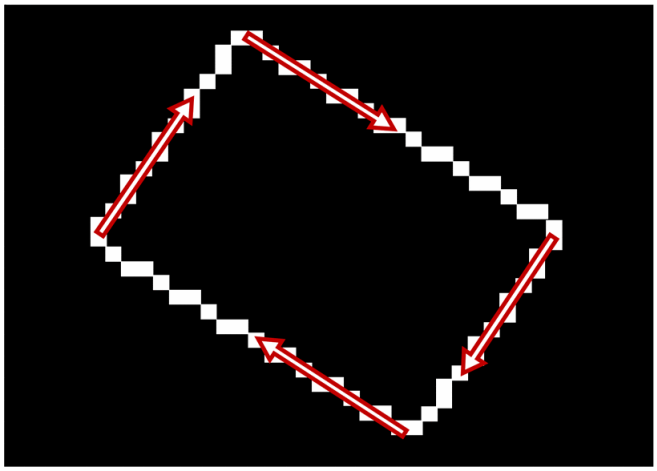


Figura 3.11: Ilustração da extração de segmentos retas a partir de uma cadeia contígua de pixeis. [32]

10 pixeis) e faz-se a estimação da reta para estes pixeis pelo método *least squares line fitting*. Esta primeira reta estimada determina a direção corrente do segmento da reta e é ilustrado pela seta com contorno vermelho na figura. Os restantes pixeis pertencente ao segmento são adicionado à reta estimada enquanto o erro for inferior ao valor de *threshold*. Quando o erro for superior, conclui-se a estimação da reta e começa-se uma

Figura 3.12: Exemplo de um resultado do algoritmo EDLines.

nova reta.

3.6.2.3 Validação da reta

Similar a Desolneux [34] e ao LSD descrito na sub-secção 3.6.1.9, o método de validação da reta é baseado no cálculo do NFA. Para um segmento A de tamanho n com pelo menos k pontos que tenham a sua direção alinhada com a direção de A, numa imagem de NxN pixeis,define-se NFA de A como:

$$NFA(n,k) = N^4 * \sum_{i=k}^{n} p^i (1-p)^{n-i} \tag{3.25}$$

Tal como no LSD, um segmento é chamado de significativo se o seu NFA(n,k) $\leq \varepsilon$ e o valor de ε usado é de 1.

A figura 3.12 apresenta o resultado final da aplicação do algoritmo do EDLines.

3.6.3 CannyLines

O método de CannyLines foi desenvolvido por Xiaohu Lu em conjunto com Jian Yao†, Kai Li e Li Li [37] e efetua a deteção de segmentos de retas em 4 etapas:

- Operador de Canny livre de parâmetros;
- Cluster de edges;

- Agregação dos segmentos de retas;
- Validação das retas.

3.6.3.1 Operador Canny

O operador de Canny consiste em quatro etapas, onde as primeiras duas etapas do algoritmo seguem a mesma abordagem apresentada no EDLines em que consiste no uso de um kernel gaussiano para reduzir o ruído da imagem e o uso do operador de Sobel para calcular a magnitude e orientação do gradiente. A terceira etapa, chamada de *non-maximum suppression*, determina se o pixel é um melhor candidato como edge que os seus vizinhos. A quarta etapa, consiste em aplicar uma histerese de *threshold* que encontra os segmentos de edges.

non-maximum suppression

O objetivo deste passo é de identificar qual o pixel que apresenta uma magnitude superior face (pixel forte) ao seus pixeis vizinhos. Para tal, isto é feito preservando todos os máximos locais na imagem do gradiente. O algoritmo 5 detalha o o procedimento. Para cada pixel da imagem do gradiente é efetuado os seguintes passos:

1. Estima-se a direção do gradiente em relação ao 45^{o} mais próximo, correspondendo ao uso dos 8 vizinhos mais próximos.
2. Comparar o valor da magnitude do edge com os valores dos pixeis na direção positiva e negativa do gradiente. Por exemplo, se a direção do gradiente for 90^{o}, compara se com os pixeis a 90 e -90 graus.
3. Se a magnitude do edge for superior, preserva-se o valor. Se não, remove-se o valor.

A figura 3.13 é um exemplo simples da etapa de *non-maximum suppression*. Quase todos os pixeis têm direção de 90^{o}. Eles são então comparados com os pixeis superior e inferior. O resultado é dado pelos pixeis marcados com a borda branca.

Histerese de *threshold*

O algoritmo de Canny usa um duplo *threshold*. Os pixeis com magnitude superior ao *threshold* superior são marcados como fortes. Os pixeis com magnitude inferior ao *threshold* inferior são removidos e os pixeis entre os dois *threshold* são marcados como fracos.

Algorithm 5 : Algorithm to proceed left of an anchor at (x,y)

Symbols used in the algorithm:
(x,y): Pixel being processed
G: Gradient map
D: Direction map

for each (x,y) $\in$ G **do**
  $Dir \leftarrow$ Get_dir_estimate(D(x,y))
  if $Dir == 0$ **then**
    $G_1 = G[x-1, y]$
    $G_2 = G[x+1, y]$
  else if $Dir == 45$ **then**
    $G_1 = G[x-1, y-1]$
    $G_2 = G[x+1, y+1]$
  else if $Dir == 90$ **then**
    $G_1 = G[x, y-1]$
    $G_2 = G[x, y+1]$
  else if $Dir == 135$ **then**
    $G_1 = G[x+1, y-1]$
    $G_2 = G[x-1, y+1]$
  end if
  if $G[x, y] \leq G_1$ **or** $G[x, y] \leq G_2$ **then**
    $G[x, y] = 0$
  end if
end for

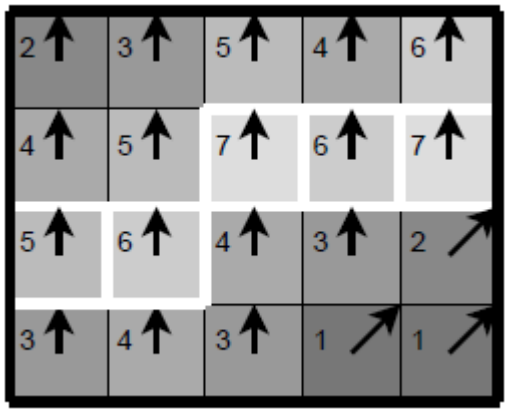


Figura 3.13: Ilustração da supressão não-máxima. A magnitude do edge é indicada pela cor e pelos números, enquanto que a direção é representada pelas setas. Os edges resultantes estão marcados com bordas brancas.

Os pixeis maracados como fortes são imediatamente incluídos na lista final de edges. Os pixeis fracos são incluídos se, e apenas se, estiverem conectados a pixeis fortes.

Operador de Canny livre de parâmetros

Os autores do CannyLines consideram que os dois *thresholds* do Canny não devem ser definidos como constantes para todas as imagens, devem ser ajustados adaptativamente de acordo com as imagens de entrada, usando para isso os princípios de Helmholtz, que diz que, uma estrutura geométrica observada é perceptivamente significativa, se o número de ocorrências for bastante pequeno numa situação aleatória. Neste método, é utilizado de forma a obter a magnitude do gradiente mínima significativa e a magnitude do gradiente máxima sem significado, que resultam no valor dos *thresholds* inferior e superior do operador de Canny.

Considerando um histograma H da magnitude do gradiente de todos os pixeis com um passo de 1 e a distribuição de probabilidade $P(i)$ em que i está associado ao índice de H, com $i = 1, 2, ..., N_h$, onde N_h significa o número de classes de H.

O número total de partes conectadas N_p é dada por

$$N_p = \sum_{i=1}^{N_h} (H(i) * (H(i) - 1)) \tag{3.26}$$

A probabilidade mínima g_{min} e máxima g_{max} é inicializada por

$$g_{min} = \frac{1}{\exp\left(\frac{\log(N_p)}{\sqrt{R*C}}\right)} \tag{3.27}$$

$$g_{max} = \frac{1}{\exp\left(\frac{\log(N_p)}{mfl}\right)} \tag{3.28}$$

com o comprimento significativo mfl,

$$mfl = \frac{2 * \log(R * C)}{\log(8)} + 0.5 \tag{3.29}$$

e R e C a serem a altura e largura da imagem.

O algoritmo 6 descreve em detalhe o cálculo dos *thresholds* para aplicação no operador de Canny.

Algorithm 6 : Parameter-Free Canny Edge Detection

Input: The input image I
Output: The edge map E
Apply Gaussian kernel on I.
Calculate gradient magnitudes via Sobel operator.
Build the histogram H.
Calculate the probability distribution $P(i)$.
Compute N_p.
Compute g_{min} and g_{max}.
$p = 0$
for $i = N_h : -1 : 1$ **do**
  $p = p + P(i)$
  if $p \geq g_{min}$ **then**
    $g_{min} = i$
    break
  end if
end for
for $i = N_h : -1 : 1$ **do**
  $p = p + P(i)$
  if $p \geq g_{max}$ **then**
    $g_{max} = i$
    break
  end if
end for
Revise $g_{max} = \sqrt{\lambda * g_{max}}$
$E \leftarrow CANNY(..., g_{max}, g_{min}, ...)$

3.6.3.2 Cluster de edges

O processo de cluster de segmentos de edges está dividido em três etapas. Numa primeira fase é efetuada a ordenação de todos os edges consoante os seus valores de magnitude do gradiente. De seguida, é efetuada a junção de edges. Começando pelo edge com maior magnitude do gradiente, o processo de junção procura nos 8 vizinhos mais próximos e seleciona os que apresentam uma orientação semelhante com uma tolerância de $\pi/8$. Após o processo terminar para este segmento de edge, o mesmo processo de junção é efetuado para os restantes edges.

A última etapa do processo consiste separação dos segmentos. Todos os segmentos com um comprimento maior que um *threshold* θ_s é separado, em dois segmentos, no ponto com um desvio máximo maior que um pixel à reta formada pelas duas extremidades do segmento de edges.

3.6.3.3 Agregação dos segmentos de retas

Nesta etapa do algoritmo, tem-se um conjunto de segmentos de edges constituído por pontos quase colineares, que são enviados para uma função de *least squares line fitting* para obter um número largo de segmentos de retas iniciais.

Dado um segmento de reta L_i, seleciona-se o primeiro e último pixel pertencente à reta, como pontos iniciais para extensão ao longo da sua direção. Usando o pixel inicial como exemplo, seleciona-se os 3 pixeis vizinhos na direção da reta. Se estes pixeis forem um edge, aceita-se como "hipótese de extensão", de outra forma, considera-se este caso como um "falha".

Se algum edge encontrado já pertencer a uma reta e a direção das duas retas forem semelhantes, estas duas retas são fundidas, se o erro médio quadrado, baseado no método *Least Squares line fit*, não for maior que 1 pixel.

Continuando a estender o segmento de reta L_i ao longo da sua direção, para-se o processo quando houver 2 "falhas"em 5 operações de extensão.

Quando o número de "hipóteses de extensão"é maior que o comprimento significativo mfl calculado na equação 3.29, é efetuada novamente a estimação da reta à reta estendida L_i.

Quando a direção de extensão é terminada, começa-se a outra direção de extensão, usando o mesmo método.

Figura 3.14: Exemplo de um resultado do algoritmo CannyLines.

3.6.3.4 Validação da reta

A validação da reta do CannyLines acrescenta à equação de validação de retas, usadas no LSD e no EDLines, informação referente à magnitude do gradiente de cada segmento. Assim, a definição de número de falsos positivos é dado por

$$NFA(n,k) = N^4 * \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i} * N_p * H(u)^n \tag{3.30}$$

onde u denota o gradiente de magnitude mínimo dos pixeis pertencentes ao segmento e N_p o número total de pixeis conectados em todos os segmentos.

A figura 3.14 mostra um exemplo do resultado final do algoritmo do CannyLines.

Capítulo 4

Comparativo de deteção de linhas elétricas

Neste capítulo será detalhado o desempenho de 3 métodos: EDLines, LSD e Canny-Lines, que baseado no estado da arte foram identificados como técnicas promissoras para deteção de linhas elétricas. A análise foi efetuada com 82 imagens reais de linhas elétricas adquiridas de diferentes ângulos a partir de um UAV ao longo de várias missões de inspeção de ativos elétricos.

4.1 Byrd UAV

O UAV Byrd, apresentado na figura 4.1 é um hexacopter adaptado ao processo de inspeção de ativos elétricos e é equipado a nível de sensores de inspeção por uma câmara termográfica FLIR A65, um LiDAR velodyne VLP-16 e a câmara visível PointGrey Grasshopper3 modelo GS3-U3-91S6C-C como apresentado na figura 4.2, equipada com uma lente Spacecom de 25mm f0.95, que apresenta características apresentadas na tabela 4.1.

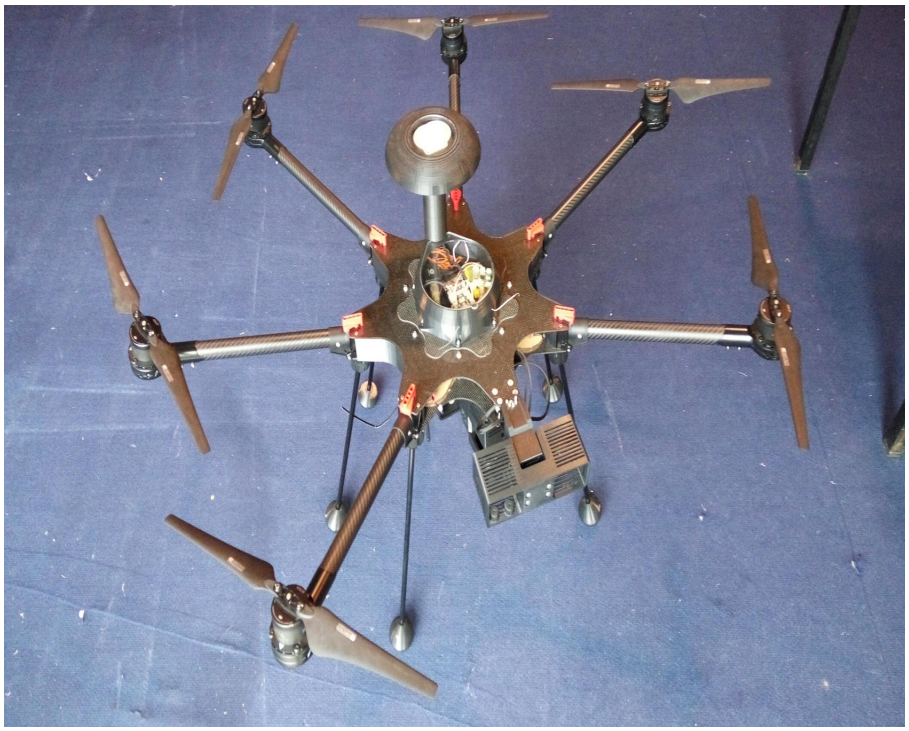


Figura 4.1: Fotografia do UAV Byrd.

Figura 4.2: Câmara de Espectro Visível Pointgrey Grasshoper3.

Tabela 4.1: Características da Pointgrey Grasshoper3.

Resolução	3376 x 2704 pixeis (9.1 MP)
Taxa de Aquisição	9Hz
Chroma	Cor
Conectividade	USB3.0
Resolução do pixel	14 bits
Tensão de Alimentação	5V a 24V
Potência Consumida Máxima	4.5W
Peso	90g (sem lente)
Dimensões	44 x 29 x 58mm
Gama de Exposição	0.04ms a 32s

4.2 Imagens de linhas elétricas

Foram selecionadas um conjunto de 82 imagens utilizando os seguintes critérios:

- Diferentes ângulos relativamente às linhas elétricas;
- Diferentes distâncias relativamente às linhas elétricas;
- Diferentes condições climatéricas(sol, nevoeiro e nuvens);
- Diferentes fundos(relva, árvores, estradas e cercas).

A figura 1.2 e 4.3 apresentam algumas das imagens selecionadas. Neste conjunto de imagens é possível verificar os vários ângulos de captura da imagem, vista superior, vista inferior e vista lateral, e a variedade de elementos nos fundos, incluído estruturas, como é o caso de uma torre eólica.

Esta diversidade de imagens permite efetuar um teste à robustez dos métodos de deteção de linhas, validando a sua possível aplicação nos mais variados cenários de linhas elétricas.

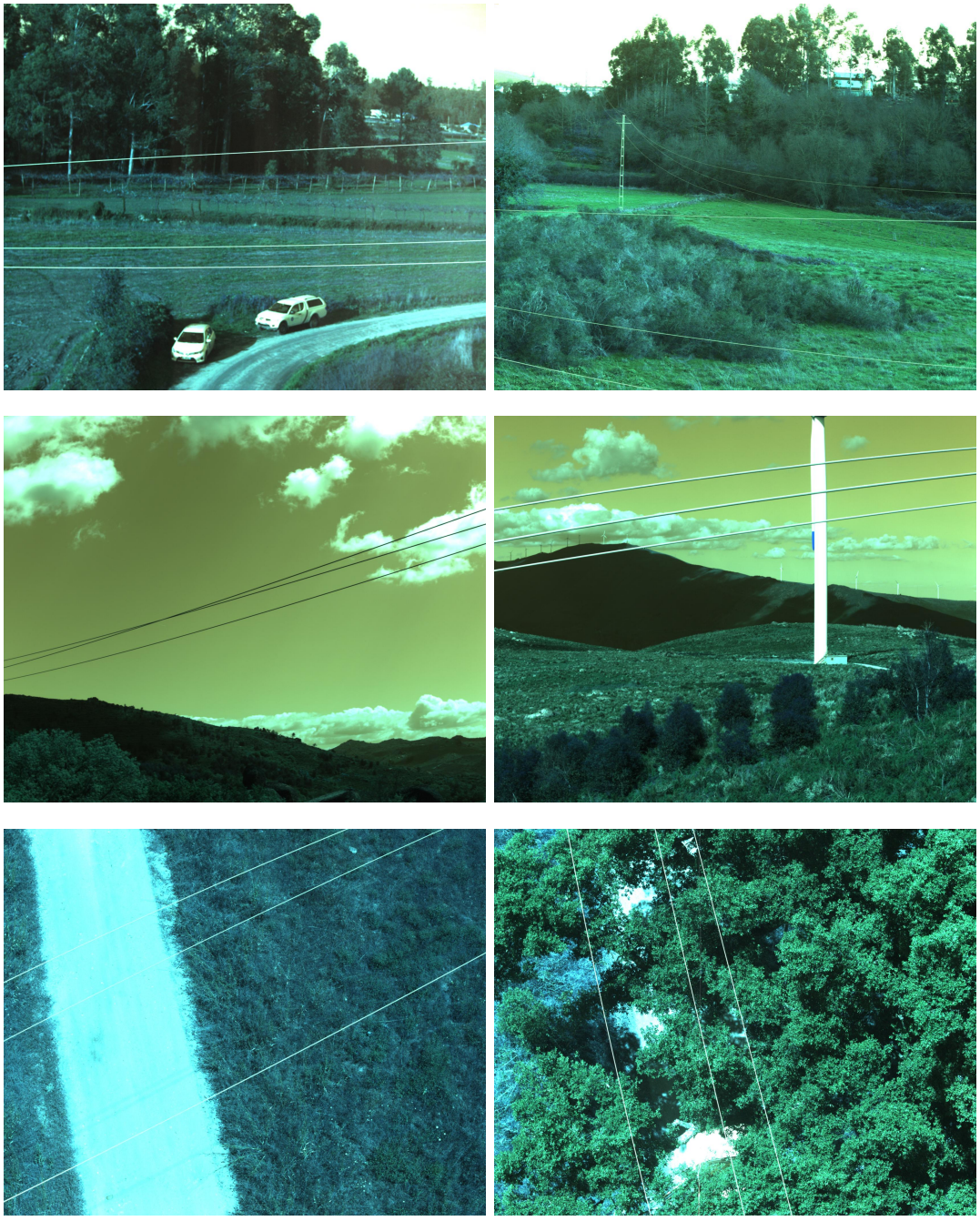


Figura 4.3: Exemplo de imagens de linhas elétricas

4.3 Critérios de avaliação

O desempenho dos métodos foi classificado em três categorias, duas de análise subjetiva e uma de análise objetiva:

- Classificação do sucesso de deteção de linhas elétricas;
- Classificação do sucesso de não geração de falsos positivos;
- Tempo de processamento de cada imagem.

A tabela 4.2 apresenta os critérios de avaliação da classificação que será atribuída a cada imagem gerada pelos métodos. Uma classificação de 1 representa uma fraca extração de linhas elétricas e um número elevado de falsos positivos.

Todos os resultados apresentados nesta dissertação foram obtidos usando a ferramenta OpenCV num computador portátil com um processador Intel Pentium T4300 dual-core com 2.1GHz, 4GB de RAM e o sistema Linux Ubuntu 14.04 LTS 64 bits.

Tabela 4.2: Critérios de avaliação

	Fraco	Não satisfaz	Satisfaz	Satisfaz Bem	Excelente
Linhas Elétricas	1	2	3	4	5
Falsos Positivos	1	2	3	4	5

4.4 Resultados da comparação

As figuras 4.4 e 4.5 são dois exemplos da comparação efetuada e mostram os resultados alcançados(classificação em Linhas Elétricas (L) e falsos positivos(FP)) por cada algoritmo nas referidas imagens originais.

A tabela 4.3 apresenta a média (μ) e desvio-padrão (σ) da classificação alcançada pelos três algoritmos. O resultado da deteção de linhas elétricas foi boa, dado a alta pontuação obtida, significando que as linhas elétricas foram detetadas com sucesso. Pelo contrário, o resultado da não deteção de falsos positivos foi má, dado a baixa pontuação alcançada, significando que foram detetado vários falsos positivos.

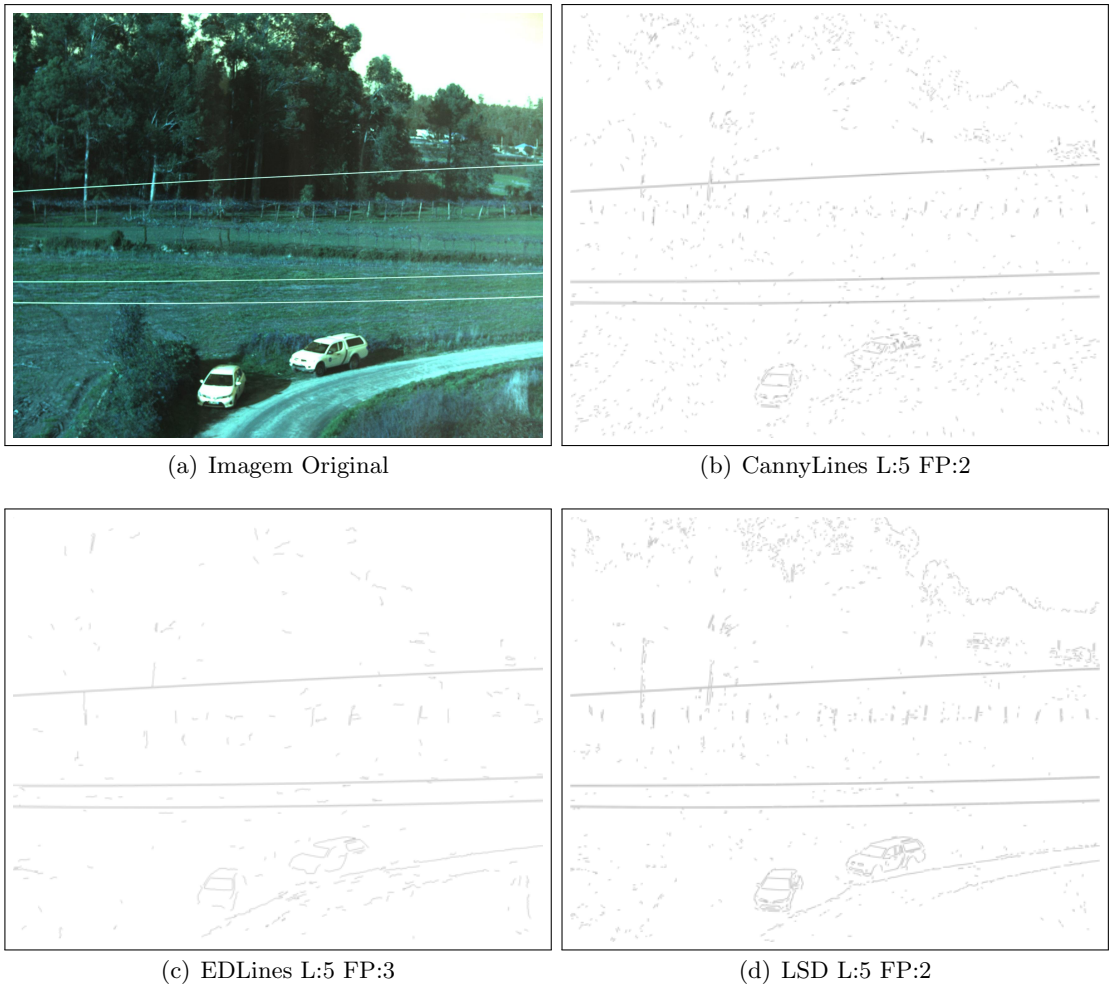
EDLines alcançou a pontuação mais baixa na deteção de linhas elétricas mas apresenta melhor desempenho no que refere à classificação da não criação de falsos positivos. Os métodos CannyLines e LSD apresentam resultados semelhantes, com o CannyLines com melhor desempenho na deteção de linhas e pior em falsos positivos quando comparado com o LSD.

Os tempos de processamento dos algoritmos para cada imagem é apresentado na figura 4.6 e a tabela 4.4 apresenta o valor médio e desvio padrão, onde se pode concluir que o EDLines apresenta maior consistência no tempo de processamento e é aproximadamente 6 vezes e 12 vezes mais rápido que o CannyLines e LSD respetivamente.

Tendo em consideração um dos objetivo principais do projeto, desenvolvimento de um método capaz de ser executado em tempo-real, o algoritmo de EDLines tem as melhores características para ser usado como base para o trabalho que será desenvolvido, pois apresenta a melhor combinação de deteção de linhas elétricas, da não geração de falsos positivos e caracteristicas de tempo-real.

Tabela 4.3: Classificação da deteção de linhas elétricas e falsos positivos

		CannyLines	EDLines	LSD
μ	L	4.56	4.33	4.55
	FP	2.50	3.75	2.67
σ	L	0.89	1.10	1.01
	FP	0.65	0.64	0.89



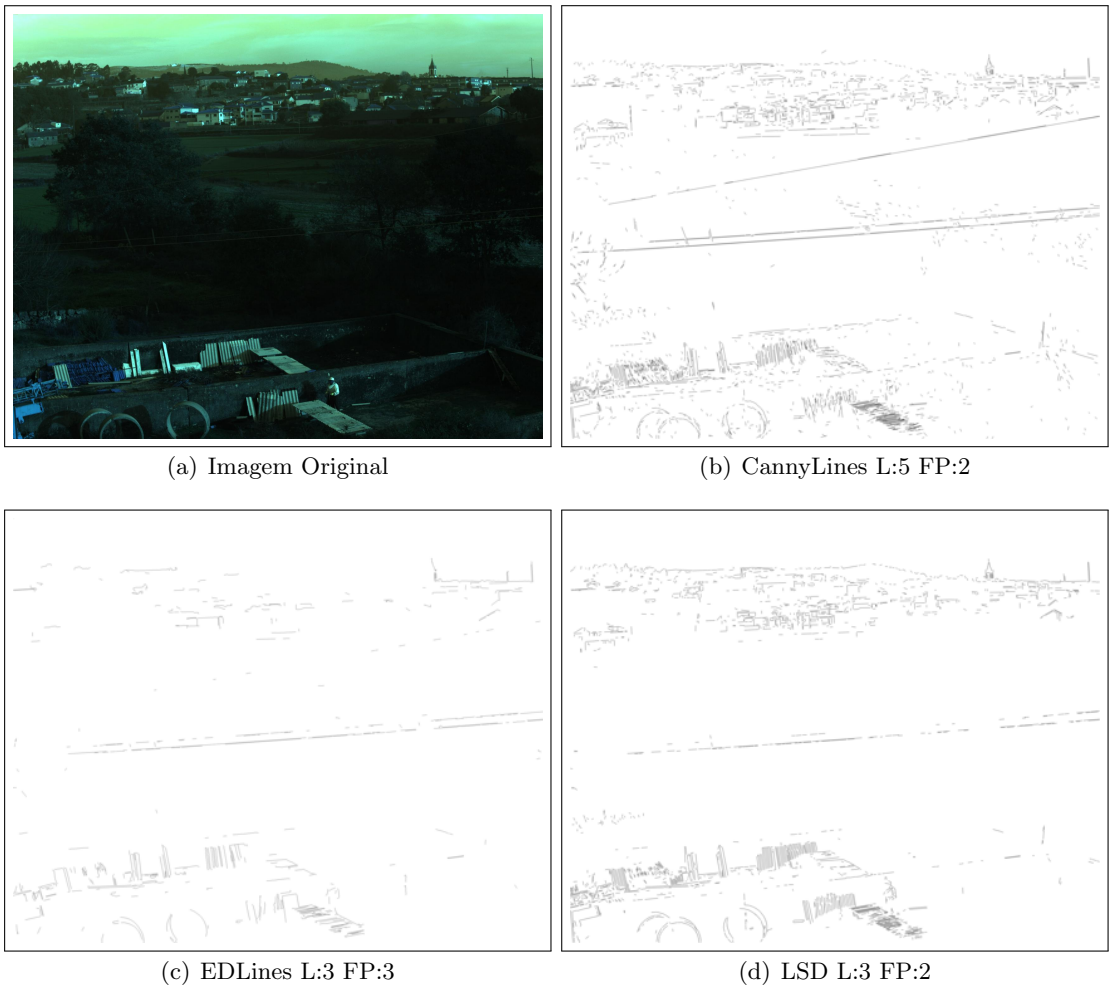
(a) Imagem Original (b) CannyLines L:5 FP:2

(c) EDLines L:5 FP:3 (d) LSD L:5 FP:2

Figura 4.4: Exemplo de classificação atribuída aos três métodos.

Tabela 4.4: Média e desvio padrão do tempo de processamento (ms).

	CannyLines	EDLines	LSD
μ	3984.03	697.39	8201.93
σ	1442.86	270.79	3423.57



(a) Imagem Original

(b) CannyLines L:5 FP:2

(c) EDLines L:3 FP:3

(d) LSD L:3 FP:2

Figura 4.5: Exemplo de classificação atribuída aos três métodos.

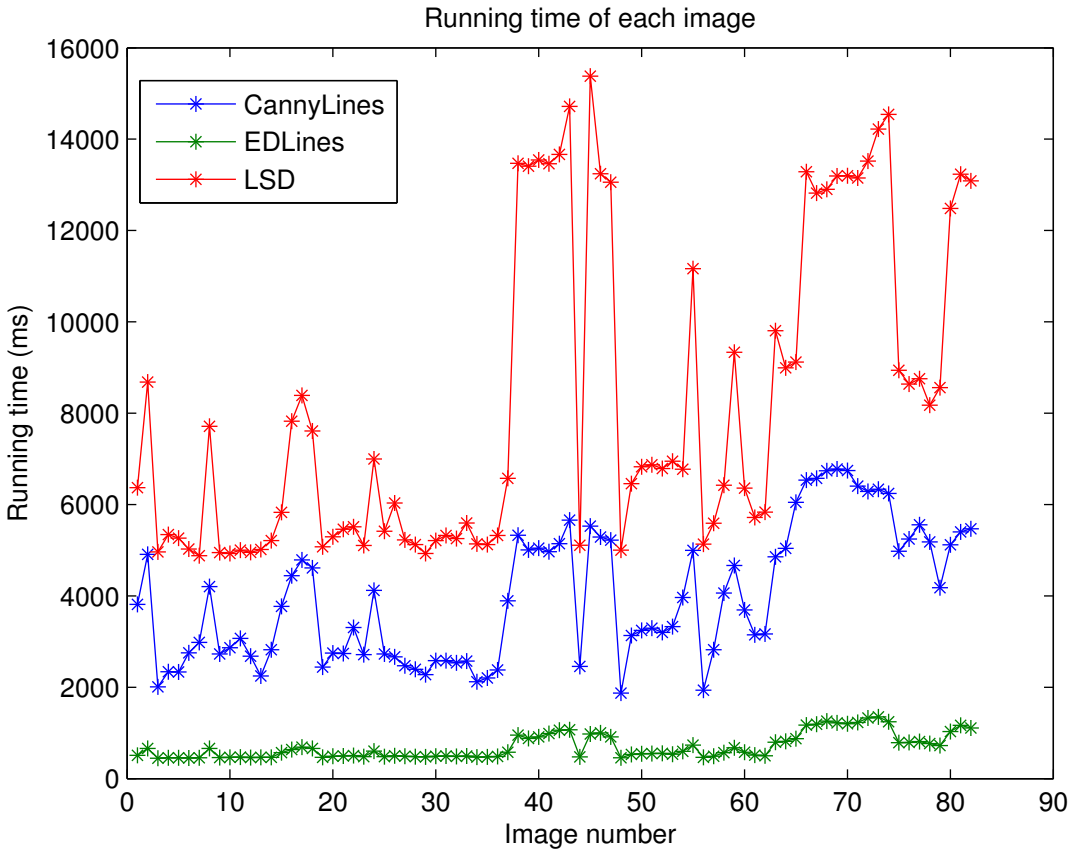


Figura 4.6: Comparação dos tempos de processamento de cada imagem (ms).

Capítulo 5

Algoritmo

Neste capítulo vai ser descrito o algoritmo desenvolvido que pretende responder ao problema da extração de linhas elétricas a partir de imagens. Para tal, inicialmente é explicado a ideia geral do que o algoritmo pretende efetuar e com isso demonstrando a integração do EDLines no trabalho desenvolvido. Depois, é explicado detalhadamente todos os passos referentes ao Algoritmo desenvolvido, chamado de PLineD.

5.1 Conceito

O EDLines, e como já explicado previamente, pode ser dividido em dois módulos principais, o algoritmo de deteção de segmentos de edges (ED) e a extração de segmentos de linhas (least squares line fit).

Atendendo às características das linhas elétricas, estas podem ter curvaturas ao longo do seu comprimento, e à complexidade do método de extração de segmentos de linhas usado pelo EDLines, decidiu-se utilizar o algoritmo ED como detetor de edges, que alimenta uma *pipeline* que refine e corta segmentos, carateriza e agrupa segmentos, de forma a extrair apenas os segmentos de edges que definem as linhas elétricas.

Os segmentos de edges pertencentes às linhas elétricas são, normalmente, segmentos longos e que, devido ao paralelismo entre linhas elétricas, são acompanhados por outros segmentos paralelos igualmente longos. Na figura 5.1 é possível visualizar, para dois exemplos, um gráfico que mostra o ângulo de cada segmento detetado face à distância, em pixeis, da origem. O tamanho do X representa o número de pixeis pertencente ao segmento. No exemplo superior ve-se que existem 3 conjuntos de segmentos longos em torno de 0-360º, onde, um dos quais, encontra-se mais afastado dos outros dois, tal como a imagem original mostra. No exemplo inferior os 3 conjuntos de segmentos de linhas,

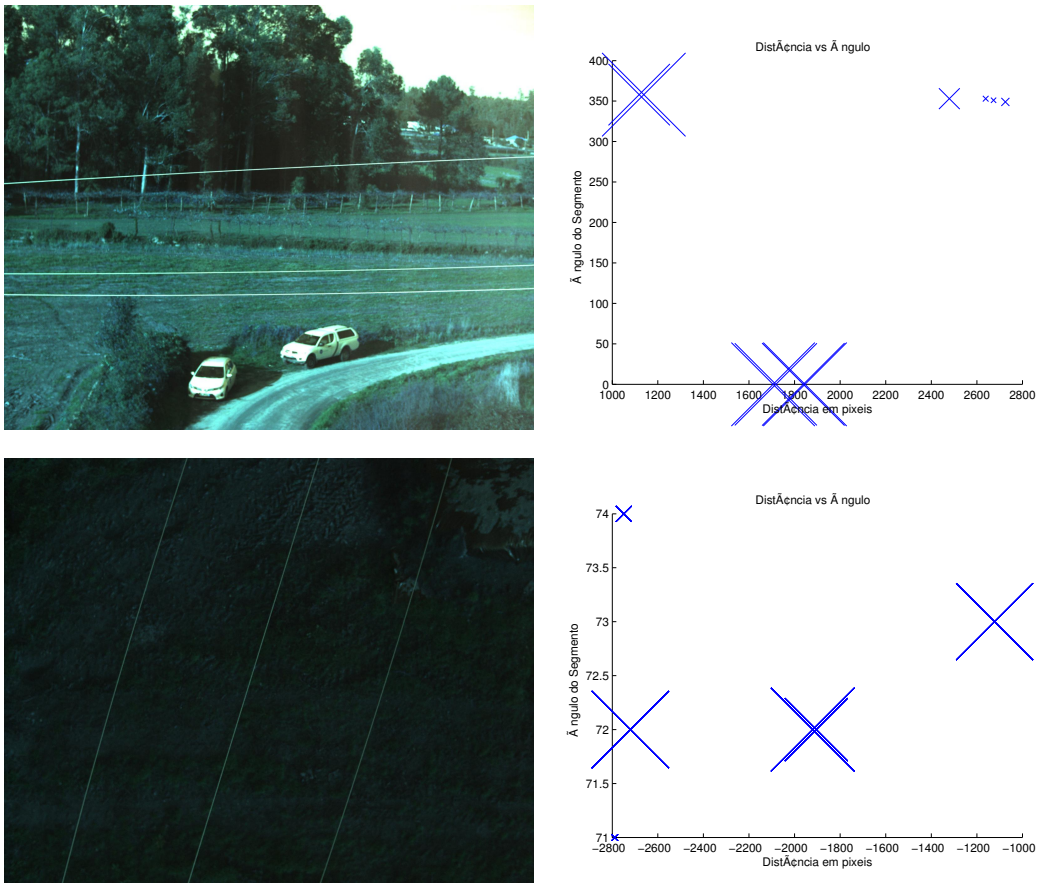


Figura 5.1: Relação entre o ângulo e a distância das linhas elétricas numa imagem.

com 72^{o} e 73^{o} que representam as 3 linhas elétricas, estão bem explícitos.

Assim, demonstra-se que utilizando as propriedades das linhas elétricas, segmentos longos e paralelos, é possível encontrar as linhas elétricas numa imagem.

5.2 Arquitetura de software

O algoritmo desenvolvido para extrair linhas elétricas a partir de imagens em tons de cinza encontra-se dividido em 5 fases, tal como demonstrado na figura 5.2. À imagem em tons de cinza é aplicado o algoritmo ED que permite obter os segmentos de edge. De seguida seguem-se os passos Corte de Segmentos e Covariância dos Segmentos que pretendem dividir segmentos que façam ângulos acentuados e eliminar segmentos pequenos ou com curvas acentuadas. Por fim, os segmentos são agrupados de forma a que todos os segmentos pertencentes a uma linha elétrica fiquem juntos, para que, no último passo,

apenas os grupos de segmentos paralelos sejam considerados.

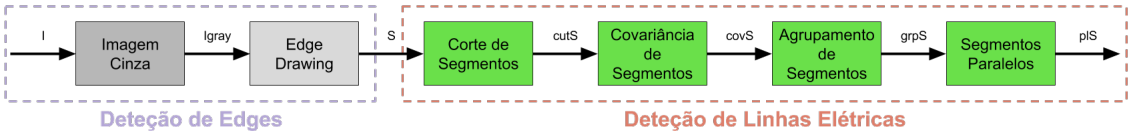


Figura 5.2: Pipeline de deteção de linhas elétricas do PLineD.

5.2.1 Corte de Segmentos

ED retorna os segmentos de edges detetados na imagem, resultando em segmentos que descrevem as linhas elétricas mais segmentos com ruído de fundo. Em alguns segmentos pode ser possível que o mesmo segmento pertença às linhas elétricas e ao fundo.

A figura 5.3 exemplifica o objetivo deste passo do algoritmo, onde, do lado esquerdo acontece de estar um segmento de uma linha elétrica horizontal conectado com um segmento de ruído de fundo vertical (segmento vermelho), que precisa de ser cortado. Este passo corta-o em dois segmentos, o segmento horizontal (azul-claro) e o segmento vertical (vermelho).

Desta forma, consegue-se aproveitar um maior número de segmentos de edges, que caso contrário, poderiam ser eliminados no passo seguinte.

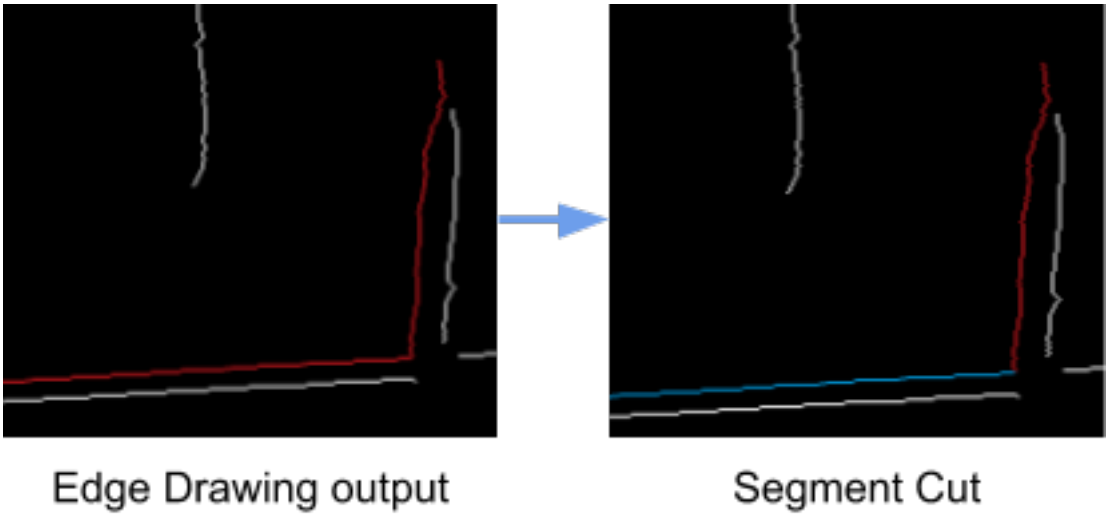


Figura 5.3: Ilustração do passo de corte dos segmentos

O algoritmo desenvolvido, detalhado no algoritmo 7, produz a lista de segmentos

cortados $cutS$, onde verifica $step$ em $step$ $Pixel$ se o ângulo entre os vetores que vai de $Pixel$ para $Pixel + step$ e $Pixel - step$ é maior que θ.

Algorithm 7 $cutS \leftarrow \text{segCUT}(S, step, \theta)$

$setP \leftarrow step\mathbb{Z}^+$
$i \leftarrow 0$
for each $s \in S$ **do**
  for each $Pixel \in s$ **do**
    Add $Pixel$ to $cutS[i]$
    if $Pixel \in s \cap setP$ **then**
      $\overrightarrow{V_1} = \overrightarrow{P_{nPixel} - P_{nPixel-step}}$
      $\overrightarrow{V_2} = \overrightarrow{P_{nPixel} - P_{nPixel+step}}$
      $A = \arccos(\overrightarrow{V_1} \cdot \overrightarrow{V_2} / \left\|\overrightarrow{V_1}\right\| \left\|\overrightarrow{V_2}\right\|)$
      if $A > \theta$ **then**
        $i++$
      end if
    end if
  end for
end for
return $cutS$

A figura 5.4, exemplo demonstrativo de um segmento com ângulo de 90º, pode ser usada para perceber o que acontece no algoritmo. Considerando o pixel vermelho (1,4) o pixel a analisar ($Pixel$) e um $step$ de 4, $Pixel - step$ é o pixel azul (1,0) e $Pixel + step$ é o pixel verde (5,4), criando os vetores $\overrightarrow{V_1}$ = (0,4) $\overrightarrow{V_2}$ = (-4,0). O ângulo resultante entre os dois vetores é de 90º.

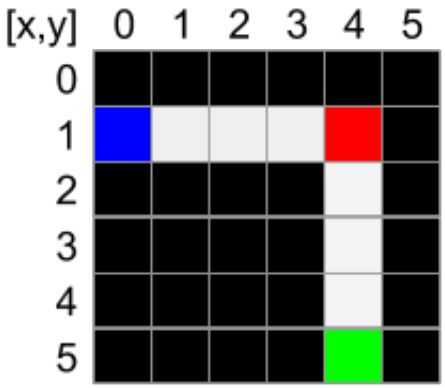


Figura 5.4: Exemplo para demonstração do corte de um segmento com ângulo de 90º.

5.2.2 Covariância dos Segmentos

O passo da covariância dos segmentos é responsável por apagar segmentos pequenos e segmentos com curvaturas acentuadas. Seguindo o algoritmo detalhado no algoritmo 8, para cada segmento calcula-se os valores próprios da matriz de covariância e o seu rácio decide se o segmento é apagado da lista de segmentos.

Algorithm 8 $covS \leftarrow \text{segCOV}(cutS, ratio)$

```
for each s ∈ cutS do
  COV ← Get_covar_matrix(s)
  λ ← eigen(COV)
  if λ1/λ2 < ratio then
    Remove s from cutS
  end if
end for
return cutS
```

A matriz de covariância é calculada pelas coordenadas x, y de cada pixel pertencente ao segmento, dado por:

$$\begin{bmatrix} \sum x^2 - (\sum x)^2 & \sum xy - \sum x \sum y \\ \sum yx - \sum y \sum x & \sum y^2 - (\sum y)^2 \end{bmatrix} \tag{5.1}$$

No exemplo demonstrativo apresentado na figura 5.5 tem se um segmento branco, que pela matriz de covariância obtém-se a elipse vermelha, definida pelo comprimento das linhas azul e verde (valores próprios).

5.2.3 Agrupamento dos Segmentos

Nesta fase do algoritmo global desenvolvido, a lista de segmentos irá conter vários segmentos que pertencem às linhas elétricas e um baixo número, que podem ser zero, de segmentos de ruído de fundo (outros segmentos grandes sem curvaturas acentuadas).

Este passo de agrupamento de segmentos vai formar um grupo de segmentos para cada linha elétrica, como detalhado no algoritmo 9, onde, começando pelo segmento com o maior número de pixeis($LongSeg$), a distância em pixeis entre esse segmento e o pixel central (s_{half}), dos segmentos restantes que têm uma diferença de ângulo inferior ao máximo (maA), é calculado. Se esta distância for inferior ao *threshold* (dS), os segmentos são agrupados. Todos os grupos de segmentos criados que tenham um número total de pixeis inferiores a miP são eliminados.

Algorithm 9 $grpS \leftarrow \text{segGRP}(covS, miL, maA, dS, miP)$

$i \leftarrow 0$
while $covS! = 0$ **do**
  $LongSeg \leftarrow$ Get_longer_seg($covS$)
  if $LongSeg > miL$ **then**
    while $s \notin grpS$ **do**
      if $diffAngle(s, LongSeg) < maA$ **then**
        $\overrightarrow{V_1} = \overrightarrow{LongSeg_{max} - LongSeg_{min}}$
        $\overrightarrow{V_2} = \overrightarrow{s_{half} - LongSeg_{max}}$
        $Dist = \perp\overrightarrow{V_1} / \left\|\overrightarrow{V_1}\right\| \cdot \overrightarrow{V_2}$
        if $Dist < dS$ **then**
          Add s to $grpS[i]$
          Remove s from $covS$
        end if
      end if
    end while
  else
    break
  end if
  if $length(s \in grpS[i]) < miP$ **then**
    Remove $grpS[i]$
  else
    $i++$
  end if
end while
return $grpS$

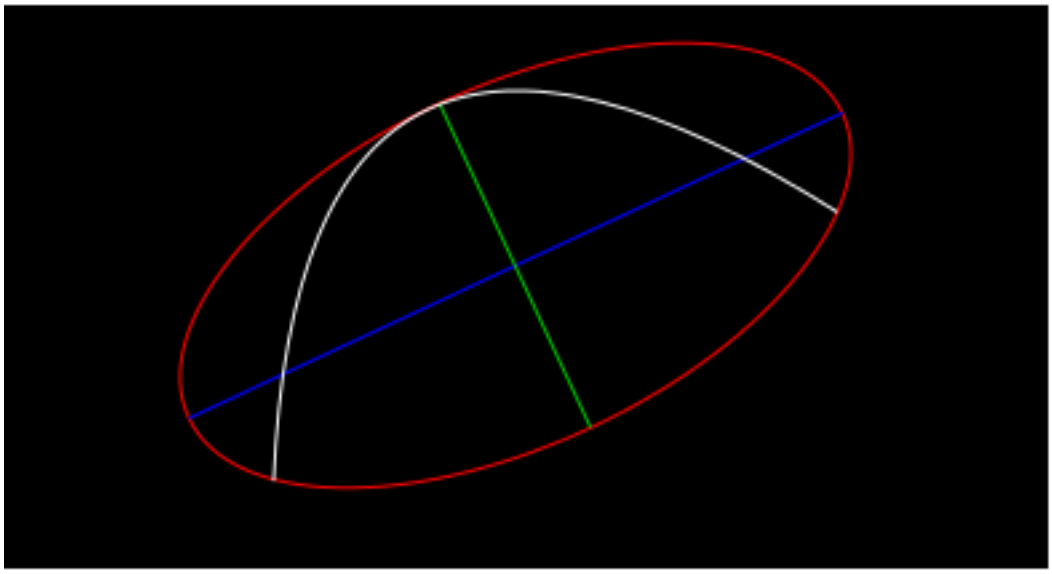


Figura 5.5: Exemplo demonstrativo do uso da covariância para obter os valores próprios.

A figura 5.6 serve de exemplo para a explicação do agrupamento de segmentos. A branco, existem dois segmentos, o segmento com maior comprimento(com P1 como pixel mínimo e P2 como pixel máximo) e o segmento a ser testado (com o pixel central P3). A distância em pixeis entre os dois segmentos é dado no eixo y do sistema de coordenadas vermelho com origem no pixel P2, que é resultante do vetor perpendicular $\overrightarrow{PV_1}$.

5.2.4 Segmentos Paralelos

O passo final do PLineD é descrito no algoritmo 10, onde se procura por grupos de segmentos paralelos, e se o número de grupos paralelos for maior que o *threshold* $minPPL$, esses grupos são considerados como segmentos das linhas elétricas.

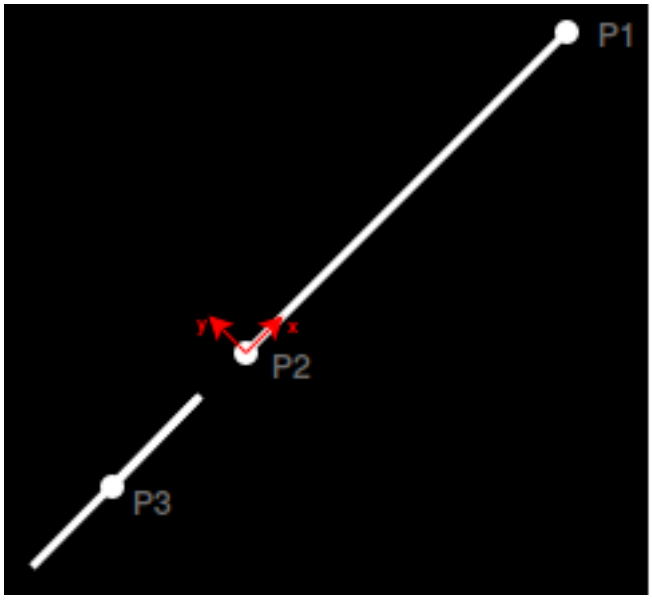


Figura 5.6: Exemplo para demonstração de agrupamento de dois segmentos.

Algorithm 10 $plS \leftarrow$ segPRLL($grpS, tAngle, minPPL$)

for each $gs \in gprS$ **do**
  if $diffAngle(gs, prllS) < tAngle$ **then**
    Add gs to $plS[i]$
  end if
end for
for each $ps \in plS$ **do**
  if $ps < minPPL$ **then**
    Remove ps from plS
  end if
end for
return plS

Capítulo 6

Resultados e Análise

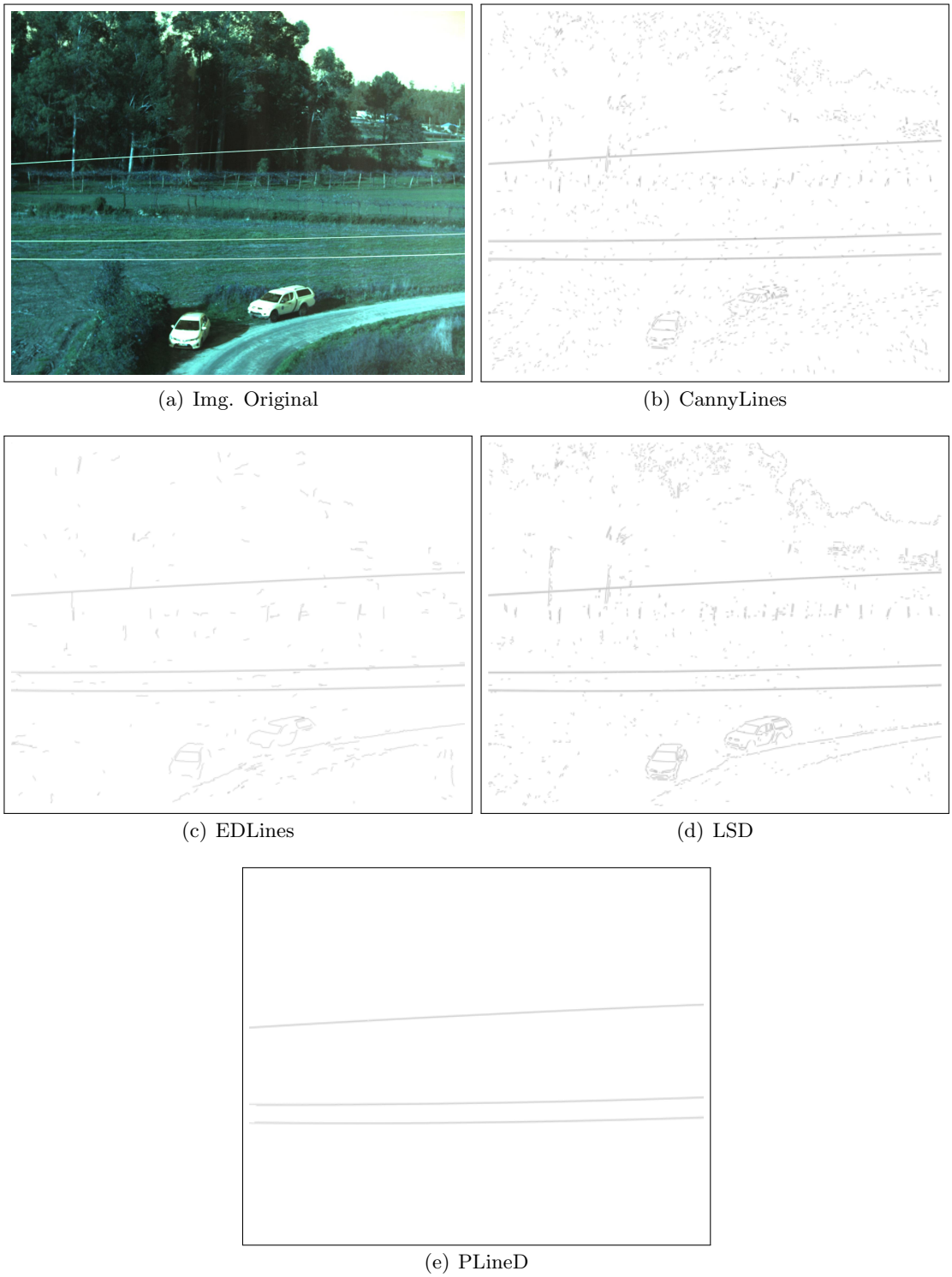
O algoritmo PLineD foi testado no conjunto de 82 imagens apresentados na Secção 4.2.As figuras 6.1, 6.2, 6.3, 6.4 e 6.5 mostra lado-a-lado vários exemplos dos resultados obtidos usando o CannyLines, EDLines, LSD e PLineD.

O PLineD comparando aos outros métodos é capaz de extrair apenas as linhas elétricas, excluindo a figura 6.5, pois, como já visto anteriormente na figura 4.5, os segmentos de edges detetados pelo ED não é satisfatória, acabando assim por não se conseguir extrair as linhas elétricas.

Analisando mais detalhadamente o algoritmo desenvolvido, a figura 6.6 mostra todos os passos efetuados no algoritmo, tal como explicado na Secção 5. Como mostrado na figura 6.6 (a), existem muitos elementos lineares na imagem original: três linhas elétricas, uma torre eólica, árvores e uma montanha. O resultado do ED, figura 6.6 (b), mostra segmentos em quase todos os elementos previamente descritos da imagem original. Os passos do corte e covariância dos segmentos, mostrado na figura 6.6 (c), trata de remover a maioria do ruído de fundo, devido aos seus segmentos de tamanho reduzido, deixando apenas as linhas elétricas e alguns segmentos longos, que por causa da sua orientação e falta de segmentos paralelos, são removidos pelos passos de agrupamento de segmentos e segmentos paralelos, como mostrado pelas figuras 6.6 (d) e 6.6 (e).

A figura 6.7 mostra um exemplo de uma imagem com vista de cima, onde, por baixo das linhas elétricas tem uma estrada de areia, uma cerca metálica e um apoio de tensão. O resultado do algoritmo demonstra que a extração é bem sucedida, onde apenas as linhas elétricas foram extraídas e o ruído de fundo foi removido.

A figura 6.8 mostra mais um resultado do algoritmo, usando uma imagem com um apoio de tensão por baixo de uma interseção de linhas elétricas. Esta imagem é um caso

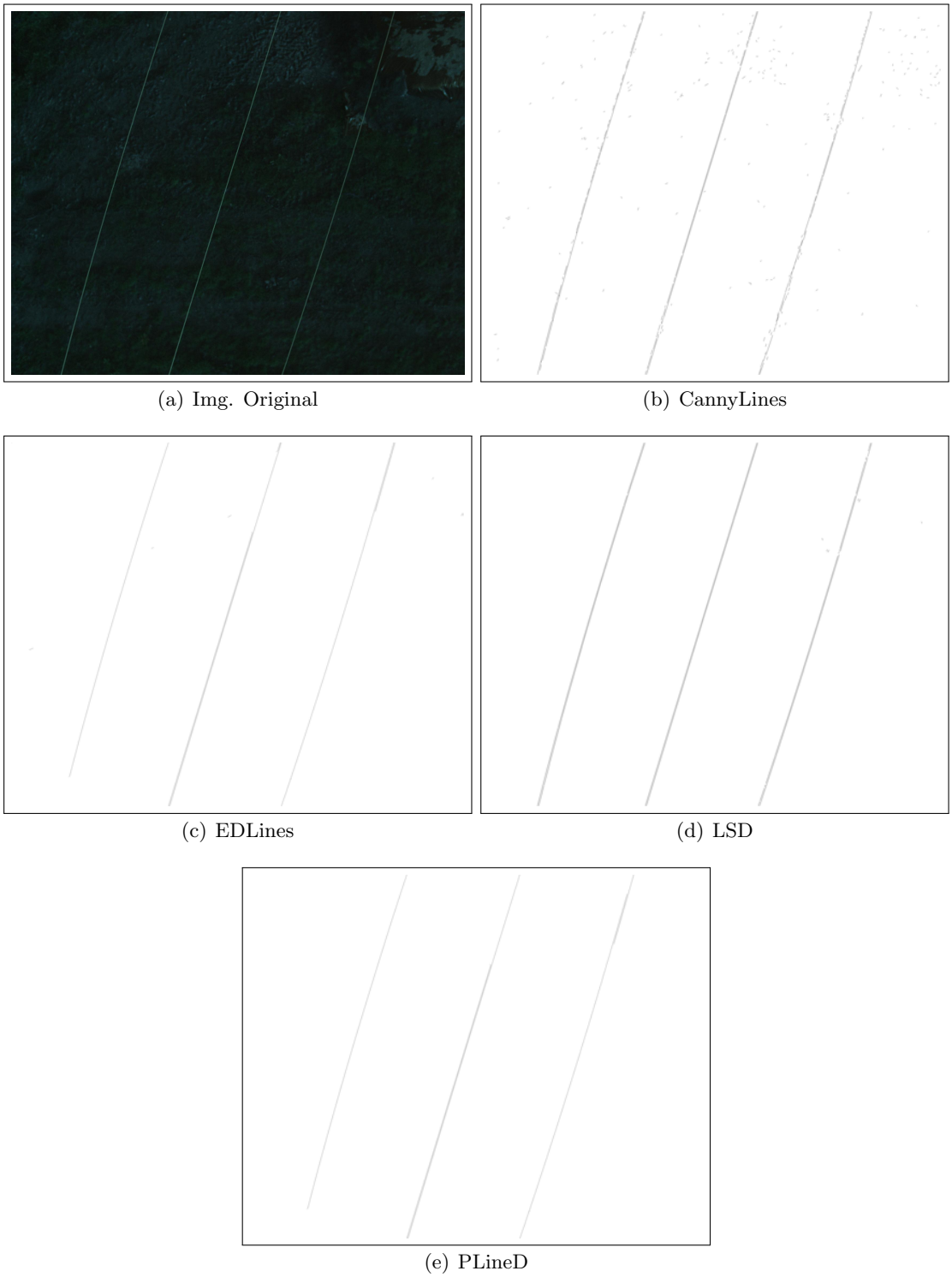


(a) Img. Original (b) CannyLines

(c) EDLines (d) LSD

(e) PLineD

Figura 6.1: Resultados da deteção de linhas elétricas de (b) CannyLines, (c) EDLines, (d) LSD e (e) PLineD.



(a) Img. Original

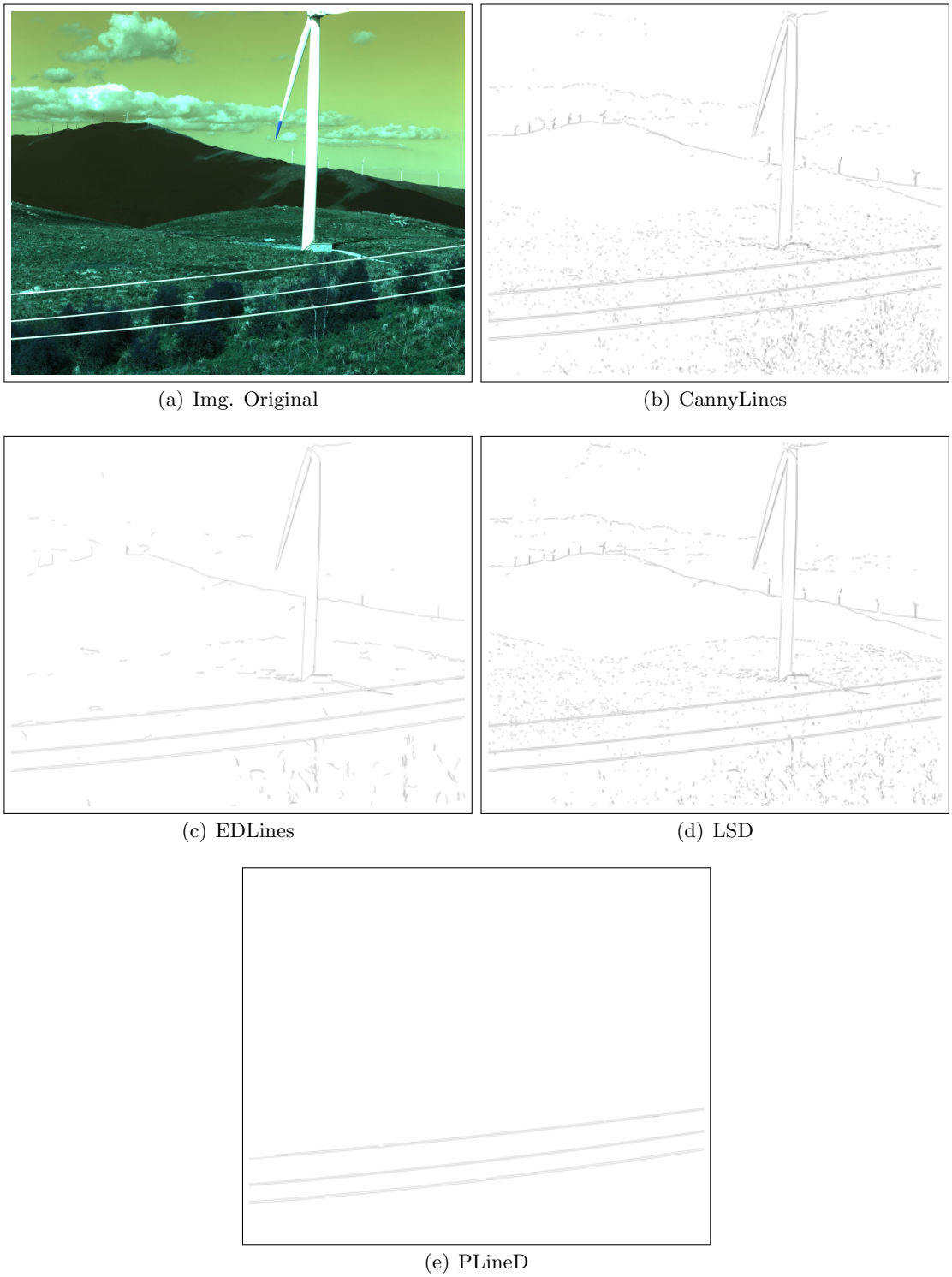
(b) CannyLines

(c) EDLines

(d) LSD

(e) PLineD

Figura 6.2: Resultados da deteção de linhas elétricas de (b) CannyLines, (c) EDLines, (d) LSD e (e) PLineD.



(a) Img. Original

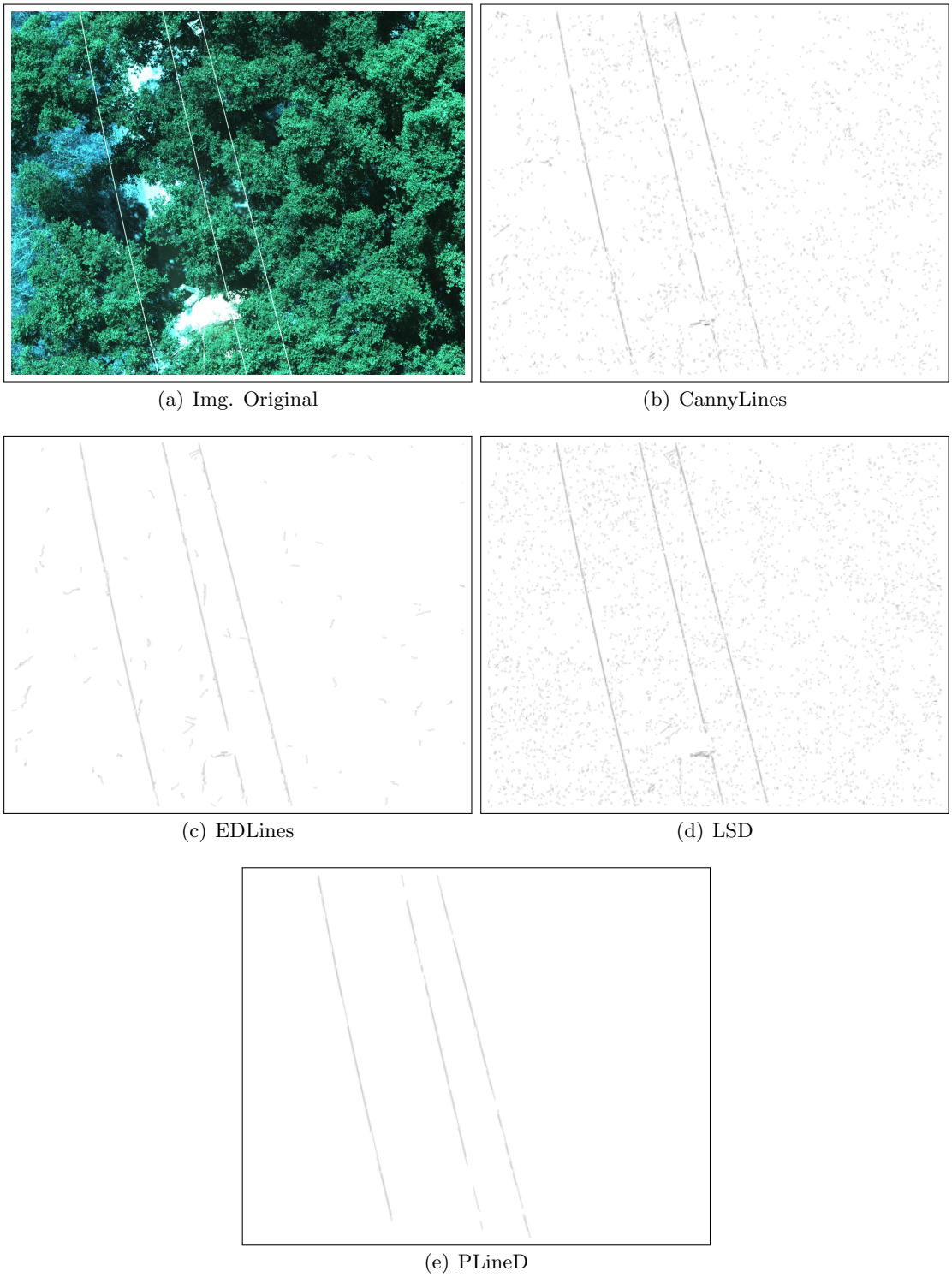
(b) CannyLines

(c) EDLines

(d) LSD

(e) PLineD

Figura 6.3: Resultados da deteção de linhas elétricas de (b) CannyLines, (c) EDLines, (d) LSD e (e) PLineD.

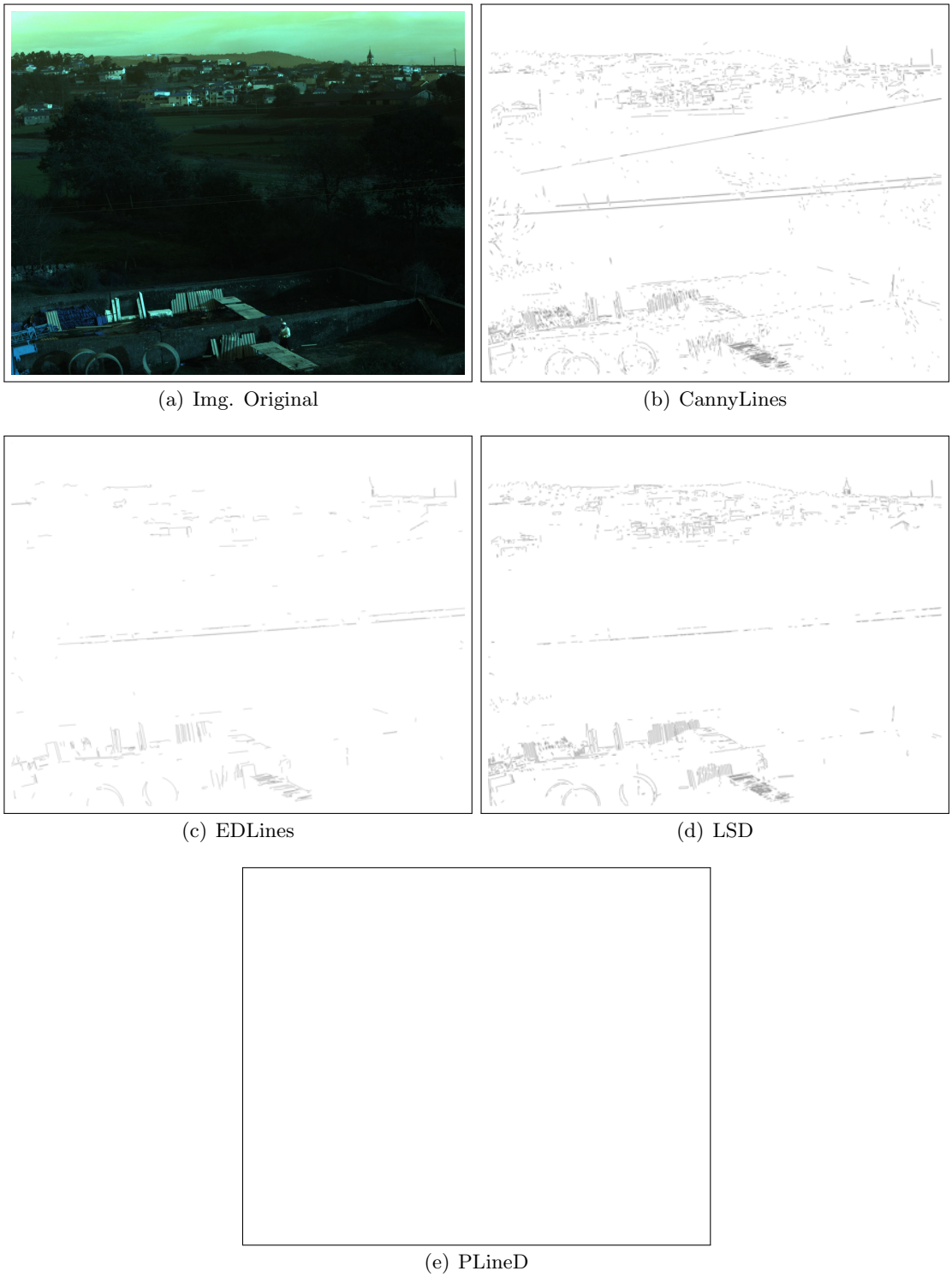


(a) Img. Original (b) CannyLines

(c) EDLines (d) LSD

(e) PLineD

Figura 6.4: Resultados da deteção de linhas elétricas de (b) CannyLines, (c) EDLines, (d) LSD e (e) PLineD.



(a) Img. Original

(b) CannyLines

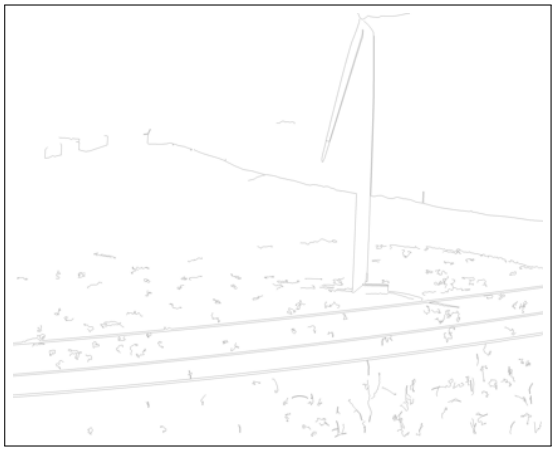
(c) EDLines

(d) LSD

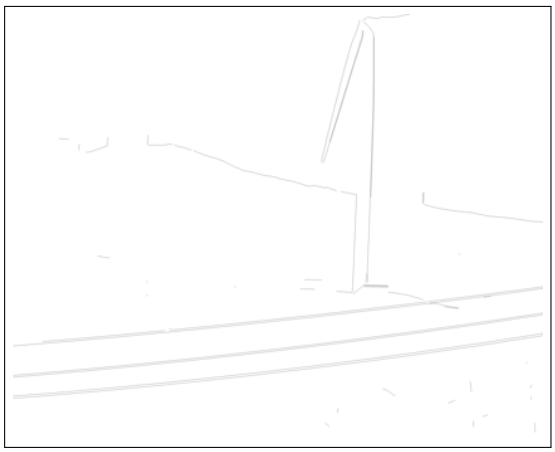
(e) PLineD

Figura 6.5: Resultados da deteção de linhas elétricas de (b) CannyLines, (c) EDLines, (d) LSD e (e) PLineD.

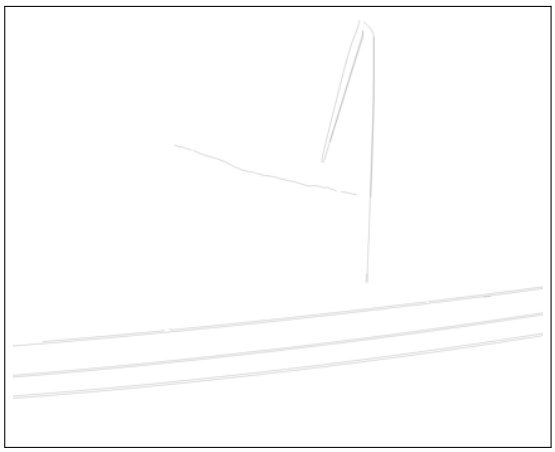
(a) Imagem Original



(b) Edge Drawing



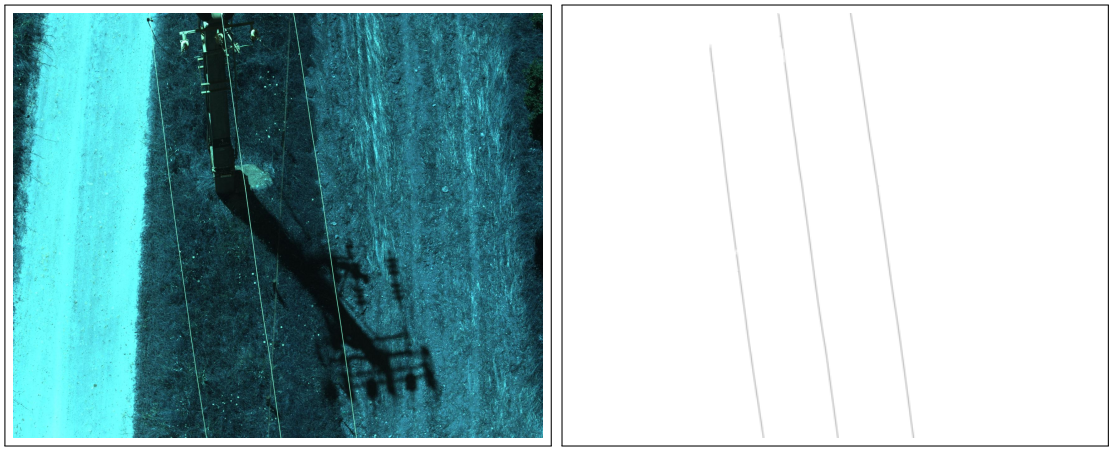
(c) Corte + Covariância



(d) Agrupamento

(e) Paralelos

Figura 6.6: Resultados de cada passo do PLineD.



(a) Imagem Original (b) PLineD

Figura 6.7: Resultado de deteção de linhas elétricas do PLineD.

especial na deteção de linhas elétricas pois além de ter duas direções de linhas elétricas, também tem um apoio de tensão que interrompe as linhas elétricas. No estado da arte analisado na Secção 2, este é um caso que não é suficientemente analisado.

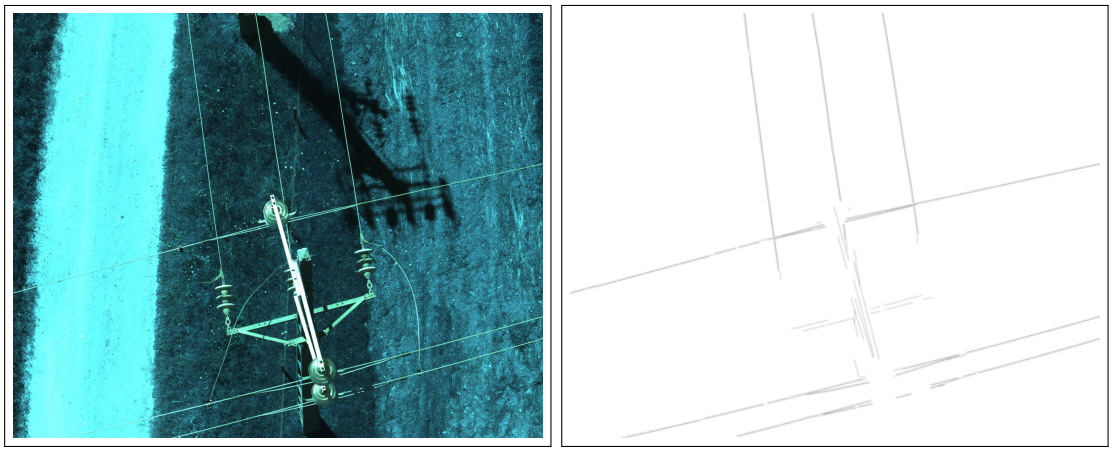
O algoritmo deteta com sucesso as linhas elétricas nas duas direções, mas tem também alguns segmentos detetados no apoio de tensão. Isto acontece devido à proximidade e paralelismo entre estes segmentos e as linhas elétricas.

Continuando nos caso em que o algoritmo não é perfeito, a figura 6.9 mostra um caso em que uma das linhas elétricas não é extraída perfeitamente, devido ao contraste baixo entre a linha elétrica e a estrada de areia do fundo. Mesmo assim, existe um número de alto de pixeis que representam as linhas elétricas.

Outro problema que acontece nesta imagem, é que a cerca metálica que é paralela às linhas elétricas é também extraída. Como a cerca é detetada no passo do ED, o passo dos segmentos paralelos não é capaz de eliminar esses segmentos. Este problema deverá ser abordado no futuro em camadas de perceção de alto nível do UAV, onde esta informação é fundida com outros sensores e informações prévias num mapa do ambiente robusto.

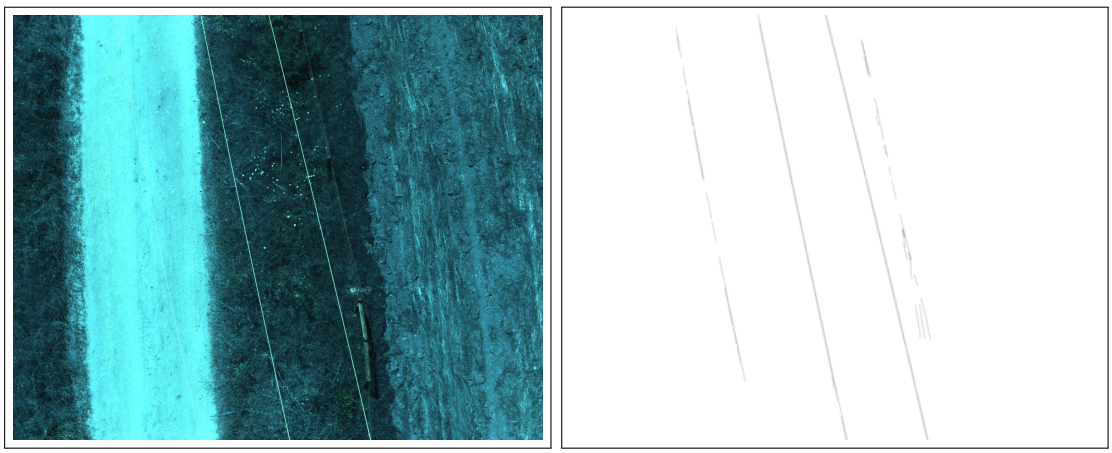
A figura 6.10 mostra um exemplo de uma imagem com vista lateral das linhas elétricas. Apesar das linhas serem extraídas sucessivamente, devido à proximidade entre as linhas elétricas, o passo de agrupamento de segmentos é incapaz de fazer a distinção entre as três linhas elétricas.

O tempo de execução do PLineD para cada uma das 82 imagens é apresentado na figura 6.11, acompanhado da comparação com os resultados do EDLines apresentado na Secção 4. O PLineD é mais rápido no processamento de todas as imagens, sendo quase



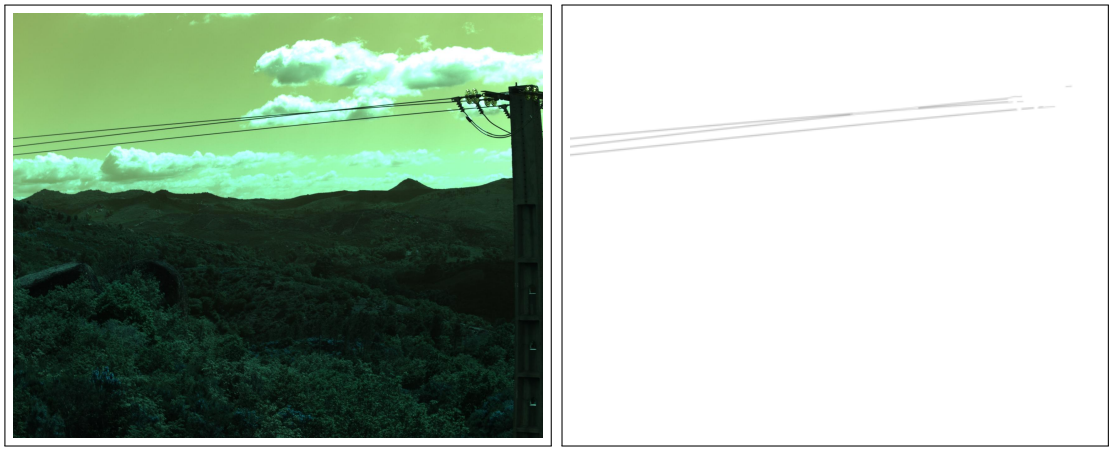
(a) Imagem Original (b) PLineD

Figura 6.8: Resultado de deteção de linhas elétricas do PLineD numa imagem com interseção de linhas num apoio de tensão.



(a) Imagem Original (b) PLineD

Figura 6.9: Resultado de deteção de linhas elétricas do PLineD numa imagem com uma cerca metálica paralela.



(a) Imagem Original (b) PLineD

Figura 6.10: Resultado de deteção de linhas elétricas do PLineD numa imagem de vista lateral.

Tabela 6.1: Média e desvio padrão do tempo de processamento (ms).

	CannyLines	EDLines	LSD	PLineD
μ	3984.03	697.39	8201.93	428.17
σ	1442.86	270.79	3423.57	44.41

duas vezes mais rápido que o EDLines e é também o algoritmo que apresenta maior consistência, tal como demonstrado pelo valor da média e desvio-padrão apresentado na tabela 6.1.

A figura 6.12 mostra a dissecação do tempo de processamento do PLineD. O passo de Edge Drawing, dissecado na tabela 6.2, é claramente o que demora mais tempo, mas com uma implementação otimizada, (processamento no GPU) o tempo de execução pode sofrer uma grande redução.

Além disso, os resultados presentes nesta dissertação foram alcançados por um processador antigo, resultando em tempos de execução inflacionados. A atual unidade de processamento do UAV é capaz de correr o PLineD 2 a 4 vezes mais rápido que estes

Tabela 6.2: Média e desvio padrão do tempo de processamento do EdgeDrawing (ms).

	Gaussiano	Sobel	*Anchors*	Ligação
μ	44.92	220.67	68.20	71.36
σ	0.80	5.50	12.84	27.01

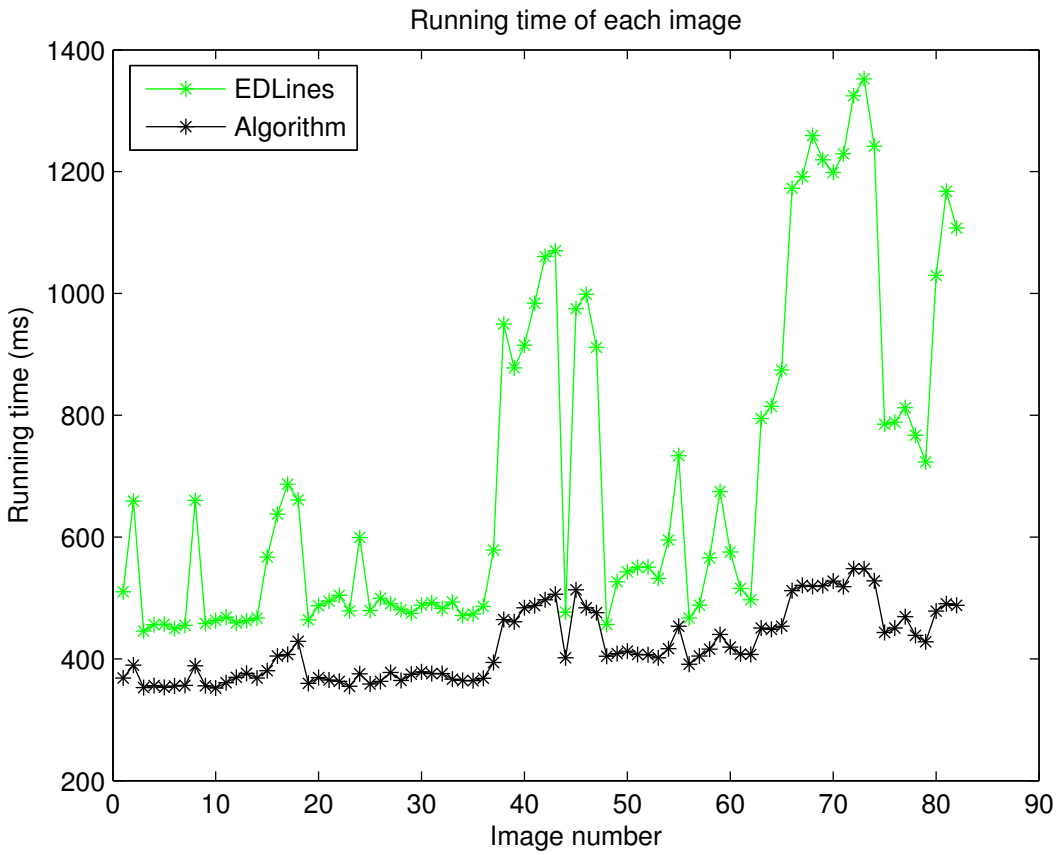


Figura 6.11: Comparação dos tempos de processamento (ms) de cada imagem do EDLines e PLineD.

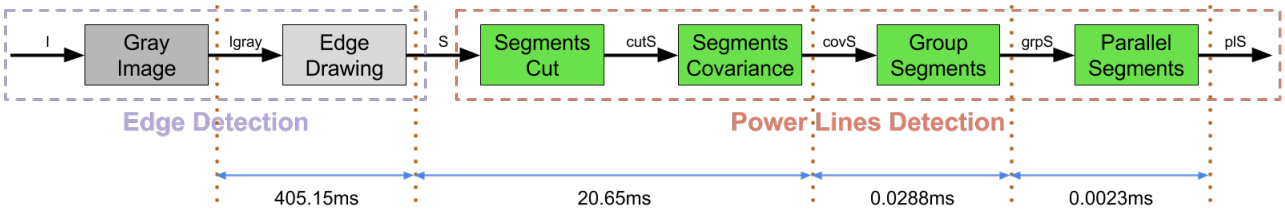


Figura 6.12: Dissecação dos tempos de processamento (ms) do PLineD

resultados apresentados.

Esta página foi intencionalmente deixada em branco.

Capítulo 7

Conclusão e Trabalho Futuro

Esta dissertação abordou o desenvolvimento de um algoritmo de deteção de linhas elétricas para veículos aéreos não tripulados, sendo testado em imagens reais de linhas elétricas em Portugal.

Numa fase inicial, realizou-se um estudo com o objetivo de analisar os métodos e abordagens existentes para extração de linhas elétricas a partir de uma câmara visível. Esta análise permitiu conhecer vários métodos que foram úteis para perceber qual a melhor abordagem ao problema e a definir a arquitetura do algoritmo. Deste estudo sobre os métodos existentes foram selecionados três para participarem em um estudo experimental, com o intuito de determinar qual seria o método que serviria como base para o algoritmo desenvolvido. Com base no resultado final do estudo, decidiu-se optar pelo método Edge Drawing, uma vez que este cumpre os requisitos delineados.

Na abordagem desenvolvida procurou-se tirar proveito da forma e paralelismo das linhas elétricas, de forma a extrair apenas os segmentos pertencentes às linhas elétricas. Utilizando o Edge Drawing para obter segmentos de edges foram aplicados métodos que removem todos os segmentos que são pequenos, curvos e que não têm outros segmentos paralelos.

A validação do método desenvolvido consistiu na análise dos resultados obtidos nas imagens reais de linhas elétricas, onde a sua extração foi alcançada com sucesso na maioria das imagens. Comparativamente aos métodos existentes que foram analisados, demonstrou-se que o algoritmo desenvolvido apresenta um tempo de execução inferior e que é muito mais robusto na deteção de falsos positivos.

O trabalho desenvolvido na dissertação resultou na publicação do artigo "PLineD: Vision-based Power Lines Detection for Unmanned Aerial Vehicles"na conferência 17º International Conference on Autonomous Robot Systems and Competitions (ICARSC

2017).

Como trabalho futuro pretende-se melhorar o passo de agrupamento de linhas, de forma a que funcione em casos mais extremos de proximidade e para linhas elétricas que apresentem uma curvatura mais acentuada, que não possam ser aproximadas por retas.

Uma outra linha de trabalho consiste no desenvolvimento de um passo que permita distinguir os segmentos de linhas elétricas de outros segmentos gerados por objetos semelhantes, p. ex. cercas metálicas.

A integração desta abordagem num UAV permitirá também validar o desempenho do método proposto devido ao movimento ao longo da linha elétrica. A continuidade da linha elétrica também poderá ser usada como melhoria do algoritmo, limitando a área de procura das linhas elétricas com informação das linhas elétricas detetadas na imagem anterior.

Bibliografia

[1] Leena Matikainen, Matti Lehtomäki, Eero Ahokas, Juha Hyyppä, Mika Karjalainen, Anttoni Jaakkola, Antero Kukko, and Tero Heinonen. Remote sensing methods for power line corridor surveys. {*ISPRS*} *Journal of Photogrammetry and Remote Sensing*, 119:10 – 31, 2016.

[2] A. Pagnano, M. Höpf, and R. Teti. A roadmap for automated power line inspection. maintenance and repair. *Procedia* {*CIRP*}, 12:234 – 239, 2013. Eighth {CIRP} Conference on Intelligent Computation in Manufacturing Engineering.

[3] `http://www.inmr.com/examples-insulator-failure/2/#!prettyPhoto`. [acedido a 02-Fevereiro-2017].

[4] `http://www.foresightdiagnostics.com/proven-experience/`. [acedido a 02-Fevereiro-2017].

[5] J. Zhang, L. Liu, B. Wang, X. Chen, Q. Wang, and T. Zheng. High speed automatic power line detection and tracking for a uav-based inspection. In *2012 International Conference on Industrial Control and Electronics Engineering*, pages 266–269, Aug 2012.

[6] H. Sharma, R. Bhujade, V. Adithya, and P. Balamuralidhar. Vision-based detection of power distribution lines in complex remote surroundings. In *2014 Twentieth National Conference on Communications (NCC)*, pages 1–6, Feb 2014.

[7] T. W. Yang, H. Yin, Q. Q. Ruan, J. D. Han, J. T. Qi, Q. Yong, Z. T. Wang, and Z. Q. Sun. Overhead power line detection from uav video images. In *2012 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 74–79, Nov 2012.

[8] G. Yan, C. Li, G. Zhou, W. Zhang, and X. Li. Automatic extraction of power lines from aerial images. *IEEE Geoscience and Remote Sensing Letters*, 4(3):387–391, July 2007.

[9] I. Golightly and D. Jones. Visual control of an unmanned aerial vehicle for power line inspection. In *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, pages 288–295, July 2005.

[10] M. Gerke and P. Seibold. Visual inspection of power lines by u.a.s. In *2014 International Conference and Exposition on Electrical and Power Engineering (EPE)*, pages 1077–1082, Oct 2014.

[11] A. Cerón, I. F. Mondragón B., and F. Prieto. Power line detection using a circle based search with uav images. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 632–639, May 2014.

[12] Wei-guo Tong, Bao-shu Li, Jin-sha Yuan, and Shu-tao Zhao. Transmission line extraction and recognition from natural complex background. (July):12–15, 2009.

[13] Z. Li, Y. Liu, R. Hayward, J. Zhang, and J. Cai. Knowledge-based power line detection for uav surveillance and inspection systems. In *2008 23rd International Conference Image and Vision Computing New Zealand*, pages 1–6, Nov 2008.

[14] Yuee Liu, Luis Mejias, and Zhengrong Li. Fast power line detection and localization using steerable filter for active uav guidance. In *12th International Society for Photogrammetry & Remote Sensing (ISPRS2012)*, Melbourne Convention and Exhibition Centre, Melbourne, VIC, 2012.

[15] Y. Liu and L. Mejias. Real-time power line extraction from unmanned aerial system video images. In *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, pages 52–57, Sept 2012.

[16] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979.

[17] N. Aggarwal and W. C. Karl. Line detection in images through regularized hough transform. *IEEE Transactions on Image Processing*, 15(3):582–591, March 2006.

[18] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[19] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[20] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[21] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, Sep 1991.

[22] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.

[23] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London B: Biological Sciences*, 207(1167):187–217, 1980.

[24] R. Eckhorn, H. J. Reitboeck, M. Arndt, and P. Dicke. Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex. *Neural Comput.*, 2(3):293–307, September 1990.

[25] Leandro A.F. Fernandes and Manuel M. Oliveira. Real-time line detection through an improved hough transform voting scheme. *Pattern Recognition*, 41(1):299 – 314, 2008.

[26] T. S. Chan and R. K. K. Yip. Line detection algorithm. In *Proceedings of the 13th International Conference on Pattern Recognition - Volume 2*, ICPR '96, pages 126–, Washington, DC, USA, 1996. IEEE Computer Society.

[27] F. Tupin, H. Maitre, J. F. Mangin, J. M. Nicolas, and E. Pechersky. Detection of linear features in sar images: application to road network extraction. *IEEE Transactions on Geoscience and Remote Sensing*, 36(2):434–453, Mar 1998.

[28] J. Radon. Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten. *Akad. Wiss.*, 69:262–277, 1917.

[29] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 465–470, Jun 1996.

[30] Irvin Rock and Stephen Palmer. The legacy of Gestalt psychology. *Scientific American*, 263(6), 1990.

[31] R. Grompone von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, April 2010.

[32] C. Akinlar and C. Topal. Edlines: Real-time line segment detection by edge drawing (ed). In *2011 18th IEEE International Conference on Image Processing*, pages 2837–2840, Sept 2011.

[33] Ö. E. Yetgin, Z. Şentürk, and Ö. N. Gerek. A comparison of line detection methods for power line avoidance in aircrafts. In *2015 9th International Conference on Electrical and Electronics Engineering (ELECO)*, pages 241–245, Nov 2015.

[34] Agnès Desolneux, Lionel Moisan, and Jean-Michel Morel. Meaningful alignments. *International Journal of Computer Vision*, 40(1):7–23, 2000.

[35] C. Topal, C. Akinlar, and Y. Genc. Edge drawing: A heuristic approach to robust real-time edge detection. In *2010 20th International Conference on Pattern Recognition*, pages 2424–2427, Aug 2010.

[36] C. Topal, O. Ozsen, and C. Akinlar. Real-time edge segment detection with edge drawing algorithm. In *2011 7th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 313–318, Sept 2011.

[37] X. Lu, J. Yao, K. Li, and L. Li. Cannylines: A parameter-free line segment detector. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 507–511, Sept 2015.