UNIVERSIDADE
FEDERAL RURAL
DE PERNAMBUCO

Jeremias Leite da Silva

Detecção de linhas que separam o mar da areia e o mar do céu em imagens de praia

Recife

2018

Jeremias Leite da Silva

Detecção de linhas que separam o mar da areia e o mar do céu em imagens de praia

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Computação

Curso de Bacharelado em Ciência da Computação

Orientador: Valmir Macário Filho

Recife

2018

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Biblioteca Central, Recife-PE, Brasil

S586d Silva, Jeremias Leite da.
Detecção de linhas que separam o mar da areia e o mar do céu em imagens de praia / Jeremias Leite da Silva. - Recife, 2018.
92 f.: il.

Orientador(a): Valmir Macário Filho.
Trabalho de Conclusão de Curso (Graduação) – Universidade Federal Rural de Pernambuco, Departamento de Computação, Recife, BR-PE, 2019.
Inclui referências.

1. Detecção de linha do horizonte 2. Detecção de linha da costa 3. Programação dinâmica I. Macário Filho, Valmir, orient. II. Título

CDD 004

MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO (UFRPE)
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

http://www.bcc.ufrpe.br

FICHA DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

Trabalho defendido por Jeremias Leite da Silva às 10 horas do dia 15 de janeiro de 2019, no Auditório do CEAGRI-02 – Sala 07, como requisito para conclusão do curso de Bacharelado em Ciência da Computação da Universidade Federal Rural de Pernambuco, intitulado " **Detecção de linhas que separam o mar da areia e o mar do céu em imagens de praia** ", orientado por Valmir Macario Filho e aprovado pela seguinte banca examinadora:

Valmir Macario Filho

DC/UFRPE

Filipe Rolim Cordeiro
DC/UFRPE

”Dedico este trabalho aos meus pais Rute Leite e Joabe José, irmãos, e a todos os professores que me acompanharam durante a graduação, em especial ao Prof. Valmir Macário Filho, responsável pela realização deste trabalho. E o que dizer a vocês Sandrelynn e Jonathas? Obrigado pela paciência, pelo incentivo, pela força e principalmente pelo carinho. Valeu a pena toda distância, todo sofrimento, todas as renúncias... Valeu a pena esperar... Hoje estamos colhendo, juntos, os frutos do nosso empenho! Esta vitória é muito mais de vocês do que minha!!!”

Agradecimentos

Primeiramente a gradeço a Deus, que sempre esteve ao meu lado me dando forças e saúde para continuar.

Ao Prof. Dr. Valmir Macário Filho pela oportunidade e apoio na elaboração deste trabalho, e aos demais professores do departamento de computação da Universidade Federal Rural de Pernambuco.

Agradeço a Sandra Xavier que foi minha guia durante essa graduação e que sempre estava lá para orientar e dá força quando precisei.

Aos meus amigos de curso, companheiros de trabalhos e irmãos na amizade que fizeram parte da minha formação e sempre me ajudaram e me motivaram a continuar.

Agradeço também aos meus pais que sempre estiveram orando por mim.

Agradeço a minha esposa Sandrelynn, heroína paciente que me deu apoio, incentivo nas horas difíceis, de desânimo e cansaço. Por fim ao meu filho Jonathas que mesmo pequeno foi compreensível abdicando de brincadeiras, e foi o motivo de eu chegar até o fim do curso.

”Uma visão não é apenas uma imagem do que poderia ser; é um apelo ao nosso melhor, um chamado para se tornar algo mais.”

Rosabeth Moss Kanter

Resumo

A orla da região metropolitana de Pernambuco tem apresentado vários incidentes com tubarões e alguns desses casos foram fatais. Visando reduzir os incidentes, este trabalho propõe uma abordagem para segmentação do mar como parte de um sistema de monitoramento de banhistas através de câmeras. Uma vez que são identificadas uma ou mais pessoas dentro da zona de risco, o sistema emitirá um alerta a central de monitoramento, e o guarda-vida mais próximo seria alertado para se deslocar até o local. Para que o sistema identifique as pessoas na imagem, o sistema deve ser capaz de identificar a região de praia na imagem, afim de segmentá-las e identificar os banhistas. A faixa de água é formada por duas fronteiras, uma com o céu e a outra com a areia. A fronteira com o céu é uma linha reta horizontal denominada linha do horizonte, e a fronteira com a areia é um contorno formado pelo limite da água com a areia que é chamado de linha da costa. Esse trabalho visa propor algoritmos para detecção das linhas do horizonte e da costa para segmentação do mar, que representa uma das etapas principais para o sistema de monitoramento de banhista em imagens de praia. Neste trabalho foram analisados quatro algoritmos de detecção da linha do horizonte para avaliar quais desses obtém o melhor resultado na detecção. Dois algoritmos do estado da arte foram analisados: o de Lie *et al.* e o de Ahmad *et al.*. Ambos são trabalhos para detecção de linha do horizonte em imagens de montanhas, outros dois algoritmos foram contribuições desse trabalho: o Detecção da Linha do horizonte com *Canny Edge Detection* e Grafo Multiestágios (DLHCGME) e o Detecção da Linha do Horizonte com *Sobel* e transformada de *Hough* (DLHSTH). Para detecção da linha da costa foram propostos dois novos algoritmos: o Detecção da Linha da Costa com *Canny Edge Detection* e Grafo Multiestágios (DLCCGME) e Detecção da linha da costa baseado em contornos do canal *hue* (DLCCCH). Na detecção da linha do horizonte em imagens sem oclusões os experimentos demostram que o DLHCGME obteve o melhor resultado com uma taxa de erro de 0,47 e o segundo foi o DLHSTH com 1,11, e para imagens com oclusões o DLHSTH obteve o melhor resultado com taxa de erro de 1,98 e o DLHCGME foi o segundo melhor resultado com 2,62.

Palavras-chave: detecção de linha do horizonte, detecção de linha da costa, programação dinâmica.

Abstract

The border of the metropolitan region of Pernambuco has been presented several incidents from sharks and some of these cases were fatal. In order to reduce the incidents, this work proposes an approach for the segmentation of the sea as part of a monitoring system for bathers through cameras. Once one or more persons are identified within the risk zone, the system will issue an alert to central monitoring, and the nearest lifeguard would be alerted to move to the location. In order for the system to identify the people in the image, the system must be able to identify the beach region in the image in order to segment them and identify the bathers. The strip of water is formed by two borders, one with the sky and the other with sand. The boundary with the sky is a straight horizontal line called the horizon line, and the boundary with the sand is a contour formed by the boundary of the water with the sand that is called the shoreline. This work aims to propose algorithms for the detection of the horizon and coastlines for sea segmentation, which represents one of the main steps for the monitoring system of beach bather images. In this work four horizon-line detection algorithms were analyzed to evaluate which of these obtains the best detection result. Two state-of-the-art algorithms were analyzed: Lie et al. and that of Ahmad et al. Both are works for the detection of the horizon line in mountain images. Two other algorithms were contributions of this work: the Canny Edge Detection and Multi-Stage Graph Detection (DLHCGME) and Detection of Horizon Line with Sobel and Hough Transform (DLHSTH). Two new algorithms were proposed for Coastline Detection: Coastline Detection with Canny Edge Detection and Multi-Stage Graphs (DLCCGME) and Coastline Detection based on Contour of the Hue Channel (DLCCCH). In the detection of the horizon line in images without occlusions the experiments show that the DLHCGME obtained the best result with an error rate of 0.47 and the second was the DLHSTH with 1.11 and for images with occlusions the DLHSTH obtained the best result with an error rate of 1.98 and the DLHCGME was the second best result with 2.62.

Keywords: horizon line detection, coastline detection, dynamic programming.

Lista de ilustrações

Figura 1 – Placa de alerta de ataques de tubarões. 17
Figura 2 – Exemplo de imagem de praia, a linha do horizonte em preto e linha da costa em verde. 20
Figura 3 – (a) Luminosidade, (b) oclusões por coqueiros, (c) oclusões por guarda-sóis. 21
Figura 4 – Cubo RGB. 22
Figura 5 – Decomposição de imagem RGB. 23
Figura 6 – (a) imagem em RGB e em (b) imagem convertida para tons de cinza. 24
Figura 7 – Cone HSV. 24
Figura 8 – Exemplo de imagem RGB à esquerda e mesma imagem no HSV na imagem da direita. 25
Figura 9 – Separação dos canais HSV. 25
Figura 10 – Exemplo de limiarização. 26
Figura 11 – Imagem original em escala de cinza na primeira, G_x na segunda, o G_y na terceira e a magnitude do gradiente na quarta. 27
Figura 12 – Exemplo de uso do operador *Canny*. 29
Figura 13 – Representação normal da reta. 30
Figura 14 – Espaço de *Hough*. 31
Figura 15 – Exemplo de aplicação da transformada de *Hough*. 31
Figura 16 – Grafo multi-estágios com cinco estágios. 32
Figura 17 – Caminho mais curto (A, C, E, D, F) entre os vértices A e F no grafo direcionado com peso. 33
Figura 18 – Exemplo de execução do algoritmo de Dijkstra. 35
Figura 19 – Exemplo de detecção de bordas 42
Figura 20 – Topologia do grafo multiestágios para um mapa de bordas 8×8 . . 43
Figura 21 – Expansão de região para um vértice p, que não possui vértices de bordas no estágio seguinte 44
Figura 22 – Exemplos de detecção da linha do horizonte 45
Figura 23 – Exemplos de classificação da linha do horizonte para uma imagem de praia. 46
Figura 24 – mDCSI aplicado na imagem da Figura 23. 47
Figura 25 – Exemplo de detecção da linha do horizonte. 48
Figura 26 – Exemplo de detecção de bordas. 50
Figura 27 – Exemplo de detecção de linhas usando a transformada de *Hough*. . 51
Figura 28 – Exemplo de detecção da linha do horizonte usando o DLHSTH. . . 51
Figura 29 – Exemplo de imagem que não pode se aplicado o DLHCGME. 53

Figura 30 – Etapas da detecção das bordas. 54
Figura 31 – Etapas da detecção das bordas. 55
Figura 32 – Topologia do grafo multiestágios para um mapa de borda 8×6. . . 55
Figura 33 – Exemplo de detecção da linha do horizonte utilizando o DLHCGME. 56
Figura 34 – *Clusters* binarizados à esquerda e preenchimento do contorno externo de grandes áreas à direita. 58
Figura 35 – *Clusters* obtidos e seus histogramas referentes ao canal H. 59
Figura 36 – (1) Imagem original; (2) Segmentação temporária com erros; (3) Expansão de borda e ajuste das regiões faltantes; (4) Segmentação final de praia. 60
Figura 37 – Máscara da segmentação de praia final (esquerda) e região de água detectada (direita). 60
Figura 38 – Algoritmo de segmentação de praia. 61
Figura 39 – (A) Imagem original; (B) Mapa de bordas a partir de uma imagem em tons de cinza; (C) Mapa de bordas a partir do canal H do HSV. . . . 62
Figura 40 – Exemplos de detecções da linha da costa com o DLCCGM. 63
Figura 41 – Exemplos de imagem de praia no sistema de cor HSV, onde (a) é a imagem original em RGB e (b) é a imagem em HSV. 64
Figura 42 – Exemplos de extrações do canal H em imagens de praia. 65
Figura 43 – Limiarizações do canal H de imagens de praia. 66
Figura 44 – Imagem de entrada a esquerda e resultado após subtrações dos pequenos contornos brancos na imagem à direita. 67
Figura 45 – A imagem de entrada à esquerda e resultado após subtrações dos pequenos contornos brancos na imagem no centro e resultado na direita. 67
Figura 46 – Exemplo de imagem de praia com oclusão. 70
Figura 47 – Imagem rotulada. 71
Figura 48 – Imagem rotulada para treinamento do DCSI. 72
Figura 49 – Exemplo de imagem sem oclusão. 74
Figura 50 – Exemplos de imagens com oclusões. 75
Figura 51 – Exemplo de rotulagem de imagem com oclusões para linha do horizonte. 76
Figura 52 – Exemplo de rotulagem de imagem com oclusões para linha da costa. 77
Figura 53 – Exemplos de detecções da linha do horizonte sem oclusões por árvores 79
Figura 54 – Exemplos de detecções da linha do horizonte com oclusões por árvores 81
Figura 55 – Exemplos de detecções da linha da costa sem oclusões, em verde linha detectada e em preto linha rotulada 82

Figura 56 – Exemplos de detecções da linha da costa com oclusões por árvores 84

Lista de tabelas

Tabela 1 – Visualização dos algoritmos abordados nesse trabalho 41
Tabela 2 – Visualização dos resultados dos experimentos de detecção da linha do horizonte para base de dados sem oclusões. 78
Tabela 3 – Visualização dos resultados dos experimentos de detecção da linha do horizonte para base de dados com oclusões. 80
Tabela 4 – Visualização dos experimentos de detecção da linha da costa para base de dados sem oclusões. 82
Tabela 5 – Visualização dos experimentos de detecção da linha da costa para base de dados com oclusões. 83
Tabela 6 – Visualização da média dos experimentos H-1 e H-2. 85
Tabela 7 – Visualização da média dos experimentos C-1 e C-2. 85

Lista de abreviaturas e siglas

ASPC	Algoritmo de segmentação da praia de Carrera
Cemit	Comitê Estadual de Monitoramento de incidentes com tubarões
DLCCCH	Detecção da Linha da Costa Baseado em Contornos do Canal *Hue*
DLCCGME	Detecção da Linha da Costa com Canny e Grafo Multi-Estágios
DCSI	Dense Classifier Score Image
DLHCGME	Detecção da Linha do Horizonte com Canny Edge Detector e Grafo Multi-Estágios
DLHSTH	Detecção da Linha do Horizonte com Sobel e Transformada de Hough
GPU	Graphics Processing Unit
H-HC	Edge detection and Hough transform based algorithm
HSV	Hue, Saturation, Value
mDCSI	Reduced Dense Classifier Score Image
OpenCv	Open Source Computer Vision Library
RGB	Red, Green e Blue
SVM	Máquina de Vetores de Suporte
UFRPE	Universidade Federal Rural de Pernambuco

Sumário

Lista de ilustrações **9**

1 INTRODUÇÃO **17**
1.1 Problema de Pesquisa **19**
1.2 Objetivos **21**
1.3 Estrutura do trabalho **21**

2 CONCEITOS BÁSICOS **22**
2.1 Processamento de imagem **22**
2.1.1 Espaço de cores 22
2.1.1.1 Modelo RGB 22
2.1.1.2 Tons de cinza 23
2.1.1.3 Modelo HSV 24
2.1.2 Limiarização 25
2.1.3 Detectores de bordas 26
2.1.3.1 Operador *Sobel* 26
2.1.3.2 *Canny* 27
2.1.3.3 Transformada de *Hough* 29
2.2 Grafo multi-estágios **31**
2.3 Programação dinâmica **32**
2.3.1 Problema do caminho mais curto 33
2.3.2 O algoritmo de *Dijkstra* 33

3 TRABALHOS RELACIONADOS **36**

4 DETECÇÃO DE LINHA DO HORIZONTE E LINHA DA COSTA . **41**
4.1 Algoritmos para detecção de linha do horizonte **41**
4.1.1 Algoritmo Lie *et al.* 41
4.1.1.1 Detecção de bordas 41
4.1.1.2 Grafo multiestágios 42
4.1.1.3 Expansão de região 44
4.1.1.4 Encontrar a linha do horizonte 44
4.1.2 *Dense classifier score image* (DCSI) 45
4.1.2.1 Classificação dos *pixels* 46
4.1.2.2 *Reduced dense classifier score image* (mDCSI) 47
4.1.2.3 Detecção da linha do horizonte como um problema de caminho mais curto . . 47

4.1.3 DLHSTH . . . 49
4.1.3.1 Detecção de bordas . . . 49
4.1.3.2 Encontrar a linha do horizonte . . . 50
4.1.4 DLHCGME . . . 53
4.1.4.1 Detecção das bordas . . . 54
4.1.4.2 Encontrar a linha do horizonte . . . 55
4.2 Algoritmos para detecção da linha da costa . . . 57
4.2.1 Algoritmo de segmentação da praia de Carrera (ASPC) . . . 57
4.2.2 DLCCGME . . . 61
4.2.2.1 Detecção de bordas . . . 62
4.2.2.2 Encontrar a linha da costa . . . 62
4.2.3 DLCCCH . . . 64
4.3 Observações . . . 68

5 AVALIAÇÃO EXPERIMENTAL . . . 70
5.1 Base de dados . . . 70
5.1.1 Base de dados para treinamento do DSCI . . . 71
5.2 Ambiente . . . 72
5.3 Métrica de avaliação . . . 72
5.3.1 Média dos erros médios absolutos . . . 72
5.3.2 Desvio padrão do erros médios absolutos . . . 73
5.3.3 Tempo médio de processamento . . . 73
5.4 Experimentos H . . . 73
5.4.1 Experimento H-1 . . . 74
5.4.2 Experimento H-2 . . . 74
5.5 Experimentos C . . . 76
5.5.1 Experimento C-1 . . . 76
5.5.2 Experimento C-2 . . . 76

6 RESULTADOS . . . 78
6.1 Algoritmos de detecção da linha do horizonte . . . 78
6.1.1 Experimento H-1 . . . 78
6.1.2 Experimento H-2 . . . 80
6.2 Algoritmos de detecção da linha da costa . . . 81
6.2.1 Experimento C-1 . . . 81
6.2.2 Experimento C-2 . . . 83
6.2.3 Discussão . . . 84
6.2.3.1 Limitações . . . 86

7 CONCLUSÃO . . . 88

7.1 Trabalhos futuros 89

REFERÊNCIAS 90

1 Introdução

A orla da região metropolitana de Pernambuco vem apresentado vários incidentes com tubarões e alguns desses casos foram fatais. Desde 1992 a contagem vem sendo feita pelo Comitê Estadual de Monitoramento de incidentes com tubarões (Cemit) o qual registrou 65 incidentes e nesse mesmo período foram 25 vítimas fatais (COMMERCIO, 2018).

Uma das medidas preventivas realizadas pelo Cemit são as instalações de placas de alerta do risco de ataques de tubarões a fim de orientar sobre os perigos de ataques naquela região. Apesar das instalações das placas de advertência vários banhistas a ignoram e se colocam em situações de risco. Outra medida para reduzir os incidentes são os guardas-vidas que é o profissional apto para realizar medidas preventivas, educacionais e salvamento. Uma grande dificuldade é ter o número suficiente de guardas-vidas para realizações de monitoramentos ao longo da costa pernambucana e o custo financeiro que necessitará ser investido com esse pessoal.

Figura 1 – Placa de alerta de ataques de tubarões.

Fonte – *Autor*

Visto isso, uma parceria entre o Cemit, a secretaria de defesa social (SDS) e a Universidade Federal Rural de Pernambuco (UFRPE) propõe uma abordagem para o monitoramento de banhistas através de câmeras e seriam instaladas várias em locais estratégicos cobrindo as áreas de risco. Uma vez identificadas uma ou mais pessoas

dentro da zona de risco que neste caso seria a região de água do mar, o sistema emitirá um alerta a central de monitoramento, e o guarda-vida mais próximo será alertado para se deslocar até esses indivíduos a fim de orientá-los.

O sistema de monitoramento automático de banhistas na região de praia possui 3 principais etapas: segmentação, classificação e rastreamento. Na segmentação, as áreas de interesse são destacadas na imagem que podem conter os banhistas, na etapa de classificação as áreas segmentadas são classificadas como pessoa ou não pessoa, e na etapa de rastreamento as pessoas identificadas na etapa anterior são rastreadas.

Para que o sistema identifique os banhistas na imagem, a etapa de segmentação é uma das mais importantes pois nela são destacados regiões da imagem que podem conter ou não banhistas. A imagem de praia é bastante complexa, nela pode conter diversos elementos e diferentes iluminações, o ângulo filmado também deve ser levado em conta e entre outros fatores que pode deixar uma imagem bastante diferente uma da outra. Quanto mais limpo o resultado da segmentação mais chance do algoritmo apresentar resultados melhores. Para que os banhistas sejam identificados, achamos que seria muito útil para o sistema separar apenas a região de mar na imagem, pois essa região contém os banhistas que são o principal objetivo do projeto e também possui menos interferências de objetos estranhos que a areia da praia. Além do mais, o processamento realizado numa área menor da imagem pode acelerar o desempenho do algoritmo de encontrar o banhista.

Desse modo, esse trabalho propõe um método capaz de identificar a área referente à parte da água do mar para facilitar o processamento posterior. A zona de água é formada por duas fronteiras: uma com o céu e a outra com a areia. A fronteira com o céu é uma linha reta horizontal denominada linha do horizonte, e a fronteira com a areia é um contorno formado pelo limite da água com a areia que chamamos de linha da costa.

A segmentação do mar se torna um processo difícil devido à diversidade de textura que o mesmo pode assumir tais como: diferentes tipos de praia, ondas, iluminação, clima, cor, formato e outros elementos adicionados pela ação do homem. Devido à essas características é proposta uma abordagem para segmentação do mar que tem como base encontrar as linhas que delimitam essa região, que são a linha do horizonte e a linha da costa.

Há vários métodos na literatura para encontrar linhas do horizonte em imagens (GERSHIKOV TZVIKA LIBE, 2013), algumas abordagens fazem uso de algoritmos de aprendizagem de máquina, não supervisionados, e fusão entre supervisionado e não supervisionado (YAZDANPANAH et al., 2015), e outros apenas aplicam técnicas de processamento de imagens. Nos métodos supervisionados um conjunto de dados de

treinamento é usado para treinar o algoritmo. Nesta proposta podemos usar para classificar as regiões das imagens em céu, mar, areia, linha do horizonte e linha da costa. Os algoritmos não supervisionados são algoritmos baseados em formarem grupos, útil quando se deseja segmentar uma imagem de acordo com algumas características similares extraídas da imagem. Também se pode segmentar imagens apenas com processamento de imagem, onde podemos aplicar filtros, binarização, limiarização, detectores de bordas, operações morfológicas e etc (AHMAD et al., 2015).

Com a finalidade de reduzir os ataques de tubarões na Região Metropolitana do Recife esse trabalho propõe uma metodologia para a identificações das linhas de costa e horizonte com o intuito de segmentar a região de água em imagens de praia.

Para solucionar o problema de encontrar as linhas do horizonte e da praia em imagens de praia, é proposto o estudo de alguns algoritmos para encontra as linhas da costa e horizonte, onde esses algoritmos são divididos em dois grupos: algoritmos para detecção da linha do horizonte e algoritmos para detecção da linha da costa. Para os de detecção da linha do horizonte foram analisados o algoritmos de detecção de linha do horizonte em imagens montanhas de (AHMAD et al., 2015) e o algoritmo de (LIE et al., 2005) baseado em grafo multiestágios. Dois algoritmos foram propostos para contribuições desse trabalho, um baseado no detector de bordas *Canny Edge Detection* e grafo multiestágios denso, abordado em (AHMAD et al., 2015), e o outro baseado no detector de bordas *Sobel* e na transformada de *Hough*.

Quanto à detecção da linha da costa, visto que não foram encontrados trabalhos publicados na literatura, dois algoritmos foram propostos para encontrar a linha da costa e o algoritmo de segmentação de praia da monografia de (CARRERA, 2017) foi utilizado para comparações, o primeiro algoritmo proposto é baseado em detectores de bordas e grafo multiestágios, e o segundo é baseado em encontrar contornos das principais regiões da imagem.

1.1 Problema de Pesquisa

O problema de pesquisa proposto é: ”como detectar linhas que separam o mar da areia e do céu em imagens de praia”. Reconhecer essas linhas é uma das etapas principais para o sistema de detecção de banhista, pois é o passo fundamental para segmentar a região de mar, e é nessa segmentação que o sistema vai encontrar os banhistas. As imagens de praia são imagens extraídas de câmeras instaladas na orla para monitorar banhistas, em geral são imagens frontais da praia como a imagem da Figura 2.

A linha que separa o mar e o céu é a fronteira que separa essas regiões que é denominada linha do horizonte que nesse contexto se aproxima de uma linha reta.

A linha da costa é definida como o limite da região de água do mar que separa o mar da areia, mesmo as entradas de águas na areia que se apresentam como uma fina camada de água é considerado dentro do limite do mar.

Figura 2 – Exemplo de imagem de praia, a linha do horizonte em preto e linha da costa em verde.

Fonte – O autor

A segmentação do mar em imagem de praia torna-se uma tarefa mais complexa devido a variedades de cenários de praias, variedade de luminosidade, reflexo da luz na água, movimentações das ondas, guarda-sol, oclusões e uma fina camada de água sobre a água, que as características da cor pode ser confundida com a do mar e identificando pessoas nessa área, como se estivesse no mar. Apresar das complexidades encontradas em imagens de praias, o algoritmo deve ser capaz de encontrar as linhas do horizonte e linhas da costa com ótimas acurácias.

A luminosidade pode afetar drasticamente o tom da água, podendo ser total abrangendo toda região do mar ou parcial apenas parte dela, essa luminosidade pode ser causada pela reflexão da luz do sol na câmera. As oclusões proporciona um desafio na para detecção das linhas pois podem ocorrer devido aos guarda-sóis, e as pessoas que podem interceptar a linha da costa causando um desvio na linha da costa, também tem as grandes oclusões como coqueiros que podem se estender da borda inferior até a borda superior da imagem. Para esse problema apenas as imagens diurnas foram consideradas. Também apenas imagens frontais ou com pequena inclinação, onde as

linhas do horizonte e da costa devem se estender entre as bordas lateral da imagem. Figura 3 apresenta três exemplos desse problema.

Figura 3 – (a) Luminosidade, (b) oclusões por coqueiros, (c) oclusões por guarda-sóis.

Fonte – O autor

1.2 Objetivos

Objetivo Geral:

O trabalho proposto tem por objetivo implementar um método para extração da linha do horizonte e linha da costa em imagens de praia.

Objetivos Específicos:

- Avaliar diversos sistemas de cores, para escolher o que melhor diferencia os elementos na imagem de praia.
- Avaliar várias técnicas de visão computacional, a fim de extrair as linhas do horizonte e costa, para avaliar a que obtêm os melhores resultados.

1.3 Estrutura do trabalho

O Capítulo 2 desse trabalho apresentará uma introdução sobre as técnicas utilizadas, no contexto envolvendo processamento de imagens e programação dinâmica.

O Capítulo 3 traz os trabalhos relacionados, que foram utilizados como justificativas desse trabalho.

O Capítulo 4 apresenta os algoritmos utilizados para a detecção das linhas do horizonte e da costa.

O capítulo 5, apresenta a metodologia utilizada para realização desse projeto.

O Capítulo 6 descreve os resultados obtidos para cada experimento realizado.

No Capítulo 7, contém a conclusão e os trabalhos futuros.

2 Conceitos básicos

Neste Capítulo será feita uma introdução aos conceitos básicos de processamento de imagens e programação dinâmica que foram utilizados para chegar ao resultado deste trabalho, tais como espaço de cores, conversões entre espaços de cores, detectores de bordas, segmentação, limiarização, equalização do histograma e também serão abordados conceitos de programação dinâmica.

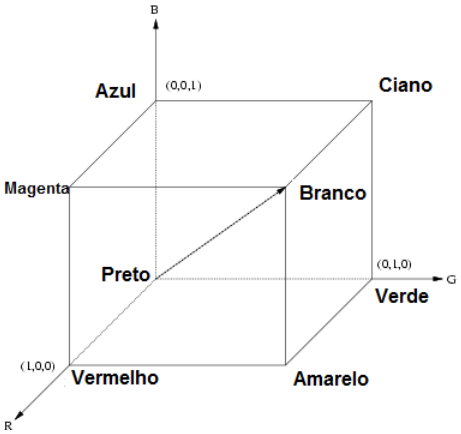
2.1 Processamento de imagem

2.1.1 Espaço de cores

Através das cores percebemos o mundo e temos a noção de como os objetos e textura são representados. A cor pode ser definida como a percepção humana de diversos espectros de luz. Um sistema de cores ou espaço de cores pode ser definido como um modelo matemático abstrato que descreve como as cores devem ser representada digitalmente. Existe dois sistemas de cores na natureza: os sistemas aditivos e os subtrativos. Os sistemas aditivos representam cor enquanto luz, e representam as cores primárias do espectro visível (azul, verde, vermelho), são usados em dispositivos que emitem luz. Os sistemas subtrativos representam cor enquanto pigmento, representam as cores primárias de impressão (amarelo, ciano e magenta). (GONZALEZ; WOODS, 2010). Para os sistemas de cores existentes, existe diversos modelos de cores que podem ser representados digitalmente.

2.1.1.1 Modelo RGB

Figura 4 – Cubo RGB.

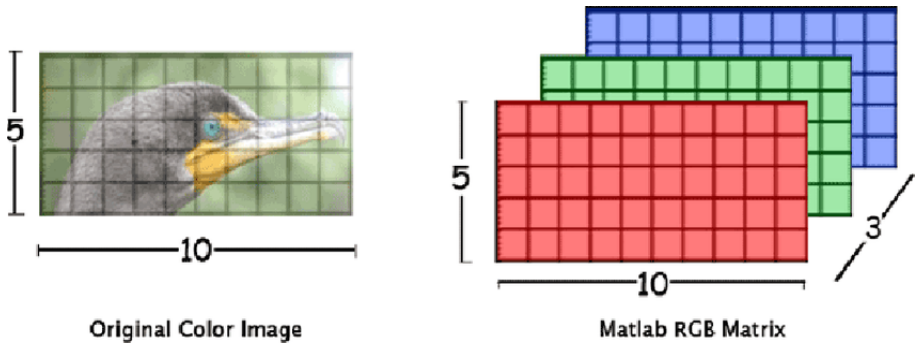


Fonte – (GONZALEZ; WOODS, 2010)

O modelo RGB (*Red, Green, Blue*) é baseado em sistemas de coordenadas cartesianas tridimensionais, que pode ser visto como um cubo, onde cada eixo cartesiano é uma cor primária, e para representar uma cor basta pegar um ponto dentro desse cubo, o vértice junto a origem corresponde ao tom preto e o mais afastado da origem corresponde a cor branca e a escala de cinza se estende através da diagonal do cubo que sai da origem (preto) até o vértice mais distante dela (branco) (GONZALEZ; WOODS, 2010), como pode ser visto na Figura 4.

As imagens representadas por esse modelo de cor possuem três componentes para representar uma imagem de modo que cada componente utiliza uma cor primária. Uma imagem colorida é formada a partir das combinações desses canais (componentes) como ilustrar a Figura 5.

Figura 5 – Decomposição de imagem RGB.



Fonte – (SINGH, 2015)

2.1.1.2 Tons de cinza

A conversão de uma imagem colorida em tons de cinza é muito útil em para várias aplicações que envolve processamento de imagem, tais como: detecção de bordas, processamento de imagem em um único canal, e impressões preto e branco. Vários trabalhos encontrados na literatura utilizam imagens em tons de cinza, por isso faz se necessário entender a conversão de uma imagem em RGB para tons de cinza. Muitas vezes é suficiente trabalhar com imagens de um único canal ao invés de três canais como RGB, pois diminui o tempo de processamento. Na Equação 2.1 podemos ver o método da combinação linear, onde as variáveis são as intensidades de cada canal de um *pixel* e as constantes são os valores utilizado pelo *Opencv* (OPENCV, 2015) para conversão de uma imagem em RGB para escala de cinza.

$$Y = 0.299R + 0.587G + 0.114B \tag{2.1}$$

Uma imagem em tons de cinza é obtida aplicando a formula 2.1 para cada *pixel* da imagem de entrada e o resultado de uma conversão pode ser vista na Figura 6

abaixo.

Figura 6 – (a) imagem em RGB e em (b) imagem convertida para tons de cinza.

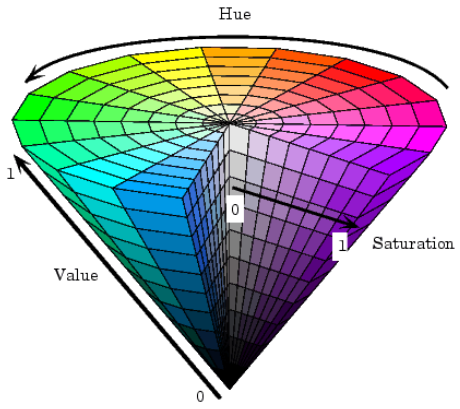
(a) Imagem original em RGB (b) Imagem em tons de cinza

Fonte – O autor

2.1.1.3 Modelo HSV

A representação tridimensional do espaço de cor HSV é uma pirâmide hexagonal invertida em que o eixo vertical central representa o Brilho (*Value*), um ângulo no intervalo [0,2π] define a Matiz (*Hue*) que representa um tipo de cor do espectro e a saturação (*Saturation*) é o grau de pureza da cor que é medida como uma distância radial do eixo central com valor entre 0 no centro e 1 na superfície externa (GANESAN et al., 2014). Na Figura 7 temos a pirâmide HSV.

Figura 7 – Cone HSV.



Fonte – (IBRAHEEM et al., 2012)

Esse modelo é baseado na ideia do sistema da visão humana. Na Figura 8 abaixo podemos ver uma imagem convertida de RGB para HSV.

Figura 8 – Exemplo de imagem RGB à esquerda e mesma imagem no HSV na imagem da direita.

Fonte – O Autor

O modelo HSV assim como o RGB possui três canais, assim sendo representado de forma análoga através de matrizes. Com isso nos possibilita analisar cada canal separadamente podemos observar na Figura 9, onde a partir da imagem HSV da Figura 8 foram extraídos os seus canais.

Figura 9 – Separação dos canais HSV.

Fonte – O autor

2.1.2 Limiarização

A limiarização é o processo de segmentação baseado na diferença dos valores de intensidade de cor de uma imagem, onde a partir de um limiar estabelecido de acordo com as características do objeto ou região que se deseja isolar, a imagem é binarizada obtendo duas classes: uma com valores acima do limiar e a outra com valores abaixo do limiar. Essa limiarização pode ser descrita pela Equação 2.2 em que T é o limiar estabelecido.

$$g(x,y) = \begin{cases} 1, & Sef(x,y) > T \\ 0, & Sef(x,y) \leq T \end{cases} \tag{2.2}$$

Na Figura 10 temos um exemplo de limiarização com valor de limiar $T = 100$.

Figura 10 – Exemplo de limiarização.

Fonte – O autor

2.1.3 Detectores de bordas

Bordas é uma característica muito útil em visão computacional, através dela podemos definir limites de um objeto em uma imagem ou uma característica, uma borda é definido como o limite entre duas regiões com propriedades relativamente distintas de níveis de cinza (GONZALEZ; WOODS, 2010), elas podem ser causadas por descontinuidade de profundidade, descontinuidade de cor, descontinuidade de orientação de uma superfície, mudança nas propriedades da superfície do material, descontinuidade na iluminação (sombras), para extração dessas bordas existe diversas técnicas na literatura e algumas delas serão apresentadas a seguir.

2.1.3.1 Operador *Sobel*

O filtro *Sobel* é uma operação utilizada na detecção de contornos. Ele calcula a magnitude do gradiente da imagem em cada ponto para a função $f(x, y)$. Este operador utiliza duas matrizes 3x3, através de uma convolução que é deslocar a máscara sobre a imagem e calculando a soma dos produtos em cada local, é calculada as aproximações das derivadas, sendo uma para variações horizontais e outra para as verticais. As mudanças na horizontal é calculada pela convolução com um *kernel* G_x mostrado na equação 2.3, onde I é a imagem de entrada.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * I \tag{2.3}$$

Mudanças na vertical é calculado pela convolução com um *kernel* G_y mostrado na equação 2.4.

$$G_y = \begin{bmatrix} +1 & 2 & +1 \\ -0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \tag{2.4}$$

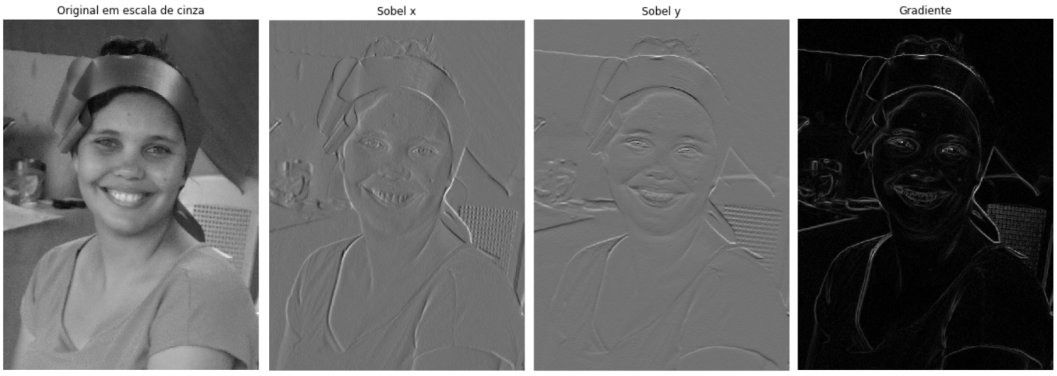
Para cada ponto da imagem calculamos a magnitude do gradiente G e a direção do gradiente Θ nesse ponto combinando os de $G_x x$ e G_y, como mostra as Equações 2.5 e 2.6.

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.5}$$

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{2.6}$$

Na Figura 11 abaixo podemos observar o resultado do operador *Sobel*.

Figura 11 – Imagem original em escala de cinza na primeira, G_x na segunda, o G_y na terceira e a magnitude do gradiente na quarta.



Fonte – O autor

2.1.3.2 *Canny*

O detector de bordas *Canny* é um dos mais popular operador de detecção de bordas, foi desenvolvido por John F. Canny em 1986, ele utiliza um algoritmo de múltiplos estágios para detectar uma ampla quantidades de bordas (CANNY, 1986). A ideia desse algoritmo é obter um método de detecção ótimo das bordas, e uma detecção ótima da borda significa que:

- o algoritmo deve marcar tantas bordas quanto possível;
- as bordas marcadas devem estar tão perto quanto possível da borda real; e
- cada borda na imagem deve ser marcada uma vez, e o ruído da imagem não deve criar bordas falsas.

O processo desse algoritmo de detecção de borda pode ser dividido em 5 etapas diferentes (MAIA; PORFÍRIO, 2002):

1. Aplicação do filtro Gaussiano para suavizar e remover ruídos. Como a detecção de bordas é suscetível a ruídos na imagem, o primeiro passo é remover os ruídos na imagem com um filtro gaussiano 5x5 como ilustra a Equação 2.7, para evitar a detecção de bordas falsas causada pelos ruídos.

$$F = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{2.7}$$

2. Uma borda de uma imagem pode apontar para várias direções, deste modo quatro máscara para detectar bordas horizontais, verticais e diagonais (45 e 135 graus) são utilizadas. O operador de detecção de borda muitas vezes é utilizado o *Sobel*, que retorna um valor para a primeira derivada na direção horizontal G_x e na direção vertical G_y. Então o gradiente e a direção da borda pode ser determinado pelas respectivas equações 2.8 e 2.9.

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.8}$$

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{2.9}$$

3. *Non-maximum suppression* é uma técnica para redução de bordas onde apenas máximos locais são considerados bordas, depois de obter a magnitude e a direção do gradiente para cada *pixel*, então ele é verificado se é um máximo local em sua vizinhança na direção do gradiente. Os passos para cada *pixel* no gradiente da imagem é:
   - compare a intensidade da borda do *pixel* atual com a intensidade da borda do *pixel* nas direções de inclinações positiva e negativa;

- se a intensidade da borda do *pixel* atual for maior em comparação com os outros *pixel* da máscara com a mesma direção o valor será preservado. Caso contrário, o valor será suprimido.

4. Após a aplicação da *Non-maximum suppression*, os *pixel* da borda remanescente fornecem uma representação mais precisa das bordas reais de uma imagem. No entanto alguns *pixel* da borda permanecem, causados pela variação de ruído e cor. Para responder por essas respostas falsas é essencial filtrar os *pixel* da borda com um valor de gradiente fraco e preservar os *pixel* da borda com um valor de gradiente alto. Isso é feito selecionando valores limite altos e baixos, se o valor do gradiente de um *pixel* de borda for maior que o valor de limite alto então será marcado como um *pixel* de borda forte, se o valor do gradiente de um *pixel* de borda for menor que o valor de limite alto e maior que o valor de limite baixo, ele será marcado como um *pixel* de borda fraca, se o valor de um *pixel* de borda for menor que o valor de limite baixo então será suprimido.

5. *Hysteresis*: esta etapa decide dentre todas as bordas, quais são realmente bordas. Bordas finas são determinadas suprimindo as bordas que não estão conectadas a bordas fortes. Na Figura 12 podemos ver o resultado do operador *Canny*.

Figura 12 – Exemplo de uso do operador *Canny*.

Fonte – O autor

2.1.3.3 Transformada de *Hough*

A transformada de *Hough* é uma técnica de detecção de formas geométricas que podem ser parametrizadas, usada em análise de imagens, visão computacional e processamento digital de imagens, foi desenvolvida por Paul Hough em 1962 (ILLINGWORTH; KITTLER, 1988) . Tem como objetivo encontrar instâncias imperfeitas

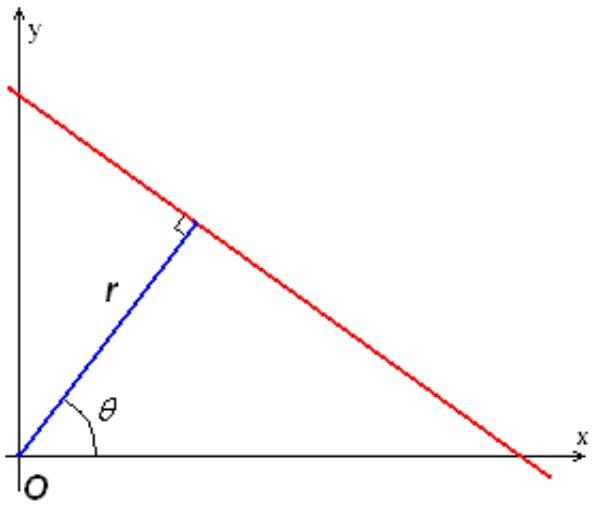
de objetos dentro de uma determinada classe de formas por um procedimento de votação. Este método de votação é realizado em um espaço de parâmetros, a partir do qual os candidatos a objetos são obtidos como máximos locais em um espaço acumulador que é construído pelo algoritmo para calcular a transformada.

O algoritmo funciona partindo da suposição que para uma imagem com n pontos, queiramos encontrar subconjuntos deste pontos que sejam colineares. O caso mais comuns de utilização desse método é a detecção de linhas restas em que a linha reta é determinada pelo ponto (b, m) no espaço de parâmetros da Equação $y = mx + b$. Entretanto linhas verticais representam um problema, eles teria valores e origens que tendem ao infinito na inclinação m. Uma alternativa é considerar a representação normal da reta em coordenada polares como pode-se ver na Equação 2.10.

$$\rho = x \cos\theta + y \sin\theta \tag{2.10}$$

Na Equação 2.10 ρ é a distância perpendicular da reta à origem do plano xy e θ é o ângulo desta reta perpendicular em relação ao eixo x como podemos ver na Figura 13.

Figura 13 – Representação normal da reta.

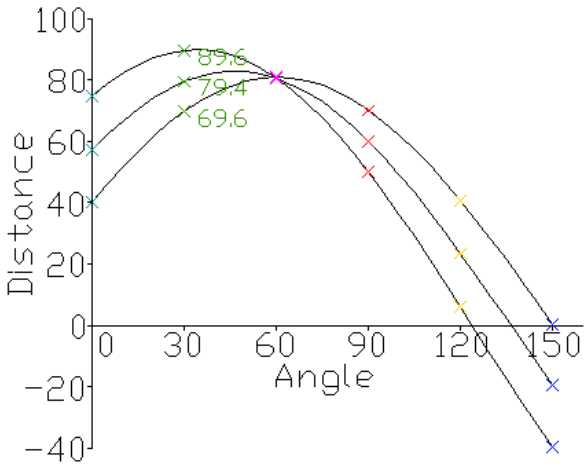


Fonte – (WIKIPEDIA.ORG, a)

Cada curva senoidal no espaço de parâmetros representa a família de retas que passam por um ponto no plano xy, e o ponto de interseção (ρ,θ) corresponde à reta que passa por dois ponto no plano xy.

Na Figura 14, podemos ver o espaço de *Hough*, o ponto em que as curvas se cruzam dá uma distância e um ângulo. Esta distância e ângulo indicam uma linha que intercepta os pontos sendo testados.

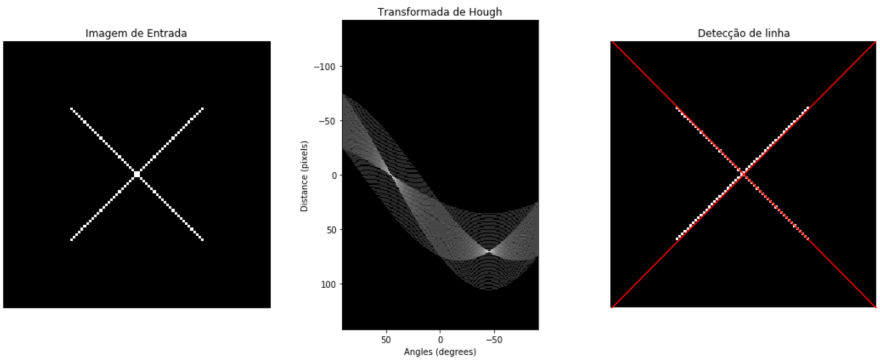
Figura 14 – Espaço de *Hough*.



Fonte – (WIKIPEDIA.ORG, c)

Na Figura 15 podemos ver o resultado da aplicação desse algoritmo, na imagem central podemos observar as senoides que são formadas pelos pontos das linhas da imagem da esquerda, e também podemos notar na imagem central dois pontos de intersecção que representam duas restas que são encontradas na imagem da direita.

Figura 15 – Exemplo de aplicação da transformada de *Hough*.



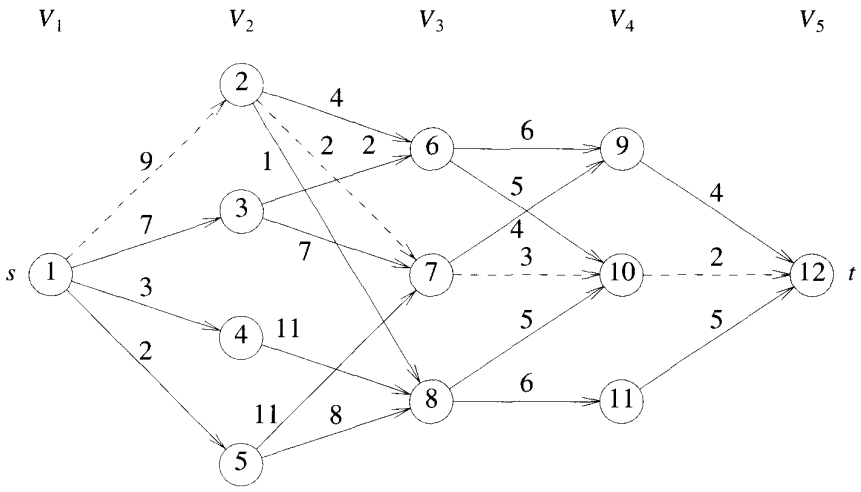
Fonte – (IMAGE.ORG,)

2.2 Grafo multi-estágios

Segundo Horowitz, Sahni e Rajasekaran (2007), um grafo multi-estágio $G(V, E)$ é um grafo direcionado, em que os vértices são particionados em $k \geq 2$ V_i conjuntos disjuntos, onde $1 \leq i \leq k$. Se $\langle u, v \rangle$ é uma aresta pertencente a E, então $u \in V_i$ e $v \in V_{i+1}$ para qualquer $1 \leq i < k$. Os conjuntos V_1 e V_k são qual que $|V_1| = |V_k| = 1$, isso indica que o número de vértices pertencente a V_1 e a V_k devem ser igual a 1. O vértice que pertence a V_1 é a origem e é representado por s e o vértice pertencente a V_k é chamado de destino. A aresta $\langle i, j \rangle$ tem custo $c(i, j)$. O custo de um caminho de s

para t é a soma dos custos das arestas no caminho. O problema do grafo multi-estágio é para encontrar um caminho de custo mínimo. Na Figura 16 abaixo podemos ver um exemplo de grafo multi-estágios com cinco estágios.

Figura 16 – Grafo multi-estágios com cinco estágios.



Fonte – (HOROWITZ; SAHNI; RAJASEKARAN, 2007)

2.3 Programação dinâmica

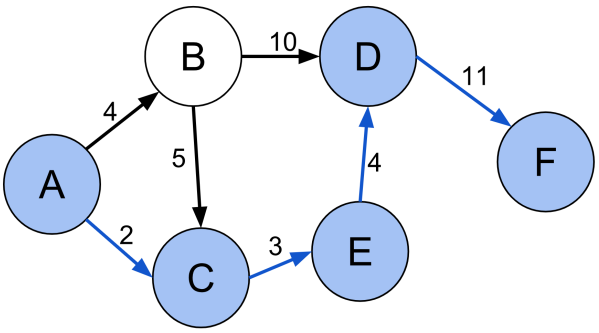
A programação dinâmica é um método usado quando a solução para um problema pode ser vista como o resultado de um sequência de decisões. Segundo (HOROWITZ; SAHNI; RAJASEKARAN, 2007), esta técnica procura resolver o problema de otimização através da resolução de vários subproblemas para chegar à solução do problema principal. A programação dinâmica em geral é aplicada a problemas de otimizações em que um problema pode ter várias soluções e que cada solução possui um valor de custo onde podemos encontrar uma solução ótima podendo ser mínima ou máxima. Vários problemas podem ser resolvidos por programação dinâmica tais como:

1. O problema da mochila, onde o objetivo é maximizar a soma dos valores dos itens que serão colocados na mochila respeitando sua capacidade.
2. O problema do Caixeiro-viajante, que consiste em determinar o caminho mais curto que passa exatamente uma vez por cada vértice e retorna ao vértice de partida.
3. O problema do caminho mais curto, que consiste em encontrar o caminho mais curto entre dois nós qualquer de um grafo direcionado.

2.3.1 Problema do caminho mais curto

Na teoria dos grafos o problema de caminhos mais curtos é o problema para encontrar o menor caminho entre dois vértices (nós) em um grafo ponderado tal que o somatório dos pesos das arestas constituintes seja o mínimos (CORMEN et al., 2002). Na Figura 17 podemos observar o menor caminho formado de vértices A para F em um grafo direcionado com peso nas arestas, portando os vértices (A,C,E,D,F) é a solução para esse problema.

Figura 17 – Caminho mais curto (A, C, E, D, F) entre os vértices A e F no grafo direcionado com peso.



Fonte – (WIKIPEDIA.ORG, b)

Uma variante do problema do caminho mais curto chamado de problema do caminho mais curto de uma única origem: dado uma grafo $G(V, E)$ queremos encontrar um caminho mais curto de um vértice de origem s pertencente a V até todos os vértices v pertencente a V. Muitos problemas podem ser resolvidos pelo algoritmo do problema do caminho mais curto de única origem.

2.3.2 O algoritmo de *Dijkstra*

O algoritmo de *Dijkstra* é um algoritmo para encontrar solução do problema do caminho mais curto de única origem em um grafo $G(V, E)$ ponderado não negativo. A partir de um vértice inicial ele calcula o custo mínimo deste vértice para todos os demais vértices do grafo. Ele pode ser usado para grafo orientado ou não orientado e admite que todas as arestas possuem pesos positivos, podendo ser usado para achar o menor caminho entre dois vértices quaisquer.

Este algoritmo mantém um conjunto S de vértices cujos pesos finais de caminhos mais curtos desde a origem s já foram determinados. Esse método seleciona repetidamente o vértice $u \geq V - S$ com a estimativa mínima de caminhos mais curtos, adiciona u e s e relaxa todas as arestas que saem de u. Na implementação do Pseudocódigo 1 é mantida uma fila de prioridade mínima Q de vértices tendo como chaves

os valores de d (CORMEN et al., 2002).

Algoritmo 1: Algoritmo de Dijkstra

Entrada: G, w, s

Saída: d (Vetor com a distância de todos os vértices em relação a s), π (vetor com os antecessores)

```
1  início
2      para cada v ∈ V[G] faça
3          d[v] ← ∞
4          π[v] ← Nulo
5      fim
6      d[s] ← 0
7      S ← ∅
8      Q ← V[G]
9      enquanto Q ≠ ∅ faça
10         u ← Extrair-Minimo(Q)
11         S ← S ∪ {u}
12         para cada vértice v ∈ Adj[u] faça
13             Relaxar(u, v, w)
14         fim
15     fim
16 fim
17 retorna d, π
```

Algoritmo 2: Relaxar

Entrada: u, v, w

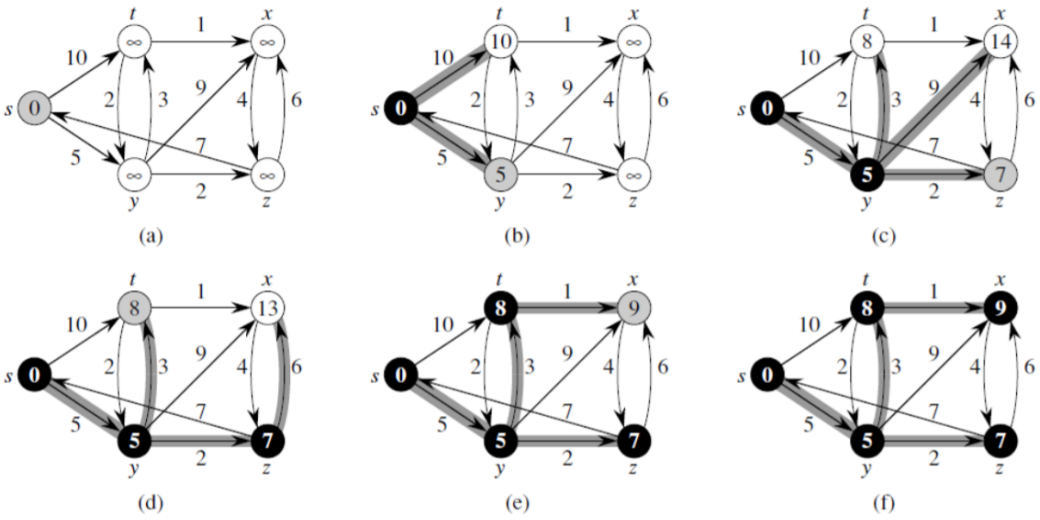
```
1  início
2      se d[v] > d[u] + w(u, v) então
3          d[v] ← d[u] + w(u, v)
4          π[v] ← u
5      fim
6  fim
```

Fazendo uma análise no Algoritmo 1 da linha 2 até a linha 6 é executada a inicialização dos valores de d e π, a linha 7 inicializa o conjunto S como conjunto vazio, linha 8 inicializa a fila de prioridade mínima Q para conter todos os vértices em V; em cada passagem pelo *loop* das linhas 9 a 15, um vértice u é extraído de $Q = V - S$ e inserido no conjunto S, mantendo o invariante. Em seguida as linhas 12 e 13 relaxam cada aresta (u, v) que saem de u como mostra o Algoritmo 2, atualizando assim a estimativa $d[v]$ e o predecessor $\pi[v]$ se o caminho mais curto até v pode ser melhorado

mediante a passagem por u (CORMEN et al., 2002).

Na Figura 18 apresentamos um exemplo de execução do algoritmo de *Dijkstra* onde a origem é o vértice s e as estimativas de caminhos mais curtos são mostradas dentro dos vértice, as arestas sombreadas indicam valores de predecessores, e os vértices pretos estão no conjunto S e vértices brancos estão na fila de prioridade mínima $Q = V - S$ (CORMEN et al., 2002). Na imagem (a) da Figura 18 as distâncias são inicializadas onde o vértice de início s tem valor 0 e os demais vértices recebem valores infinitos. Na imagem (b) o vértice s é retirado da fila de prioridades e são realizados relaxamentos nos vértices adjacentes a s, no próximo passo da imagem (c) como o vértice y possui o menor valor, então é removido da fila de prioridades e é realizado relaxamento nos vértices adjacentes, podemos observar que t foi atualizado pois possui um valor menor passando por y, na imagem (d) seguinte o vértice z é removido e é realizado um relaxamento no vértice adjacente x, que tem o valor atualizado, na imagem (e) o vértice t é removido da fila de prioridade e atualiza o valor do vértice adjacente x, no passo da imagem (f) o vértice x é removido e apresenta a solução para o grafo encontrando o menor caminho de s para os demais vértices.

Figura 18 – Exemplo de execução do algoritmo de Dijkstra.



Fonte – (CORMEN et al., 2002)

3 Trabalhos relacionados

As linhas do horizonte e da costa são fundamentais para segmentação do mar e identificação de banhistas em áreas de risco. Uma vez que as linhas são encontradas podemos desconsiderar o céu acima da linha do horizonte e a areia abaixo da linha da costa e focar apenas nas pessoas dentro da região formada por essas linhas.

Para linha da costa em imagens de praia não foram encontrados trabalhos na literatura. Foram encontrados vários trabalhos em linha da costa para imagem obtida pelo sensor *Synthetic Aperture Radar* (SAR) que é uma técnica de radar que permite imagem com alta resolução a grande distâncias em geral são imagens obtidas por aeronave ou satélite (DEMIR; KAYNARCA; OY, 2016).

Bing *et al.* (BING et al., 2016) apresentam um método de encontrar a linha do horizonte em imagens de montanhas baseado na fusão de detector de bordas e gradiente de adjacência. Ele tem como entrada uma imagem, na qual são aplicadas algumas operações morfológicas de processamento de imagens, como abertura, fechamento, erosão e dilatação. O operador *Canny* é aplicado para detectar as bordas na imagem, o terceiro passo do método encontra o ponto inicial na primeira coluna da imagem que é encontrada combinando o gradiente local e as informações da borda. O último passo do método aplica um algoritmo de detecção de linha do céu, onde a partir do ponto inicial o algoritmo vai avaliando os *pixels* adjacentes e formando o caminho da linha do horizonte até chegar na última coluna. Na aplicação desse algoritmo para detecção da linha do horizonte em imagens de praia, não foram obtidos bons resultados devido ao fato que no passo de encontrar o ponto inicial na primeira coluna nem sempre o *pixel* da linha do horizonte tem a força do gradiente maior que os demais, obtendo assim muitas vezes ponto de partida que não corresponde a linha do horizonte. Um outro problema são oclusões pois a linha do horizonte algumas vezes é interrompida por árvores impossibilitando o algoritmo de continuar a busca da linha do horizonte.

O método abordado por Ma e Ma (MA; MA, 2016) para detecção da linha do horizonte em imagens oceânicas e baseado em um algoritmo de detecção de segmentos de linha e na transformada de *Hough*. O objetivo da detecção do segmento de linha é encontrar as regiões candidatas a linha que separa o céu e o mar agrupando os *pixels* com o mesmo ângulo do gradiente e encontra o segmento que se aproxima de cada região de suporte de linha, então utiliza a transformada de *Hough* para extrair a posição precisa da linha horizonte. Esse método é preciso e rápido em algumas imagens de praia não seria possível aplicar essa metodologia para detecção da linha da costa, dada a existência de curvas e irregularidades, visto que esse método busca linhas

retas.

Foram analisadas técnicas de detecção de linhas do horizonte e segmentação do céu, várias técnicas encontradas fazem uso dos algoritmos de aprendizagem de máquina, onde cada *pixel* da imagem é classificado em céu e não-céu, gerando uma imagem levando em consideração a probabilidade de um *pixel* pertencer a uma determinada classe. Esse método é usado por Yazdanpanah *et al*. (YAZDANPANAH et al., 2015) onde é feito a fusão de uma classificador com um algoritmo de agrupamento, para extrair as características é usado o descritor de textura *Gray-Level Co-Occurrence Matrix* (GLCM). Essa abordagem possui um custo computacional muito alto devido a extração e a classificação de cada *pixel*, tornando inviável para aplicar em um sistema em tempo real e com baixo custo computacional.

Ahmad *et al.* (AHMAD et al., 2015) apresentam uma abordagem para detectar linha do horizonte em imagens de cenários da natureza como montanhas. Essa abordagem se baseia na ideia de atribuir um *score* para cada *pixel*, e pode ser representada como a probabilidade de um *pixel* pertencer à linha do horizonte. Um grafo é usado para representa o mapa de classificação. Para classificar cada *pixel* foi utilizado um classificador supervisionado: *Support Vector Machine* (SVM) e o Rede Neural Convolucional (CNN). Cada *pixel* foi rotulado em duas classes: horizonte e não-horizonte. No treinamento a linha do horizonte foi rotulada manualmente e para extrair a característica foi usada uma janela 16x16 ao redor do *pixel* e a partir dessa janela é extraído um vetor de intensidade do *pixel* normalizado entre -1 e 1 que serve de entrada para o classificador. O *Dense Classifier Score Image* (DCSI) é gerado como resultado da classificação de uma imagem de teste dada, onde o *score* da classificação está associado ao *pixel*. Uma imagem é gerada a partir dos valores do *score*, no qual pode ser interpretada como o mapa da probabilidade de um *pixel* pertencer a linha do horizonte. Com o DCSI gerado é aplicado outro procedimento chamado mDCSI (*Reduced Dense Classifier Score Image*), onde apenas os m melhores *scores* de cada coluna é mantido, pois os maiores *scores* estão em pequena faixa ao redor da linha do horizonte. Com o resultado do mDCSI é gerado um grafo multi-estágio. A detecção da linha do horizonte é interpretada como um problema do caminho mais curto, e esse caminho é encontrado utilizando o algoritmo de *Dijkstra*. Esse método pode ser aplicado no nosso problema para encontrar a linha do horizonte. Como esse algoritmo utiliza o grafo multi-estágios denso, onde cada *pixel* possui três arestas para os *pixels* adjacentes do estágio (coluna) seguinte, possibilita encontrar a linha do horizonte mesmo desconectada por oclusões, pois sempre há caminho mesmo apesar do custo alto.

Outro trabalho de Ahmad *et al*. (AHMAD et al., 2015), apresenta um método para encontrar a linha do horizonte baseado no acoplamento entre programação dinâmica e aprendizagem de máquina. Nesse trabalho ele compara vários descritores

de textura e também suas combinações, tais como: SIFT, LBP, HOG, SIFT-HOG e SIFT-LBP-HOG, onde esses descritores são usados para treinar um classificador de borda horizontal. Também foram analisados vários custos nodais para a programação dinâmica. Para a extração dos recursos é utilizado uma janela (16X16) ao redor de cada *pixel*. Ele funciona tendo uma imagem como entrada onde é aplicado o detector de bordas *Canny* e são mantidas apenas as bordas dentro de um intervalo de limiares. Nas bordas restantes é aplicado o classificador *Support Vector Machine* (SVM) para reduzir o número de bordas, classificando em horizontais e não-horizontais. Nas arestas classificadas como horizontais é aplicado à programação dinâmica para extrair a linha do horizonte.

O trabalho de Verbickas e Whitehead (2014) apresenta um método para detecção do céu e da terra usando redes neurais convolutivas (CNN) para o cenário obtido através de câmera de veículos aéreo autônomo, onde cada *pixel* na imagem pode ser classificado como céu ou terra. Para treinamento da CNN, é criada uma máscara da imagem original. Essa máscara é uma imagem binária que possui a mesma dimensão da imagem original onde os *pixels* brancos representam o céu e os *pixels* pretos representam a terra. Neste trabalho a rede neural convolutiva é responsável por classificar a imagem, onde a saída é uma imagem binária que representa a segmentação do céu e o que não é céu.

Gershikov (GERSHIKOV, 2014) apresenta uma avaliação de métodos de detecção de linha do horizonte para analisar se a cor é importante para detecção da linha. Ele avalia dois grupos de abordagem: uma baseada em métodos que usa todos os componentes da cor e outro que usa somente um dos componentes e descarta os demais, e tem como objetivo verificar o impacto de usar as informações das cores. Os métodos baseados em tons de cinza, que são métodos baseados na luminância, são divididos em duas categorias: métodos que usam recurso local, como valores de bordas, valores da média local e gradiente, e métodos que usam recurso global tais como estatística regional. Alguns métodos que usam as cores foram os que tiveram melhores resultados, o uso desse método é justificado para situações onde a confiabilidade é fundamental, já os métodos baseados em intensidades tiveram resultados muito próximos dos descritos anteriormente, suas vantagens são a simplicidade, a menor complexidade e que não há necessidade de imagem em cores.

Ahmad *et al.* (AHMAD et al., 2014) apresentam uma avaliação experimental de diferentes descritores e custos para o nó para detecção da linha do horizonte de um conjunto de imagens de montanhas, onde nesse trabalho, eles exploram diversos descritores de textura local e suas combinações para reduzir o número de classificações falsas. Os descritores explorados no trabalho foram: *Scale Invariant Feature Transform* (SIFT), *Local Binary Patterns* (LBP), *Histogram of Oriented Gradients* (HOG) bem como

suas combinações SIFT-LBP, SIFT-HOG, LBP-HOG, SIFT-LBP-HOG, e a combinação é realizada concatenando os vetores de descritores. Os três descritores de textura foram escolhidos como opções de descritores para treinar o classificador *Support vector machine* (SVM), que classifica um *pixel* com valores entre [0-1], representando a probabilidade de um *pixel* pertencer à linha do horizonte. Para encontrar a linha do horizonte é usada a programação dinâmica aplicada ao grafo gerado pela classificação.

O trabalho de Gershikov *et al*. (GERSHIKOV TZVIKA LIBE, 2013), apresenta e avalia cinco métodos para detectar linhas do horizonte em imagens marinhas. O primeiro método é um algoritmo baseado na covariância regional da luminância da imagem, onde o algoritmo recebe uma imagem como entrada e procura uma partição ideal da imagem em duas regiões céu e mar, em que o critério de otimização é baseado nos determinantes e na matriz de covariância das duas regiões. O segundo algoritmo é baseado na detecção de bordas e na transformada de *Hough*, esse algoritmo funciona da seguinte forma: primeiro aplica uma operação morfológica de erosão para reduzir a probabilidade de encontrar bordas fracas, aplica o operador *Canny* para detectar as bordas, aplicar a transformada de *Hough* para encontrar as linhas e o último passo é encontrar a linha mais longa. A terceira abordagem é a detecção de bordas e algoritmo baseado em calibração de mínimos quadrados, usa pré-processamento, detecção de borda local máxima e calibração pela abordagem do método de mínimos quadrados. Na quarta abordagem é a filtragem mediana e algoritmo baseado em regressão linear, esse método emprega filtro mediano de vários estágios, ele procura pela borda máxima na direção vertical com base na vizinhança estendida. A quinta abordagem é o de arestas regionais de magnitudes e algoritmos baseados em calibração de mínimos quadrados, divide a imagem em faixas verticais e procura a magnitude da borda regional máxima em cada faixa, seguida pela técnica de mínimos quadrados para estimar a melhor linha que passa pelas coordenadas máximas da borda.

Lie *et al*. (LIE et al., 2005) propõe um método para detecção da linha do horizonte em imagens de montanhas baseado em grafo multi-estágios e programação dinâmica, onde ele parte da suposição que a linha do horizonte é completa, isso é, ela se estende da esquerda para direita da imagem, com isso o algoritmo de programação dinâmica encontra o caminho mais curto da esquerda para direita, as bordas encontradas a partir da imagem original é a entrada do algoritmo do grafo multi-estágios, onde os *pixels* brancos em uma determinada coluna possuem uma aresta com um custo ligado a outros *pixels* brancos na coluna seguinte, é um algoritmo robusto que pode ser aplicado tanto na detecção da linha do horizonte quanto da detecção da linha da costa. Como o grafo multi-estágio apresenta um custo computacional alto para geração do grafo, pois esse algoritmo implementa um algoritmo para expandir a região de buscar caso não tenha *pixels* borda no estágio buscado.

Os trabalhos relacionados descritos anteriormente apresentam aplicabilidade específica a um determinado cenário, tais como: encontrar linha do céu em imagens oceânicas (MA; MA, 2016), encontrar linha do horizonte em imagens montanhosas (YAZDANPANAH et al., 2015) (AHMAD et al., 2015) (AHMAD et al., 2015) (BING et al., 2016). Apesar dos algoritmos que fazem usos de algoritmos de aprendizagem de máquina apresentarem bons resultados de acertos, possui um custo computacional muito elevado, visto também que os algoritmos de grafos possuem alto custo computacionais para geração do grafo, alguns algoritmos fazem usos desses dois métodos, elevando muito o custo computacional.

4 Detecção de linha do horizonte e linha da costa

Neste Capítulo serão apresentadas as descrições dos algoritmos utilizados nos experimentos. Ao todo foram analisados sete algoritmos, quatro para detecção da linha do horizonte e três para detecção da linha da costa, dentre os quais alguns da literatura e outros são contribuições desse projeto, na Tabela 1 podemos ver os algoritmos que serão abordados.

Tabela 1 – Visualização dos algoritmos abordados nesse trabalho

	Linha do Horizonte	**Linha da Costa**
Estado da Arte	DCSI (AHMAD et al., 2015)	ASPC (CARRERA, 2017)
	Lie *et al.* (LIE et al., 2005)	
Abordagens Propostas	DLHCGME	DLCCCH
	DLHSTH	DLCCGME

4.1 Algoritmos para detecção de linha do horizonte

A detecção da linha do horizonte em imagens de praia é encontrar a linha que separa o céu do mar, geralmente é uma linha reta que se estende da esquerda para direita da imagem. Encontrar essa linha é uma tarefa complexa devido a variedade de tons de cores que o céu e o mar podem apresentar, em alguns momentos muito claro, em outros escuro e outras vezes tons muito próximos entre si.

4.1.1 Algoritmo Lie *et al.*

O algoritmo Lie *et al.* (LIE et al., 2005) formula o problema de detecção de linha do horizonte como um problema de grafo multiestágios e usa programação dinâmica para encontrar menor caminho que se estende da esquerda para direita. Dado uma imagem de tamanho $M \times N$, o primeiro passo é computar o mapa binário de bordas usando algum operador.

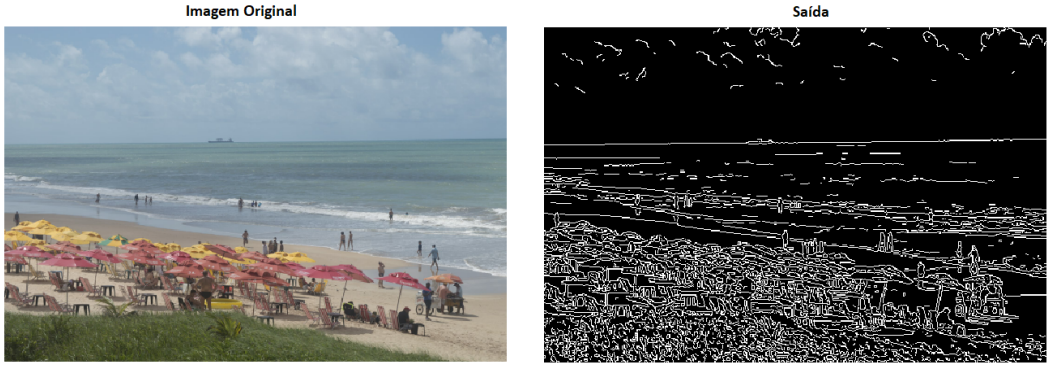
4.1.1.1 Detecção de bordas

Para aplicar esse algoritmo ao problema de detecção da linha do horizonte, várias técnicas de detecção de bordas foram aplicadas para destacar a linha do horizonte.

Tendo uma imagem em tons de cinza como entrada, a mesma é suavizada com um filtro gaussiano para eliminar pequenas bordas, logo em seguida o operador *Canny*

é aplicado para obter o mapa de bordas. Podemos ver a saída dessa etapa na Figura 19.

Figura 19 – Exemplo de detecção de bordas



Fonte – O autor

4.1.1.2 Grafo multiestágios

O mapa de bordas binarizado é usado para construir um grafo multiestágios $G(V, E, \Psi, \Phi)$ de tamanho $M \times N$, as etapas para construção do grafo está descrita logo abaixo.

1. Cada ponto b_{ij} no mapa de bordas corresponde a um vértice $v_{ij} \in V$ no estágio $j-th$ do grafo, cada vértice v_{ij} pode ser atribuído como vértice de borda se $b_{ij} = 1$ e um não *pixel* de borda se $b_{ij} = 0$.

2. Dois vértices virtuais s e t, são adicionados no início e no fim respectivamente. O s representa o estágio $0th$ e o t o estágio $(N + 1)th$. Na Figura 20 podemos ver os vértices s e t.

3. Uma aresta é estabelecida entre dois vértices de bordas adjacentes de estágios consecutivos.

4. Cada vértice v_{ij} está associado uma função de custo $\Psi(ij)$, definida pela Equação 4.1 abaixo.

$$\Psi(i,j) = \begin{cases} (i+1)^2, & j = 1 \; ou \; N, \\ 0, & seno. \end{cases} \tag{4.1}$$

Essa função de custo do vértice, é diretamente proporcional ao valor de i em $\Psi(ij)$, esse custo é aplicado somente ao estágio 1 e N, pois esse algoritmo parte da suposição que a linha do horizonte está na parte superior da imagem.

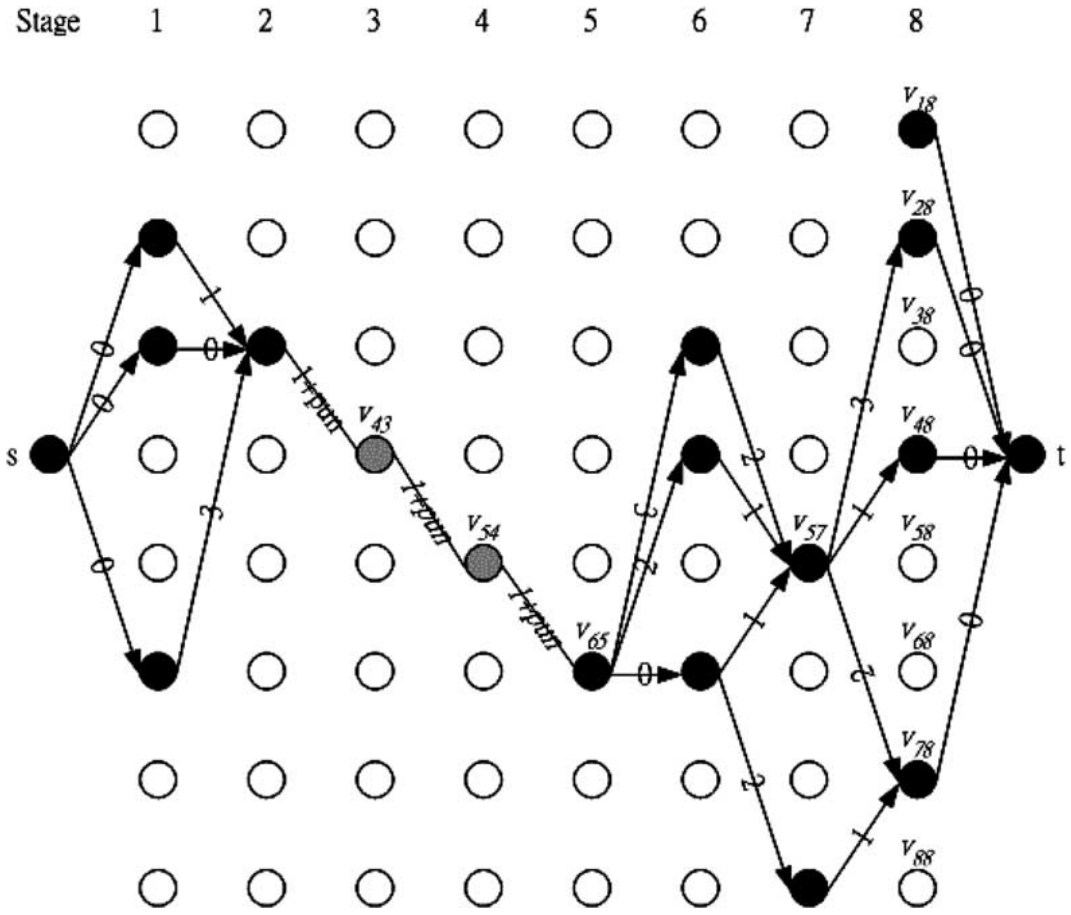
5. Cada aresta $l_{h,k,j}$ do grafo multiestágios está associado a uma função custo $\Phi(h,k,j)$, representada na Equação 4.2, onde h é a valor da linha do *pixel* no estágio j e k é o valor da linha do *pixel* de borda no estágio $j+1$.

$$\Phi(h,k,j) = \begin{cases} \mid h-k \mid & \text{se } (b_{h,j} = b_{k,j+1} = 1 \text{ e } \mid h-k \mid \leq \delta) \\ \infty, & \text{senão,} \end{cases} \tag{4.2}$$

   Na Função de custo 4.2 acima, o δ representa um limiar predefinido, onde dois vértices de borda de estágios consecutivos com uma distância vertical maior que δ não pode ser ligado, a escolha do valor de δ afeta a detecção de curvas íngremes da linha do horizonte.

6. Qualquer aresta conectada aos vértices s ou t, possui o valor da função $\Phi(h,k,j)$ igual a 0, sendo assim $\Phi(h,k,0) = \Phi(h,t,N) = 0$.

Figura 20 – Topologia do grafo multiestágios para um mapa de bordas 8×8

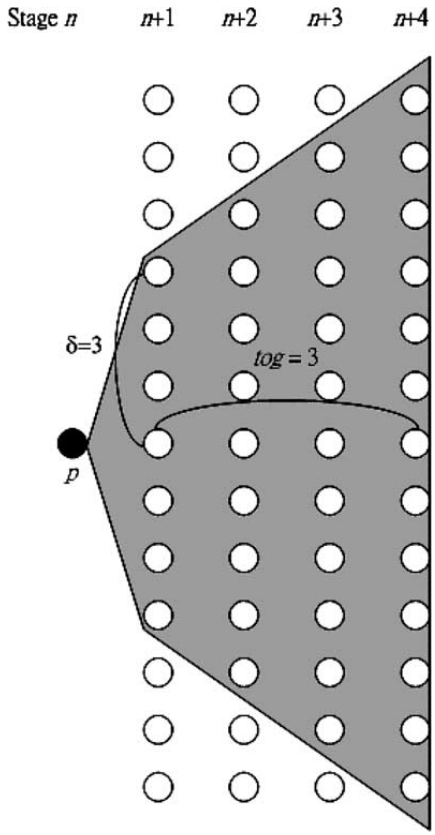


Fonte – (LIE et al., 2005)

4.1.1.3 Expansão de região

A extração de bordas nem sempre é perfeita deixando falhas nas bordas encontradas, se um vértice i no estágio j não poder ser conectado há nenhum vértice no estágio $j+1$ com vizinhança δ, para manter o algoritmo robusto uma tolerância razoável de falha (denominada como *tog*) é permitida uma expansão da região de pesquisa até o estágio j+*tog* pode ser realizada para preencher as lacunas como podemos ver na Figura 21:

Figura 21 – Expansão de região para um vértice p, que não possui vértices de bordas no estágio seguinte



Fonte – (LIE et al., 2005)

Essa etapa é muito útil para detecção de linha do horizonte em imagens de praia devido a oclusões que podem ocorrer, tais como folhas de árvore, coqueiro e entre outros.

4.1.1.4 Encontrar a linha do horizonte

Com o grafo multiestágios criado e os custos $\Psi(ij)$ e $\Phi(h,k,j)$ calculados, um algoritmo de de programação dinâmica é executado para encontrar o caminho de menor custo de s até t. Na Figura 22 podemos ver o resultado da detecção da linha do horizonte com o algoritmo de (LIE et al., 2005).

Figura 22 – Exemplos de detecção da linha do horizonte

Fonte – O autor

Na Figura 22 (b), podemos verificar que o algoritmo foi capaz de encontrar a linha do horizonte mesmo com oclusão por um coqueiro.

Algoritmo 3: Algoritmo Lie *et al.*

Entrada: I

Saída: $LinhaDetectada(I_l)$

início

- $corLinha \leftarrow (0, 0, 0)$
- $imagemCinza \leftarrow Cinza(I)$
- $bordas \leftarrow Canny(imagemCinza, minVal = 10, maxVal = 50)$
- $grafoMultiEstagio \leftarrow CriarGrafoMultiEstagioLie(bordas)$
- $linha[] \leftarrow Dijkstra(grafoMultiEstagio, s, t)$
- $I_l \leftarrow Copiar(I)$
- **para cada** $p \in linha$ **faça**
  - $I_i[p.linha][p.coluna] \leftarrow corLinha$
- **fim**

fim

retorna I_l

4.1.2 *Dense classifier score image* (DCSI)

O algoritmo DCSI é o algoritmo proposto por Ahmad *et al.* (AHMAD et al., 2015), que é uma algoritmo para detecção de linha do horizonte em imagens de montanhas. Ele é uma extensão do algoritmo de Lie *et al.* (LIE et al., 2005), mas ao invés de extrair a linha do horizonte de um mapa de bordas, ele extrai a partir de um mapa de classificação. A ideia chave o algoritmo é classificar cada *pixel* da imagem como pertencente a linha do horizonte ou não, esse mapa de classificação é chamado de *Dense Classifier Score Image* (DCSI). Essa classificação é realizada usando os *pixels*

que pertence a linha do horizonte e os que não pertence de um conjunto pequeno de treinamento. No mapa de classificação obtido cada *pixel* está associado a um *score* de classificação que fornece a informação sobre a probabilidade de um *pixel* pertencer à linha do horizonte.

Abaixo apresentamos as etapas desse algoritmo:

4.1.2.1 Classificação dos *pixels*

Para classificar uma imagem de entrada foi utilizado o classificador SVM (CORTES; VAPNIK, 1995). Ele foi treinado usando duas classes uma com *pixels* pertencentes a linha do horizonte e outras com *pixels* não pertencentes a linha do horizonte, extraídos de um conjunto de imagens de treinamento rotulado manualmente. Para cada imagem da base de treinamento foram escolhidas uma quantidade N de pontos (pixel) pertencentes a linha do horizonte e a mesma quantidade com pontos não pertencente a linha do horizonte. Para cada ponto é selecionado uma janela da imagem com tamanho 16×16 ao redor desse ponto. Essa janela com a intensidade do *pixel* é normalizada, resultando num vetor de característica de 256 atributos que são usados para treinar o classificador. Foram extraídos 300 pontos para linha do horizonte e 300 pontos para não-horizonte de 16 imagens do conjunto de treinamento. Para os experimentos desse projeto foi utilizado o classificador SVM com *kernel* linear. Após o classificador ter sido treinado o DCSI pode ser gerado para uma imagem de teste. Na Figura 23 podemos ver o resultado da classificação da linha do horizonte para uma imagem de praia formando o DCSI.

Figura 23 – Exemplos de classificação da linha do horizonte para uma imagem de praia.

Fonte – O autor

4.1.2.2 *Reduced dense classifier score image* (mDCSI)

Com o intuído de aumentar a velocidade algoritmo o mDCSI mantém apenas os m maiores *scores* por coluna. Como consequência o caminho para percorrer é menor sem prejudicar a precisão da detecção da linha do horizonte, na Figura 24 abaixo podemos ver o resultado do mDCSI.

Figura 24 – mDCSI aplicado na imagem da Figura 23.

Fonte – O autor

4.1.2.3 Detecção da linha do horizonte como um problema de caminho mais curto

Para detectar a linha do horizonte a partir do mDCSI é utilizado uma abordagem semelhante a de Lie *et al.* (LIE et al., 2005), onde a partir do mDCSI ele cria uma grafo multiestágios $M \times N$ dado por $G(V, E, \Psi, \Phi)$. Como essa abordagem não utiliza mapa de bordas não precisa se preocupar com preenchimento de lacunas, pois todo o grafo está conectado. O custo de cada nó inicializado é dado por $mD(x, y)$ como pode-se ver na Equação 4.3, onde o mD corresponde ao mDCSI.

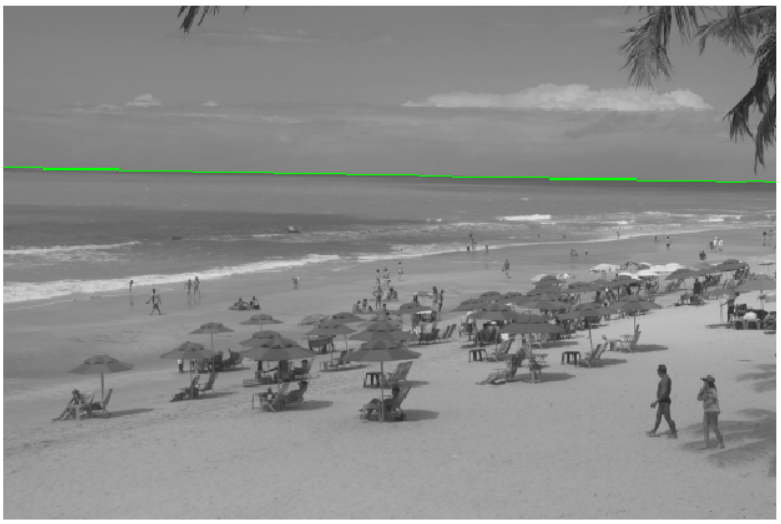
$$\Psi(i, j) = mD(x, y) \tag{4.3}$$

O custo da aresta é dado levando em consideração que cada vértices i em um estágio (coluna) j está conectado a três vértices no estágio $j+1$ o i, $i-1$ e $i+1$, essas arestas são definidas com custo zero como mostra a Equação 4.4 abaixo.

$$\Phi(i, k, j) = \begin{cases} 0 & |i - k| \leq \delta) \\ \infty, & \text{senão.} \end{cases} \tag{4.4}$$

Assim como o algoritmo de Lie *et al.* (LIE et al., 2005) são definidos dois vértices s e t, eles são introduzidos à esquerda do primeiro estágio e a direita do estágio N, respectivamente. Sendo que o custo da aresta s para o primeiro estágio e do estágio N para t possuem peso zero. O caminho mais curto desse grafo pode ser encontrado usando o algoritmo de Dijkstra. Na Figura 25, mostra o resultado da detecção da linha do horizonte para uma imagem de praia.

Figura 25 – Exemplo de detecção da linha do horizonte.



Fonte – O autor

Algoritmo 4: Algoritmo DCSI

Entrada: I
Saída: $LinhaDetectada(I_l)$

início
- $corLinha \leftarrow (0, 0, 0)$
- $imagemCinza \leftarrow Cinza(I)$
- $imagemDCSI \leftarrow DCSI(imagemCinza)$
- $imagemMDCSI \leftarrow mDCSI(imagemDCSI)$
- $grafoMultiEstagio \leftarrow CriarGrafoMultiEstagio(imagemMDCSI)$
- $linha[] \leftarrow Dijkstra(grafoMultiEstagio, s, t)$
- $I_l \leftarrow Copia(I)$
- **para cada** $p \in linha$ **faça**
  - $I_i[p.linha][p.coluna] \leftarrow corLinha$
- **fim**

fim

retorna I_l

4.1.3 DLHSTH

O DLHSTH é um método para detecção da linha do horizonte em imagens de praia, esse método é uma extensão do algoritmo *Edge detection and Hough transform based algorithm* (H-HC) de Gershikov *et al.* (GERSHIKOV TZVIKA LIBE, 2013), visando obter um melhor resultado na detecção da linha do horizonte para imagens de praia, é proposto uma nova abordagem. Como imagens de praia contém mais elementos que imagens marítima (céu e mar) se tornar difícil encontrar um conjunto de parâmetros que se ajuste perfeitamente a todas as imagens da base de dados. Muitas vezes em alguns experimentos um conjunto de parâmetro se ajustava a uma determinada quantidade de imagens com valores ótimo e com valores péssimos para as demais imagens da base de dados. A transformada de *Hough* pode encontrar milhares de linhas e algumas dessas linhas com maiores acumuladores da transformada de *Hough* que a linha do horizonte. Visto isso foram realizados vários experimentos com sistemas de cores e técnicas de limiarização para encontrar os parâmetros que salientasse a linha do horizonte para reduzir o número de linhas detectadas e que essas linhas encontradas se aproxime da linha do horizonte. Um dos experimento foi extrair as bordas da imagem em tons de cinzas. No entanto ele encontrar um número muito grande de bordas e nem sempre era encontrado a borda da linha do horizonte favorecendo para uma taxa erro maior quando aplicado a base de dados. Após esses experimentos foi constatado que o canal *value* do sistema de cores HSV foi o que obteve o melhor resultado uma vez que o brilho dar do céu é diferente do mar, obtendo um contraste maior para linha do horizonte que os demais opções dos experimentos. Para detecção de bordas necessária para entrada do detector de linha, foram analisados vários operadores mas o que apresentou melhores resultados foi o *Sobel* no eixo Y que detecta bordas horizontais, e visando melhorar a acurácia foram feitos testes com vários filtros morfológicos, o filtro morfológico de erosão mostrou melhores taxas de acertos, pois deixou as bordas obtida pelo *Sobel* mais fina fazendo a linha detectada chegar mais próxima da linha do horizonte.

4.1.3.1 Detecção de bordas

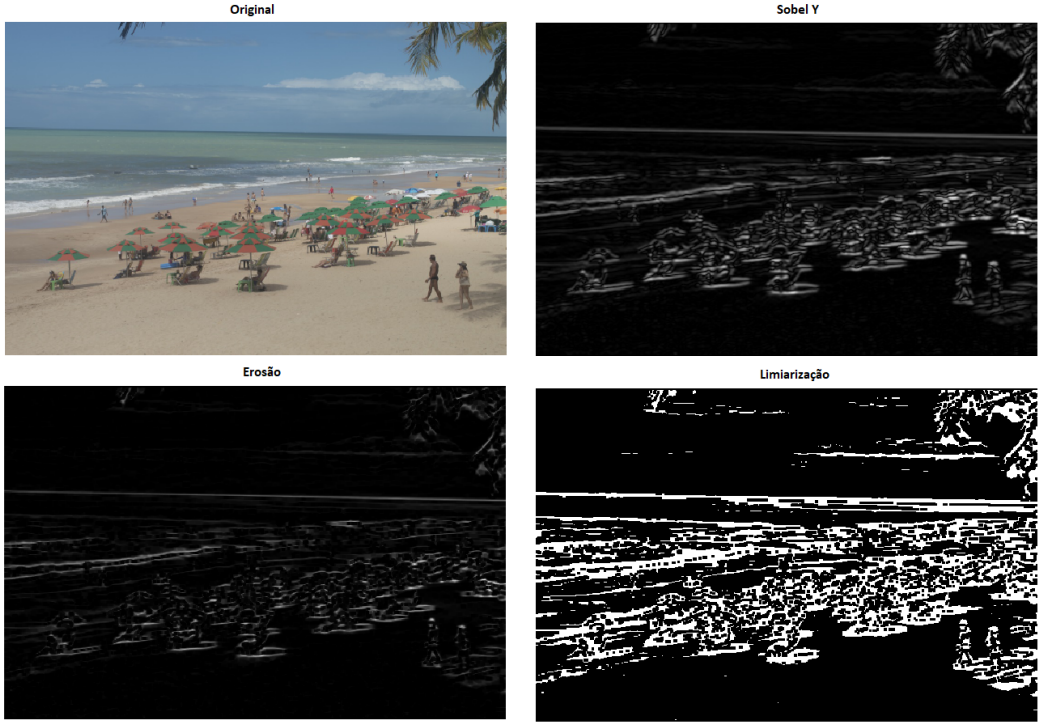
Nesta etapa, a imagem de entrada é convertida para o sistema de cor HSV, depois é suavizada com um filtro gaussiano 3×3 para redução de ruídos, e consequentemente bordas falsas, e em seguida o operador de *Sobel* no eixo Y é aplicado no canal V (*value*) do sistema de cor HSV. Depois com o intuito de destacar as bordas horizontais, é aplicado um filtro morfológico de erosão nas bordas obtidas pelo *Sobel* para obter uma borda mais fina. Outra etapa é a limiarização descrito pela Equação 4.5 o intuito de binarizar as bordas obtida pelo *Sobel*, os parâmetros utilizados foram obtidos através de vários experimentos tando para o filtro gaussiano quando para o

operador *Sobek*.

$$g(x,y) = \begin{cases} 0, & Se\ 190 \leq f(x,y) \\ 255, & \text{senão.} \end{cases} \tag{4.5}$$

Na Figura 26, podemos algumas dos passos descrito acima.

Figura 26 – Exemplo de detecção de bordas.

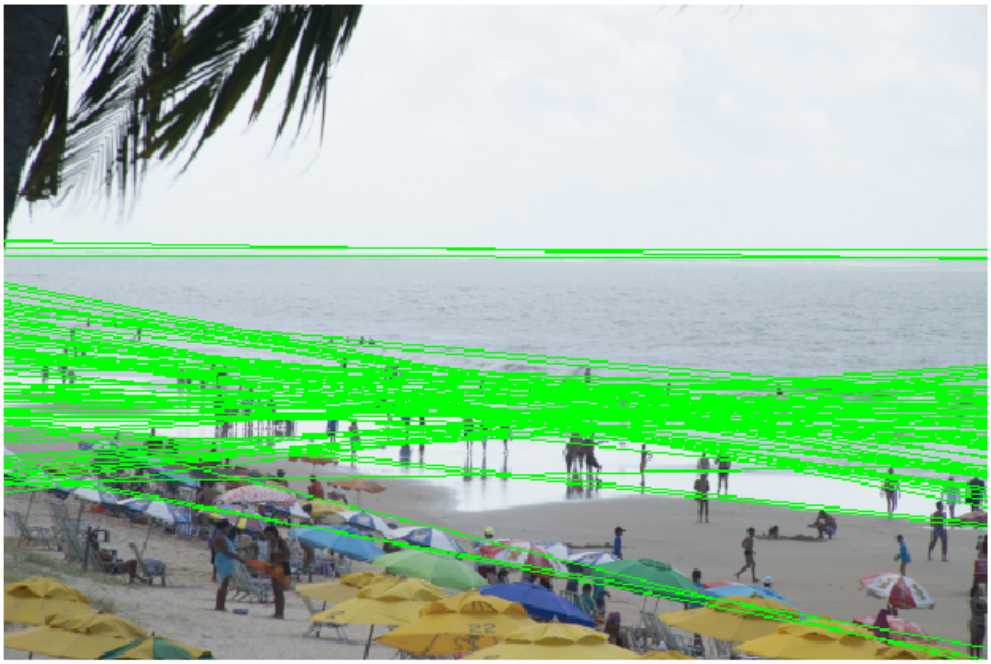


Fonte – O autor

4.1.3.2 Encontrar a linha do horizonte

Para detectar a linha da horizonte essa abordagem utiliza a transformada de *Hough* para detecção de linhas retas a partir de bordas onde quando mais *pixels* colineares uma borda tiver maior vai ser o acumulador da transformada, e como a borda da linha do horizonte tende a serem colineares e consequentemente tem a maior probabilidade de acumuladores maiores, algumas bordas tende a terem acumuladores próximo ou maiores que o da linha do horizonte, é o caso da borda formada pelas ondas, que muitas vezes forma bordas com colineares que se estende da esquerda para direita da imagem, na Figura 27 abaixo podemos observar a detecção de linhas usando a transformada de *Hough*, onde são setados parâmetros que para encontrar centenas de linhas a partir do mapa de bordas, e que posteriormente será selecionada a linha com características mais próxima das características da linha do horizonte.

Figura 27 – Exemplo de detecção de linhas usando a transformada de *Hough*.



Fonte – O autor

Figura 28 – Exemplo de detecção da linha do horizonte usando o DLHSTH.

Fonte – O autor

Como visto na Figura 27 a transformada de *Hough* é aplicado na imagem binari-

zada, obtendo diversas linhas, onde podemos ver que são encontradas algumas linhas que mais se aproxima da linha do horizonte, os parâmetros foram escolhidos para obter centenas de linhas. Para isso é necessário escolher entre elas qual se aproxima mais da linha do horizonte, entretanto para encontrar essas linhas foi utilizado a função $cv2.HoughLines$ do *Opencv* (OPENCV, 2015), o qual retorna uma lista contendo as linhas encontradas ranqueada em ordem decrescente pelo valor do acumulador, onde as linhas com maiores acumuladores estão no começo da lista. Para cada linha é retornado dois valores *theta* e *rho* em que o primeiro é o angulo formado pela linha em radianos e o segundo é a normal formada pela posição do primeiro *pixel* da imagem com a reta e esse valor é dado em pixels. A linha escolhida leva em consideração a inclinação da reta(valor de *theta*) o valor do acumulador, pois linha do horizonte deve ter acumulador alto, e a aproximação com a borda superior, como podemos observar na Figura 27 não foram encontradas linhas a região do céu para essa imagem, mas há caso em que as nuvens criam bordas suficientes para o detector encontrar linhas, mas essas linhas contem valores de acumuladores baixo possibilitando encontrar a linha do horizonte corretamente como mostra a Figura 28.

Algoritmo 5: Algoritmo DLHSTH

Entrada: I

Saída: $LinhaDetectada(I_l)$

início

$corLinha \leftarrow (0, 0, 0)$

$imagemHsv \leftarrow HSV(I)$

$imagemHsvSuavisada \leftarrow FiltroGaussiano(imagemHsv, (3 \times 3), 0)$

$imagemCanalV \leftarrow ExtrairCanalV(imagemHsvSuavisada)$

$imagemEqualizada \leftarrow EqualizarHistograma(imagemCanalV)$

$sobely \leftarrow Sobel(imagemEqualizada)$

$gradiente \leftarrow Gradiente(sobelY)$

$bordas \leftarrow Canny(gradiente, minVal = 90, maxVal = 100)$

$linhas[] \leftarrow DetectorHough(bordas)$

$linhasMaioresAcumuladores \leftarrow linhas[0 : 6]$

$indice \leftarrow LinhaMaisProximaBordaSuperior(linhasMaioresAcumuladores)$

$I_l \leftarrow Copia(I)$

$DesenhaLinha(I_l, linhasMaioresAcumuladores[indice])$

fim

retorna I_l

4.1.4 DLHCGME

Método para detecção da linha do horizonte de imagens de praia, baseado na abordagem de grafo multiestágios apresentado em Lie et al. (2005) e Ahmad et al. (2015), porém diferente desse último por não utiliza o DCSI como entrada e sim uma imagem com as bordas obtida por um detector de bordas. Uma das principais características desse algoritmo é a utilização do sistema de cores HSV, onde é possível obter uma maior diferença do mar e céu, em relação ao ao tons de cinza. O uso de grafo para detecção de linhas do horizonte inicialmente foi proposto por Lie *et al.* em (LIE et al., 2005), onde ele utiliza grafo multi-estágio para representar o mapa de bordas de uma imagem, onde os *pixels* são os vértices e dois *pixels* adjacentes de borda representa um aresta no grafo, e a partir do grafo gerado utiliza o algoritmo de encontrar o caminho de menor custo em grafo ponderado. A utilização de grafo para detecção linha do horizonte partir do princípio que a linha do horizonte se estende da esquerda para direita da imagem, A Figura 29 ilustra a um exemplo que não poderia se aplicar a esse algoritmo, pois não existe na imagem a linha que divide o mar e céu que estende da esquerda para direita.

Figura 29 – Exemplo de imagem que não pode se aplicado o DLHCGME.

Fonte – O autor

Como a linha do horizonte se estende da esquerda para direita e geralmente é totalmente encontrada pelo detector de bordas, a estratégia do caminho de menor custo aplicado ao grafo multi-estágio denso na maioria dos casos resulta na linha do horizonte, e para aumentar a chance de encontrar a linha do horizonte foram feitos

vários experimentos envolvendo a adição de peso as arestas Ahmad *et al.* formula o peso do nó como a probabilidade do *pixel* pertencer a linha do horizonte obtido do algoritmo de classificação, diferente desse, como nossa abordagem utiliza o mapa de borda binaria o peso da aresta é dado através da soma da intensidade dos pixels. Como a linha do horizonte se encontra na parte superior se aproximando da borda superior da imagem, é adicionado peso as arestas que liga o primeiro estágio ao segundo, onde o peso é proporcional a posição da linha do *pixel* aumentando a probabilidade de encontrar a linha do horizonte. Abaixo será apresentado as etapas desse método:

4.1.4.1 Detecção das bordas

O objetivo dessa abordagem é um algoritmo tão preciso quanto o abordado em (AHMAD et al., 2015) mas com um tempo de processamento bem inferior e que não necessitasse de treinamento para detecção da linha do horizonte em imagens de praia. A partir daí foram realizado vários experimentos a fim de encontrar os melhores parâmetros que melhor destacasse a borda que forma a linha do horizonte. Ao usar um detector de bordas para o problema de detecção da linha do horizonte, ele deve ser capaz de encontrar a borda que representa a linha do horizonte nas imagens de praia avaliadas neste projeto.

Os passos dessa etapa são:

1. Converte a imagem de entrada do RGB para o HSV.
2. Aplica um filtro gaussiano 3×3, para reduzir ruídos e bordas falsas.
3. O algoritmo *Canny* é aplicado no canal V do HSV, pois o céu possui o tom do brilho maior que o do mar, obtendo um contraste maior entre as duas regiões, facilitando que o operador *Canny* encontre essas bordas.
4. Após obter o mapa de bordas, o mesmo é invertido para entrada do grafo multi-estágios.

Nas Figuras 30 e 31 podemos observar o resultado dessa etapa.

Figura 30 – Etapas da detecção das bordas.

Fonte – O autor

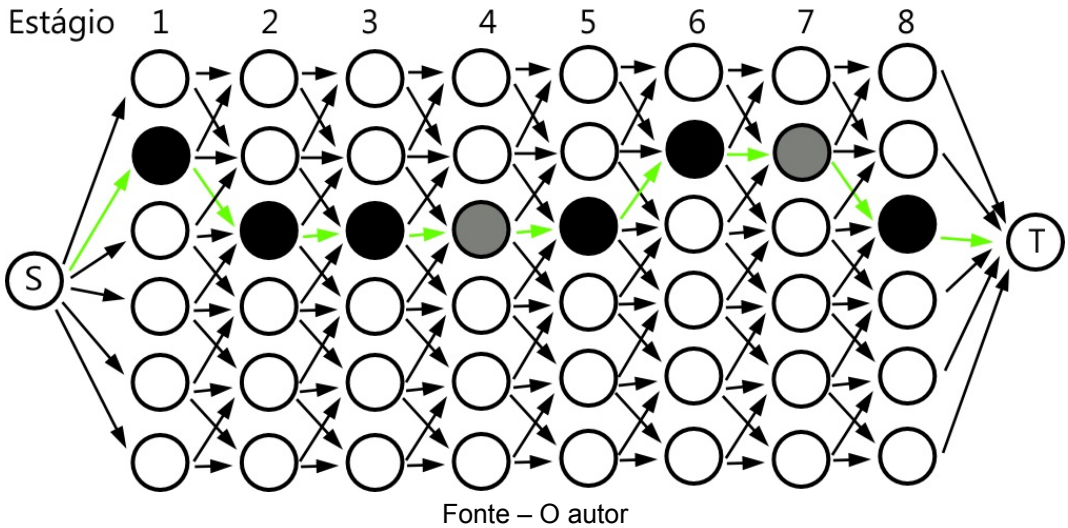
Figura 31 – Etapas da detecção das bordas.

Fonte – O autor

4.1.4.2 Encontrar a linha do horizonte

A detecção da linha do horizonte a partir do mapa de bordas gerado pelo operador *Canny*, é resolvido como um problema de grafo multiestágios como apresentado por (LIE et al., 2005) e utilizando a ideia em (AHMAD et al., 2015) de grafo multiestágios denso, onde todos os *pixels* estão conectados aos três *pixels* do estágio seguinte. Com isso não há necessidade de utilizar um passo para preenchimento de lacunas como abordado em (LIE et al., 2005). Nessa abordagem o mapa de bordas invertido é utilizado para criar um grafo multiestágios $M \times N$ dado por $G(V, E, \Phi)$, onde cada *pixel* é um vértice no grafo. Na Figura 32 podemos ver um exemplo de um grafo multiestágios 8×6, e um caminho encontrado usando um algoritmo para busca pelo menor caminho, do vértice (s) até o vértice (t), que são vértices virtuais adicionados antes do estágio 1 e no estágio $N + 1$, respectivamente. Mesmo com lacunas como mostra a Figura 32, os vértices em cinza não são *pixels* de bordas, mas através deles é possível encontrar o caminho mesmo que tenha um custo mais alto, pois essa é a característica desse grafo multiestágios denso.

Figura 32 – Topologia do grafo multiestágios para um mapa de borda 8×6.



Fonte – O autor

Como o podemos notar na Figura 32 cada nó i no estágio (coluna) j está conectado a três nós, i, $i-1$ e o $i+1$ no estágio $j+1$, tendo assim o valor de $\delta = 1$ que representa a distância vertical de vértice no estágio i ao vértice do estágio seguinte. O custo das arestas Φ é dado pela Equação 4.6, em que m representa o mapa de bordas da imagem. Como a imagem é um mapa de bordas invertido os valores dos *pixels* de bordas são 0 e os não bordas são 255, portanto uma aresta ligando dois vértices de bordas terá um custo $\Phi = 0$.

$$\Phi(i,k,j) = m(i,j) + m(k,j+1) \tag{4.6}$$

O custo das arestas do vértice (s) para os vértices do estágio 1 e dos vértices do estágio N até o vértice (t), é dado por $(i+1)^2$, para poder encontrar a caminho mais curto e mais próximo da borda superior, a medida que as bordas vão se aproximando da margem inferior da imagem os custos dessas arestas são maiores. Para encontrar o caminho de custo mínimo no grafo, é utilizado um algoritmo de encontrar caminhos mínimo, que pra esse trabalho foi utilizado o algoritmo de *Dijkstra*, que possibilita encontrar o caminho mais curto em um grafo de arestas com pesos positivos. Na Figura 33 temos um exemplo da detecção da linha do horizonte utilizando esse algoritmo, onde o algoritmo foi capaz de encontrar uma linha (em verde) que se aproxima da linha do horizonte com uma alta precisão.

Figura 33 – Exemplo de detecção da linha do horizonte utilizando o DLHCGME.

Fonte – O autor

Algoritmo 6: Algoritmo DLHCGME

Entrada: I

Saída: $LinhaDetectada(I_l)$

início

$corLinha \leftarrow (0, 0, 0)$

$imagemHsv \leftarrow HSV(I)$

$imagemHsvSuavisada \leftarrow FiltroGaussiano(imagemHsv, (3 \times 3), 0)$

$imagemCanalV \leftarrow ExtrairCanalV(imagemHsvSuavisada)$

$bordas \leftarrow Canny(imagemCanalV, minVal = 10, maxVal = 50)$

$bordasInv \leftarrow NOT(bordas)$

$grafoMultiEstagio \leftarrow CriarGrafoMultiEstagio(bordasInv)$

$linha[] \leftarrow Dijkstra(grafoMultiEstagio, s, t)$

$I_l \leftarrow Copia(I)$

para cada $p \in linha$ **faça**

$I_i[p.linha][p.coluna] \leftarrow corLinha$

fim

fim

retorna I_l

4.2 Algoritmos para detecção da linha da costa

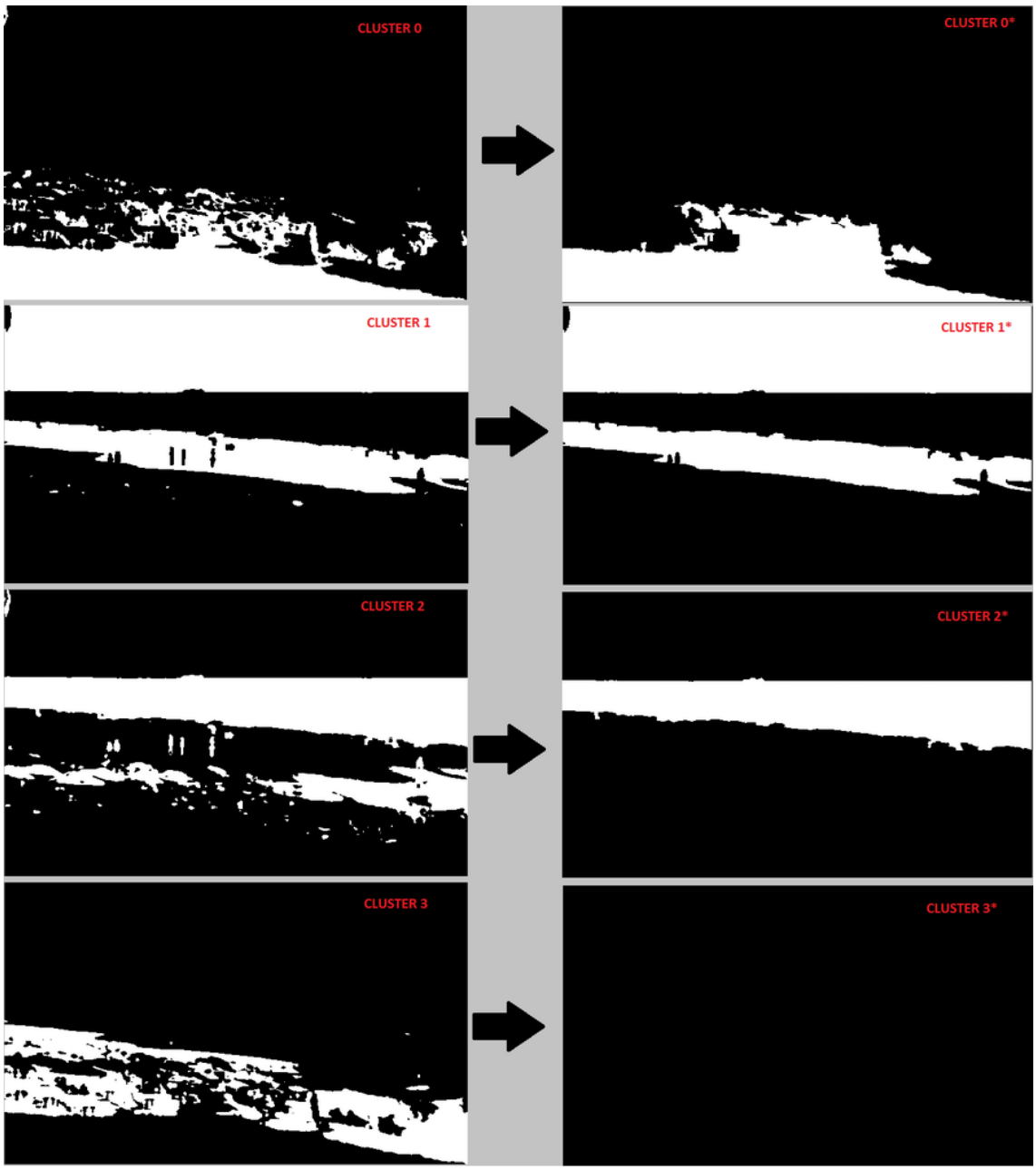
Nesta seção serão apresentados os algoritmos que separam o mar da areia da praia construindo a linha da costa.

4.2.1 Algoritmo de segmentação da praia de Carrera (ASPC)

Este algoritmo foi proposto por Carrera na monografia (CARRERA, 2017) que é apresentada uma abordagem para segmentação da praia que se baseia em segmentações por cores onde é utilizado o sistema de cor HSV. Esse algoritmo possui a premissa de que o canal H do HSV apresenta um boa separação da região de água e areia, devido a grande diferença de brilho entre eles. Um algoritmo de segmentação é utilizado para agrupar essas regiões semelhantes. Para isso foi utilizado o algoritmo de agrupamento *K-Means* (HARTIGAN; WONG, 1979), tendo como entrada uma imagem HSV suavizada com um filtro gaussiano. O grupos gerados pelo *K-Means* são separado da imagem, e são binarizados com um limiar $T = 0$, de modo que o grupo fique com a cor branca e a demais área fique em preto. Os *clusters* maiores que um certo limiar a, são considerados para a próxima etapa, dessa forma, são eliminados ruídos que possam surgir com a segmentação do *k-means*. Os *clusters* resultantes vizinhos são agrupados em um único *cluster*. Na Figura 34, podemos verificar os *clusters*

binarizados à esquerda e preenchimento do contorno externo de área maior que a à direita.

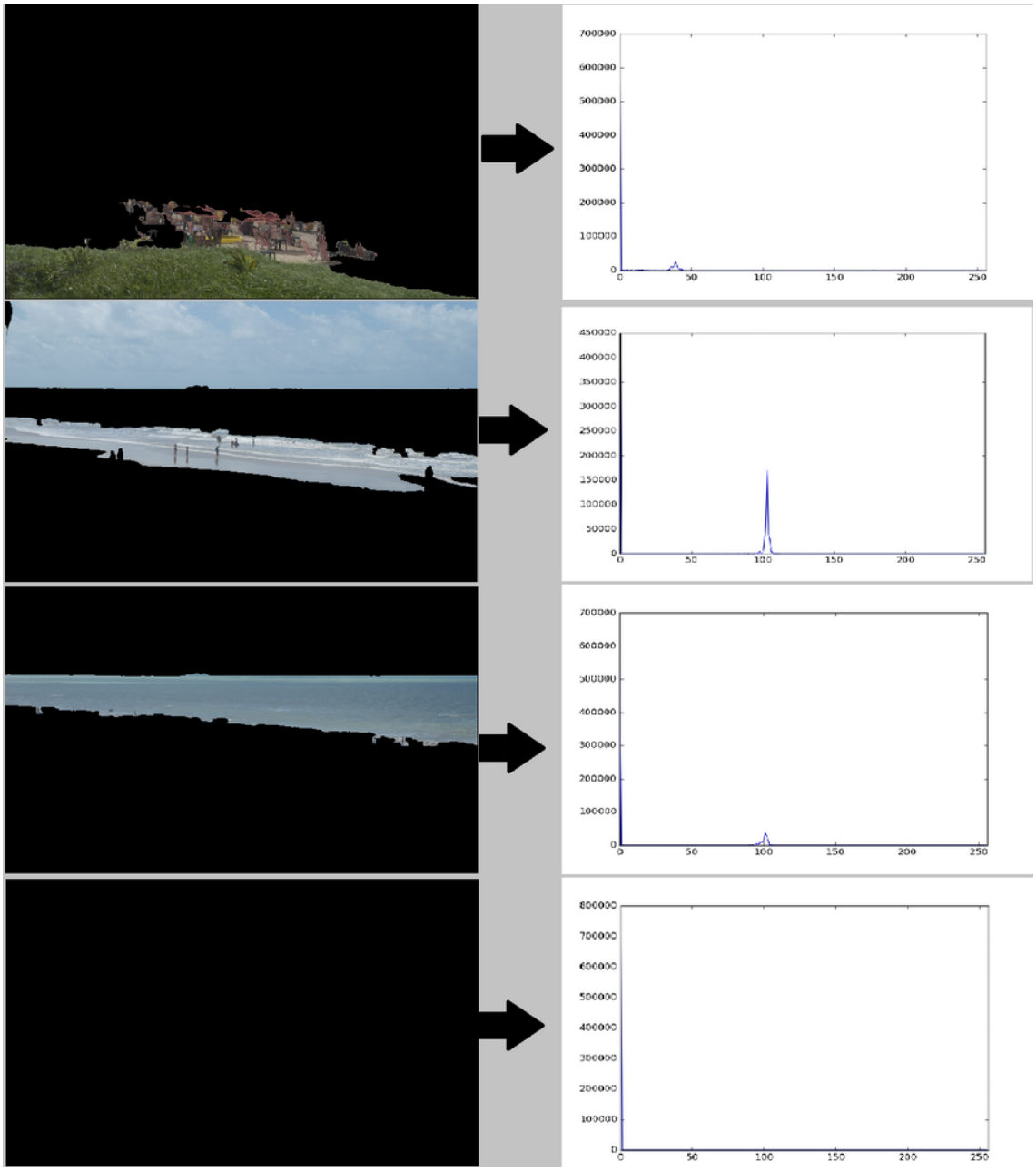
Figura 34 – *Clusters* binarizados à esquerda e preenchimento do contorno externo de grandes áreas à direita.



Fonte – (CARRERA, 2017)

Após obter as máscaras definida pelo contorno externos de áreas maiores que a, as imagens originais são recuperadas através da operação de *AND* entre a máscara e a imagem original. Depois é realizado uma operação para detectar quais *clusters* possuem regiões de água, logo em seguida são convertidos para o HSV e o histograma referente ao canal H é calculado, como mostra a Figura 35.

Figura 35 – *Clusters* obtidos e seus histogramas referentes ao canal H.



Fonte – (CARRERA, 2017)

Os *clusters* com desvio padrão maior que σ, na faixa de cor entre 81 a 139, são agrupados uns aos outros por uma operação de *OR* gerando uma máscara de segmentação de praia temporária. Como em alguns casos o de agrupamento junta regiões de água a partir das bordas, gerando falhas intermediárias, então o algoritmo estende em 1 *pixel* as bordas laterais e recalcula os contornos externos para corrigir essa falha.

A Figura 36 mostra o resultado desse algoritmo através de quatro passos, na imagem (1) é a imagem de entrada, na (2) é a imagem segmentada com erros, na (3)

é a expansão de bordas e ajuste das regiões faltantes e na (4) a segmentação final da praia.

Figura 36 – (1) Imagem original; (2) Segmentação temporária com erros; (3) Expansão de borda e ajuste das regiões faltantes; (4) Segmentação final de praia.

Fonte – (CARRERA, 2017)

Depois da expansão de borda, a máscara da segmentação de praia é obtida através de uma operação de AND com a imagem original 37.

Figura 37 – Máscara da segmentação de praia final (esquerda) e região de água detectada (direita).

Fonte – (CARRERA, 2017)

Na Figura 38 podemos observar o pseudocódigo do algoritmo dessa abordagem.

Figura 38 – Algoritmo de segmentação de praia.

Algoritmo 1: SEGMENTAÇÃO DE PRAIA

Entrada: $\mathrm{I}, k, b, a, \sigma$
Saída: Máscara_Segmentação $(\mathrm{M}_i), Imagem\_Segmentada(I_s)$

início
  $mascara\_preenchida \leftarrow \phi$
  $imagem\_temporaria \leftarrow \phi$
  $imagem\_final \leftarrow \phi$
  $imagem\_hsv \leftarrow HSV(I)$
  $imagem\_borrada \leftarrow FiltroGaussiano(I, (9x9), \sigma = 5)$
  $clusters[] \leftarrow MiniBatchKMeans(imagem\_borrada, k, b)$
  para *cada* $c \in clusters$ **faça**
    $bin \leftarrow Threshold(c, 0)$
    $bin\_externo[] \leftarrow EncontrarContornosExternos(bin)$
    para *cada* $ex \in bin\_externo$ **faça**
      $area \leftarrow Area(ex)$
      se $(area > a)$ **então**
        $mascara\_preenchida \leftarrow PreencheContornoExterno(ex)$
      fim
    fim
    $imagem\_temporaria \leftarrow AND(I, mascara\_preenchida)$
    $imagem\_hsv\_temp \leftarrow HSV(imagem\_temporaria)$
    $histograma \leftarrow CalculaHistograma$(imagem_hsv_temp, 0)
    $histograma\_azul \leftarrow histograma[81 : 139]$
    $std \leftarrow STD(histograma\_azul)$
    se $(std > \sigma)$ **então**
      $imagem\_final \leftarrow OU$(imagem_final, mascara_preenchida)
    fim
  fim
  $imagem\_borda \leftarrow AdicionaBordaLateral$(imagem_final, 1, 255)
  $imagem\_final \leftarrow EncontraPreencheContorno$(imagem_borda)
  $M_i \leftarrow RemoveBordaLateral$(imagem_final)
  $I_s \leftarrow AND$(I, M_i)
fim
retorna M_i, I_s

Fonte – (CARRERA, 2017)

4.2.2 DLCCGME

Esse algoritmo proposto nesse trabalho se baseia nos algoritmos de grafo multiestágios apresentados em (LIE et al., 2005) e em (AHMAD et al., 2015). Visando encontrar uma algoritmo para detecção da linha da costa que oferecesse uma boa taxa de acertos e um bom tempo de processamento, é formulado uma abordagem onde é utilizado um detector de bordas que destaque bem a linha da costa e o grafo multiestágios denso semelhante a proposta por Ahmad *et al.*, essa abordagem de grafo multi-estágio denso foi utilizado ao invés ao de Lie *et al.* por oferecer melhor desempenho e ao fato

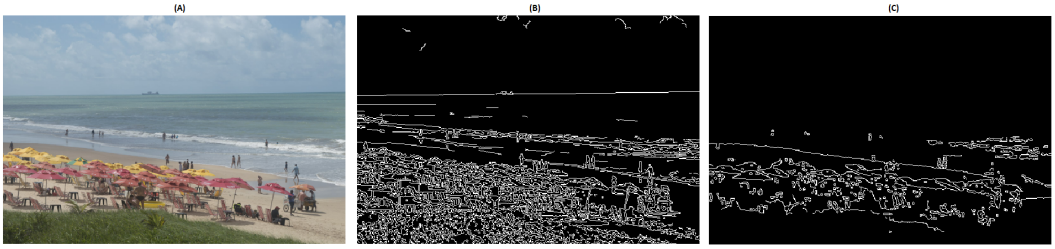
de que sempre há um caminho mesmo que tenha um custo alto, favorecendo encontrar linha da costa mesmo com oclusões.

Ao utilizar o detector de bordas *Canny* em tons de cinza, nem sempre é possível encontrar uma boa detecção da linha da costa, então a imagem foi convertida para o sistema de cor HSV, e utilizando o canal H, o qual obtém uma maior contraste entre areia e o mar, que uma imagem em tons de cinza, também foram testados outros sistemas de cores, mas esse mostrou melhor resultado.

4.2.2.1 Detecção de bordas

Para detecção das bordas é utilizado o *Canny* por ser um detector de bordas ótimo ele consegue encontrar uma ampla quantidade de bordas, Lie *et al.* usa um detector de bordas a partir de uma imagem em tons cinza, a proposta difere dessa em que ao invés de usar uma imagem em tons de cinza é utilizado o canal H da imagem no sistema de cor HSV. Os parâmetros foram definidos através de vários experimentos com a base de dados e o melhor resultado foi o que definiu os parâmetros do operador *Canny*. Na Figura 39 podemos comparar a detecção de bordas para uma imagens em tons de cinza e para o canal H, usando esse canal possibilitou manter os parâmetros de $minVal$ e $maxVal$ do algoritmo do *Canny* fixo para todas as imagens da base de dados, obtendo a linha da costa com menos ruídos que o obtido por uma imagem em tons de cinza.

Figura 39 – (A) Imagem original; (B) Mapa de bordas a partir de uma imagem em tons de cinza; (C) Mapa de bordas a partir do canal H do HSV.



Fonte – O autor

4.2.2.2 Encontrar a linha da costa

A detecção da linha da costa é uma técnica para encontrar a linha que separa a região de água da praia da areia. Para isso esse algoritmo utiliza o mapa de bordas invertido obtido na seção anterior. Então o algoritmo de grafo multiestágios semelhante ao algoritmo apresentado na Subseção 4.1.4.2 é utilizado para encontrar a linha mais próxima da linha da costa, onde o mapa de borda invertido é usado para criar um grafo multiestágios $M \times N$ dado por $G(V, E, \Phi)$, onde cada *pixel* representa um vértice no grafo multiestágios. A diferença desta abordagem proposta para a utilizada no algo-

ritmo DLHCGME descrito na Seção 4.1.4.2, é que no DLHCGME é utilizado peso nas arestas do vértice virtual s para o primeiro estágio e dos vértice no estágio N para o vértice virtual t dado por $(i+1)^2$ onde i é a linha a qual ele pertence, e nessa abordagem a linha central da imagem tem peso mínimo, e a medida que se afasta do centro o valor do preso cresce até o valor máximo nas duas extremidade, e pode ser definido pela equação $(r+1)^2$, onde $r = [i - (i/2)]$. Essa modificação tem o objetivo de aumentar a probabilidade de encontrar a linha que se aproxima da região central da imagem, visto que a linha da costa geralmente se encontra mais próxima do centro da imagem.

Na Figura 40, podemos observar alguns exemplos de detecções da linha da costa em imagens de praia com o DLCCGME.

Figura 40 – Exemplos de detecções da linha da costa com o DLCCGM.

Fonte – O autor

Algoritmo 7: Algoritmo DLCCGME

Entrada: I

Saída: $LinhaDetectada(I_l)$

início

- $corLinha \leftarrow (0, 0, 0)$
- $imagemHsv \leftarrow HSV(I)$
- $imagemHsvSuavisada \leftarrow FiltroGaussiano(imagemHsv, (3 \times 3), 0)$
- $imagemCanalH \leftarrow ExtrairCanalH(imagemHsvSuavisada)$
- $bordas \leftarrow Canny(imagemCanalH, minVal = 10, maxVal = 50)$
- $bordasInv \leftarrow NOT(bordas)$
- $grafoMultiEstagio \leftarrow CriarGrafoMultiEstagio(bordasInv)$
- $linha[] \leftarrow Dijkstra(grafoMultiEstagio, s, t)$
- $I_l \leftarrow Copia(I)$
- **para cada** $p \in linha$ **faça**
  - $I_i[p.linha][p.coluna] \leftarrow corLinha$
- **fim**

fim

retorna I_l

4.2.3 DLCCCH

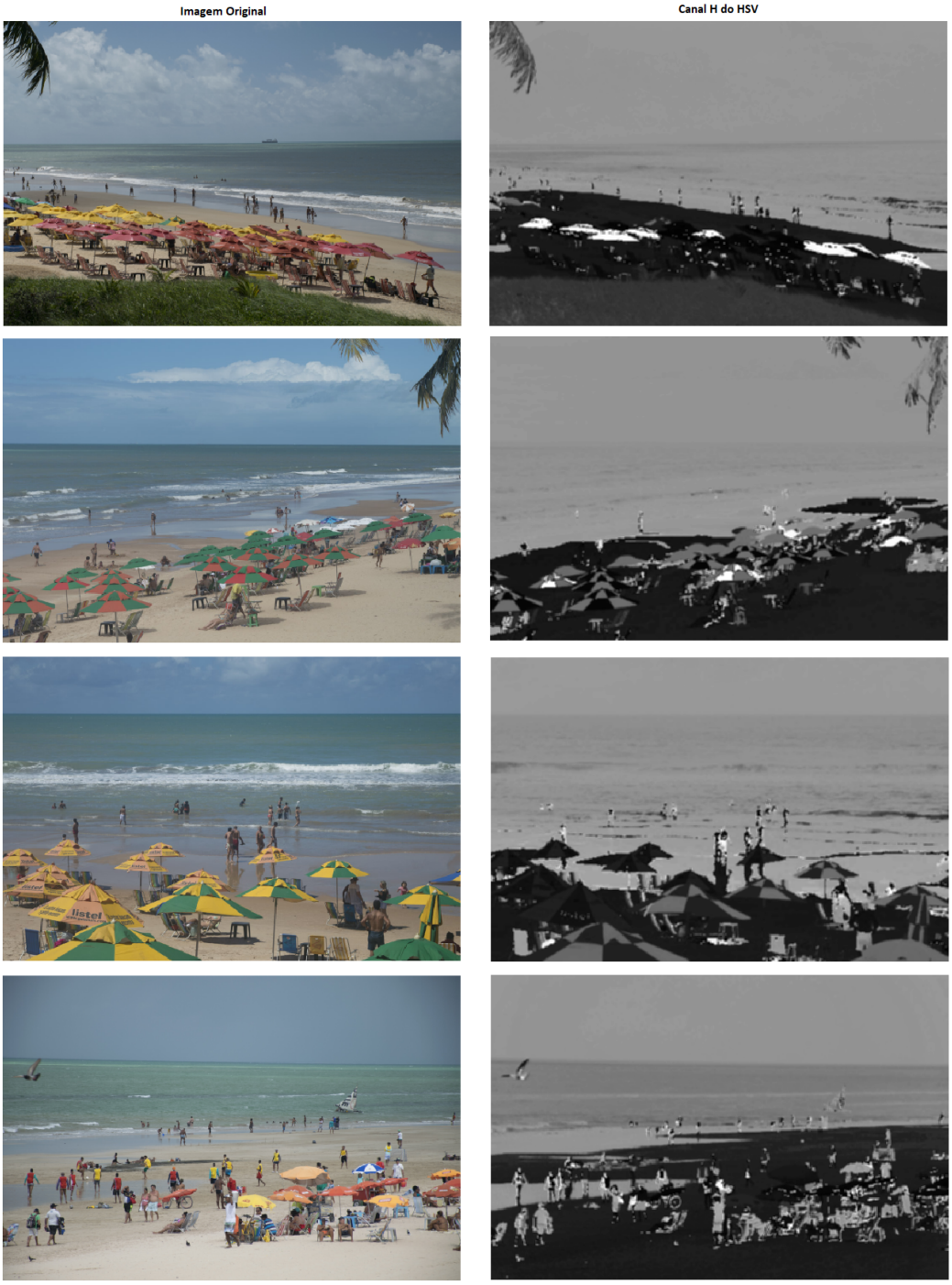
Algoritmo para detecção da linha da costa em imagens de praia, baseado em segmentar a imagens em duas regiões, uma contendo a faixa da areia e a outra contendo o céu e o mar. Imagens de praia são compostas de três grandes regiões: céu, mar e areia. No entanto essas regiões são formadas por várias regiões menores assim como o céu é composto de várias regiões com tom diferentes de azuis ou cinza, e também por nuvem, no mar podemos ter várias tonalidades, seja formadas pelas ondas ou iluminação de uma região pelo sol, ou por uma sombra ocasionada pelas nuvens e também por pessoas dentro da mar. Na areia podemos ter pessoas, guarda-sol, cadeiras, árvore e etc. Todos esses fatores contribuem para dificultar a segmentação da imagem naquelas duas regiões.

Dentre os sistemas de cores avaliados o sistema de cor HSV é o que melhor segmentar essas regiões, entretanto para isso apenas o canal *Hue* é utilizado, visto que ele apresenta um contraste maior entre a areia e as demais regiões como o céu e o mar. Na Figura 42, pode-se observar na segunda coluna o canal *Hue* de algumas imagens de praia.

Figura 41 – Exemplos de imagem de praia no sistema de cor HSV, onde (a) é a imagem original em RGB e (b) é a imagem em HSV.

Fonte – O autor

Figura 42 – Exemplos de extrações do canal H em imagens de praia.



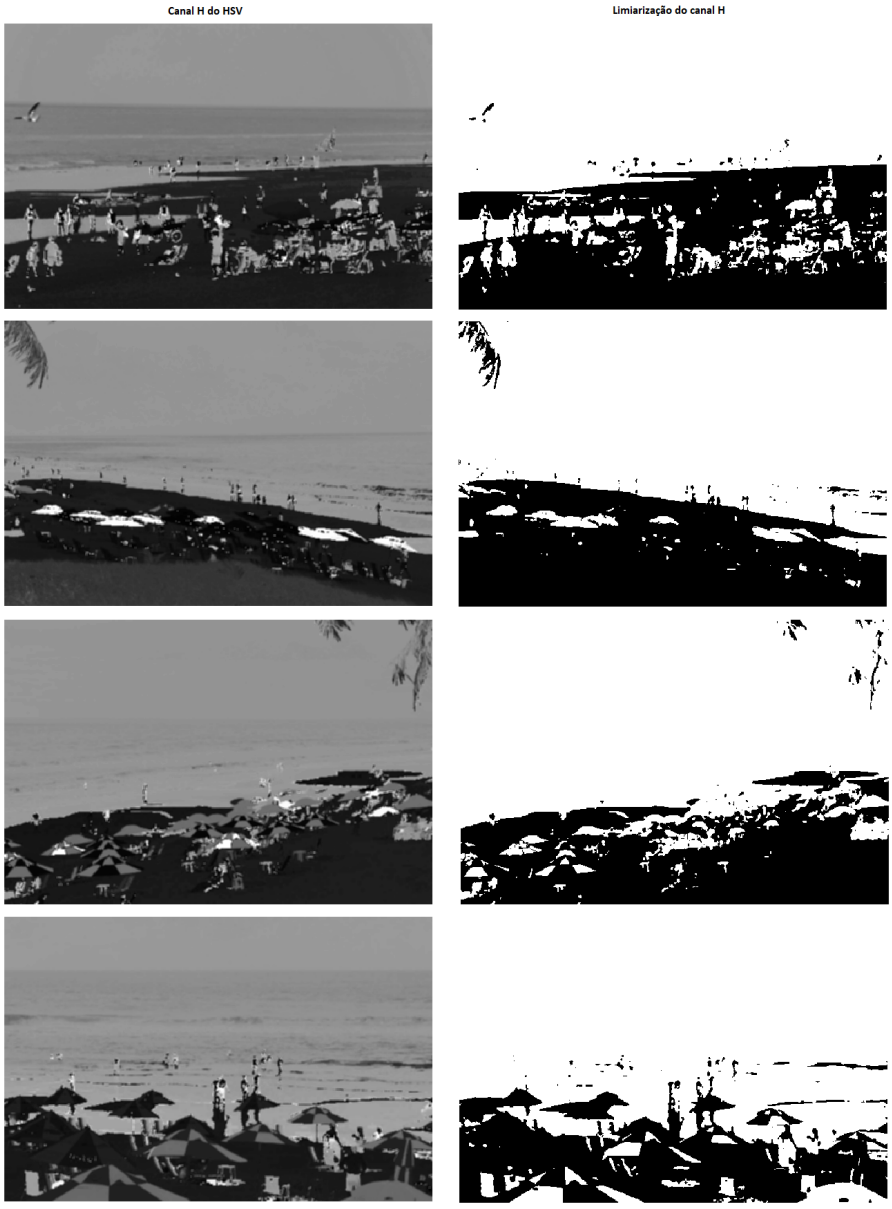
Fonte – O autor

O próximo passo é binarizar a imagem obtida do passo anterior, utilizando o valor de limiar de 65 como mostra a Equação 4.7.

$$g(x,y) = \begin{cases} 0, & Se\ 65 \leq f(x,y) \\ 255, & \text{senão.} \end{cases} \tag{4.7}$$

Após a limiarização é obtido uma imagem onde o mar e o céu se funde como céu em um tom branco e a região da areia em preto, mas na região da areia pode conter pequenas regiões brancas devido a alguns elementos com tons próximo aos do céu e mar, como guarda-sol e outros elementos, assim como na região do mar e céu podemos encontrar pequenas regiões pretas como mostrar a Figura 43.

Figura 43 – Limiarizações do canal H de imagens de praia.



Fonte – O autor

A ideia desse algoritmo é separar a imagem em duas regiões, onde a parte inferior da imagem em preto seria a areia e na parte superior o mar e o céu. Para isso as pequenas regiões pretas dentro da região grande branca, como também as pequenas regiões brancas dentro da região preta, devem ser eliminadas, sobrando

apenas as duas regiões (areia e céu com o mar).

Primeiro são eliminadas as pequenas regiões brancas dentro da região preta (areia) realizando a contagem da área de cada região branca, e assim é selecionada a região de maior área. A área selecionada é pintada com cor branca sobre uma imagem preta do mesmo tamanho da imagem de entrada como mostra a Figura 44. Para encontrar as regiões foram utilizadas as funções do *OPENCV*: *findContours contourArea* para calcular a área do contorno e *drawContours* para desenhar o contorno.

Figura 44 – Imagem de entrada a esquerda e resultado após subtrações dos pequenos contornos brancos na imagem à direita.

Fonte – O autor

O próximo passo é remover os pequenos contornos pretos na região branca (céu e mar), para isso a imagem é invertida através da operação *NOT*, e repte-se o mesmo passo anterior buscando contornos e calculando as áreas selecionadas. Assim é criada uma nova imagem com esse contorno, como mostra a Figura 45.

Figura 45 – A imagem de entrada à esquerda e resultado após subtrações dos pequenos contornos brancos na imagem no centro e resultado na direita.

Fonte – O autor

Após esse resultado, é extraída a linha que separa as duas regiões, obtendo assim uma linha que se aproxima da linha da costa como mostra a terceira imagem da Figura 45.

Algoritmo 8: Algoritmo DLCCCH

Entrada: I

Saída: $LinhaDetectada(I_l)$

início

$corLinha \leftarrow (0, 0, 0)$

$imagemHsv \leftarrow HSV(I)$

$imagemCanalH \leftarrow ExtrairCanalH(imagemHsv)$

$th \leftarrow Threshold(imagemCanalH, 65)$

$contornos[] \leftarrow EncontrarContornos(th)$

$areas[] \leftarrow \varnothing$

para cada $contorno \in contornos$ **faça**

  $areas.adicionar(CalcularArea(contorno))$

fim

$indiceAreaMaior \leftarrow IndiceMaiorArea(areas)$

$imagemMaiorArea \leftarrow CriaImagem(imagemCanalH.dimesao, 0)$

$imagemMaiorArea \leftarrow PreencherContorno(contornos[indiceAreaMaior], 255)$

$imagemInv \leftarrow NOT(imagemMaiorArea)$

$contornos2[] \leftarrow EncontrarContornos(imagemInv)$

$areas2[] \leftarrow \varnothing$

para cada $contorno \in contornos2$ **faça**

  $areas2.adicionar(CalcularArea(contorno2))$

fim

$imagemDoisContornos \leftarrow CriaImagem(imagemCanalH.dimesao, 0)$

$indiceAreaMaior2 \leftarrow IndiceMaiorArea(areas2)$

$imagemDoisContornos \leftarrow$
$PreencherContorno(contornos[indiceAreaMaior2], 255)$

$I_l \leftarrow ExtrairLinha(imagemDoisContornos)$

fim

retorna I_l

4.3 Observações

O objetivo desse projeto é desenvolver um estudo sobre a detecções linha do horizonte e da costa em imagens de praias e tem por motivação que não foi encontrado nenhum trabalho na literatura com essa proposta. Para detecção da linha do horizonte são abordados quatro algoritmos sendo dois do estado da arte o DCSI (AHMAD et al., 2015) e Lie *et al.* (LIE et al., 2005) os quais foram propostos para detecções de linha do horizonte em imagens de montanhas, os outros dois algoritmos são contribuições

desse trabalho, sendo o DLHCGME baseado em segmentações, detectores de bordas e grafo multiestágios, e o DLHSTH visa encontrar a linha por meio de detectores de bordas horizontais e transformada de *Hough*. Para detecção da linha da costa não foram encontrado trabalho publicado na literatura, apenas uma monografia de Carrera (CARRERA, 2017) que aborda o algoritmo ASPC para segmentação de praias, os outros dois algoritmos são contribuições desse projeto o DLCCCH que é um algoritmo que encontrar a linha a partir de segmentações na imagem e o DLCCGME algoritmo que utiliza segmentações, detectores de bordas e grafo multiestágios. Essas abordagens foram apresentadas para realizar uma analise desses algoritmos visando encontrar os que apresentarem melhores resultados para mitigar o problema dos risco apresentado pela orla da praia de Boa Viagem.

5 Avaliação Experimental

Com o objetivo de avaliar os algoritmos propostos e da literatura, neste Capítulo é descrita metodologia dos experimentos realizados neste trabalho com imagens de praia.

5.1 Base de dados

Para realiza os experimento, a base de dados foi dividida em duas partes: uma formada por imagens com oclusões e outra sem oclusões. Imagens com oclusão são imagens que possuem algum objeto atravessando algumas das regiões (areia, mar ou céu). Na maioria das vezes, são coqueiros, que prejudicam a detecção das linhas. Na Figura 46 pode-se ver exemplos de imagens de praia com oclusão, onde um tronco do coqueiro atravessa a imagem verticalmente. Um outro critério para composição da base de dados é que as linhas devam se estender da esquerda para direita das bordas laterais da imagem.

Figura 46 – Exemplo de imagem de praia com oclusão.

Fonte – O autor

A base de dados de imagens sem oclusões é composta por 126 imagens e a com oclusão é de 16 imagens de praia obtidas na praia de Boa Viagem em 2016 e

2017, com resolução de 542×360. As imagens formam rotuladas manualmente para identificar as linhas do horizonte e da costa. A rotulação consiste em criar duas linha com tons de cores específicos para representar as linhas do horizonte e da costa como mostra a Figura 47. A linha do horizonte é a linha em preto da Figura 47 que define a fronteira entre o mar e o céu e a linha da costa em verde é a fronteira entre a água do mar e a areia incluindo a fina camada de água sobre a areia.

Para todas as imagens da base de dados as linhas do horizonte e da costa se estendem da borda esquerda para borda direita da imagem, tendo assim imagens mais frontais com pequenos ângulos de inclinações. Também não foram utilizadas imagens noturnas e com chuvas.

Figura 47 – Imagem rotulada.

Fonte – O autor

Essa base de dados é utilizada para avaliações dos algoritmos abordados no projeto.

5.1.1 Base de dados para treinamento do DSCI

Em (AHMAD et al., 2015) é utilizado um conjunto de 9 imagens de cenários de montanhas para treinar o classificador. Entretanto neste trabalho é utilizado um conjunto formado por 12 imagens de praias para treinamento do algoritmo DCSI, que utiliza aprendizagem de máquina, como descrito na Seção 4.1.2. Para treinar o classificador foram utilizados 300 amostras positivas (linha do horizonte) e 300 amostras negativas

(não horizonte) de cada imagem, onde uma amostra corresponde a um *pixel* rotulado da imagem, a partir do *pixel* rotulado uma janela ao redor do *pixel* é extraída. A Figura 48 mostra exemplos de *pixels* da linha do horizonte (preto) e não horizonte (verde).

Figura 48 – Imagem rotulada para treinamento do DCSI.

Fonte – O autor

5.2 Ambiente

Os experimentos formam realizados em uma máquina equipada com processador *Intel(R) Core(TM) i5-6300HQ* de 2.30GHz com 8GB de memória *RAM*. Os algoritmos foram implementados em *Python 3.6* (FOUNDATION,). As bibliotecas utilizadas foram: *Opencv* (OPENCV, 2015) para processamento de imagem, *Numpy* (DEVELOPERS, b), *NetworkX* (DEVELOPERS, a) para utilização de grafos e *Scikit-learn* (PEDREGOSA et al., 2011) para algoritmos de aprendizagem de máquina utilizado no algoritmo DCSI.

5.3 Métrica de avaliação

5.3.1 Média dos erros médios absolutos

Com o objetivo de medir a quão distante a linha encontrada está da linha verdadeira rotulada, se um método gera uma solução precisa, a linha encontrada deve ser consistente com a linha verdadeira, ou seja, a distância vertical dos *pixels* pertencen-

tes aos dois limites deve ser minimizada. Isso é medido através da distância vertical absoluta em *pixels*, que é uma métrica denominada de média das distâncias absoluta do *pixel* ou erro médio absoluto, utilizada por Ahmad *et al.* nos seguintes trabalhos: (AHMAD et al., 2017) e (AHMAD et al., 2015). Onde matematicamente pode ser representado através da Equação 5.1.

$$S = \frac{1}{N_{set}} \sum_{i=1}^{N_{set}} \left(\frac{1}{N} \sum_{j=1}^{N} |{P_{d(j)}}^{i} - {P_{g(j)}}^{i}| \right) \tag{5.1}$$

Onde $P_{d(j)}$ e $P_{g(j)}$ são posições das linhas detectada e verdadeira (rotulada) respectivamente. Na coluna j e N é o número de coluna da imagem de teste, e N_{set} é o número de imagens na base de dados de teste. Um alinhamento perfeito da linha encontrada com a rotulada deve resultar em zero.

5.3.2 Desvio padrão do erros médios absolutos

O desvio padrão é calculado a partir dos erros médios absolutos obtido de da imagem como mostrar a Equação 5.2 onde x_i é o valor do erro médio absoluto de uma imagem, $\bar{x}$ é a média dos erros médios absolutos e n é o número de elemento na base de dados em questão.

$$\sigma = \sqrt{\sum \frac{(x_i - \bar{x})^2}{n}} \tag{5.2}$$

5.3.3 Tempo médio de processamento

O tempo médio de processamento é o tempo médio gasto para processamento de uma amostra, e pode ser descrita pela Equação 5.3, onde t_i é o tempo de processamento de uma imagem da base de dados.

$$T = \frac{1}{N_{set}} \sum_{i=1}^{N_{set}} (t_i) \tag{5.3}$$

5.4 Experimentos H

O experimento H visa encontrar os melhores resultados para a detecção da linha do horizonte. Para a avaliação dos algoritmos foram analisados três parâmetros: média dos erros médios absolutos descrito na Seção 5.3, desvio padrão dos erros médios absolutos e tempo médio de processamento. Para esse experimento foram analisados os seguintes algoritmos para detecção da linha do horizonte: DCSI (AHMAD et al., 2015), Lie *et al.* (LIE et al., 2005), DLHCGME e DLHSTH.

5.4.1 Experimento H-1

Os experimentos do tipo H-1 são experimentos para detecção da linha do horizonte para base de dados de imagens sem oclusões. São consideradas nesse trabalho imagem sem oclusão como sendo as imagens que não possui grandes oclusões onde geralmente atravessa a imagem e interrompe a linha do horizonte como mostra Figura 49 que é um exemplo de imagens sem oclusão.

Figura 49 – Exemplo de imagem sem oclusão.

Fonte – O autor

Para esse experimento foram utilizadas 126 imagens com resoluções de 542×360 e as linhas do horizonte de cada imagem foram rotuladas manualmente.

5.4.2 Experimento H-2

Os experimentos do tipo H-2 são experimentos de detecção da linha do horizonte para base de dados de imagens com oclusões, como mencionado na seção anterior imagens com oclusões são imagens em que possui uma grande oclusão onde uma boa parte da linha do horizonte é interrompida por essa oclusão, na Figura 50 abaixo temos alguns exemplos de imagens com oclusões onde a linha do horizonte é parcialmente interrompida parte de construção na imagem superior e por um coqueiro na imagem inferior.

Figura 50 – Exemplos de imagens com oclusões.

Fonte – O autor

Nesse experimento foram utilizados 16 imagens com oclusão de 542×360 que foram rotuladas manualmente para avaliação, nessa rotulagem a linha rotulada atravessa a oclusão como se a mesma não existisse como mostra a Figura 51, caso ocorra oclusões por navios, os mesmos são desconsiderados e a linha do horizonte rotulada é continua, assim como os coqueiros.

Figura 51 – Exemplo de rotulagem de imagem com oclusões para linha do horizonte.

Fonte – O autor

5.5 Experimentos C

Nos experimentos do tipo C são abordados os algoritmos de detecção da linha da costa em imagens de praia, os seguintes algoritmos foram analisados: ASPC (CARRERA, 2017), DLCCGME e DLCCCH. Para as avaliações foram analisados três parâmetros: média dos erros médios absolutos 5.3, desvio dos erros médios absolutos e o tempo médio de processamento.

5.5.1 Experimento C-1

O experimento do tipo C-1 visa avaliar os algoritmos para detecção da linha da costa em imagens de praia para base de dados de imagens sem oclusões, essa base de imagens sem oclusões contém pequenas oclusões tais como: por pessoas, guarda-sol e entre outras pequenas oclusões. Para esse experimento foram utilizadas 126 imagens com dimensões de 542$\times$360 e a linha da costa de cada imagem foi rotulada manualmente.

5.5.2 Experimento C-2

O experimento do tipo C-2 tem como objetivo avaliar os algoritmos para detecção da linha da costa em imagens de praia para base de dados de imagens com oclu-

sões. Para esse experimento foram utilizadas 16 imagens com dimensões de 542×360 e a linha da costa de cada imagem foi rotulada manualmente. Onde há grande oclusão a linha da costa rotulada atravessa a oclusão como se a mesma não existisse, entretanto pequenas oclusões como por: pessoa, guarda-sol e outros, onde são possível para os algoritmos contornarem, essas oclusão foram contornadas, como mostra a Figura 52 que apresenta um exemplo de rotulagem da linha da costa em imagens com oclusões.

Figura 52 – Exemplo de rotulagem de imagem com oclusões para linha da costa.

Fonte – O autor

6 Resultados

Este Capítulo mostra os resultados dos algoritmos apresentados na Seção 4. Os resultados foram divididos em algoritmos de detecção da linha do horizonte e algoritmos de detecção da linha da costa, usando a métrica apresentada da Seção 5.3.

Os resultados serão apresentados em duas seções, onde a primeira será dedicada aos algoritmos de detecção da linha do horizonte e a segunda será para algoritmos detecção da linha da costa, utilizando a métrica da média dos erros absolutos, onde S é o erro médio absolutos, σ é o desvio padrão e t é o tempo médio de processamento em segundos.

6.1 Algoritmos de detecção da linha do horizonte

6.1.1 Experimento H-1

A Tabela 2 apresenta os resultados obtidos a partir dos algoritmos de detecção da linha do horizonte descrito na Seção 4.1. Nela são apresentados os resultados dos algoritmos para base de dados sem oclusões. Onde foram analisados para cada algoritmo os seguintes parâmetros: média dos erros absolutos S, o desvio padrão dos erros absolutos σ e o tempo médio de processamento em segundos t. As linhas representam os resultados de um algoritmo e as colunas os resultados por parâmetros.

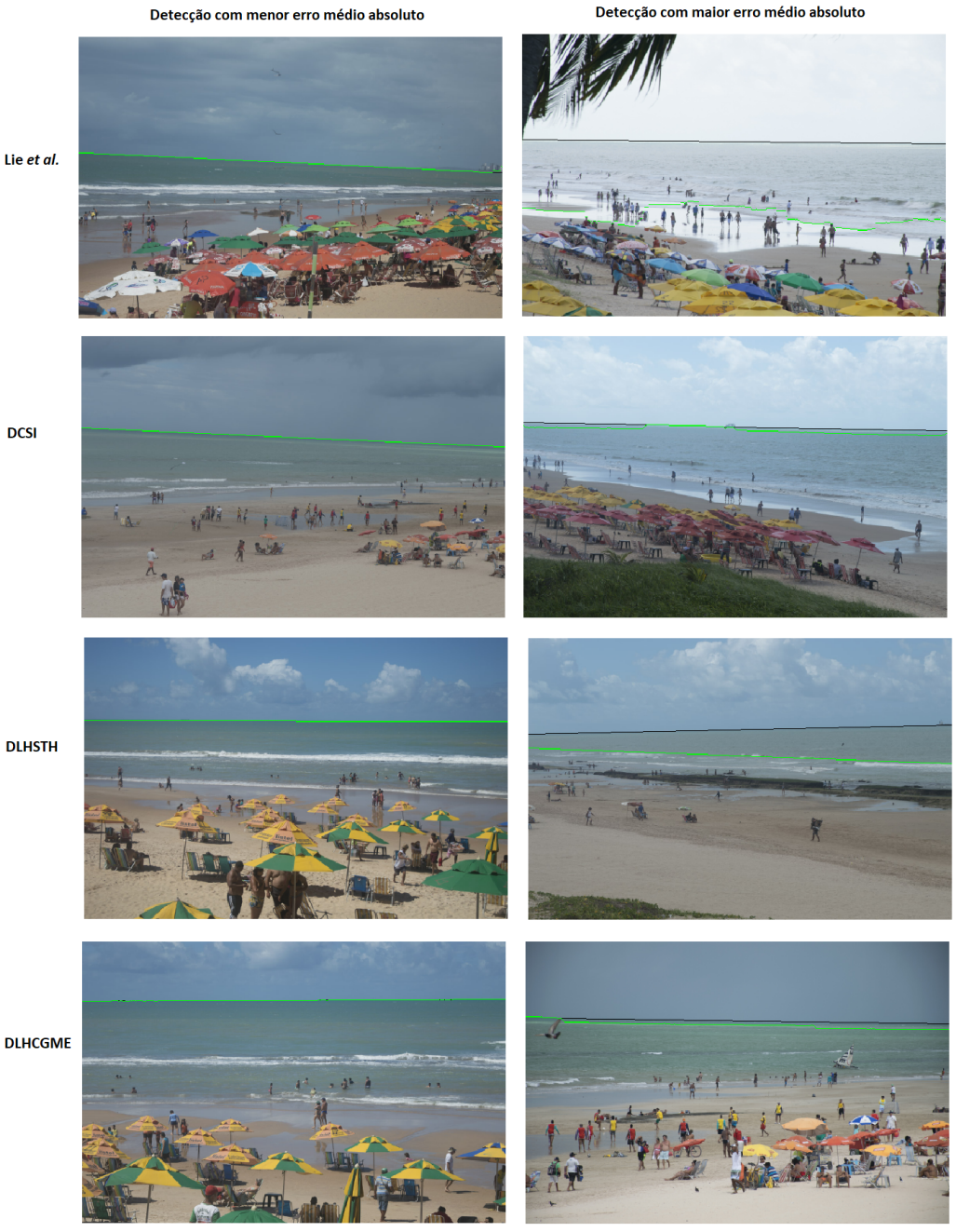
Tabela 2 – Visualização dos resultados dos experimentos de detecção da linha do horizonte para base de dados sem oclusões.

Resultado Para Base de Dados Sem Oclusões				
Parâmetros Analisados		S	σ	t
Algoritmos do Estado da Arte	***DCSI***	0,69	0,82	70,8
	Lie et al.	1,29	8,36	157,79
Algoritmos Propostos	***DLHCGME***	**0,47**	0,52	6,62
	DLHSTH	1,11	4,9	**0,022**

Nos resultados obtidos nos experimentos em imagens sem oclusões mostrados na Tabela 2, os dois algoritmos que obtiveram os melhores resultados foram o DLHCGME e o DLHSTH. O DLHCGME obteve os melhores resultados para erro médio absoluto S e o desvio padrão do erro médio absoluto σ, mas o DLHSTH obteve o melhor resultado para o tempo médio de processamento t. Todos os resultados obtiveram resultados próximos para o erro médio absoluto com valor da diferença máxima entre eles de $0,82$, que equivale a um diferença menor que um *pixel* por coluna, na ava-

liação do desvio padrão o algoritmo de Lie *et al.* (LIE et al., 2005) foi o que apresentou maior desvio padrão. Na avaliação do tempo o algoritmo de Lie *et al.* (LIE et al., 2005) foi o que obteve o pior tempo de processamento seguido do DCSI de (AHMAD et al., 2015) esses dois algoritmos obtiveram valores de tempo discrepantes em relação a demais, o DCSI devido ao uso de algoritmos de classificação onde é feita a predição *pixel* a *pixel*, já o custo de processamento do algoritmo de Lie *et al.* é na construção do grafo multiestágios, o DLHSTH foi o que obteve o menor tempo de processamento.

Figura 53 – Exemplos de detecções da linha do horizonte sem oclusões por árvores



Fonte – O autor

Na Figura 53 acima apresentamos para cada algoritmo de detecção da linha do horizonte em imagens de praia, as imagens contendo a menor e maior taxa erro médio absoluto, na primeira coluna estão as imagens com as menores taxas de erro médio absoluto e na segunda coluna as imagens com as piores taxas de erro médio absoluto, em imagens sem oclusões, a linha detectada está em verde e a rotulada em preto, caso tenha acerto a linha rotulada é sobrescrita pela linha detectada.

6.1.2 Experimento H-2

A Tabela 3 apresenta os resultados obtidos a partir dos algoritmos de detecção da linha do horizonte descrito na Seção 4.1 para base de dados com oclusões. Onde foram analisados para cada algoritmo os seguintes parâmetros: média dos erros absolutos S, o desvio padrão dos erros absolutos σ e o tempo médio de processamento em segundos t. As linhas representa os resultados de uma algoritmo e as colunas os resultados por parâmetros.

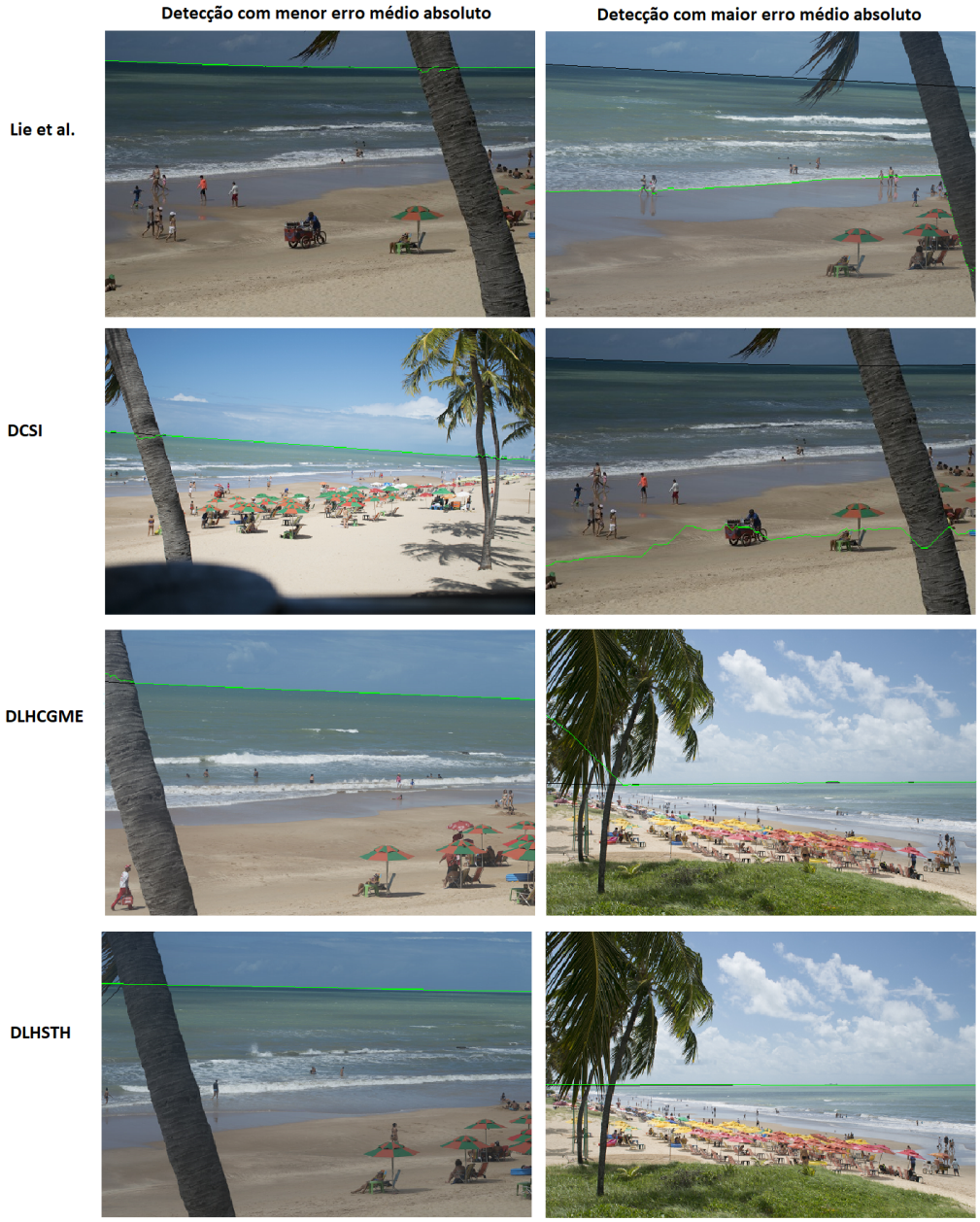
Tabela 3 – Visualização dos resultados dos experimentos de detecção da linha do horizonte para base de dados com oclusões.

Resultado Para Base de Dados Com Oclusões				
Parâmetros Analisados		S	σ	t
Algoritmos do Estado da Arte	***DCSI***	50,21	61,5	75,82
	Lie et al.	59,07	58,1	168,2
Algoritmos Propostos	***DLHCGME***	2,62	2,04	6,58
	DLHSTH	**1,98**	1,20	**0,028**

Nos experimentos com a base de dados de imagens com oclusões, geralmente por coqueiros, o algoritmo DLHSTH obtive o melhor erro médio absoluto e o desvio padrão de erro médio absoluto. Na avaliação do tempo médio de processamento o DLHSTH obteve um tempo de tempo de processamento muito melhor que o DLHCGME que possui o segundo melhor tempo médio. Os algoritmo DCSI e Lie *et al.* obtiveram os piores resultados para todos os parâmetros observados.

Na Figura 54 apresentamos para cada algoritmo de detecção da linha do horizonte em imagens de praia, a linha detectada está em verde e a rotulada em preto, caso tenha acerto a linha rotulada é sobrescrita pela linha detectada, as imagens contendo a menor e maior taxa erro médio absoluto, na primeira coluna estão as imagens com as menores taxas de erro médio absoluto e na segunda coluna as imagens com as piores taxas de erro médio absoluto, em imagens com oclusões. Podemos perceber uma analise visual da primeira coluna que todos obtiveram ótimos resultados, e todos os quatro algoritmos conseguiram atravessar as oclusões por coqueiros, na analise da segunda coluna é possível observar que o DLHCGME e o DLHSTH apresentaram bons resultados mesmo para os piores casos.

Figura 54 – Exemplos de detecções da linha do horizonte com oclusões por árvores



Fonte – O autor

6.2 Algoritmos de detecção da linha da costa

6.2.1 Experimento C-1

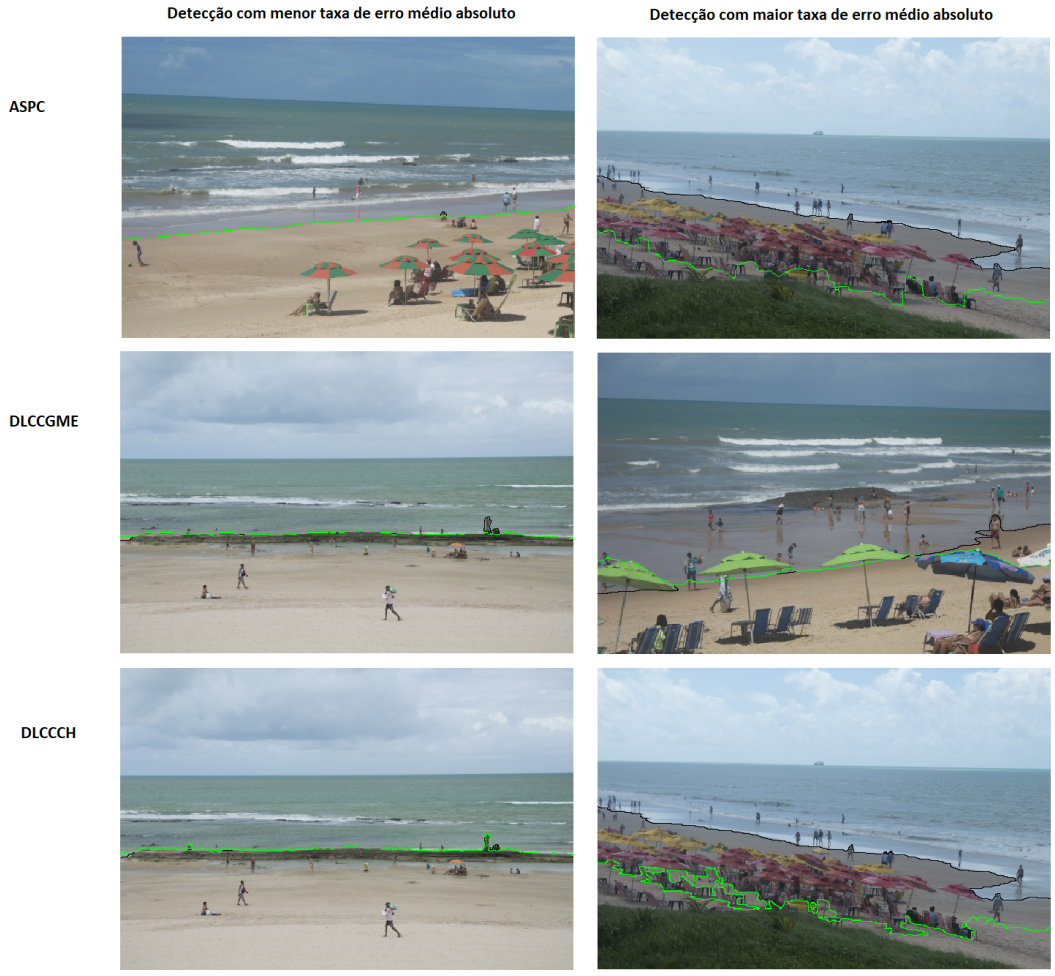
A Tabela 4 apresenta os resultados dos algoritmos de detecção da linha da costa descrito na Seção 4.2 utilizando a base de dados sem oclusões, utilizando a métrica da média dos erros absolutos, onde S é o erro médio absoluto, σ é o desvio padrão e t é o tempo médio de processamento em segundos. As linhas representa os resultados de uma algoritmo e as colunas os resultados por parâmetros.

Tabela 4 – Visualização dos experimentos de detecção da linha da costa para base de dados sem oclusões.

Resultado Para Base de Dados Sem Oclusão				
Parâmetros Analisados		S	σ	t
Algoritmo do Estado da Arte	***ASPC***	**9,45**	11,81	0,48
Algoritmos Propostos	***DLCCGME***	12,7	13,92	6,97
	DLCCCH	10,18	11,1	**0,21**

Como podemos observar a Tabela 4, o algoritmo de ASPC (CARRERA, 2017) apresentou o melhor resultado para média dos erros absolutos. O DLCCCH apresentou um resultado bem próximo e um desvio padrão um pouco inferior. No quesito tempo médio de processamento, o DLCCCH foi o que mostrou melhor resultado. O DLCCGME foi o que demonstrou pior resultado para o tempo de processamento.

Figura 55 – Exemplos de detecções da linha da costa sem oclusões, em verde linha detectada e em preto linha rotulada



Fonte – O autor

Na Figura 55 acima a linha detectada está em verde e a rotulada em preto, caso tenha acerto a linha rotulada é sobrescrita pela linha detectada, nessa imagem podemos observar na primeira coluna exemplos de cada algoritmo de imagens que obtiveram as menores taxas de erro médio absoluto e na segunda coluna exemplos de imagens que obtiveram as maiores taxas de erro médio absoluto. Em uma analise visual dessas imagens podemos observar na primeira coluna que os três algoritmos apresentaram ótimos resultados, na segunda coluna com os exemplos com maiores erros médios absolutos, podemos observar que o DLCCGME foi o mais se aproximou da linha da costa. Analisando essa imagem visualmente é possível perceber na primeira coluna que os três algoritmos apresenta um resultado ótimo, encontrando a linha do horizonte perfeitamente, na segunda coluna é possível perceber que os algoritmos DCSI e DLHCGME apresentou bons resultados mesmo para exemplos com piores resultados.

6.2.2 Experimento C-2

Na Tabela 5 é mostrado os resultados dos algoritmos de detecção da linha da costa descrito na Seção 4.2 utilizando a base de dados com oclusões, utilizando a métrica da média dos erros absolutos, onde S é o erro médio absoluto, σ é o desvio padrão e t é o tempo médio de processamento em segundos. As linhas representa os resultados de uma algoritmo e as colunas os resultados por parâmetros.

Tabela 5 – Visualização dos experimentos de detecção da linha da costa para base de dados com oclusões.

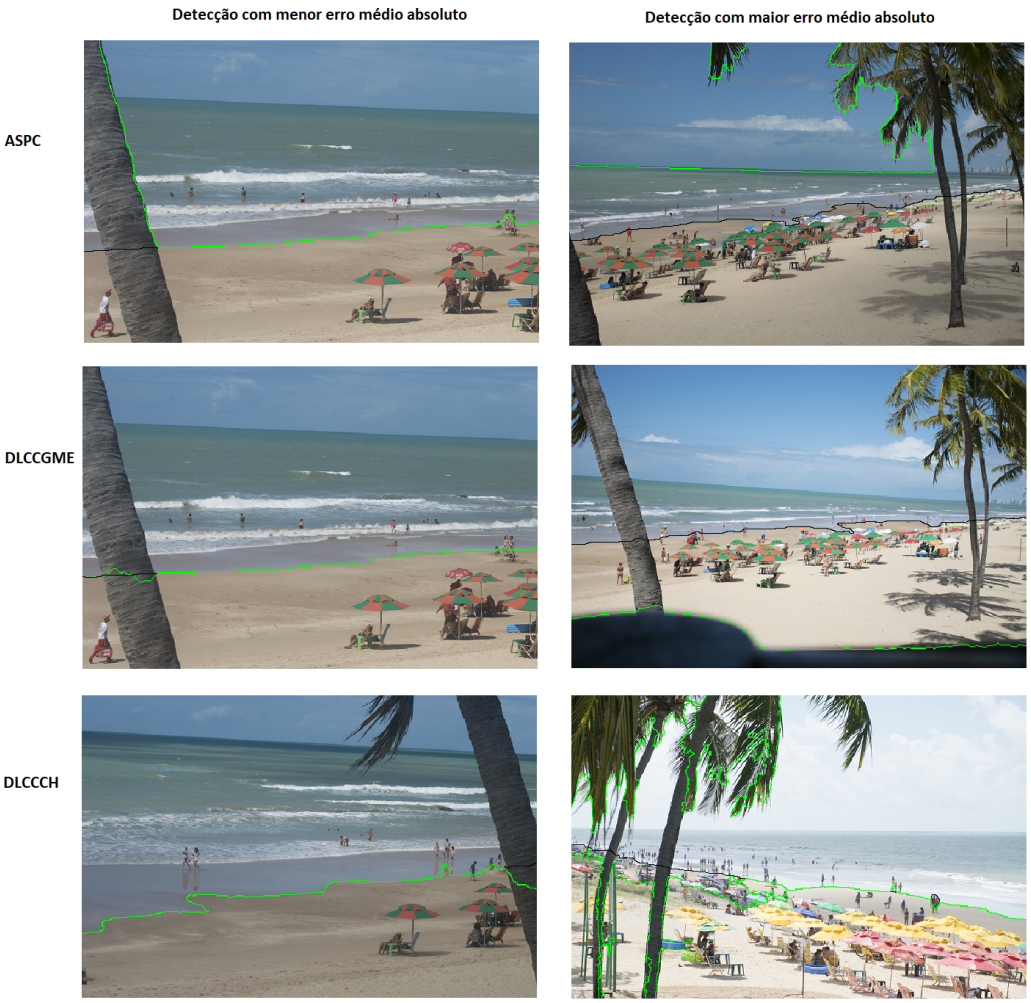
Resultado Para Base de Dados Com Oclusão				
Parâmetros Analisados		S	σ	t
Algoritmo do Estado da Arte	***ASPC***	68,21	30,62	0,41
Algoritmos Propostos	***DLCCGME***	**11,77**	29,18	6,08
	DLCCCH	49,79	32,0	**0,16**

Analisando a Tabela 5 com resultados para base de dados com oclusões o algoritmo que apresentou melhor resultado para media dos erros absolutos foi o DLCCGME e também com o melhor desvio padrão. Com relação ao tempo médio de processamento o melhor resultado foi o DLCCCH.

Na Figura 56 as linhas detectadas estão em verde e as rotuladas em preto, caso tenha acerto, a linha rotulada é sobrescrita pela linha detectada, nessa figura são apresentados exemplos de imagens que tiveram as menores taxas de erro médio absoluto na primeira coluna e as maiores taxas na segunda coluna, em uma analise visual podemos constatar que na primeira coluna os três algoritmos apresentaram ótimos resultados com uma pequena desvantagem para o ASPC que não atravessa o coqueiro

perdendo uma pequena parte da linha da costa, na segunda coluna os três algoritmos não apresentam bons resultados para os piores casos, o DLCCGME apresentou um resultado ruim devido a sobra que aparece em primeiro plano.

Figura 56 – Exemplos de detecções da linha da costa com oclusões por árvores



Fonte – O autor

6.2.3 Discussão

No desenvolvimento dos algoritmos propostos foram realizados vários experimentos com o intuito de selecionar os melhores parâmetros que obtivessem os melhores resultados os quais foram mostrados nas seções anteriores.

Ao analisar os resultados obtidos, é percebível que os algoritmos que utilizam grafos multiestágios denso (AHMAD et al., 2015) e o sistema de cores HSV, foram os que obtiveram os melhores resultados dentre todos os experimentos para o erro médio absoluto, mas em relação ao custo computacional ele se mostrou um algoritmo bastante custoso, e a maior parte desse custo está na geração do grafo quando a

parti da imagem é gerado o grafo multiestágio denso, gastando em média cerca de cinco segundo para uma imagem de 542×360, o qual para essa imagem são gerados 195.122 vértices e 584.818 arestas, a medida que o tamanho da imagem cresce o custo computacional também cresce.

Nos experimentos do tipo H, o algoritmo DLHSTH obteve um diferença de tempo muito grande em relação aos demais algoritmos, obtendo uma tempo médio de processamento na ordem de dezena de milissegundo, esse desempenho se deve ao fato de apenas utilizar métodos de processamento de imagens da biblioteca *Opencv* que é uma biblioteca otimizada para processamentos de imagens.

Tabela 6 – Visualização da média dos experimentos H-1 e H-2.

Média dos Experimentos H1 e H2			
	Média S	**Média** σ	**Média** t
DCSI	26,45	31,16	73,31
Lie *et al.*	30,18	33,23	160,49
DLHCGME	**1,54**	2,56	6,6
DLHSTH	**1,54**	12,26	**0,025**

A Tabela 6 mostra o a média dos experimentos do tipo H, levando em considerações apenas a média dos erros médios absolutos e o desvio padrão dos erros médios absolutos o método que apresenta melhor resultado na detecção da linha do horizonte para as duas bases de dados como mostra a Tabela 6 é DLHCGME pois ele apresentar melhores resultados S e σ, e a segunda melhor média para os dois experimentos foi o DLHSTH que apresenta a melhor média de tempo de processamento.

Tabela 7 – Visualização da média dos experimentos C-1 e C-2.

Média dos Experimentos C1 e C2			
	Média S	**Média** σ	**Média** t
ASPC	38,84	21,21	0,44
DLCCGME	**12,23**	21,55	6,52
DLCCCH	30,12	21,79	**0,17**

Para de detecção da linha da costa apesar do DLCCGME e DLCCCH obterem resultados inferiores ao ASPC para base de dados de imagens sem oclusões os três algoritmos apresentaram valores próximos com uma diferença máxima de 3,25 *pixels* por coluna de erro médio absoluto e para base de dados com oclusões, o DLCCGME mostrou a melhor taxa de erro com um diferença bastante significativa em relação ao demais. Observando a média dos experimentos C-1 e C-2 na Tabela 7 o DLCCGME apresentou a melhor média para S, na coluna da média do σ as três abordagens apresentaram valores próximos com um vantagem para o ASPC, em ralação a média de tempo t o DLCCCH apresentou melhor desempenho.

Através das Tabelas 6 e 7 podemos notar que os algoritmos propostos nesse trabalho tiveram resultados significativos, mostrando a importância dos métodos propostos.

A abordagens proposta por Lie *et al.* (LIE et al., 2005) por usar apenas tons de cinza para gerar o mapa de bordas é requerido que a imagem apresente um alto contraste entre o céu e o mar. Outro fato é o grafo multiestágio, esse grafo não é denso, e há possibilidade de não encontrar o caminho para linha do horizonte, caso o mapa de bordas tenha uma descontinuidade muito grande como ocorre por oclusão por coqueiros.

6.2.3.1 Limitações

O algoritmo DCSI abordado nos experimentos H1 e H2 é o único abordado nesse trabalho que utiliza aprendizagem de máquina, ele apresentou bons resultados no experimento H1, mas nos experimentos H2 não obteve bons resultados, devido a oclusão que criar uma grande área sem *pixels* pertencente a linha do horizonte, criando um região com vértices de custo alto, na detecção da linha do horizonte o algoritmo do caminho mínimo pode seguir outro caminho de menor custo mas que não corresponde a linha do horizonte. Como esse algoritmo realiza a classificação dos *pixels* e depois gerar o grafo multiestágios denso o tempo médio de processamento é degradado devido a essas duas abordagens.

O DLHCGME é baseado no operador *Canny* para isso é necessário que a imagem apresente um bom contraste entre o mar e o céu, caso o contraste entre essas regiões seja pequeno ao ponto do detector de bordas não identificar, a linha do horizonte não seria detectada corretamente. Outro caso são as oclusões que podem realizar desvio na linha detectada, devido a criação de outras bordas.

O método DLHSTH a presenta ótimos resultados mesmo para imagens com oclusões devido ao uso da transformada de *Hough*, que encontra uma linha reta, mas um problema para esse algoritmo são as ondas formadas pelo mar, onde muitas vezes as ondas podem se estender da borda esquerda para borda direita da imagem, e são identificadas como bordas pelos detectores de bordas, e apresentando acumuladores maiores que o da linha do horizonte, fazendo assim o algoritmo identificar uma onda como a linha do horizonte.

Analisando o ASPC, o primeiro problema identificado é na posição da câmera, pois cada câmera estará em posições diferentes, podendo ter distâncias para água diferentes, e em algumas imagens apresentarem a região de água de tamanhos muito discrepantes, o que altera os valores de k e do desvio padrão, podendo assim perder regiões de água na avaliação do desvio padrão de uma região. Outo problema é a perda de regiões de água entre as árvores devido ao método do contorno externos.

No DLCCGME, um problema que pode contribuir para identificação incorreta da linha da costas são arrecifes, pois caso tenha região de água antes do arrecife essa região pode ser perdida, devido ao método encontrar o caminho mínimo, pois identificaria o arrecife como um caminho de menor custo. Um outro problema é a ocorrência de sombra, que pode ocorrer se estender da esquerda para direita da imagem criando bordas com caminho de menor custo que a borda da linha da costa. Uma limitação desse algoritmo é a exigência que as linhas do horizonte e da costa devam se estender da borda esquerda para borda direita da imagem.

Um problema enfrentado pelo DLCCCH é limiarização, pois devido a variedade de cenários, nem sempre os parâmetros utilizados no operador de limiarização é capaz de separar a região da areia da região do mar e céu, muitas vezes parte da areia é identificada como sendo de outra região. Um outro problema são as oclusões por árvores, pois o mar é divido em regiões menores, e essas regiões menores do mar são descartadas, perdendo essas regiões que muitas vezes contém pessoas.

7 Conclusão

Este trabalho aborda a detecção de linhas do horizonte e da costa em imagens de praia, visando segmentar a região de mar como parte do sistema de detecção de banhista da praia de Boa Viagem, em Recife. Foram analisados vários algoritmos a fim de encontrar os melhores algoritmos para detecção das linhas do horizonte e da costa, localizar essas linhas é uma tarefa complexa devido ao cenário de praia ser dinâmico e está em constante mudanças, um câmera na mesma posição obtém imagens diferente em luminosidade, características do mar, clima e entre outras.

Esse projeto abordou sete algoritmos, sendo quatro para detecção da linha do horizonte e três para detecção da linha da costa, os abordados para detecção da linha do horizonte foram: DCSI (AHMAD et al., 2015), Lie *et al.* (LIE et al., 2005), DLHCGME e DLHSTH, esses dois últimos são contribuições desse trabalho, para os algoritmos de detecção da linha da costa não foram encontrado trabalho publicado na literatura apenas uma monografia de Carrera (CARRERA, 2017) o qual apresenta um algoritmo de segmentação da praia apresentado nesse trabalho por ASPC, outros dois algoritmos são contribuições desse trabalho o DLCCGME baseado em detectores de bordas e grafo multi-estágios denso e o DLCCCH baseado em detectores de bordas e transformada de *Hough*.

Os resultados para detecção da linha do horizonte mostram que o DLHCGME obteve a menor taxa de erro médio absoluto e desvio padrão, mas quando ao tempo de processamento o DLHSTH seria mais indicado para sistema de monitoramento em tempo real. Mesmo com resultado muito bom para o erro médio absoluto, o DCSI exige um tempo de processamento muito grande, foram necessário cerca de 70,8 segundos para processar uma única imagem.

A partir dos resultados dos experimentos para detecção da linha da costa para base de dados sem oclusão, ficou constatado que os três algoritmos obtiveram taxas de erros próximas, o ASPC obteve a menor taxa de erro médio absoluto com 9,45, o DLCCCH obteve 10,18 e o DLCCGME 12,7. Para imagens com oclusões o DLCCGME obteve o melhor resultado com um erro médio absoluto de 11,77, o DLCCCH apresentou um resultado de 49,79 e o ASPC foi o obteve o pior resultado com uma taxa erro de 68,21.

É importante notar que os algoritmos propostos neste trabalho obtiveram resultados satisfatórios, pois para detecção da linha do horizonte foram os que obtiveram os melhores resultados e na detecção da linha da costa em imagens com oclusões o algoritmo DLCCGME obteve melhor resultado uma vez que é o único abordado que

consegue atravessar oclusões e conseguindo um resultado bem próximo para as duas bases de dados.

Apesar dos bons resultados dos algoritmos propostos, os baseados em grafo multi-estágio sofrem com o custo computacional os limitando ao processamento em tempo real, que poderia ser resolvido com programação paralela em trabalhos futuros.

7.1 Trabalhos futuros

Para trabalhos futuros uma proposta seria usar programação paralela tanto para melhorar o tempo de processamento dos algoritmos de grafo multi-estágios existente quanto para propor novas abordagens, abrindo possibilidade para uso GPU (*Graphics Processing Unit*).

Avaliar uso de algoritmos de rede neurais profundas em GPU para segmentação do mar, com objetivo de obter melhores taxas de erros médios absoluto e tempo de processamento, onde esse algoritmos devem ser simples o suficientes para funcionar em tempo real.

Referências

AHMAD, T. et al. An edge-less approach to horizon line detection. In: IEEE. *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. [S.l.], 2015. p. 1095–1102. Citado 15 vezes nas páginas 19, 37, 40, 41, 45, 53, 54, 55, 61, 68, 71, 73, 79, 84 e 88.

AHMAD, T. et al. An experimental evaluation of different features and nodal costs for horizon line detection. In: SPRINGER. *International Symposium on Visual Computing*. [S.l.], 2014. p. 193–205. Citado na página 38.

AHMAD, T. et al. Coupling dynamic programming with machine learning for horizon line detection. *International Journal on Artificial Intelligence Tools*, World Scientific, v. 24, n. 4, p. 1540018, 2015. Citado 2 vezes nas páginas 37 e 40.

AHMAD, T. et al. Comparison of semantic segmentation approaches for horizon/sky line detection. In: IEEE. *2017 International joint conference on neural networks (IJCNN)*. [S.l.], 2017. p. 4436–4443. Citado na página 73.

BING, T. et al. Based on fusion of edge extraction and adjacent gradient algorithm of skyline detection research. *Journal of Residuals Science & Technology*, v. 13, n. 7, 2016. Citado 2 vezes nas páginas 36 e 40.

CANNY, J. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, Ieee, n. 6, p. 679–698, 1986. Citado na página 27.

CARRERA, A. *Segmentação de multidões para sistema de monitoramento de banhistas com foco no baixo custo computacional*. 2017. Departamento de estatística e informática, Universidade Federal Rural de Pernambuco. Citado 11 vezes nas páginas 19, 41, 57, 58, 59, 60, 61, 69, 76, 82 e 88.

COMMERCIO, J. do. *Ataque de tubarão em Piedade é o 65° registrado em Pernambuco*. 2018. Acessado em 16 de novembro de 2018. Disponível em: <https://jconline.ne10.uol.com.br/canal/cidades/geral/noticia/2018/06/03/ataque-de-tubarao-em-piedade-e-o-65-registrado-em-pernambuco-341825.php>. Citado na página 17.

CORMEN, T. et al. *Algoritmos: Tradução da Segunda Edição Americana*. [S.l.]: Campus, 2002. Citado 3 vezes nas páginas 33, 34 e 35.

CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995. Citado na página 46.

DEMIR, N.; KAYNARCA, M.; OY, S. Extraction of coastlines with fuzzy approach using sentinel-1 sar image. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Copernicus GmbH, v. 41, p. 747, 2016. Citado na página 36.

DEVELOPERS, N. *NetworkX*. Acessado em 2 de outubro de 2018. Disponível em: <https://networkx.github.io/>. Citado na página 72.

DEVELOPERS, N. *NumPy*. Acessado em 20 de outubro de 2018. Disponível em: <https://www.numpy.org/>. Citado na página 72.

FOUNDATION, P. S. *Python 3.6*. Acessado em 28 de Novembro de 2018. Disponível em: <https://www.python.org/>. Citado na página 72.

GANESAN, P. et al. Hsv color space based segmentation of region of interest in satellite images. In: IEEE. *Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on*. [S.l.], 2014. p. 101–105. Citado na página 24.

GERSHIKOV, E. Is color important for horizon line detection? In: IEEE. *Advanced Technologies for Communications (ATC), 2014 International Conference on*. [S.l.], 2014. p. 262–267. Citado na página 38.

GERSHIKOV TZVIKA LIBE, S. K. E. Horizon line detection in marine images: Which method to choose? In: IEEE. *International Journal on Advances in Intelligent Systems, v. 6, n. 1, 2013*. [S.l.], 2013. p. 1–10. Citado 3 vezes nas páginas 18, 39 e 49.

GONZALEZ, R. C.; WOODS, R. *Processamento digital de imagens. tradução: Cristina yamagami e leonardo piamonte*. [S.l.]: São Paulo: Pearson Prentice Hall, 2010. Citado 3 vezes nas páginas 22, 23 e 26.

HARTIGAN, J. A.; WONG, M. A. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, JSTOR, v. 28, n. 1, p. 100–108, 1979. Citado na página 57.

HOROWITZ, E.; SAHNI, S.; RAJASEKARAN, S. *Computer Algorithms*. 2nd. ed. Summit, NJ, USA: Silicon Press, 2007. ISBN 0929306414, 9780929306414. Citado 2 vezes nas páginas 31 e 32.

IBRAHEEM, N. A. et al. Understanding color models : A review. In: . [S.l.: s.n.], 2012. Citado na página 24.

ILLINGWORTH, J.; KITTLER, J. A survey of the hough transform. *Computer vision, graphics, and image processing*, Elsevier, v. 44, n. 1, p. 87–116, 1988. Citado na página 29.

IMAGE.ORG scikit. *Straight line Hough transform*. Disponível em: <http://scikit-image.org/docs/stable/auto_examples/edges/plot_line_hough_transform.html#sphx-glr-auto-examples-edges-plot-line-hough-transform-py>. Citado na página 31.

LIE, W.-N. et al. A robust dynamic programming algorithm to extract skyline in images for navigation. *Pattern recognition letters*, Elsevier, v. 26, n. 2, p. 221–230, 2005. Citado 16 vezes nas páginas 19, 39, 41, 43, 44, 45, 47, 48, 53, 55, 61, 68, 73, 79, 86 e 88.

MA, T.; MA, J. A sea-sky line detection method based on line segment detector and hough transform. In: IEEE. *Computer and Communications (ICCC), 2016 2nd IEEE International Conference on*. [S.l.], 2016. p. 700–703. Citado 2 vezes nas páginas 36 e 40.

MAIA, G.; PORFÍRIO, V. A. O processo de detecção de bordas de canny: fundamentos, algoritmos e avaliação experimental. 2002. Citado na página 28.

OPENCV. *OpenCV: Color conversions*. 2015. Acesso em: 15/05/2018. Disponível em: <http://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html>. Citado 3 vezes nas páginas 23, 52 e 72.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 72.

SINGH, B. P. The project is based on emerging field image processing, in this project a graphical user interface has been designed using the software labwindows. which can process both type of real time image processing and also many formats of images like .jpg, .dat , .bmp etc. In: . [S.l.: s.n.], 2015. Citado na página 23.

VERBICKAS, R.; WHITEHEAD, A. Sky and ground detection using convolutional neural networks. In: . [S.l.: s.n.], 2014. Citado na página 38.

WIKIPEDIA.ORG. *Hough transform*. Disponível em: <https://en.wikipedia.org/wiki/Hough_transform>. Citado na página 30.

WIKIPEDIA.ORG. *Shortest path problem*. Disponível em: <https://en.wikipedia.org/wiki/Shortest_path_problem>. Citado na página 33.

WIKIPEDIA.ORG. *Transformada de Hough*. Disponível em: <https://es.wikipedia.org/wiki/Transformada_de_Hough>. Citado na página 31.

YAZDANPANAH, A. P. et al. Real-time horizon line detection based on fusion of classification and clustering. In: IEEE. *International Journal of Computer Applications, v. 121, n. 10*. [S.l.], 2015. p. 1–7. Citado 3 vezes nas páginas 18, 37 e 40.