UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

JOSÉ MENDONÇA NETO

DETECÇÃO SEMIAUTOMÁTICA DA LINHA D'ÁGUA E DAS MARCAÇÕES DE CALADO DE NAVIOS

VITÓRIA
2022

JOSÉ MENDONÇA NETO

DETECÇÃO SEMIAUTOMÁTICA DA LINHA D'ÁGUA E DAS MARCAÇÕES DE CALADO DE NAVIOS

Parte manuscrita do Projeto de Graduação do aluno **José Mendonça Neto**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Klaus Fabian Côco

VITÓRIA
2022

JOSÉ MENDONÇA NETO

DETECÇÃO SEMIAUTOMÁTICA DA LINHA D'ÁGUA E DAS MARCAÇÕES DE CALADO DE NAVIOS

Parte manuscrita do Projeto de Graduação do aluno **José Mendonça Neto**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovada em 18 de agosto de 2022.

COMISSÃO EXAMINADORA:

Prof. Dr. Klaus Fabian Côco
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Patrick Marques Ciarelli
Universidade Federal do Espírito Santo
Examinador

Prof. Dra. Thaís Pedruzzi do Nascimento
Universidade Federal do Espírito Santo
Examinador

AGRADECIMENTOS

Agradeço em primeiro lugar, à Deus pelas bençãos derramadas sobre mim e por me guiar até aqui. Aos meus pais, por me auxiliarem e me fornecerem todo suporte que precisei, em especial nos momentos mais difíceis. À minha irmã, por estar sempre ao meu lado. À toda minha família, avô e avós, tios e tias, primos e prima, por todo apoio. Também agradeço aos amigos que estiveram por perto e torceram comigo.

Agradeço ao professor Klaus por ter aceitado ser meu orientador e por todo auxílio e disponibilidade prestados no decorrer do trabalho.

Agradeço também à Universidade Federal do Espírito Santo e a todos os professores e funcionários que tive o prazer de conhecer, especialmente do Departamento de Engenharia Elétrica, por contribuírem na minha formação.

RESUMO

A medição de calado é o processo utilizado em navios para determinar a carga a granel contida neles. Durante muito tempo, esse processo foi e ainda segue sendo realizado manualmente, necessitando de mão de obra e demandando bastante tempo, além do risco ao qual o trabalhador é exposto. Outro ponto relevante para a automatização desse processo é o fato das medições serem realizadas por 3 a 4 entidades distintas para fins de auditoria, sendo feitas em dois momentos separados, um na chegada do navio no porto e outra na sua saída. Assim, implementando um sistema de filmagem com detecção automática não haverá necessidade de tantas medições para auditoria, visto que as imagens estarão disponíveis a qualquer momento. Dessa forma, o projeto visa aprimorar e testar métodos de automatização de parte do processo: a detecção das posições da linha d'água e das marcações de escala do calado. Os métodos utilizados envolverão processamento digital de imagens, visto que o banco de dados disponível não possui informações suficientes para que um sistema de inteligência artificial seja implementado. Atualmente existem diversos estudos focados nessa área, porém este busca analisar adversidades presentes nas medições do dia a dia, como reflexos no mar, falta de luz, presença de lodo no casco do navio, entre outros, além de focar a análise em processamentos de imagens em vídeos, que será o caso real. Utilizando das tecnologias de processamento digital de imagens e do *software MATLAB*, com conceitos envolvendo detecção de bordas, transformada de *Hough* e tratamento de iluminação e saturação das imagens, o trabalho busca obter resultados satisfatórios que auxiliem para que essa tecnologia seja utilizada brevemente.

Palavras-chave: Calado. Processamento de imagens. Linha d'água. Marcações de escala.

ABSTRACT

Draft measurement is the process used on ships to determine the bulk cargo contained on them. For a long time, this process was and still is carried out manually, requiring people and demanding a lot of time, besides the risk which the worker is exposed. Another relevant point for the automatization of this process is the fact that the measurements are performed by 3 to 4 different entities for auditing purposes, being made in two separated moments, one in the arrival of the ship at the port and another in its departure. Thus, by implementing a filming system with automatic detection, there will be no need for so many measurements for the auditing, as the images will be available at any time. In this way, the project aims to improve and test methods to automate part of the process: the detection of the waterline and the position of draft scale marks. The methods used will involve digital images processing, since the available data base does not have enough information for the implementation of an artificial intelligence system. There are currently several studies focused on this area, but this one seeks to analyse adversities present in measurements made during the day, such as reflections in the sea, lack of light, presence of sludge on the ship's hull, among others, in addition to focusing the analysis on image processing in videos, which will be the real case. Using image processing technologies and the software MATLAB, with concepts involving edge detection, Hough transform and image treatment, the work seeks to obtain satisfactory results that help this technology to be used soon.

Key-words: Draft. Image processing. Waterline. Scale marks.

LISTA DE FIGURAS

Figura 1 - Representação de um navio e conceitos relacionados a sua estrutura 17
Figura 2 - Marcas de escala do calado utilizando o Sistema Internacional e algarismos arábicos 18
Figura 3 - Exemplo de imagem com baixo contraste e seu histograma 23
Figura 4 - Exemplo de imagem com histograma equalizado e seu histograma..... 24
Figura 5 - Operadores de Sobel 25
Figura 6 - Conversão entre plano cartesiano e plano polar utilizado na Transformada de *Hough* 27
Figura 7 - Fluxograma simplificado do funcionamento do programa 28
Figura 8 - Exemplos de vídeos com baixa qualidade para detecção 30
Figura 9 - Primeiro frame de um dos vídeos do banco de dados..... 32
Figura 10 - *Frame* de um vídeo (a) antes e (b) depois da operação de estabilização 33
Figura 11 – Imagem (a) transformada para escala de cinza e (b) após passar pelo filtro Butterworth de ordem 2 especificado 35
Figura 12 - Imagem com histograma equalizado 36
Figura 13 - Bordas encontradas a partir do filtro de Sobel..... 39
Figura 14 - Exemplo de (a) marcações de escala inclinada e (b) de linha d'água inclinada... 40
Figura 15 - Exemplo de capturas com navios (a) próximos e (b) distantes da câmera 43
Figura 16 - Fluxograma detalhado do funcionamento do programa 45
Figura 17 - Resultados encontrados pelo programa após as detecções 46
Figura 18 - Exemplo do corte realizado na imagem para sequência da análise 47
Figura 19 - Linhas verticais retornadas pela transformada de Hough 48
Figura 20 - Linhas horizontais retornadas pela transformada de Hough..... 49
Figura 21 - Validação após determinada quantidade de linhas detectadas..... 50
Figura 22 - (a) Conjunto das linhas d'água encontradas nos frames do vídeo estabilizado e (b) o resultado final do programa após todas as detecções 51

LISTA DE GRÁFICOS

Gráfico 1 - Histograma do 1º frame do vídeo com os limiares de validação 37
Gráfico 2 - (a) Histograma e (b) histograma equalizado da Figura 8 38
Gráfico 3 - Plano θ-ρ das linhas verticais 48
Gráfico 4 - Plano θ-ρ das linhas horizontais com inclinação (a) 70° a 89,9° e (b) -90° a -70° 49

LISTA DE TABELAS

Tabela 1 - Frota mundial por tipo de navio em 2020-2021 13

SUMÁRIO

1 INTRODUÇÃO **12**
1.1 Objetivos **16**
1.1.1 Objetivo Geral 16
1.1.2 Objetivos Específicos 16
1.2 Organização do trabalho **16**

2 REFERENCIAL TEÓRICO **17**
2.1 Medição do calado do navio **17**
2.2 Conceitos matemáticos **19**
2.3 Tratamento das imagens **20**
2.3.1 Filtros passa-alta 20
2.3.2 Filtragem no domínio da frequência 21
2.3.3 Equalização de histograma 22
2.4 Filtro de Sobel **24**
2.5 Transformada de Hough **25**

3 METODOLOGIA **28**
3.1 Banco de dados **29**
3.2 Estabilização do vídeo **30**
3.3 Tratamento das imagens **33**
3.4 Detecção das linhas **38**

4 RESULTADOS E DISCUSSÕES **46**

5 CONCLUSÃO **53**

REFERÊNCIAS BIBLIOGRÁFICAS **55**

APÊNDICE A – ESTABILIZA.M **57**

APÊNDICE B – HOMOFILTER.M **60**

APÊNDICE C – DETECTA_LINHAS.M **62**

APÊNDICE D – VALIDA_LINHAS.M **66**

APÊNDICE E – MAIN.M 70

1 INTRODUÇÃO

No estágio de globalização no qual o mundo se encontra, diversos produtos precisam ser transportados entre os diferentes países participantes do seu mercado, e muitos desses produtos possuem vendedor e comprador localizados em posições geográficas distantes. Assim sendo, são necessários meios de transporte que tornem possíveis o deslocamento desses produtos independentemente das distâncias e das grandes quantidades que são submetidos.

Nesse contexto, aparecem os navios graneleiros (*bulk carriers*), que são responsáveis pelo transporte de grandes quantidades da carga em granel, ou seja, cargas que não requerem trabalho prévio ao seu transporte. Assim, os processos de carga e descarga do navio são feitos de forma simplificada, por meio de escotilhas que permitem a movimentação de alta parcela da mercadoria e que serão realizados em terminais portuários específicos para estes serviços, facilitando a logística da operação. Mostrando a participação desse tipo de transporte dentro do transporte marítimo, a Tabela 1 apresenta que os navios graneleiros representaram cerca de 42,77% de toda a frota mundial em 2021 (UNCTAD, 2021).

Devido ao seu método de carregamento, existe uma dificuldade ao utilizar esses processos, que é referente à medição da carga que será transportada ou recebida em uma negociação, visto que não é viável a utilização de balanças ou outros métodos gerais, como os utilizados em navios porta-contêineres.

No mercado atual, tal medição é realizada em um processo manual, nomeado Arqueação, ou *Draft Surveying*, que consiste na estimativa do peso da carga, a partir da leitura do calado do navio juntamente com as propriedades do material em questão e do meio em que a embarcação se encontra, utilizando o princípio de Arquimedes, o qual diz que um corpo parcialmente ou totalmente imerso em um fluido está submetido à uma força de empuxo que é igual ao peso do volume de fluido deslocado pelo corpo.

A leitura do calado do navio, além dessa importante função na determinação da quantidade de carga, também é fundamental para a análise da estabilidade do navio, por isso, essa medição deve ser realizada em seis diferentes pontos: na proa, meia-nau e na popa, sendo que cada um

deles está posicionado em ambos os lados da embarcação, tanto à estibordo quanto à bombordo (FONSECA, 2005).

Tabela 1 - Frota mundial por tipo de navio em 2020-2021

Table 2.1	World fleet by principal vessel type, 2020–2021 (thousand dead-weight tons and percentage change)		
Principal types	2020	2021	Percentage change 2021 over 2020
Bulk carriers	879 725	913 032	3.79%
	42.47%	*42.77%*	
Oil tankers	601 342	619 148	2.96%
	29.03%	*29.00%*	
Container ships	274 973	281 784	2.48%
	13.27%	*13.20%*	
Other types of ship:	238 705	243 922	2.19%
	11.52%	*11.43%*	
Offshore supply	84 049	84 094	0.05%
	4.06%	*3.94%*	
Gas carriers	73 685	77 455	5.12%
	3.56%	*3.63%*	
Chemical tankers	47 480	48 858	2.90%
	2.29%	*2.29%*	
Other/not available	25 500	25 407	-0.36%
	1.23%	*1.19%*	
Ferries and passenger ships	7 992	8 109	1.46%
	0.39%	*0.38%*	
General cargo ships	76 893	76 754	-0.18%
	3.71%	*3.60%*	
World total	**2 071 638**	**2 134 640**	**3.04%**

Fonte: Citado em (UNCTAD, 2021). Cálculos da UNCTAD (*United Nations Conference on Trade and Development*), baseados nos dados da pesquisa de Clarkson (Clarksons Research, 2021).

Dessa maneira, a eficiente leitura do calado traz ao mercado dos navios graneleiros uma melhor exatidão aos processos, gerando além de resultados econômicos positivos, um aumento da segurança de todos os envolvidos, devido à toda sua importância no correto funcionamento do navio.

Para o processo realizado para a obtenção da medida do calado, atualmente é necessário que um profissional se aproxime do navio utilizando outra embarcação e avalie a posição de encontro entre a linha d'água e as marcas de escala do calado em cada um dos pontos de

medição. Como os processos de carga e descarga são demorados, podendo levar diversas horas dependendo da quantidade de mercadoria, a medição deve ser realizada repetidas vezes para que uma análise possa ser feita enquanto os procedimentos estão sendo realizados.

Além disso, essas medições são feitas pelo comprador da carga, pelo vendedor, pela receita federal e, quando for o caso, pelo órgão de controle ambiental. Cada qual com seu registro próprio, podendo divergir entre eles causando problemas.

Devido a essa medição ser feita a olho humano e sob qualquer condição, algumas variações indesejadas podem ocorrer, como falta de precisão e de segurança no valor medido. Também deve ser levado em consideração as interferências climáticas que podem aparecer durante os momentos de medição, como agitação da água, chuva, iluminação, entre outros. Nesses casos, o profissional fica submetido a maiores riscos, pois pode ser necessário se aproximar ainda mais do navio devido à falta de visibilidade, ou ainda, não ser possível obter bons resultados da medição.

Avaliando todos esses pontos chega-se ao contexto da inserção de visão computacional e processamento digital de imagens a esses cenários. A partir de câmeras localizadas em posições estrategicamente avaliadas, imagens das posições da linha d'água e das marcações do calado do navio podem ser obtidas, e em conjunto com o processamento das mesmas para interpretação dos dados, é obtida a medida do calado do navio. Com isso, retira-se o perigo enfrentado pelo profissional, diminui o custo da operação, unifica a medição entre os diferentes órgãos, retirando a necessidade dos variados medidores e traz uma maior confiabilidade devido a disponibilidade das imagens em qualquer momento de interesse, como eventuais auditorias.

Outros métodos de aprimoramento dessa medição já foram pensados, como alguns baseados na utilização de diferentes sensores (ZHANG, YU, *et al.*, 2019), de sensores em fibra ótica (IVčE, JURDANA e MOHOVIć, 2011), entre outros. Avaliando essas aplicações, foi notado que fatores como o alto custo relacionado à manutenção, o impacto considerável sofrido pelos resultados devido as variações climáticas e a instalação desses recursos em navios existentes ser cara e trabalhosa, dificultam a utilização desses métodos.

Dessa forma, a implementação do processamento de imagens nesse contexto é de extrema ajuda, em especial no quesito de execução, visto que, os recursos necessários para tal processo são, além de um computador para realizar o processamento digital, uma câmera com capacidade de realizar capturas suficientemente boas para a interpretação das mesmas, iluminação e um sistema de transmissão de dados para que o computador processe as imagens feitas pela câmera localizada próxima ao navio.

Logo, por esses fatores, essa tecnologia se torna muito interessante, já que ela também se mostra, além de menos custosa do que os métodos envolvendo sensores, altamente escalável, pois, com a instalação de um conjunto de poucos equipamentos, em qualquer porto de navios graneleiros, é possível realizar a medição dos calados dos navios que lá atracarem.

Métodos utilizando processamento de imagens também já foram estudados por diferentes pesquisadores e alguns deles inclusive serviram como inspiração para o desenvolvimento do presente trabalho (OKAMOTO, YOSHIDA, *et al.*, 2014), (TSUJII, YOSHIDA e IIGUNI, 2016), (MACHADO, SANTOS e VASSALO, 2018), (MACHADO, 2020).

A utilização de processamento de imagens para a leitura do calado de navios é dividida em basicamente duas etapas, na qual, uma é a detecção da linha d'água e da localização das marcas de escala e a outra é a interpretação dos caracteres das marcações de escala do calado, e então, com essas duas informações em conjunto, conclui-se onde ocorre o cruzamento entre a linha d'água e as marcas de escala, estabelecendo, assim, qual a carga presente no navio.

Em especial, o presente trabalho visa a elaboração de um programa focado justamente na detecção da linha d'água e da região das marcas de escala do calado em vídeos de situações reais, obtendo o ponto de interseção entre as linhas e considerando também adversidades como, por exemplo, iluminação e limo nas embarcações, visto que, muito do que se tem de informação foi concluído com base em amostras contendo imagens equilibradas.

Assim, a ideia principal é desenvolver algo a partir de uma base de dados pequena contendo vídeos de navios em que ocorrem esses procedimentos, de maneira semiautomática, permitindo uma pequena interação com o usuário de forma que o mesmo aprove uma primeira detecção da linha d'água e da região das marcações de escala do calado para, então, o programa seguir com

a análise, tomando como base dados já validados. Isso, porque a realização da detecção utilizando aprendizado de máquinas é dificultada, devido à indisponibilidade de uma larga quantidade de vídeos.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral do trabalho aqui proposto é desenvolver um código que realize a detecção da linha d'água e da região das marcações de escala do calado em vídeos utilizando técnicas de processamento de imagens e uma base de dados pequena para utilização de aprendizado de máquina, a partir do software MATLAB (MATHWORKS, 2020).

1.1.2 Objetivos Específicos

Os objetivos específicos que auxiliam atingir o objetivo geral estão listados a seguir:

- Implementar rotinas de detecção de bordas verticais e horizontais em imagens de navios.
- Implementar rotinas de tratamento das bordas obtidas, aprimorando as linhas encontradas.
- Validar as bordas encontradas de maneira automática e retornar o ponto de interseção entre a linha na vertical e a média das linhas na horizontal.
- Avaliar as situações que geram os maiores erros e buscar melhorias para a implementação.

1.2 Organização do trabalho

A fim de apresentar os resultados obtidos com este trabalho, o texto a seguir está estruturado da seguinte forma. A Seção 2 explora o referencial teórico, definindo conceitos que foram utilizados durante a elaboração do trabalho. A seguir, a Seção 3 apresenta a metodologia aplicada na solução proposta, descrevendo todos os passos realizados. Tendo apresentado o método, a Seção 4 detalha os resultados obtidos após a realização dos testes. Para concluir, a Seção 5 apresenta as conclusões e comenta sobre trabalhos futuros e possibilidades de melhorias.

2 REFERENCIAL TEÓRICO

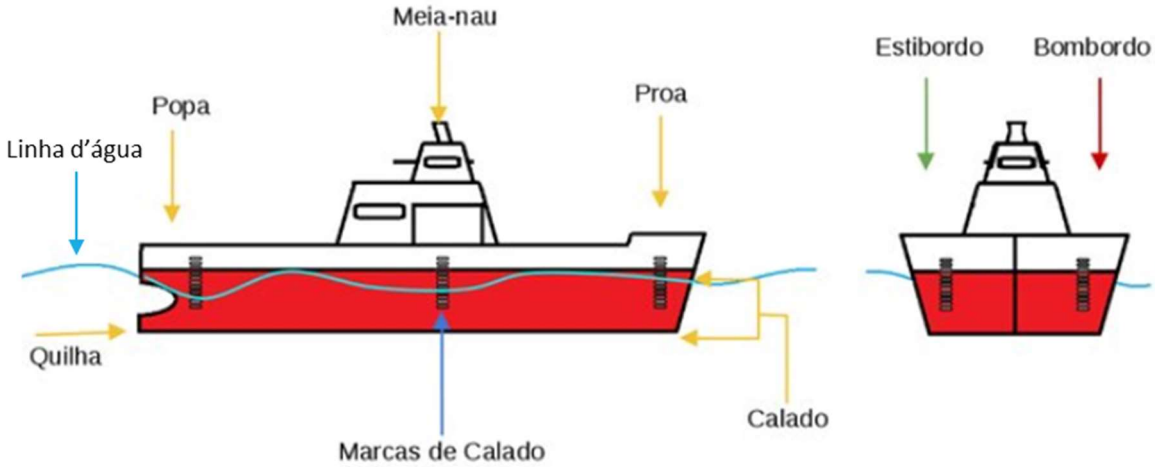
2.1 Medição do calado do navio

Calado d'água é definido pela distância vertical entre a superfície da água e a parte mais baixa do navio, no ponto escolhido para medição, ou seja, o calado se dá pelo comprimento da parte submersa do navio. Essa medida possui diferentes valores de acordo com a quantidade de carga e de equipamentos presentes na embarcação, assim, tem-se tanto um calado mínimo, correspondente à condição de deslocamento leve, quanto um calado máximo, que é referente à condição de deslocamento em plena carga (FONSECA, 2005).

A chamada linha d'água se refere a superfície do mar que se encontra em contato com o casco dos navios. Essa linha será responsável por indicar os calados nos diferentes pontos da embarcação a partir do ponto no qual essa linha intercepta as marcações de escala de calado.

Os conceitos definidos acima podem ser observados na ilustração apresentada na Figura 1.

Figura 1 - Representação de um navio e conceitos relacionados a sua estrutura



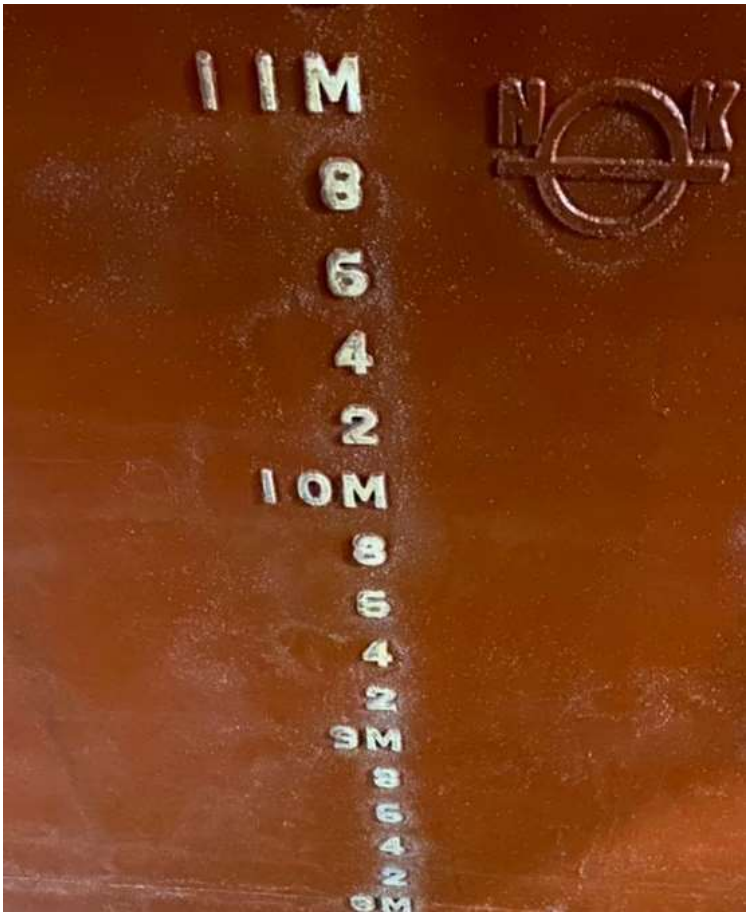
Fonte: Adaptado de (MACHADO, SANTOS e VASSALO, 2018).

Quanto às marcações de escala do calado, tem-se que se apresentam em escala alfanumérica, no qual seus algarismos são entalhados na superfície das capas, pintados ou fundidos em metal ou presos por parafusos ao casco do navio. Os algarismos presentes nas marcas de escala

possuem cores opostas as cores do casco do navio a fim de que a leitura e medição sejam facilitadas (FONSECA, 2005).

As escalas utilizadas podem seguir métricas diferentes, como o Sistema Internacional, que representa as distâncias medidas em metros e decímetros, ou o Sistema Inglês, no qual as medidas são feitas em pés e polegadas. Essas escalas também podem apresentar algarismos diferentes, sendo eles romanos ou arábicos. O caso mais comum encontrado é o que utiliza o Sistema Internacional com algarismos arábicos, em que ao lado de cada medida inteira de metro, tem-se uma indicação feita pelo caractere “M”, a Figura 2 exemplifica essa representação.

Figura 2 - Marcas de escala do calado utilizando o Sistema Internacional e algarismos arábicos



Fonte: Banco de dados do projeto.

Relacionado ao posicionamento dessas marcações, tem-se que todas as embarcações de médio e grande porte possuem de 4 a 6 marcas de escala do calado ao redor de sua estrutura, dada sua

importância (FONSECA, 2005). Um fator relevante, é que as posições e quantidades de marcas podem diferir. Em algumas embarcações são encontradas apenas as medições à estibordo e bombordo, na proa e na popa, sendo necessária uma estimativa para a medida do calado a meia-nau.

Usualmente, devido aos seus tamanhos, as distâncias entre as marcas da proa e da popa são longas e, assim, faz-se necessária a marcação também a meia-nau. Dessa maneira, em cada flutuação de navios graneleiros, podem ser realizadas medições do calado em seis posições diferentes, duas na proa, para ter a referência da flutuação da parte frontal tanto a estibordo quanto a bombordo, outras duas na popa, para referenciar a parte traseira do navio e outras duas a meia-nau, que são feitas na parte central da distância entre os calados na proa e na popa, como apresentado na Figura 1.

Além disso, de acordo com um profissional que atua na área de medição de calado, para que uma medição seja validada é necessário que, em casos de mares agitados, a medição seja realizada durante 10 ciclos de onda e após esses ciclos, faz-se o cálculo da média dos valores obtidos e então, tem-se a medida da linha d'água, usada para aferir a medida na marcação do navio. Nos casos de mares calmos, a contagem dos ciclos de onda não é utilizada, visto que pelos ciclos serem menos distintos entre si, as medidas obtidas são mais próximas, então, o parâmetro utilizado nesse caso é um tempo equivalente a aproximadamente 60 segundos de observação e assim a medida será obtida pela média dos valores observados.

2.2 Conceitos matemáticos

Como as operações de carga e descarga dos navios nos portos ocorrem independente das condições climáticas, existirão situações nas quais as imagens obtidas não serão as mais apropriadas. Nesses casos, pode ser necessário maior intervenção humana no processo. Por isso, o primeiro passo ao iniciar a detecção das linhas é realizar a estabilização do vídeo de forma a manter as marcações de escala do navio numa mesma posição, para que um referencial seja definido.

Para isso, são realizadas transformações geométricas entre duas imagens de forma que os pontos relacionados entre elas sirvam como referenciais para que uma se alinhe à outra. Além disso,

também é analisada a escala de cada uma das imagens e sendo diferentes, elas também são alteradas.

Para o estudo em questão as transformações geométricas utilizadas foram translação e rotação. Na qual a translação é definida pela movimentação da figura no plano com relação a um vetor, obtendo uma nova figura simétrica e congruente à original. Já a rotação é dada pela movimentação da figura em torno de um ponto, a partir de uma medida de abertura de ângulo com um sentido específico, obtendo também uma figura simétrica e congruente à original (DANTE, 2018).

Já a escala é dada pela transformação das medidas dos componentes da imagem, sendo possível aumentá-los ou reduzi-los de acordo com a variação da escala.

Outro conceito utilizado no trabalho é o de função afim, que é dada por qualquer função polinomial do primeiro grau. Possuindo dois coeficientes, linear e angular, o gráfico gerado por essas funções são retas, onde o coeficiente linear indica a posição em que a reta intercepta o eixo das ordenadas e o coeficiente angular indica a inclinação dessa reta.

2.3 Tratamento das imagens

A seguir, outro processo importante na qualificação das imagens é o tratamento delas, de forma a extrair o máximo de informação útil.

O foco do tratamento citado é evidenciar as formas mais finas na imagem, isto é, as bordas dos objetos, tais como a linha d'água e os algarismos das marcas de escala. Essas formas finas são caracterizadas por variações abruptas de intensidade, que no domínio da frequência se relacionam a altas frequências. Dessa forma, utilizando um filtro passa-alta, as bordas serão destacadas e assim a interpretação do detector das bordas será facilitada e consequentemente os esforços computacionais serão reduzidos.

2.3.1 Filtros passa-alta

Esses filtros irão remover algumas informações mais gerais, como fundos e regiões com uma textura uniforme, que são caracterizados por baixas frequências, removendo também a

componente constante (dc) da imagem e em virtude disso, a imagem filtrada terá suas bordas destacadas, porém devido a retirada do valor médio parte da iluminação da imagem será perdida (GONZALEZ e WOODS, 2008).

Os filtros estudados que realizam tal função são os filtros Ideal, Butterworth e Gaussiano, no qual o Ideal realiza uma filtragem abrupta, o Gaussiano uma filtragem mais suave e o filtro Butterworth possui um parâmetro de ordem, que com valores baixos se aproxima mais de um filtro Gaussiano e com valores altos de um filtro Ideal.

Para a situação em questão o filtro utilizado foi o filtro Butterworth, pelo fato de ser mais maleável e permitir testar diferentes ordens e avaliar qual se ajusta melhor. Como esse filtro atua no domínio da frequência é preciso realizar alguns passos para que a filtragem seja realizada corretamente.

2.3.2 Filtragem no domínio da frequência

Inicialmente, é preciso efetuar o *zero padding* da imagem a fim de evitar o erro de *wraparround*, dado pela sobreposição de uma função sobre outra ao realizar o processo de convolução, pelo fato do distanciamento entre elas não ser grande o suficiente para que uma função consiga se deslocar completamente sem sobrepor a outra. O *zero padding* se trata do preenchimento das bordas da imagem com *pixels* de intensidade 0, de forma que as dimensões da nova imagem, $f_p(x, y)$, estenda para $P\ x\ Q$, normalmente o dobro das dimensões da imagem original, $M\ x\ N$.

Em seguida, para centralizar a transformada de Fourier da imagem com o *zero padding*, multiplica-se $f_p(x, y)$ por $(-1)^{(x+y)}$, e, então, calcula-se a transformada da imagem deslocada, encontrando $F(u, v)$.

Calculada a função da imagem no domínio da frequência é necessário encontrar a função do filtro que será aplicado sobre ela. No caso do filtro Butterworth passa-baixa, sua função de transferência é dada pela Equação (1):

$$H_{LP}(u,v) = \frac{1}{1 + \left[\frac{D(u,v)}{D_0}\right]^{2n}} \tag{1}$$

onde n é a ordem do filtro, D_0 é a distância da origem até a frequência de corte e $D(u,v)$ é dado pela Equação (2):

$$D(u,v) = \sqrt{\left(u - \frac{P}{2}\right)^2 + \left(v - \frac{Q}{2}\right)^2} \tag{2}$$

porém, como o filtro desejado é um filtro passa-alta, é necessário calculá-lo antes da filtragem, realizando a operação definida na Equação (3).

$$H_{HP}(u,v) = 1 - H_{LP}(u,v) \tag{3}$$

Multiplicando a transformada de Fourier da imagem pela função de transferência do filtro passa-alta, chega-se em $G(u,v)$. Calculando a transformada inversa, e multiplicando a parte real dela por $(-1)^{(x+y)}$, chega-se na imagem filtrada no domínio espacial. Por fim, basta extrair a imagem com as dimensões originais $(M\ x\ N)$ do quadrante superior esquerdo da imagem filtrada (GONZALEZ e WOODS, 2008).

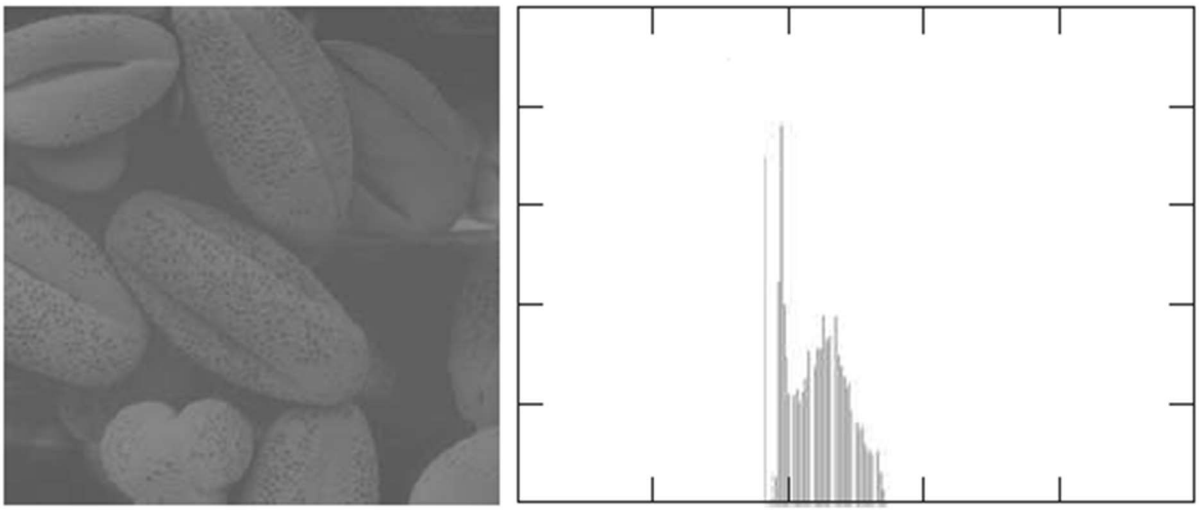
2.3.3 Equalização de histograma

Como citado anteriormente, um problema atrelado ao uso de filtros passa-alta é a perda do valor médio de intensidade na imagem filtrada e, para isso, o realce de contraste é realizado. Diversas técnicas podem ser aplicadas para essa função, como a transformação logarítmica, a transformação gama e a equalização de histograma. As duas primeiras técnicas possuem um parâmetro de ajuste que permite ao usuário definir formas de aplicação da transformação para que elas afetem a imagem focando na parte de interesse. Já a terceira utiliza informações presentes na imagem para realizar o realce.

Dessa forma, para a aplicação em questão, a técnica escolhida foi a equalização do histograma, devido a não participação do usuário e sua grande adaptação para cada imagem.

O histograma de uma imagem digital é uma função discreta que representa a distribuição de frequência para cada nível de intensidade da imagem, no qual seu eixo das abscissas é composto pelas diferentes intensidades e o eixo das ordenadas pelas quantidades de *pixels* que cada nível possui. A Figura 3 apresenta um exemplo de imagem em escala de cinza com baixo contraste e seu respectivo histograma.

Figura 3 - Exemplo de imagem com baixo contraste e seu histograma



Fonte: Adaptado de (GONZALEZ e WOODS, 2008).

Essa função é de extrema importância no processamento de imagens, visto que é a base de diversas aplicações no domínio espacial. A manipulação de histogramas pode ser utilizada para realce de contraste, que será o caso em estudo, mas também para compressão e segmentação de imagens, a partir dos dados estatísticos contidos nele.

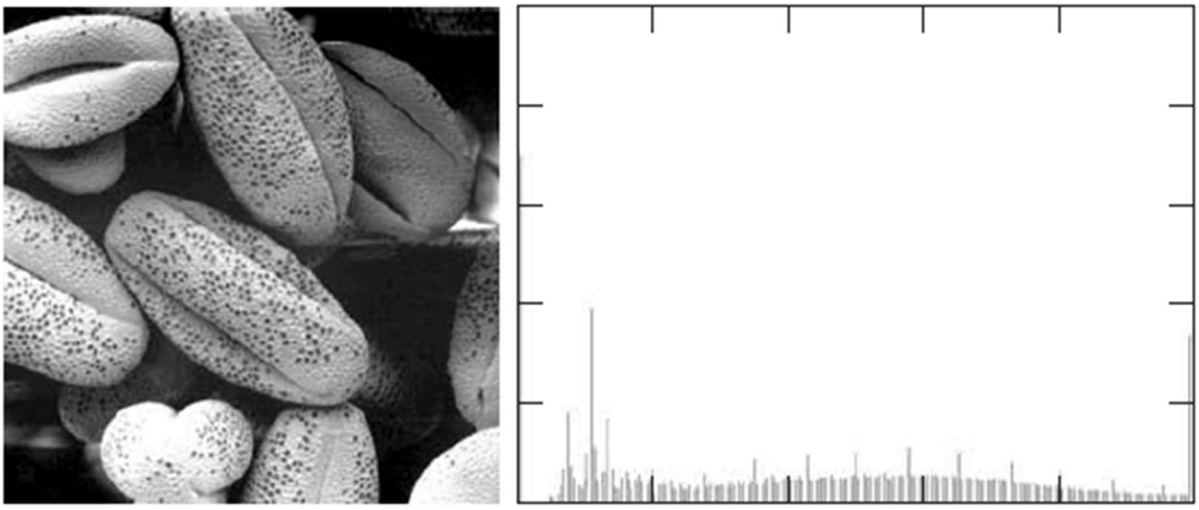
A fim de recuperar o contraste, o processo aplicado sobre o histograma é o de equalização, isto é, equilibrar as quantidades de *pixels* para cada uma das intensidades. Para isso, é necessário calcular uma função de transformação que será responsável por indicar qual a intensidade de cada pixel na imagem equalizada. Essa função, na forma discreta, é dada pela Equação (4):

$$T(r_k) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j \qquad k = 0,1,2,3,\dots,L-1 \tag{4}$$

onde r_k representa todos os níveis de intensidade da imagem original, L a quantidade de intensidades da imagem, M a dimensão vertical, N a dimensão horizontal e n_j os *pixels* com cada nível de intensidade (GONZALEZ e WOODS, 2008).

Assim, realizando o processo de transformação dos valores de intensidade para cada um dos *pixels*, chega-se na imagem equalizada. Para o caso exemplificado na Figura 3, ao realizar a equalização de histograma da mesma, chega-se no resultado apresentado na Figura 4, na qual percebe-se a imagem equalizada com um melhor contraste e seu respectivo histograma.

Figura 4 - Exemplo de imagem com histograma equalizado e seu histograma



Fonte: Adaptado de (GONZALEZ e WOODS, 2008).

2.4 Filtro de Sobel

As bordas nas imagens costumam representar características importantes delas, como transição entre o objeto e o fundo, mudança das propriedades do objeto, variações na iluminação da cena, entre outras.

Para a detecção das bordas nas imagens do presente trabalho, o método utilizado foi o filtro de Sobel, o qual utiliza técnicas de gradiente para realizar a detecção.

O gradiente de uma função $f(x, y)$ é dado pela Equação (5):

$$\nabla \mathrm{f} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{5}$$

Como a derivação de imagens realiza um efeito de destaque sobre os detalhes da mesma, a magnitude desse vetor gradiente é a base para várias abordagens de derivação de imagens.

No caso dos operadores de Sobel, as máscaras utilizadas são as apresentadas na Figura 5. Nota-se que a soma dos coeficientes de cada máscara é zero, como é esperado de um operador derivativo.

Figura 5 - Operadores de Sobel

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Fonte: (GONZALEZ e WOODS, 2008).

A partir desses cálculos, os pontos da imagem em que os gradientes calculados possuem valor máximo são retornados como bordas.

2.5 Transformada de Hough

A transformada de *Hough* é um método que foi desenvolvido inicialmente para detecção de linhas em fotografias (HOUGH, 1962) e que evoluiu para outras formas geométricas, sendo hoje utilizado para detecção de diferentes formas e ligação de linhas desconexas em uma imagem. Esse método se destaca por sua robustez quanto a ruídos presentes nas imagens e pela capacidade de sucesso na detecção de linhas mesmo em situações de descontinuidades, caracterizadas por falta de *pixels* de mesma intensidade em determinada região.

Esse método se utiliza da fórmula matemática de determinada forma, para buscar por padrões descritos pela mesma fórmula em todos os pontos da imagem, realizando uma leitura em todas

as direções de determinado *pixel*, para que se aquela forma estiver presente no ponto de analise, ela seja detectada. A partir disso, são acumulados os pontos em que as formas foram detectadas para futura análise.

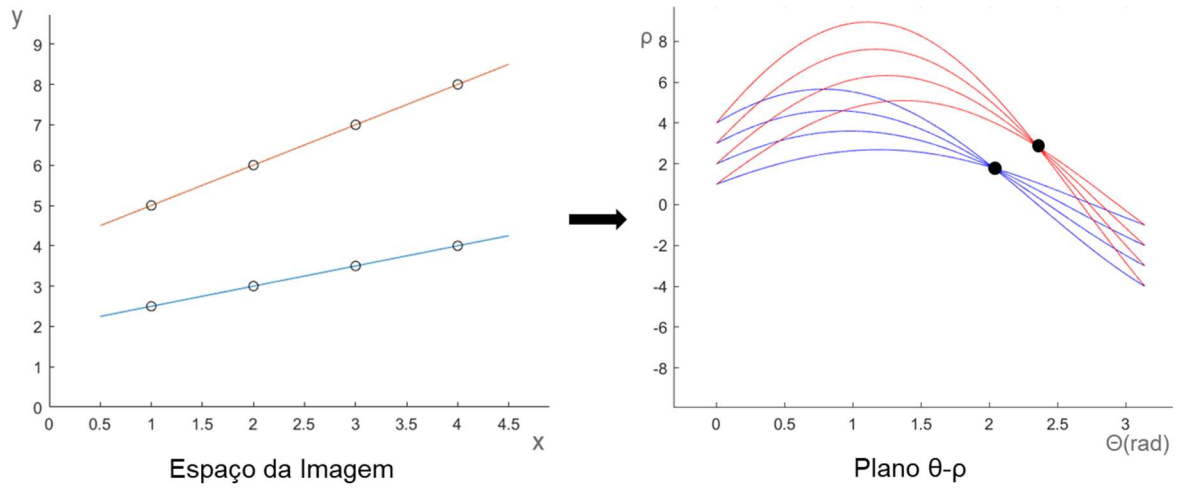
A transformada de *Hough* que será usada neste trabalho é para detecção de retas. A partir disso, pode-se definir a equação matemática que descreve uma reta e realizar uma busca a fim de encontrar essas linhas avaliando cada *pixel* das imagens.

O método aplicado nos algoritmos atuais que buscam por essas retas faz uso da parametrização proposta em (DUDA e HART, 1972), em que se propõe a utilização da equação polar da reta, descrita pela Equação (6).

$$\rho = x.\cos(\theta) + y.\mathrm{sen}(\theta) \tag{6}$$

Assim, todo ponto no plano cartesiano é descrito por uma curva no plano $\theta - \rho$ e dessa maneira, um problema que envolve detecção de pontos colineares na imagem passa a ser um problema de encontrar curvas que se interceptam no plano $\theta - \rho$. A partir disso, essas interseções de curvas sob coordenadas polares são contadas em um acumulador, e então quanto mais curvas uma interseção tiver, mais pontos colineares a linha terá na imagem. A Figura 6 apresenta como essa detecção sob coordenadas polares ocorre, visto que observando o plano polar mostrado à direita tem-se quatro curvas em azul que representam cada um dos quatro pontos destacados na reta azul do plano cartesiano, convertidos a partir da Equação (6) e utilizando um intervalo de 0 até π para o eixo θ, o mesmo acontece para a análise da reta vermelha do plano cartesiano com relação às linhas vermelhas no plano polar.

Figura 6 - Conversão entre plano cartesiano e plano polar utilizado na Transformada de *Hough*



Fonte: Produção do próprio autor.

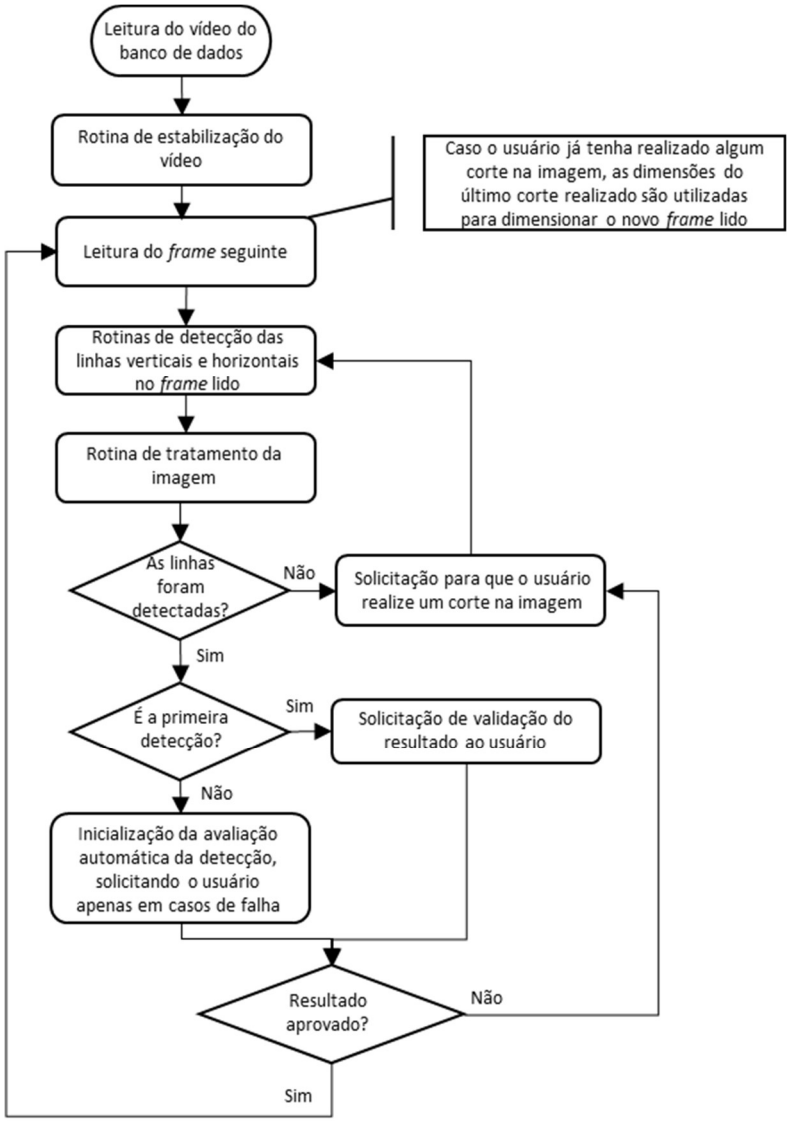
Para defeitos nas bordas causados por ruídos e iluminação irregular, assim como outros efeitos de descontinuidades em escala de cinza, a transformada de *Hough* é geralmente utilizada para converter esses *pixels* desconectados de borda em segmentos contínuos significativos.

3 METODOLOGIA

A fim de alcançar os objetivos traçados, este trabalho foi dividido em diferentes etapas nas quais foram implementados códigos na linguagem Matlab no próprio *software* (MATHWORKS, 2020) para que as detecções e análises sejam realizadas.

O fluxograma apresentado na Figura 7 explica, de forma simplificada, como ocorre o funcionamento do programa, apresentando as diferentes etapas que serão detalhadas no decorrer do trabalho.

Figura 7 - Fluxograma simplificado do funcionamento do programa



Fonte: Produção do próprio autor.

3.1 Banco de dados

O primeiro processo realizado foi a análise do banco de dados disponível para o projeto. Inicialmente, viu-se que o mesmo continha um total de 54 vídeos de diferentes ambientes, com iluminação variada e cascos de navios com características distintas. Esses vídeos foram obtidos por empresas que realizam os processos de carga e descarga de navios graneleiros e foram fornecidos como material de estudo.

Os vídeos contidos no banco de dados apresentam em torno de 1000 *frames*, com duração próxima de 20 segundos e dimensão com altura de 848 *pixels* e largura de 480 *pixels*. Também foi possível perceber que alguns desses vídeos não possuíam qualidade suficiente para realizar a detecção.

Em seguida, fez-se uma classificação desse banco de dados e concluiu-se que 36 vídeos apresentavam qualidade boa para detecção e 18 apresentavam baixa qualidade. Os vídeos classificados como de baixa qualidade, durante o processo de avaliação do algoritmo, foram desconsiderados. O primeiro *frame* de dois exemplos de vídeos com baixa qualidade são apresentados na Figura 8, sendo classificados como ruins pelo fato de as marcas de escala do calado estarem muito inclinadas.

Figura 8 - Exemplos de vídeos com baixa qualidade para detecção

Fonte: Banco de dados do projeto.

3.2 Estabilização do vídeo

Para uma correta detecção das linhas d'água é ideal que as medidas sejam feitas com base em uma mesma referência, dessa forma, foi implementada uma função, estabiliza.m, para que a cada intervalo de *frames* pré-definido busque por pontos correspondentes em *frames* subsequentes a fim de centralizar as imagens da mesma maneira e, assim, ao fazer as análises e cálculos sobre as linhas obtidas, elas retornem um resultado próximo ao esperado.

Inicialmente, são encontrados os pontos potenciais em cada uma das imagens e após armazená-los separadamente, busca-se por características que indicam os cantos de cada uma dessas imagens. Feito isso, combina-se os pontos obtidos para que sejam encontrados pares entre as

imagens em análise e, a partir deles, realiza-se uma conversão para uma função afim $3x3$ que será utilizada para reposicionar o novo *frame* de acordo com o anterior.

Para isso, utilizou-se os recursos da função estimateGeometricTransform (MATHWORKS, 2013) disponibilizada pelo *software*, que retorna uma função a partir do conjunto de pontos que são passados como pares de cada uma das duas imagens. Essa função utiliza o algoritmo *M-estimator SAmple Consensus* (MSAC) para excluir pontos classificados como *outliers*, isto é, pontos que foram detectados incorretamente e não são efetivamente pares.

Essa função afim $3x3$ encontrada é então convertida para uma transformação escala-translação-rotação que será aplicada sobre o *frame* atual do vídeo, ajustando o mesmo. Esse processo é repetido para todos os *frames* do vídeo de forma que o vídeo de saída será estabilizado.

Para a aplicação de detecção da linha d'água é necessário que diversos momentos do mar sejam analisados e, dessa forma, somente *frames* de determinados intervalos de tempo precisam ser realmente avaliados. Para a análise deste trabalho, a fim de reduzir o esforço computacional, foram selecionados apenas os primeiros *frames* de cada vídeo, sendo que a quantidade varia de acordo com a duração deles.

A fim de exemplificar o funcionamento da função, a Figura 9 apresenta o primeiro *frame* de um vídeo presente no banco de dados que será utilizado como base de análise do trabalho.

Para avaliação da aplicação da função estabiliza.m, a Figura 10(a) apresenta um outro *frame* desse mesmo vídeo antes da estabilização e a Figura 10(b) apresenta esse mesmo *frame* após a estabilização.

Figura 9 - Primeiro frame de um dos vídeos do banco de dados

Fonte: Banco de dados do projeto.

Figura 10 - *Frame* de um vídeo (a) antes e (b) depois da operação de estabilização

(a) (b)

Fonte: (a) Banco de dados do projeto. (b) Produção do próprio autor.

Observando a Figura 10(b) é possível reparar na transformação efetuada pela função para que as marcações de escala do calado se mantivessem fixas na mesma região da imagem e, dessa maneira, ao calcular a linha d'água em todos os *frames* do vídeo é possível calcular a variação da mesma, além de tornar possível a detecção da posição das marcações de escala apenas no primeiro *frame* do vídeo.

3.3 Tratamento das imagens

Feita a estabilização do vídeo e gerado o novo arquivo, o vídeo estabilizado é analisado *frame* a *frame* com a intenção de que a linha d'água seja encontrada em todas as imagens.

Para que a detecção de bordas seja facilitada e, assim, se torne mais eficiente, é realizado um trabalho de pré-processamento nas imagens, na função homofilter.m, durante o processo de análise. Primeiramente, a imagem é convertida para nível de cinza e, então, é aplicado um filtro Butterworth passa-alta para que as bordas presentes na imagem sejam destacadas, visto que, essas bordas caracterizam transições abruptas de intensidade e no domínio da frequência são representados por altas frequências. Em seguida, é feita a equalização de histograma da imagem para realce do contraste.

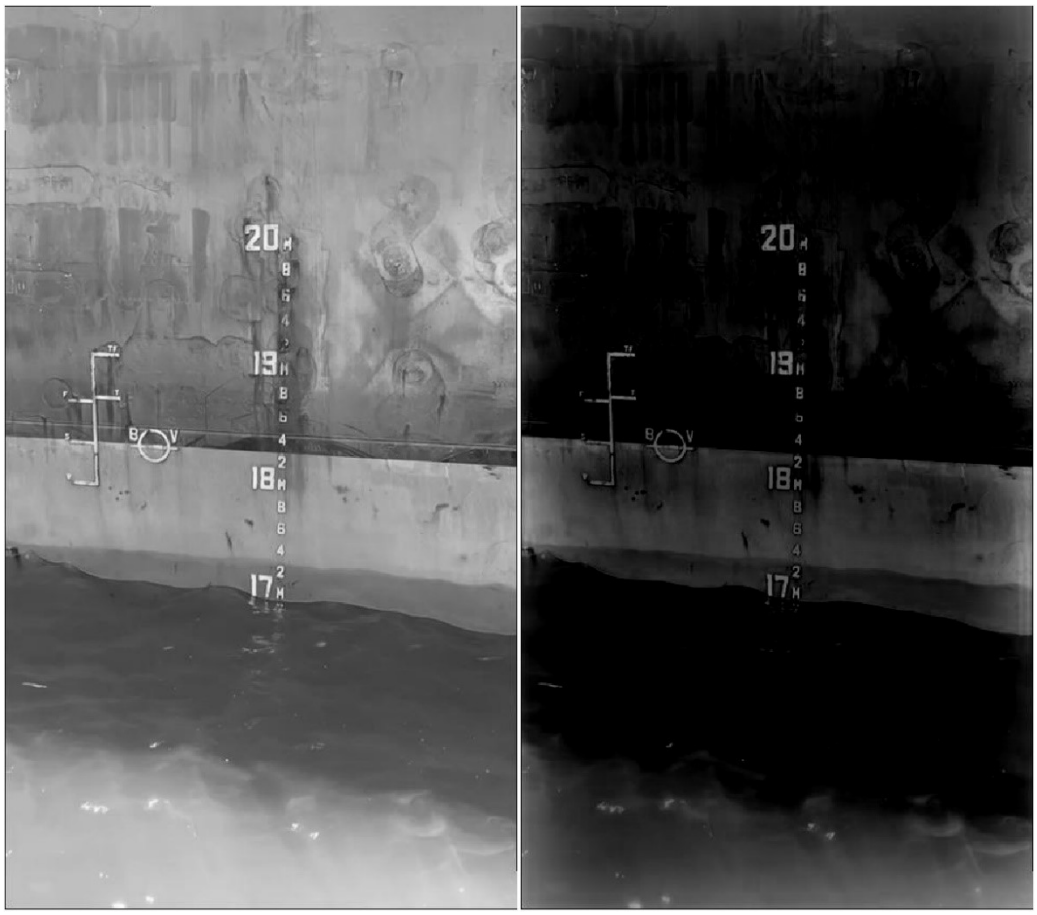
Na primeira etapa, por se tratar de um filtro no domínio da frequência, é necessário definir alguns parâmetros utilizados durante a filtragem, como a ordem e a distância da origem até a frequência de corte do filtro, como apresentado na Equação (1).

Para isso, alguns experimentos foram realizados e concluiu-se que o filtro não deveria realizar uma filtragem muito abrupta, pois poderia acabar retirando informações que seriam úteis durante o processo de detecção de bordas, com isso, optou-se pela utilização de um filtro de ordem 2.

Quanto à distância da origem até a frequência de corte, sabe-se que quanto maior esse parâmetro, mais finas são as linhas detectadas, visto que, com o aumento da distância, maior será a frequência de corte e assim apenas frequências acima dela serão mantidas. Através dos experimentos realizados, também foi possível perceber que linhas mais grossas não possuem um impacto grande na falha da detecção, enquanto que a filtragem de linhas bem finas faz com que algumas informações importantes da imagem que não estão bem destacadas possam ser omitidas, fazendo com que o detector de bordas falhe ao realizar a busca pelas linhas, não encontrando-as. Por isso, ao testar diferentes valores para esse parâmetro, percebeu-se que o valor 10 já filtrava apenas linhas bem finas e, assim, fazendo testes com outros valores definiu-se a distância D_0 como 5, um valor que não limita muito a filtragem.

Ao final do processo de filtragem, a imagem obtida como resultado está apresentado na Figura 11(b). Vale ressaltar que a imagem filtrada é resultante do processamento da imagem apresentada na Figura 9.

Figura 11 – Imagem (a) transformada para escala de cinza e (b) após passar pelo filtro Butterworth de ordem 2 especificado

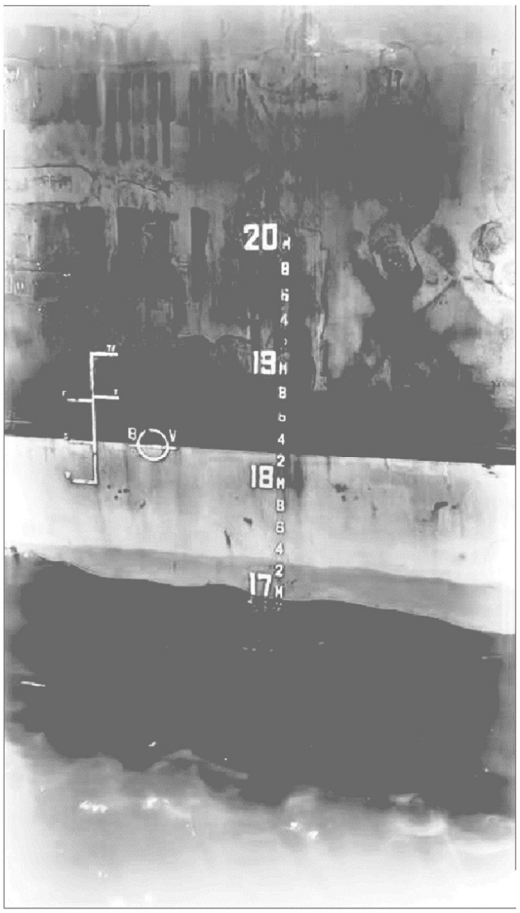


(a) (b)

Fonte: Produção do próprio autor.

Após a filtragem da imagem é possível perceber que o contraste da mesma fica deteriorado e, na intenção de corrigi-lo, é feita a equalização de seu histograma e, assim, o contraste da imagem é melhorado, como pode-se ver na Figura 12.

Figura 12 - Imagem com histograma equalizado



Fonte: Produção do próprio autor.

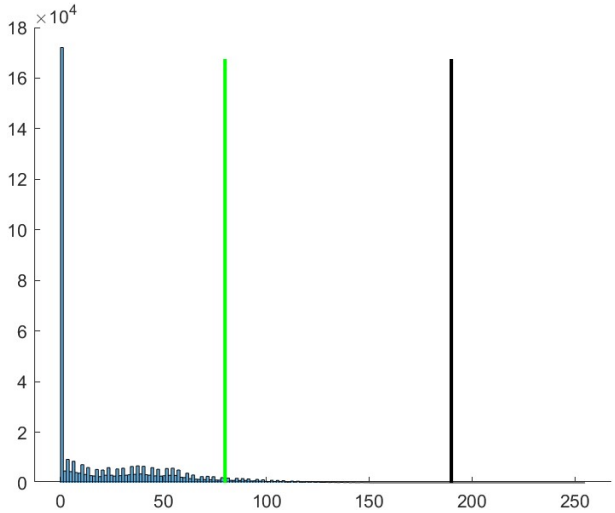
Existem imagens em que alguns de seus *pixels* possuem intensidades muito escuras enquanto que outra parcela de *pixels* possuem intensidades muito claras, e nesses casos a equalização do histograma acaba dificultando, e em alguns casos até impossibilitando a identificação das bordas nas regiões muito claras.

Por isso, fez-se necessária uma verificação para saber se a imagem em análise deveria ou não ser equalizada. Para cobrir o caso citado anteriormente, verificou-se se a quantidade de *pixels* na metade mais clara do histograma, ou seja, da intensidade média até a máxima, é superior a metade dos *pixels* da imagem. Em resumo, essa verificação avalia se a imagem possui mais regiões claras do que escuras e isso sendo verdade, a imagem em análise será equalizada, caso contrário, ela não será alterada.

Além dessa, uma outra verificação precisou ser adicionada para que imagens muito escuras também fossem equalizadas. Assim, caso a intensidade máxima da imagem seja inferior ao limiar escolhido a imagem também será equalizada. A escolha do limiar foi baseada em uma intensidade que se aproxima da cor intermediária entre o claro e escuro em uma imagem, pois dessa forma, a imagem com essa característica possui todas suas regiões com pouca iluminação o que resultará em uma imagem sem regiões excessivamente clareadas após a equalização, assim foi-se definida a intensidade 190.

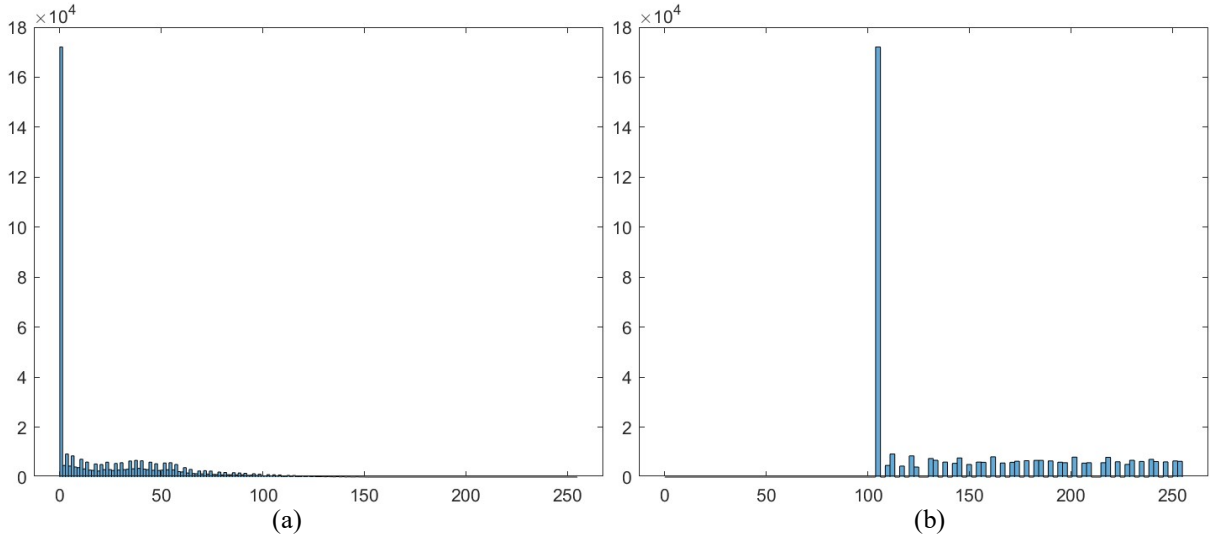
A fim de apresentar através de histogramas o que foi explicado, o Gráfico 1 mostra o histograma da Figura 9 e os limiares que foram definidos para a classificação das imagens que serão ou não equalizadas. Na linha verde, tem-se a intensidade média da imagem, e como pode-se perceber a maioria dos *pixels* estão localizados no lado esquerdo da linha, dessa forma, a imagem não seria equalizada, porém a linha preta está traçada sobre a intensidade 190 e como todos os *pixels* possuem intensidade inferior a ela, a imagem foi equalizada. Através do Gráfico 2 é possível ver o histograma e o histograma equalizado da imagem que está servindo como base de análise.

Gráfico 1 - Histograma do 1º frame do vídeo com os limiares de validação



Fonte: Produção do próprio autor.

Gráfico 2 - (a) Histograma e (b) histograma equalizado da Figura 9



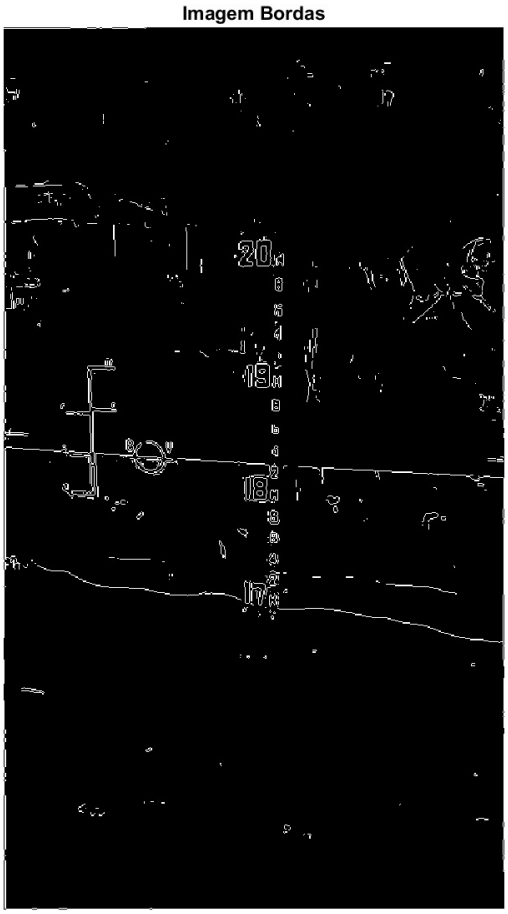
Fonte: Produção do próprio autor.

3.4 Detecção das linhas

Com o tratamento da imagem realizado, chega-se na etapa de detecção das bordas. É nessa etapa que parte fundamental da localização das marcações de escala e da linha d'água acontece e por isso, deve ser realizada com segurança. Dessa forma, durante algumas etapas dessa detecção será necessária a interferência do usuário, validando ou direcionando o funcionamento do programa e, para isso, as funções detecta_linhas.m e valida_linhas.m foram implementadas.

A fim de destacar as bordas presentes na imagem, é aplicado um filtro de Sobel sobre a imagem em análise, que já está tratada e em escala de cinza. A partir dessas bordas encontradas o programa continua a execução. Ao filtrar a imagem que está sendo utilizada como base de análise, o resultado obtido é apresentado na Figura 13.

Figura 13 - Bordas encontradas a partir do filtro de Sobel



Fonte: Produção do próprio autor.

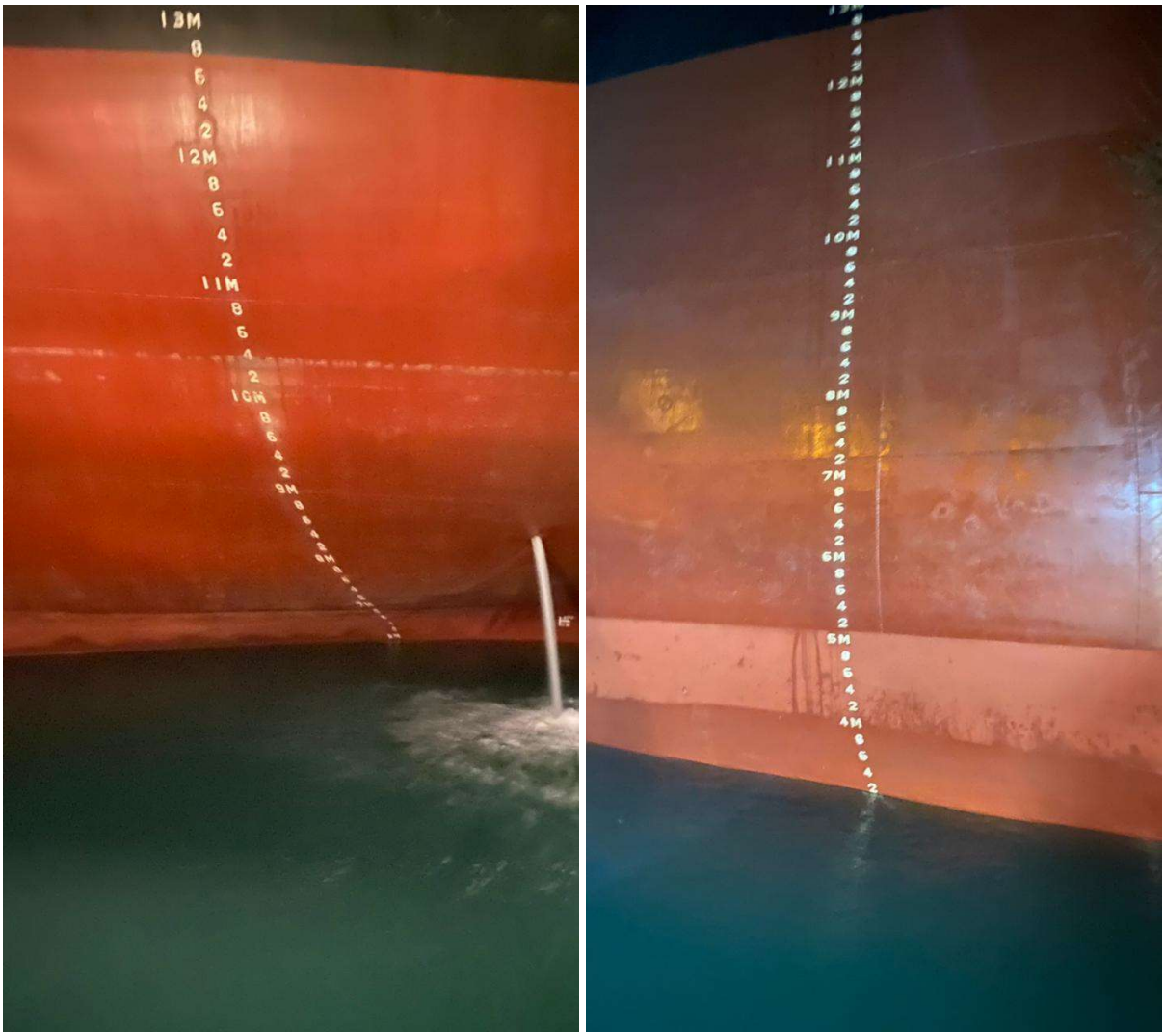
Ao realizar a filtragem, é aplicada então a transformada de Hough, a fim de que essas linhas encontradas sejam interpretadas e então designadas como região das marcações de escala e linha d'água. Como descrito na Seção 2.5, a transformada de Hough utiliza das equações matemáticas das formas geográficas para realizar a interpretação, e a partir de parâmetros que podem ser alterados, ela une e detecta as formas desejadas.

Para isso, utilizou-se a função *hough* (MATHWORKS, 2006) disponibilizada pelo *software* MATLAB, para que os pontos correspondentes as linhas desejadas fossem encontradas.

Inicialmente, definiu-se intervalos angulares para que a função buscasse por linhas tanto na vertical quanto na horizontal separadamente, e os valores definidos foram 25° na orientação vertical e 20° na orientação horizontal. Esses ângulos restringem a atuação da função, diminuindo os esforços computacionais e direcionando os resultados obtidos. O motivo da

escolha desses ângulos foi o fato de algumas imagens possuírem as marcas de escala muito inclinadas devido a posição em que a mesma foi capturada ou até mesmo pelo formato do casco do navio, no caso do intervalo de ângulos na vertical. Já para a horizontal, a razão da escolha foi o fato de que em situações nas quais as marcações não estão centralizadas nas imagens, linhas d'água mais inclinadas podem aparecer. Exemplos dos casos citados são apresentados na Figura 14.

Figura 14 - Exemplo de (a) marcações de escala inclinada e (b) de linha d'água inclinada



(a) (b)

Fonte: Banco de dados do projeto.

Dessa forma, a transformada foi dividida em 3 partes, sendo uma para a detecção de retas verticais, com inclinação variando de -25° até 25°, outra para retas horizontais com inclinação de -90° até -70° e a última, também para retas horizontais, porém com inclinação variando numa faixa de 70° até 89,9°. Todas as inclinações são referenciadas a um eixo na vertical.

Após a obtenção das transformadas de Hough para as regiões de inclinações citadas anteriormente, foram então calculados os picos presentes em cada uma das transformadas, utilizando a função *houghpeaks* (MATHWORKS, 2006). Para isso, foram definidos dois parâmetros para a função, sendo um o número máximo de picos e outro o limiar de validação dos picos.

O parâmetro NUMPEAKS indica o número máximo de picos a serem identificados na matriz gerada através da transformada de Hough, para que, assim, uma linha identificada seja considerada. No caso testado foi utilizado um máximo de 5 picos, fazendo com que apenas nos casos em que os 5 picos forem encontrados as retas localizadas serão validadas, aumentando a confiabilidade da detecção.

Outro parâmetro é o limiar para que os picos sejam validados, ou seja, somente valores superiores a esse número serão considerados picos. Assim, foi escolhido um limiar que não restrinja muito o intervalo dos picos validados, sendo considerado como limite 30% do pico máximo encontrado na matriz da transformada de Hough, valor esse definido como padrão pela própria implementação da função e aproveitado para essa análise. Limiares muito altos não são uma boa escolha no geral para esses parâmetros, pelo fato de possivelmente filtrarem alguma linha com alto potencial de acerto, porém que não está muito bem definida.

Feito isso, utilizando a função houghlines (MATHWORKS, 2006), foram então traçadas as linhas associadas aos picos localizados na transformada. Para isso, a função requer dois parâmetros para realizar essa localização das linhas, sendo um deles o tamanho das linhas a serem consideradas, para que linhas menores que as desejadas sejam descartadas na análise e o outro o espaçamento entre 2 linhas detectadas para que caso esse intervalo seja menor do que o definido, essas linhas sejam então unidas e convertidas em uma única linha.

No trabalho em questão, para o tamanho mínimo das linhas, foi utilizado um limiar de 30 *pixels* tanto para a linha das marcas de escala do calado, ou seja, para as linhas verticais quanto para a linha d'água, detectada a partir das linhas horizontais.

Para o espaçamento, cada orientação teve seu parâmetro definido separadamente. Para a orientação horizontal, esse espaçamento foi de 70 *pixels*, que é pouco maior que o dobro de uma linha com tamanho mínimo, para que linhas não muito distantes sejam consideradas, pois como as imagens retratam situações ambientes, desconexões entre as linhas são frequentemente encontradas e, assim, ao uni-las a faixa da linha d'água seja localizada.

Já para a vertical, 50 *pixels* foi o intervalo definido, menos que o dobro de uma linha com tamanho mínimo, visto que, como a região das marcas possui uma grande quantidade de informação, o intervalo necessário para que duas marcas detectadas se unam é menor quando comparadas as informações encontradas durante a detecção da linha d'água, e assim, quando unidas, representem toda a região das marcas de escala. Vale lembrar que essa detecção ocorre analisando cada ângulo do intervalo definido anteriormente, além disso, essa é apenas a detecção inicial das linhas, elas ainda serão validadas e tratadas posteriormente.

Uma outra análise pertinente para o estudo realizado é que dependendo da distância do navio até a câmera que está capturando o vídeo, os objetos contidos nas imagens terão diferentes tamanhos, isto é, a escala da imagem irá variar com essa distância. Por isso, ao definir os comprimentos mínimos e espaçamentos foi considerado tanto casos em que o navio está próximo, sendo possível localizar objetos em grandes escalas e, dessa forma, maior o comprimento das linhas detectadas. Quanto o navio está distante é mais complicada a detecção de detalhes, porém ainda é possível localizar o conjunto das marcações e os diferentes segmentos da linha d'água. Para exemplificar, são apresentadas na Figura 15, capturas com navio próximo e distante da câmera.

Figura 15 - Exemplo de captura com navio (a) próximo e (b) distante da câmera

(a) (b)

Fonte: Banco de dados do projeto.

A partir do que foi descrito até aqui, as linhas detectadas são então trabalhadas a fim de que apenas as linhas mais próximas das desejadas sejam retornadas como a linha d'água e a régua do calado.

Para as linhas verticais, dentre as retas encontradas que não estão localizadas sobre as bordas laterais da imagem, foi calculado o comprimento de cada uma e, então, o maior tamanho foi considerado a região das marcações de escala do calado.

Agora, para as retas horizontais, outra consideração precisou ser feita, sendo ela referente a conferência da posição da linha detectada em relação a orientação vertical da imagem. Assim, apenas as retas posicionadas no primeiro terço da imagem iniciando pela borda inferior foram consideradas, porém assim como nas retas verticais, desconsiderando as linhas na borda inferior. Em seguida, dentre as linhas validadas, buscou-se a maior delas, e então a linha d'água foi retornada.

Ao final da primeira análise, caso o programa detecte as linhas, ele retornará as informações para que o usuário valide os resultados, em caso afirmativo para as duas linhas, o programa irá armazenar essas linhas e repetir todo o processo para os *frames* seguintes, tomando como base para a análise automática esses resultados já validados.

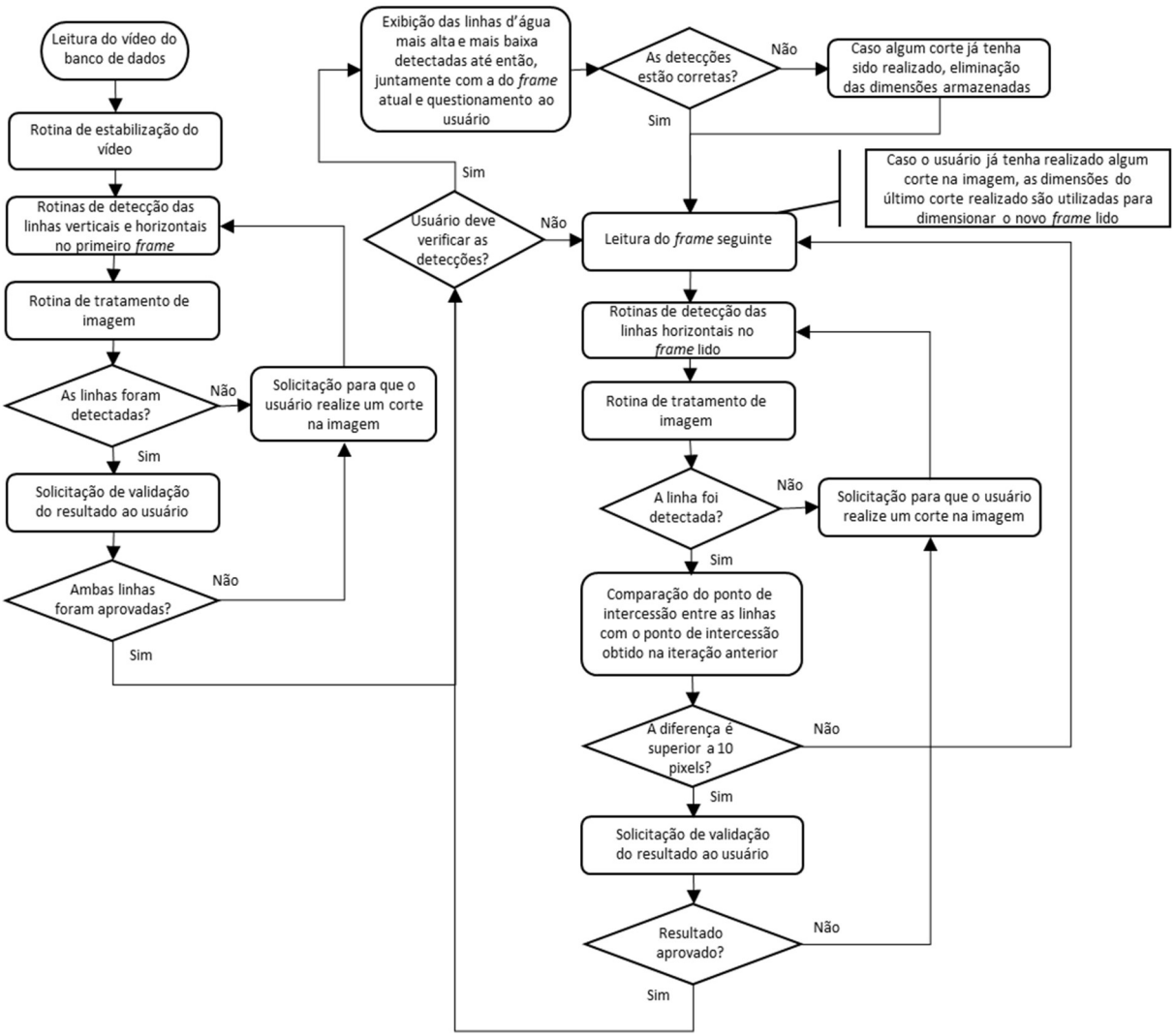
Nos casos em que o programa não consiga detectar alguma das linhas solicitadas, ele irá informar isso ao usuário e solicitará uma participação do mesmo, essa mesma solicitação ocorrerá caso alguma das respostas fornecidas pelo usuário seja negativa. Nessa solicitação, o programa irá pedir ao usuário que realize um corte na imagem, de forma a manter apenas informações úteis na imagem cortada e então o processo será repetido, considerando o corte realizado como a nova imagem de análise e, assim que as linhas forem detectadas, o usuário é novamente solicitado para que valide o resultado.

Após uma primeira validação afirmativa por completo, o programa inicia as verificações automáticas *frame* a *frame*, para isso, ele busca a interseção entre as linhas d'água e das marcas de escala e armazena esse ponto, após uma nova detecção, ele localiza novamente a interseção entre as linhas e compara com o ponto obtido anteriormente, caso ele seja menor que 10 *pixels*, segue para a próxima detecção, caso contrário solicita uma nova participação do usuário. A escolha de 10 *pixels* foi para que o programa consiga ter uma margem de variação da linha medida, até mesmo porque em mares agitados as medidas serão consideravelmente distintas.

Além disso, para que o usuário consiga acompanhar as detecções que estão sendo realizadas, o programa possui uma variável para ajuste do intervalo de interações até que uma validação manual seja solicitada e, nesse momento, o usuário irá caracterizar as detecções obtidas como certas ou erradas. Para isso, ele terá acesso a uma imagem contendo a linha da região das marcações de escala, a linha d'água do *frame* em análise e as linhas d'água mais baixa e mais alta detectadas até então. Caso os resultados sejam validados o programa seguirá com a análise dos *frames* seguintes, porém, caso o resultado seja reprovado a imagem utilizada para análise se torna o *frame* inteiro do vídeo estabilizado, desconsiderando qualquer corte realizado anteriormente.

Para que o processo fique entendido por completo, o fluxograma apresentado na Figura 16 explica de forma detalhada a interação entre todos os passos explicados.

Figura 16 - Fluxograma detalhado do funcionamento do programa



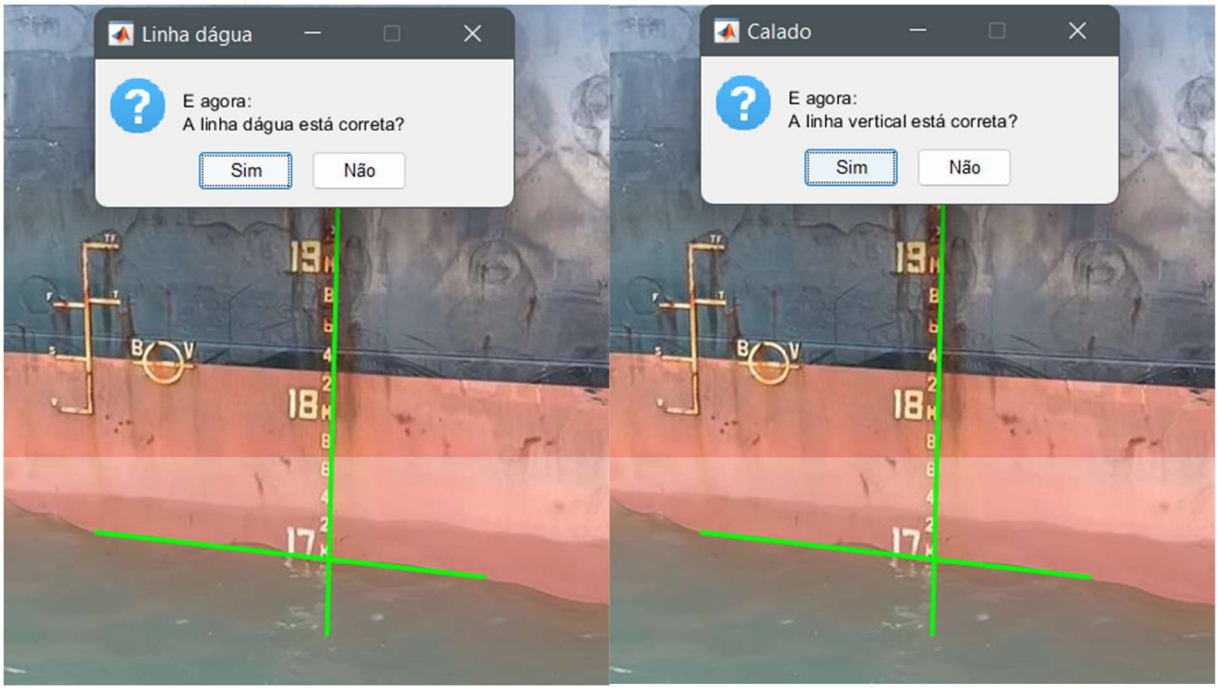
Fonte: Produção do próprio autor.

4 RESULTADOS E DISCUSSÕES

Diante do processo apresentado, chega-se nos resultados obtidos a partir da aplicação dos métodos desenvolvidos.

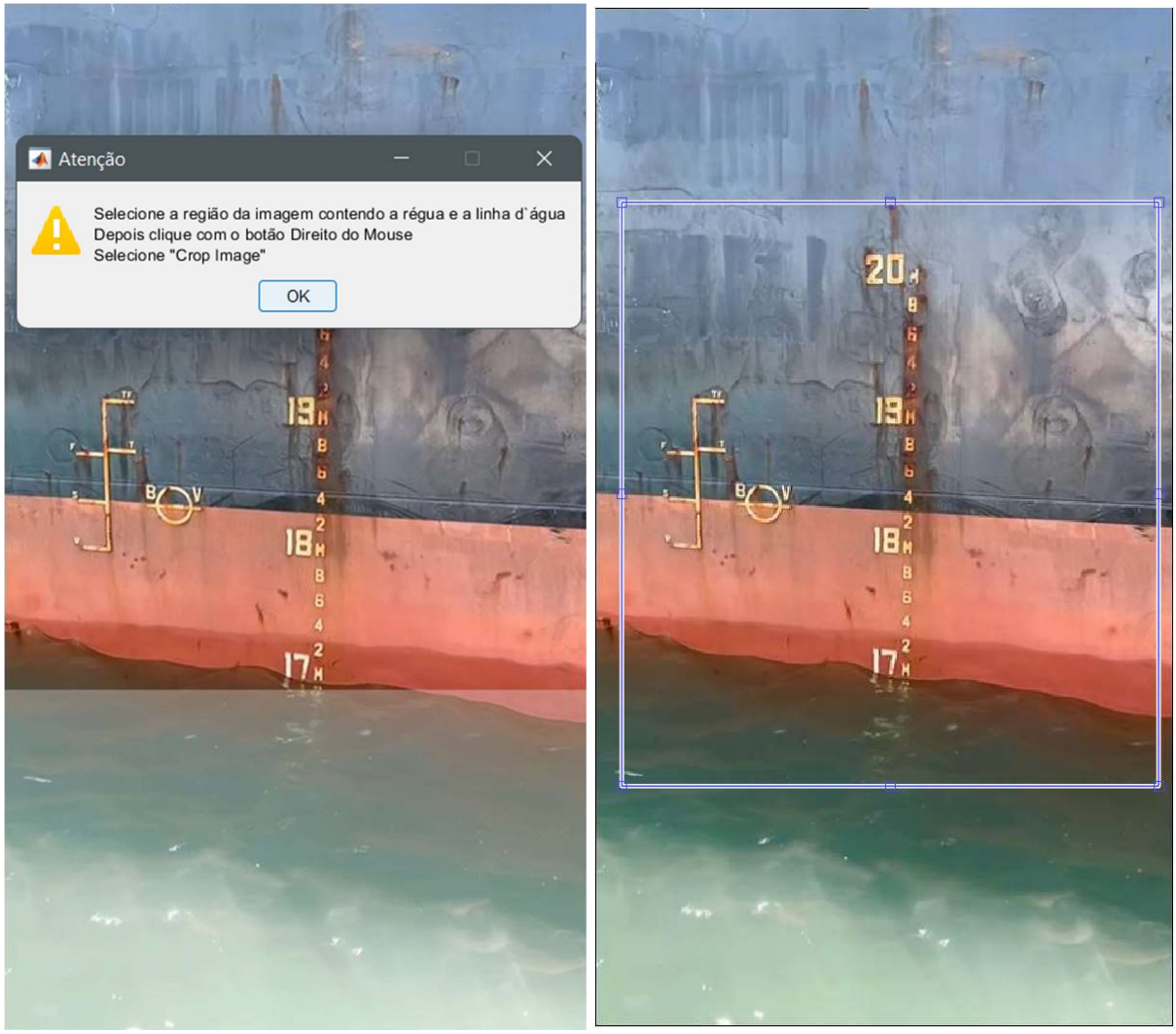
Como primeira etapa, o programa tentará detectar as linhas e em caso de sucesso retornará as linhas encontradas solicitando a confirmação do usuário, como apresentado na Figura 17, porém, em caso de falha ou caso as linhas mostradas estejam erradas, ele solicitará ao usuário que realize um corte na imagem para que a detecção seja direcionada àquela região. A forma como esse processo ocorre está apresentado na Figura 18, onde o retângulo azul na imagem da direita representa a nova imagem que será utilizada na análise.

Figura 17 - Resultados encontrados pelo programa após as detecções



Fonte: Produção do próprio autor.

Figura 18 - Exemplo do corte realizado na imagem para sequência da análise



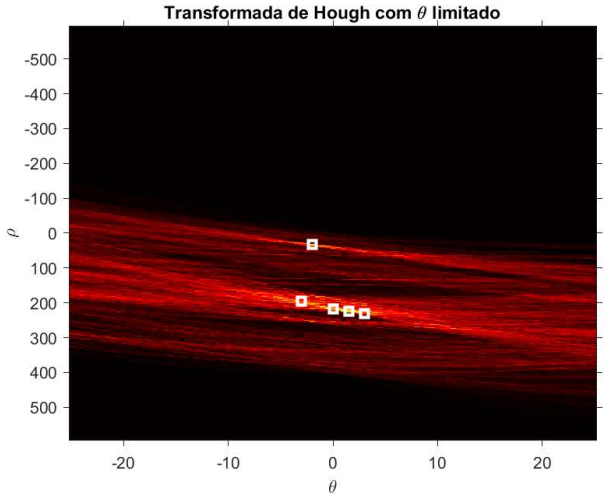
Fonte: Produção do próprio autor.

Feito isso, o programa realiza novamente a detecção das linhas e, se possível, retorna as linhas encontradas para que sejam analisadas. Para a região cortada da Figura 18 os resultados encontrados com a transformada de Hough para ambas orientações e o resultado final após o tratamento das linhas encontradas pela transformada são apresentados nas figuras a seguir.

Para o primeiro caso, que são as retas na vertical, a transformada de Hough retornou o Gráfico 3, o qual apresenta o plano $\theta - \rho$ com as linhas referentes aos pontos encontrados na imagem, destacando os cinco picos detectados, validando as linhas encontradas na imagem. Já as retas localizadas na imagem referentes às detecções da transformada são apresentadas na Figura 19, na qual todas as linhas encontradas são apresentadas, porém no próximo passo serão calculados

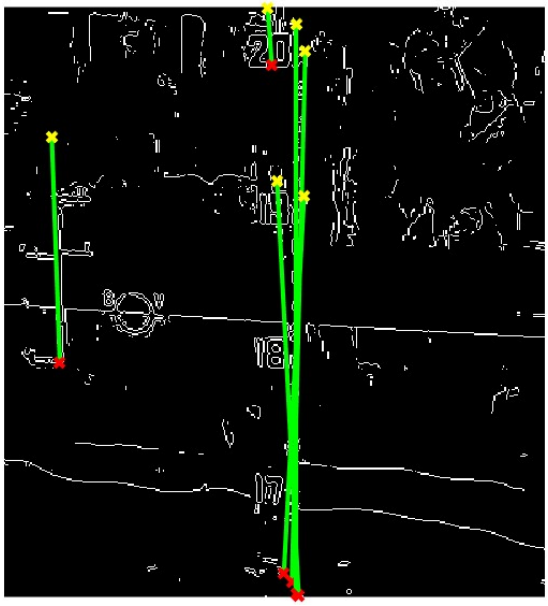
os tamanhos das linhas e a maior dela será retornada como sendo a região das marcações de escala.

Gráfico 3 - Plano θ-ρ das linhas verticais



Fonte: Produção do próprio autor.

Figura 19 - Linhas verticais retornadas pela transformada de Hough

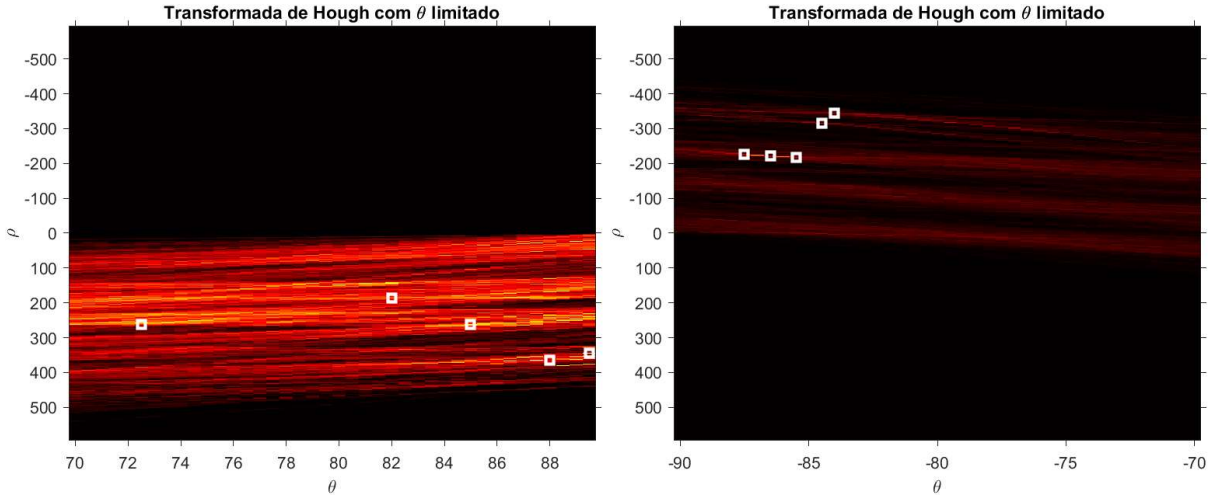


Fonte: Produção do próprio autor.

O mesmo acontece para as retas na horizontal, no qual as transformadas retornaram o Gráfico 4, sendo o Gráfico 4(a) para o intervalo de ângulos de 70° a 89,9° e o Gráfico 4(b) para o intervalo de -90° a -70°, como pode ser observado no eixo θ deles. Além disso, todas as linhas encontradas na imagem são apresentadas na Figura 20, sendo as retas na cor magenta as

detecções que foram aprovadas na validação a respeito da região do terço inferior da imagem, dentre as quais a maior delas será retornada como a linha d'água.

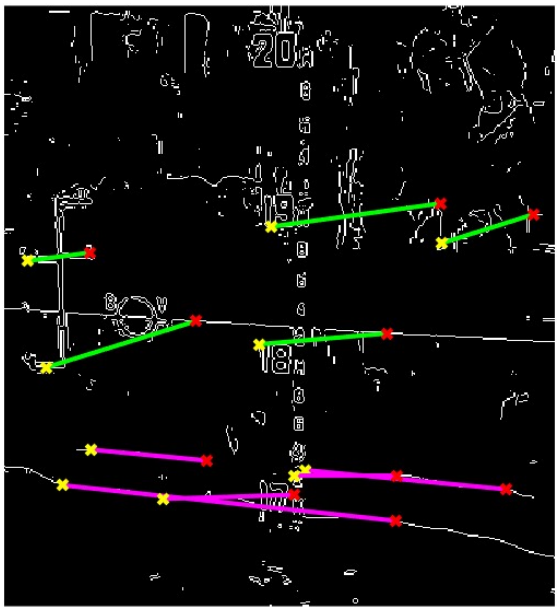
Gráfico 4 - Plano θ-ρ das linhas horizontais com inclinação (a) 70° a 89,9° e (b) -90° a -70°



(a) (b)

Fonte: Produção do próprio autor.

Figura 20 - Linhas horizontais retornadas pela transformada de Hough



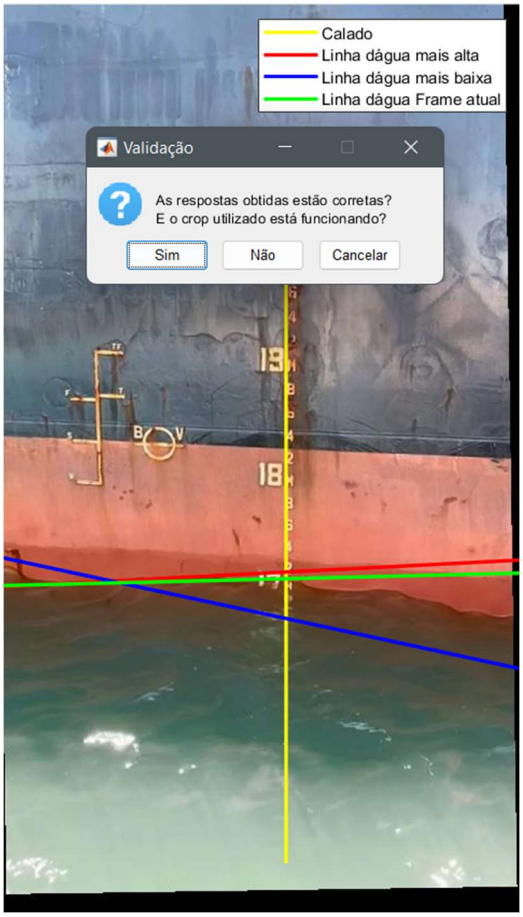
Fonte: Produção do próprio autor.

Para finalizar a primeira detecção, é realizado o cálculo das maiores linhas nas duas direções e então é retornado o resultado ao usuário, solicitando sua validação, como apresentado na Figura 17, onde uma das caixas de mensagem questiona a respeito da linha das marcas de escala e outra a respeito da linha d'água.

No caso de ambas as respostas serem afirmativas, o programa realiza a leitura do próximo *frame* e repete os processos. Vale lembrar que ao realizar o corte da imagem em um dos *frames* o programa armazena as coordenadas desse corte e o aplica nos *frames* seguintes.

No entanto, caso o programa não consiga localizar a linha d'água dentro do limiar definido ou não consiga detectar nenhuma linha em algum dos *frames*, ele voltará a solicitar a participação do usuário, porém, caso a linha seja encontrada, o usuário só será requisitado quando o intervalo de interações definido for atingido e, nesse caso, o usuário terá acesso a uma imagem como a apresentada na Figura 21, na qual 4 linhas são mostradas, sendo uma referente a região das marcações do calado, outra referente a linha d'água detectada no *frame* em análise e as outras duas referentes as linhas d'água mais alta e mais baixa encontradas.

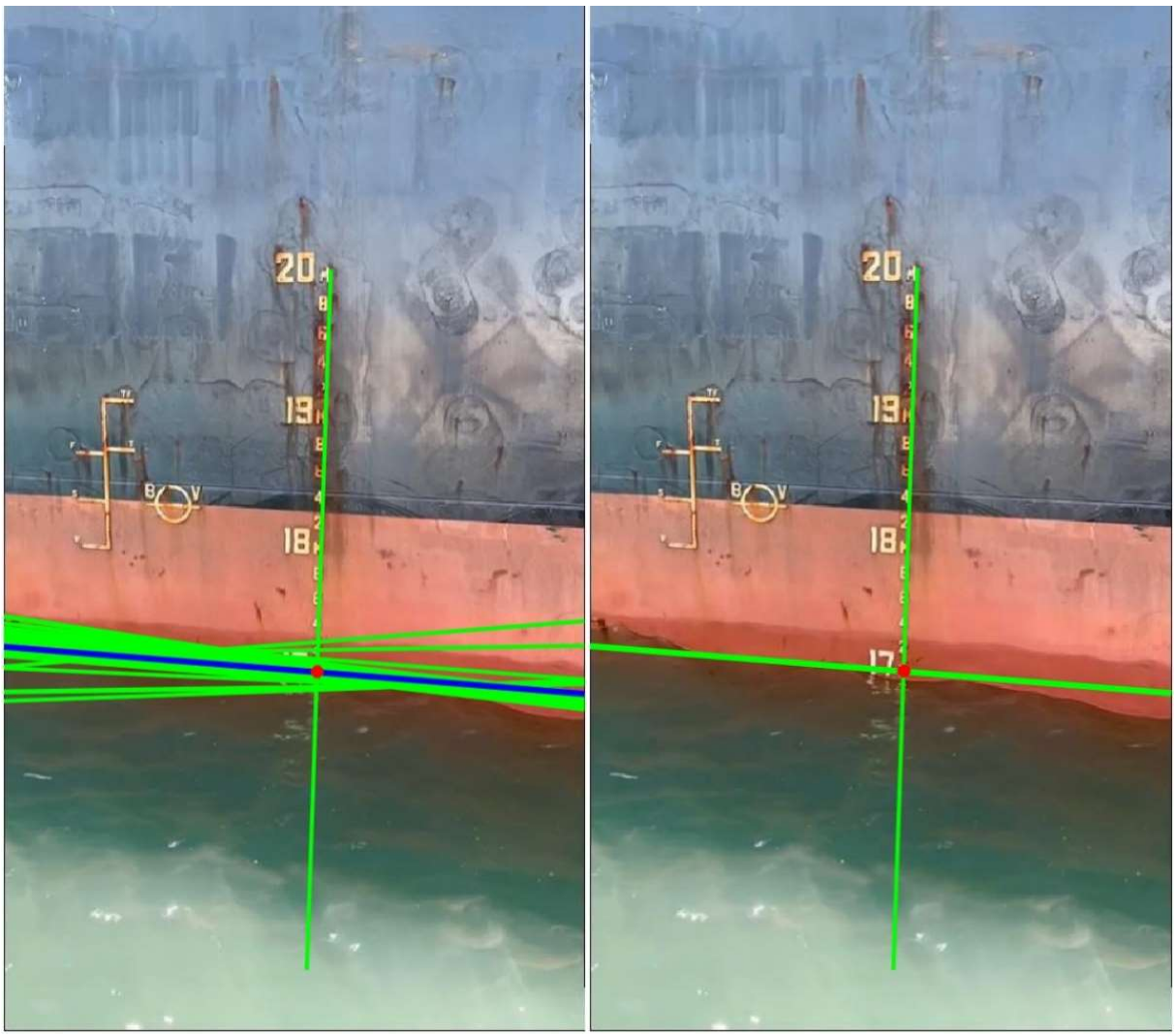
Figura 21 - Validação após determinada quantidade de linhas detectadas



Fonte: Produção do próprio autor.

E assim, ao finalizar a análise de todos os *frames* do vídeo, o programa retorna a medição realizada, na qual a linha d'água é calculada como a média de todas as linhas detectadas. O conjunto de todas as linhas encontradas pode ser visto na Figura 22(a) com destaque para linha azul dada pela média das linhas detectadas, enquanto que a Figura 22(b) mostra o resultado final retornado pelo programa.

Figura 22 - (a) Conjunto das linhas d'água encontradas nos frames do vídeo estabilizado e (b) o resultado final do programa após todas as detecções



(a) (b)

Fonte: Produção do próprio autor.

Considerando o banco de dados disponível, foi possível realizar diferentes testes e buscas por falhas no programa, procurando saber qual tinha maior recorrência, qual deteriorava mais a medição, em quais casos o programa não conseguia detectar as linhas, entre outras.

Ao testar os vídeos, o resultado encontrado foi satisfatório, visto que, considerando os 36 vídeos bons para detecção, 31 deles foram bem detectados, contabilizando 86% de sucesso. Enquanto que ao observar os resultados relacionados aos 18 vídeos com baixa qualidade, viu-se que foi possível detectar as marcas e a linha d'água em apenas 4 deles, o que ao considerar as imagens neles presentes, é um resultado aceitável, visto que o programa não foi dimensionado para esses casos.

Assim, analisando as imagens e os resultados encontrados, foi possível notar que quando marcas bem definidas de sombra, marcas d'água em cascos mais claros ou até mesmo outras marcações no próprio casco do navio, estavam localizadas próxima à linha d'água, a detecção se tornava muito difícil e até inalcançável nos casos que ambas linhas se encontravam muito próximas, isso se deve ao fato da detecção ocorrer por meio da detecção de bordas.

Um outro fator com impacto considerável nos resultados obtidos foi a iluminação, pode-se perceber que em situações com baixa luminosidade ou então quando o foco de luz era muito intenso e centralizado, as bordas acabavam não sendo detectadas, seja pela falta de luz, ou pelo excesso dela, em ocasiões nas quais as linhas ficam camufladas pela claridade, tornando difícil a detecção das marcações de escala e da linha d'água.

Outro ponto que complicou a detecção das linhas foi a análise de cascos lisos e pobre em detalhes, pois, dessa forma, a rotina de estabilização falha ao buscar pontos pares suficientes entre um *frame* e outro, não sendo possível manter o calado fixo em uma única posição com o passar dos *frames* e, assim, as medidas da linha d'água são feitas baseadas em referenciais diferentes, invalidando a medição.

Além disso, o fato da base de dados possuir poucos vídeos e, consequentemente, poucos vídeos em condições adversas, não possuindo casos com elevada quantidade de limo no casco por exemplo, fez com que o programa não pudesse ser inteiramente avaliado e, assim, casos de situações reais, mas que não foram abordados pelos vídeos, acabaram não sendo testados e analisados.

5 CONCLUSÃO

Visando cumprir os objetivos propostos, o trabalho obteve sucesso na implementação de rotinas para detecção das linhas, obtendo resultados positivos na maior parte dos vídeos analisados, detectando corretamente tanto a linha d'água quanto a região das marcações de escala do calado.

Dentro da detecção de linhas, a transformada de Hough se mostrou um grande artifício para a função, visto que, dada sua flexibilidade, permitindo o ajuste de parâmetros como ângulo de inclinação e comprimento mínimo das linhas, foi possível detectar com alta precisão bordas delicadas encontradas pelo filtro de Sobel.

Além disso, a flexibilidade que o programa disponibiliza ao usuário foi de extrema importância, considerando que muitos dos vídeos analisados corretamente tiveram que ser ajustados ao menos na primeira iteração, direcionando o funcionamento das rotinas e qualificando a imagem que seria trabalhada ao filtrar apenas os detalhes úteis. Apenas em 2 dos 36 vídeos com boa qualidade avaliados, o programa detectou corretamente as linhas e não solicitou a interferência do usuário.

Também é válido lembrar que o presente trabalho teve foco na detecção das linhas, buscando encontrar o ponto de interseção médio dentre as variadas medidas realizadas, porém, a detecção para se tornar válida deve ser feita com um intervalo em ciclos de onda e não em ciclos de tempo definidos, sendo assim, essa é uma nova abordagem a ser tratada, tendo o objetivo de encontrar os momentos corretos em que as medições devem ser realizadas.

Outro ponto no qual uma melhora é necessária é o banco de dados, pois com um banco maior e mais completo, mais casos podem ser analisados, além de tornar possível a utilização de recursos envolvendo inteligência artificial, visto que seu uso requer muitas informações disponíveis para que testes e treinamentos possam ser realizados.

Por fim, um trabalho futuro complementar a esse é o de interpretação dos caracteres das marcações de escala do calado, pois assim, obtendo a posição de interseção entre as linhas será

possível retornar a medida do calado e consequentemente realizar os cálculos necessários para descoberta do peso do navio.

REFERÊNCIAS BIBLIOGRÁFICAS

CLARKSONS Research. **World Shipyard Monitor**, v. 28, 2021.

DANTE, L. A. **Teláris matemática, 8º ano:** ensino fundamental, anos finais. 3ª. ed. São Paulo: Ática, 2018.

DUDA, R. O.; HART, P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. **Comm. ACM**, v. 15, p. 11-15, Janeiro 1972.

FONSECA, M. M. **ARTE NAVAL**. 7ª. ed. Rio de Janeiro: [s.n.], v. I, 2005. ISBN 85-7047-051-7.

GONZALEZ, R.; WOODS, R. **Digital Image Processing**. 3ª. ed. [S.l.]: Pearson Education, Inc., 2008.

HOUGH, P. V. C. **Method and means for reconizing complex patterns**. 3.069.654, 18 Dezembro 1962.

IVčE, R.; JURDANA, I.; MOHOVIć, R. Determining Weight of Cargo Onboard Ship by Means of Optical Fibre Technology Draft Reading. **Transport Engineering**, v. 23, p. 421-429, 2011.

MACHADO, F. **Medição Automática de Calado utilizando Deep Learning**. Universidade Federal do Espírito Santo. [S.l.], p. 79. 2020.

MACHADO, F.; SANTOS, C. C. D.; VASSALO, R. F. Reconhecimento de linha d'água para medição de calado. **XXII Congresso Brasileiro de Automática - CBA 2018**, 2018.

MATHWORKS. **Função houghlines**, 2006. Disponivel em: <https://www.mathworks.com/help/images/ref/houghlines.html>. Acesso em: Maio 2022.

MATHWORKS. **Função houghpeaks**, 2006. Disponivel em: <https://www.mathworks.com/help/images/ref/houghpeaks.html>. Acesso em: Maio 2022.

MATHWORKS. **Função hough**, 2006. Disponivel em: <https://www.mathworks.com/help/images/ref/hough.html>. Acesso em: Maio 2022.

MATHWORKS. **Função estimateGeometricTransform**, 2013. Disponivel em: <https://www.mathworks.com/help/vision/ref/estimategeometrictransform.html>. Acesso em: Maio 2022.

MATHWORKS. **MATLAB Versão R2020a**, 2020. Disponivel em: <https://www.mathworks.com/products/matlab.html>. Acesso em: 12 Maio 2022.

MATURANA, P. S. **Algorítmos de detecção de bordas implementados em FPGA**. FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA. São Paulo. 2010.

OKAMOTO, A. et al. A Draught Reading Method by Image Processing with the Robustness of Measurement Distance. **The Journal of Japan Institute of Navigation**, v. 130, p. 135-140, 2014.

TSUJII, T.; YOSHIDA, H.; IIGUNI, Y. Automatic draft reading based on image processing. **Optical Engineering**, v. 55, p. 104104, 2016.

UNCTAD. Review of Maritime Transport 2021, New York, p. 31, 2021. ISSN 0566-7682. Disponivel em: <https://unctad.org/system/files/official-document/rmt2021_en_0.pdf>. Acesso em: 27 fev. 2022.

ZHANG, X. et al. Self-Powered Distributed Water Level Sensors Based on Liquid–Solid Triboelectric Nanogenerators for Ship Draft Detecting. **Advanced Functional Materials**, 2019.

APÊNDICE A – ESTABILIZA.M

```
% ENTRADAS:
% FILE_IN: Nome do arquivo de vídeo original colorido com a extensão.
% OP: 0 - para gerar um vídeo em cinza
%     1 - para gerar um vídeo colorido
% N: Número de frames para processamento.
% Se N < 2 ou N > Total de Frames, então N = Total de Frames
% POS: Define a posição da linha (de cima para baixo) da imagem até onde será usada a imagem
de cada frame para calcular a matriz de transformação.
% Se POS > Numero de linhas ou POS < 10, toda a imagem será usada para o cálculo da
transformação.

% SAÍDAS:
% Arquivo contendo a letra "e" no final do nome e extensão AVI.

% Ex: estabiliza('2c.mp4',1,100,450);
% A saída será o arquivo '3ce.avi' colorido)

function [file_out] = estabiliza(file_in,op,n,pos)

% Nome do arquivo de saída
file_out = strcat(file_in(1:end-4),'e');

% Leitura do arquivo de vídeo
hVideoSrc = VideoReader(file_in);
read(hVideoSrc, 1);

% Número de frames para estabilizar
if (n < 2) || (n > hVideoSrc.NumFrames)
    n = hVideoSrc.NumFrames;
end

% Cria vídeo de saída
if exist(file_out,'file') == 2
    delete(file_out);
end
outputVideo = VideoWriter(file_out);
outputVideo.FrameRate = hVideoSrc.FrameRate;
open(outputVideo)

% Processa todos os frames no video
imgB = rgb2gray(im2single(readFrame(hVideoSrc)));

% define a região de cima da imagem para calcular a transformação
if (pos < 10) || (pos > size(imgB,1))
    lin = 1:size(imgB,1);
else
    lin = 1:pos;
end
ptThresh = 0.1; % limiar para uso no detector de cantos
ii = 2;
Hcumulative = eye(3);
while hasFrame(hVideoSrc) && ii <= n
```

```
    % guarda frame anterior
    imgA = imgB;
    % Lê em novo frame
    img = im2single(readFrame(hVideoSrc));
    imgB = rgb2gray(img);
    try
        % Estima a transformação do frame A para o frame B
        H = cvexEstStabilizationTform(imgA(lin,:),imgB(lin,:),ptThresh);
    catch
        % Caso não seja possível estimar, informa o erro e finaliza a
        % função
        warning('O programa não pode estabilizar o vídeo, devido a baixa qualidade das
imagens.')
        file_out = 0;
        return
    end
    % Interpola como um s-R-t
    HsRt = cvexTformToSRT(H);
    % Efetua a transformação
    Hcumulative = HsRt * Hcumulative;
    imgBp = imwarp(imgB,affine2d(Hcumulative),'OutputView',imref2d(size(imgB)));
    % Colorido
    if op == 1
        R = img(:,:,1);
        G = img(:,:,2);
        B = img(:,:,3);
        img(:,:,1) = imwarp(R,affine2d(Hcumulative),'OutputView',imref2d(size(R)));
        img(:,:,2) = imwarp(G,affine2d(Hcumulative),'OutputView',imref2d(size(G)));
        img(:,:,3) = imwarp(B,affine2d(Hcumulative),'OutputView',imref2d(size(B)));
        % Escreve frame no vídeo de saída colorido
        writeVideo(outputVideo,img)
    else
        % Escreve frame no vídeo de saída em cinza
        writeVideo(outputVideo,imgBp)
    end
    ii = ii+1;
end
% Fecha o arquivo de vídeo de saída
close(outputVideo)
file_out = strcat(file_out,'.avi');
end

%% H = cvexEstStabilizationTform(leftI,rightI,ptThresh)
% Retorna uma transformação afim entre leftI e rightI usando a função
"estimateGeometricTransform" e um argumento de limiar para uso no detector de cantos.
% Obtem a transformação inter-imagem e recursos de pontos alinhados.

function H = cvexEstStabilizationTform(leftI,rightI,ptThresh)
% Gera pontos potenciais
pointsA = detectFASTFeatures(leftI, 'MinContrast', ptThresh);
pointsB = detectFASTFeatures(rightI, 'MinContrast', ptThresh);

% Seleciona as correspondências pontuais
% Extrai características para os cantos
[featuresA, pointsA] = extractFeatures(leftI, pointsA);
[featuresB, pointsB] = extractFeatures(rightI, pointsB);
```

```
% Combina recursos que foram calculados a partir das imagens atuais e anteriores
indexPairs = matchFeatures(featuresA, featuresB);
pointsA = pointsA(indexPairs(:, 1), :);
pointsB = pointsB(indexPairs(:, 2), :);

% Usa o algoritmo MSAC para calcular a transformação afim
tform = estimateGeometricTransform(pointsB, pointsA, 'affine');
H = tform.T;
end

%% [H,S,ANG,T,R] = cvexTformToSRT(H)

% Retorna os parâmetros da escala, rotação e translação, e a matriz de transformação H
reconstituída.

% Converte uma transformação afim 3 por 3 em uma transformação escala-rotação-translação.
function [H,s,ang,t,R] = cvexTformToSRT(H)
% Extrai submatrizes de rotação e translação
R = H(1:2,1:2);
t = H(3, 1:2);

% Calcula theta a partir da média ods aro-tangentes estáveis
ang = mean([atan2(R(2),R(1)) atan2(-R(3),R(4))]);

% Calcula a escala a partir da media de dois calculos de média estáveis
s = mean(R([1 4])/cos(ang));

% Transformação de reconstituição
R = [cos(ang) -sin(ang); sin(ang) cos(ang)];
H = [[s*R; t], [0 0 1]'];
end
```

APÊNDICE B – HOMOFILTER.M

```
function [im_but] = homofilter(I, ordem)
    % Filtragem homomórfica
    I = rgb2gray(I);
    % Filtro Butterworth Passa-Alta
    im_fill = zeros(2*size(I,1),2*size(I,2));
    for i = 1:size(I,1)
        for j = 1:size(I,2)
            im_fill(i,j) = I(i,j);
        end
    end

    F_shift = fftshift(fft2(im_fill));
    y = linspace(-size(I,1), size(I,1), 2*size(I,1));
    x = linspace(-size(I,2), size(I,2), 2*size(I,2));
    [u, v] = meshgrid(x, y);

    D = (u.^2 + v.^2).^0.5;
    D0 = 5;

    % Define o Filtro Butterworth
    H_but = 1./(1.+(D./D0).^(2*ordem));

    % Ponto de corte do filtro HP normalizado
    alphaL = 0.00999;
    aplhaH = 1.0010;
    H_but = ((aplhaH - alphaL).*H_but) + alphaL;
    H_but = 1 - H_but;

    % Filtragem Butterworth
    G_but = F_shift .* H_but;

    im_but_fill = real(ifft2(fftshift(G_but)));
    im_but = uint8(zeros(size(I,1),size(I,2)));
    for i = 1:size(I,1)
        for j = 1:size(I,2)
            im_but(i,j) = uint8(im_but_fill(i,j));
        end
    end

    % Realiza um ajuste no contraste depois da filtragem
    im_but = ajuste_contraste(im_but);
end

%% Função ajuste_contraste
% Avalia a intensidade dos pixels da imagem de entrada e dependendo do resultado, realiza uma
equalização de histograma a fim de facilitar a detecção de bordas.
function [im] = ajuste_contraste(img)
    % Contabiliza os píxels da imagem
    [x,edge] = histcounts(img);

    % Calcula o limiar do histograma com base na média das intensidades
    limiarHist = round((max(edge)-min(edge))/2);
```

```
    % Localiza a posição do limiar calculado no vetor de intensidades
    dist = 100;
    for i = 1:size(edge,2)
        if abs(edge(i) - limiarHist) < dist
            dist = abs(edge(i) - limiarHist);
            pos = i;
        end
        if dist == 0
            break
        end
    end

    % Calcula a soma de píxels na metade mais clara da imagem
    soma = sum(x(pos:size(x,2)));
    % Calcula a metade dos píxels da imagem
    lim = sum(x)/2;

    % Caso mais da metade dos píxels esteja na região clara ou caso todos os
    % pixels tenham intensidade abaixo de 190(intensidade intermediária entre
    % o claro e escuro das imagens) então a imagem é equalizada
    if soma > lim || max(edge) < 190
        im = histeq(img);
    else
        im = img;
    end
end
```

APÊNDICE C – DETECTA_LINHAS.M

```
% [vl,hl,falha] = detecta_linhas(I, deteccao, imc, interacao)
% Retorna as melhores linhas vertical e horizontal encontradas com base nos parâmetros de
entradas.

% ENTRADA:
% I é a imagem na qual as linhas serão detectadas
% deteccao = 0 corresponde a encontrar as linhas verticais e horizontais
% deteccao = 1 corresponde a encontrar as linhas horizontais encontradas
% deteccao = 2 corresponde a encontrar as linhas verticais encontradas
% imc = true indica que a imagem de entrada é um corte da imagem original
% interacao = true corresponde a mostrar os resultados encontrados

% SAÍDA:
% v1 = hl = [x1 y1; x2 y2]
% falha = true indica que as linhas solicitadas não foram encontradas
% Usa a função HOMOFILTER

function [varargout] = detecta_linhas(I, deteccao, imc, interacao)
    limiar_inferior = floor(size(I,1)*2/3);   % posição a 1/3 da imagem a
                                              % partir da base
    vl = [];
    hl = [];
    v_zero = [0,0;0,0];
    falha = false;

    % Filtragem Homomórfica
    G = homofilter(I,2);

    % Detector de borda de Canny com threshold de 0.25 obtido
    % experimentalmente
    BW = edge(G,'sobel');

    % No caso de imagens cortadas, retira 5% da margens para eliminar
    % ruídos resultantes do corte
    if imc == 1
        bY = round(size(BW,1)*(5/100));
        bX = round(size(BW,2)*(5/100));
        BW = BW(bY:size(BW,1)-bY, bX:size(BW,2)-bX);
    end

    % Aplica a transformada de Hough com parâmetros obtidos empiricamente
    % nas bordas detectadas com o método de Sobel
    dangh = 20;     % delta angular horizontal para cada lado em graus
    dangv = 25;     % delta angular vertical para cada lado em graus
    i = 1;
    for j = 1:3
        switch j
            case 1
                [H,T,R] = hough(BW,'Theta',-dangv:0.5:dangv);  % Linha Vertical
                P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:))));
                lines = houghlines(BW,T,R,P,'FillGap',50,'MinLength',30);
            case 2
```

```
                [H,T,R] = hough(BW,'Theta',-90:0.5:-90+dangh); % Linha Horizontal
                P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:)))); % à esquerda
                lines = houghlines(BW,T,R,P,'FillGap',70,'MinLength',30);
            case 3
                [H,T,R] = hough(BW,'Theta',90-dangh:0.5:89.9); % linha Horizontal
                P = houghpeaks(H,5,'threshold',ceil(0.3*max(H(:)))); % à direita
                lines = houghlines(BW,T,R,P,'FillGap',70,'MinLength',30);
        end

        % Encontra a linha mais comprida na vertical
        if j == 1
            max_len = 0;
            for k = 1:length(lines)
                xy = [lines(k).point1; lines(k).point2];
                % Faz uma conferência para saber se as linhas detectadas
                % estão em alguma das bordas da imagem
                if xy(1,1) < 5 || xy(2,1) > size(BW,2)-5
                    continue
                end
                len = norm(lines(k).point1 - lines(k).point2);
                if (len > max_len)
                    max_len = len;
                    vl = xy;
                end
            end

        % Encontra as linhas na horizontal no terço inferior da imagem
        elseif (deteccao ~= 2)
            for k = 1:length(lines)
                xy = [lines(k).point1; lines(k).point2];
                [~,y] = intersecao(xy,vl);
                if y >= limiar_inferior && y < size(BW,1)-5 % se a reta está
                                                            % até a 1/3 da base da imagem
                        candidatas{i} = xy;
                        i = i + 1;
                end
            end
        end
    end

    % Encontra a linha mais comprida na horizontal
    if (deteccao ~= 2)
        max_len = 0;
        for k = 1:i-1
            xy = candidatas{k};
            len = norm(xy(1,:) - xy(2,:));
            if (len > max_len)
                max_len = len;
                hl = xy;
            end
        end
    end

    if imc == 1
        if ~isempty(hl)
            hl = [hl(:,1) + bX, hl(:,2) + bY];
        end
```

```
        if ~isempty(vl)
            vl = [vl(:,1) + bX, vl(:,2) + bY];
        end
    end

    % Apresenta as linhas encontradas
    if interacao
        % Destaca a região do limiar considerado
        box = [1 limiar_inferior size(I,2) size(I,1)-limiar_inferior+1];
        % [xmin ymin xmax-xmin+1 ymax-ymin+1]
        IB = insertShape(I,'FilledRectangle',box,'Color','white','Opacity', 0.3);

        figure
        imshow(IB)
        hold on
        title('Imagem Detecta Linhas')
        % plota linhas
        if deteccao == 0 && ~isempty(hl) && ~isempty(vl)
            plot(vl(:,1),vl(:,2),'LineWidth',2,'Color','green');
            plot(hl(:,1),hl(:,2),'LineWidth',2,'Color','green');
        elseif deteccao == 1 && ~isempty(hl)
            plot(hl(:,1),hl(:,2),'LineWidth',2,'Color','green');
        elseif deteccao == 2 && ~isempty(vl)
            plot(vl(:,1),vl(:,2),'LineWidth',2,'Color','green');
        else
            falha = true;
        end
    end

    % Retorna as linhas encontradas e caso as linhas solicitadas não sejam
    % detectadas, o vetor zero é retornado no lugar, juntamente com a flag
    % falha true.
    switch deteccao
    case 0 % retorna vl e hl
        if isempty(hl)
            hl = v_zero;
            falha = true;
            disp('O programa não conseguiu detectar linhas válidas')
        elseif isempty(vl)
            vl = v_zero;
            falha = true;
            disp('O programa não conseguiu detectar linhas válidas')
        end
        varargout{1} = vl;
        varargout{2} = hl;
        varargout{3} = falha;
    case 1 % retorna hl
        if isempty(hl)
            hl = v_zero;
            falha = true;
            disp('O programa não conseguiu detectar linhas válidas')
        end
        varargout{1} = hl;
        varargout{2} = falha;
    case 2 % retorna vl
        if isempty(vl)
            vl = v_zero;
```

```
            falha = true;
            disp('O programa não conseguiu detectar linhas válidas')
        end
        varargout{1} = vl;
        varargout{2} = falha;
    end
end

%% Funções
% Retorna o ponto (x,y) no qual duas retas se interceptam.
function [x,y] = intersecao(line1,line2)
    [a1,b1,c1] = eq_reta(line1);
    [a2,b2,c2] = eq_reta(line2);
    A = [a1 b1;a2 b2];
    b = -[c1;c2];
    if any(any(isinf(A))) || any(isinf(b))
        x = 0;
        y = 0;
    else
        Q = linsolve(A,b);
        x = Q(1);
        y = Q(2);
    end
end

% Retorna os parâmetros a, b e c de uma reta que passa pelos dois pontos passados como
parâmetro.
function [a,b,c] = eq_reta(xy)
    try
        x1 = xy(1,1);
        y1 = xy(1,2);
        x2 = xy(2,1);
        y2 = xy(2,2);
    catch
        a = inf;
        b = inf;
        c = inf;
        return
    end
    if x1 == x2
        a = 1;
        b = 0;
        c = -x1;
    else
        m = (y2-y1)/(x2-x1);
        a = m;
        b = -1;
        c = -m*x1+y1;
    end
end
```

APÊNDICE D – VALIDA_LINHAS.M

```
% [vl,hl] = valida_linhas(colorImage, deteccao, interacao, imc)
% Retorna as linhas vertical e horizontal validadas pelo usuário caso necessário e solicita a
intervenção do usuário em situações de falha.

% ENTRADA:
% colorImage é a imagem na qual as linhas serão detectadas
% deteccao = 0 corresponde a encontrar as linhas verticais e horizontais
% deteccao = 1 corresponde a encontrar as linhas horizontais encontradas
% deteccao = 2 corresponde a encontrar as linhas verticais encontradas
% interacao = true corresponde a solicitar a confirmação do resultado com o usuário
% imc = true indica que a imagem de entrada é um corte da imagem original

% SAÍDA:
% lineV = lineH = [x1 y1; x2 y2]
% falha = true indica que as linhas solicitadas não foram encontradas
% rec corresponde aos pontos do retângulo de crop utilizado para corte da imagem

% Usa a função DETECTA_LINHAS

function [varargout] = valida_linhas(colorImage, deteccao, interacao, imc)
    I = colorImage;
    ok = false;
    fail2 = false;
    rec = 0;
    cont = 0;
    v_zero = [0,0;0,0];
    respH = 'Não';
    respV = 'Não';
    opts.Default = 'Sim';
    opts.Interpreter = 'none';

    % Localiza as linhas com base no parâmetro deteccao
    switch deteccao
    case 0
        [lineV, lineH, fail] = detecta_linhas(I,deteccao,imc,interacao);
        if ~fail
            if interacao
                respH = questdlg({'A linha dágua está correta?'},'Linha
dágua','Sim','Não',opts);
                respV = questdlg({'A linha vertical está
correta?'},'Calado','Sim','Não',opts);
                if strcmp(respV,'Sim') && strcmp(respH,'Sim')
                    ok = true;
                    close;
                end
            else
                ok = true;
                close;
            end
        end
    case 1
        [lineH, fail] = detecta_linhas(I,deteccao,imc,interacao);
```

```
        respV = 'Sim';
        if ~fail
            if interacao
                respH = questdlg({'A linha dágua está correta?'},'Linha
dágua','Sim','Não',opts);
                if strcmp(respH,'Sim')
                    ok = true;
                    varargout{3} = rec;
                    close;
                end
            else
                ok = true;
                close;
            end
        end
    case 2
        [lineV, fail] = detecta_linhas(I,deteccao,imc,interacao);
        respH = 'Sim';
        if ~fail
            if interacao
                respV = questdlg({'A linha vertical está
correta?'},'Calado','Sim','Não',opts);
                if strcmp(respV,'Sim')
                    ok = true;
                    close;
                end
            else
                ok = true;
                close;
            end
        end
    end

    % Nos casos de falha na detecção das linhas entra no loop de validação
    % a partir do crop da imagem feito pelo usuário
    while ~ok
        uiwait(msgbox({'Selecione a região da imagem contendo a régua e a linha d`água';...
            'Depois clique com o botão Direito do Mouse';'Selecione "Crop
Image"'},'Atenção','warn','modal'));
        close;
        [I, rec] = imcrop(colorImage);
        close;

        % Caso a linha horizontal esteja correta, localiza a vertical
        if strcmp(respH,'Sim')
            % Localiza o calado
            [lineV, fail] = detecta_linhas(I,2,true,true);
            if ~fail
                lineV = [lineV(:,1) + rec(1), lineV(:,2) + rec(2)];
                % Repetir?
                respV = questdlg({'E agora:','A linha vertical está
correta?'},'Calado','Sim','Não',opts);
            end
        % Caso a linha vertical esteja correta, localiza a horizontal
        elseif strcmp(respV,'Sim')
            % Localiza a linha d'água
            [lineH, fail] = detecta_linhas(I,1,true,true);
```

```
            if ~fail
                lineH = [lineH(:,1) + rec(1), lineH(:,2) + rec(2)];
                varargout{3} = rec;
                % Repetir?
                respH = questdlg({'E agora:','A linha dágua está correta?'},'Linha
dágua','Sim','Não',opts);
            end
        % Caso nenhuma das linhas esteja correta, localiza ambas
        else
            % Localiza o calado e a linha d'água
            [lineV, lineH, fail] = detecta_linhas(I,0,true,true);
            if ~fail
                lineV = [lineV(:,1) + rec(1), lineV(:,2) + rec(2)];
                lineH = [lineH(:,1) + rec(1), lineH(:,2) + rec(2)];
                % Repetir?
                respH = questdlg({'E agora:','A linha dágua está correta?'},'Linha
dágua','Sim','Não',opts);
                respV = questdlg({'E agora:','A linha vertical está
correta?'},'Calado','Sim','Não',opts);
            end
        end
        close;

        % Faz a contagem do número de vezes que o processo foi repetido
        cont = cont+1;
        % No caso de imagens cortadas, a partir da 3ª repetição a função
        % retorna para que o processo seja reiniciado com a imagem original
        if cont > 2 && imc
            fail2 = true;
            ok = true;
            cont = 0;
        end

        % Confere se as linhas estão corretas e caso estejem finaliza o
        % loop
        if (strcmp(respV,'Sim') && strcmp(respH,'Sim'))
            ok = true;
            fail2 = false;
        end
    end

    % Retorna os parâmetros encontrados
    if ~fail2
        if ~fail && (deteccao == 0 || deteccao == 2)
            lineV = estica_linhaV(lineV, size(colorImage));
        end
        switch deteccao
        case 0
            varargout{1} = lineV;
            varargout{2} = lineH;
            varargout{3} = fail2;
            varargout{4} = rec;
        case 1
            varargout{1} = lineH;
            varargout{2} = fail2;
            varargout{3} = rec;
        case 2
```

```
            varargout{1} = lineV;
            varargout{2} = fail2;
            varargout{3} = rec;
        end
    else
        switch deteccao
        case 0
            varargout{1} = v_zero;
            varargout{2} = v_zero;
            varargout{3} = fail2;
            varargout{4} = 0;
        case 1
            varargout{1} = v_zero;
            varargout{2} = fail2;
            varargout{3} = 0;
        case 2
            varargout{1} = v_zero;
            varargout{2} = fail2;
            varargout{3} = 0;
        end
    end
end

% Retorna a linha de entrada com mesma inclinação, porém com o ponto final
% a uma distância de 50 pixels da borda vertical da imagem.
function [line] = estica_linhaV(line, size)
    try
        x1 = line(1,1);
        y1 = line(1,2);
        x2 = line(2,1);
        y2 = line(2,2);
    catch
        return
    end
    if x1 == x2
        line(2,2) = size(1)-50;
    else
        m = (y2-y1)/(x2-x1);
        a = m;
        c = -m*x1+y1;
        line(2,2) = size(1)-50;
        line(2,1) = (line(2,2)-c)/a;
    end

end
```

APÊNDICE E – MAIN.M

```
clear; close all; clc

path = strcat(pwd,'\Videos\');
video = strcat(path, '1e.mp4');
% video = strcat(path, '2a.mp4');
% video = strcat(path, '1a.mp4');
% video = strcat(path, 'WhatsApp Video 2021-03-18 at 12.43.27.mp4');
% video = strcat(path, 'WhatsApp Video 2021-03-28 at 15.07.55.mp4');

vidObj = VideoReader(video);
n_frames = vidObj.NumFrames;

figure
imshow(readFrame(vidObj))

opts.Default = 'Sim';
opts.Interpreter = 'tex';
respQ = questdlg('A imagem apresentada possui qualidade para detecção?','Análise
Imagem','Sim','Não','Cancelar',opts);

if strcmp(respQ, 'Não')
    disp('As imagens não possuem qualidade suficiente para a detecção automática.')
    close;
    return
elseif strcmp(respQ, 'Cancelar')
    disp('O usuário encerrou o programa.')
    close;
    return
end

n_frames_proc = round(n_frames/vidObj.FrameRate);

video = estabiliza(video, 1, n_frames_proc+1, 1);

if video == 0
    disp('As imagens não possuem qualidade suficiente para a detecção automática.')
    close;
    return
end

vidObj = VideoReader(video);
vidObj.CurrentTime = 0;

int_validacao = 10;
lim_linhas = 10;
resp = strings([round(n_frames_proc/int_validacao) 1]);
lH = zeros([2 2 n_frames_proc]);
comp = zeros([1 n_frames_proc]);
falha = 0;
v_zero = [0,0;0,0];
alto = 1;
baixo = 1;
```

```
opts.Default = "Sim";
opts.Interpreter = "none";

for i = 1:n_frames_proc
    frame = read(vidObj,i);
    img = frame;
    tam_frame = size(frame);

    if i == 1
        [lV, lH(:,:,i), falha, rec] = valida_linhas(img,0,true,false);
        lH(:,:,i) = estica_linha(lH(:,:,i), tam_frame);
        [~,y] = intersecao(lV, lH(:,:,i));
        max = y;
        min = y;
    else
        if ~isequal(rec,0)
            img = frame(round(rec(2)):round(rec(2)+rec(4)),
round(rec(1)):round(rec(1)+rec(3)), :);
            [lH(:,:,i), falha] = valida_linhas(img,1,false,true);
            if falha
                [lH(:,:,i), falha] = valida_linhas(frame,1,false,false);
            end
            lH(:,:,i) = [lH(:,1,i) + rec(1), lH(:,2,i) + rec(2)];
            lH(:,:,i) = estica_linha(lH(:,:,i), tam_frame);
        else
            [lH(:,:,i), falha, rec] = valida_linhas(frame,1,false,false);
            lH(:,:,i) = estica_linha(lH(:,:,i), tam_frame);
        end

        [~,y] = intersecao(lV, lH(:,:,i-1));
        [~, yN] = intersecao(lV, lH(:,:,i));

        comp(i) = abs(yN-y);
        disp('Comp:')
        disp(comp(1:i))

        if comp(i) > lim_linhas
            lH(:,:,i) = estica_linha(lH(:,:,i), tam_frame);
            figure
            imshow(frame)
            hold on
            plot(lH(:,1,i),lH(:,2,i),'LineWidth',2,'Color','green');
            hold off
            respH = questdlg({'A linha dágua está correta?'},'Linha dágua','Sim','Não',opts);
            close;
            if strcmp(respH,'Não')
                [lH(:,:,i), falha, rec] = valida_linhas(frame,1,true,false);
                lH(:,:,i) = estica_linha(lH(:,:,i), tam_frame);
            end
            [~, yN] = intersecao(lV, lH(:,:,i));
        end
        if yN > max
            max = yN;
            baixo = i;
        elseif yN < min
            min = yN;
```

```
            alto = i;
        end
    end

    if mod(i,int_validacao) == 0
        figure
        imshow(frame)
        hold on
        title('Imagem Confere Linhas')
        plot(lV(:,1),lV(:,2),'LineWidth',2,'Color','yellow');
        plot(lH(:,1,alto),lH(:,2,alto),'LineWidth',2,'Color','red');
        plot(lH(:,1,baixo),lH(:,2,baixo),'LineWidth',2,'Color','blue');
        plot(lH(:,1,i),lH(:,2,i),'LineWidth',2,'Color','green');
        legend('Calado', 'Linha dágua mais alta', 'Linha dágua mais baixa', 'Linha dágua Frame
atual')
        hold off

        opts.Default = "Sim";
        resp(i/int_validacao) = questdlg({'As respostas obtidas estão corretas?',...
        'E o crop utilizado está funcionando?'},'Validação', "Sim", "Não","Cancelar",opts);
        if strcmp(resp(i/int_validacao), "Não")
            rec = 0;
        elseif strcmp(resp(i/int_validacao), "Cancelar")
            close;
            disp('O usuário encerrou a execução do programa.')
            return
        end
        close;
    end
end

%% Finalização
frame = read(vidObj,1);
medY1 = mean(lH(1,2,:));
medY2 = mean(lH(2,2,:));
lHfinal = [0 medY1; tam_frame(2) medY2];
[x, y] = intersecao(lHfinal, lV);

% Mostra todas as linhas detectadas
% figure
% imshow(frame)
% hold on
% title('Imagem Linhas Detectadas')
% plot(lV(:,1),lV(:,2),'LineWidth',2,'Color','green');
% for i = 1:n_frames_proc
%     plot(lH(:,1,i),lH(:,2,i),'LineWidth',2,'Color','green');
% end
% plot(lHfinal(:,1),lHfinal(:,2),'LineWidth',3,'Color','blue');
% plot(lH(:,1,alto),lH(:,2,alto),'LineWidth',2,'Color','white');
% plot(lH(:,1,baixo),lH(:,2,baixo),'LineWidth',2,'Color','black');
% plot(x,y,'X','LineWidth',4,'Color','red')
% hold off
% pause(2)
% close

% Mostra o resultado final da detecção
figure
```

```
imshow(frame)
hold on
title('Imagem Avaliação Geral')
plot(lV(:,1),lV(:,2),'LineWidth',2,'Color','green');
plot(lHfinal(:,1),lHfinal(:,2),'LineWidth',3,'Color','green');
plot(x,y,'X','LineWidth',4,'Color','red')
hold off

opts.Default = "Sim";
resposta = questdlg('As respostas obtidas estão corretas?','Validação', "Sim", "Não",opts);
close;

%% Funções
function [a,b,c] = eq_reta(xy)
    x1 = xy(1,1);
    y1 = xy(1,2);
    x2 = xy(2,1);
    y2 = xy(2,2);
    if x1 == x2
        a = 1;
        b = 0;
        c = -x1;
    else
        m = (y2-y1)/(x2-x1);
        a = m;
        b = -1;
        c = -m*x1+y1;
    end
end

function [x,y] = intersecao(line1,line2)
    [a1,b1,c1] = eq_reta(line1);
    [a2,b2,c2] = eq_reta(line2);
    A = [a1 b1;a2 b2];
    b = -[c1;c2];
    if any(any(isinf(A))) || any(isinf(b))
        x = 0;
        y = 0;
    else
        Q = linsolve(A,b);
        x = Q(1);
        y = Q(2);
    end
end

function [line] = estica_linha(line, size)
    [a,~,c] = eq_reta(line);
    line(1,1) = 0;
    line(1,2) = a*line(1,1)+c;
    line(2,1) = size(2);
    line(2,2) = a*line(2,1)+c;
end
```