NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

RÚBEN FILIPE JACINTO DOS SANTOS

Licenciatura/BSc em Engenharia Informática

DETEÇÃO DE IMPACTO DE EMBARCAÇÕES NA ATRACAÇÃO A UM PORTO

OBTENÇÃO DAS FORÇAS PROVOCADAS PELA EMBARCAÇÃO NO PORTO AQUANDO DA ATRACAÇÃO DESTA AO CAIS

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Dezembro, 2022

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

DETEÇÃO DE IMPACTO DE EMBARCAÇÕES NA ATRACAÇÃO A UM PORTO

OBTENÇÃO DAS FORÇAS PROVOCADAS PELA EMBARCAÇÃO NO PORTO AQUANDO DA ATRACAÇÃO DESTA AO CAIS

RÚBEN FILIPE JACINTO DOS SANTOS

Licenciatura/BSc em Engenharia Informática

Orientador: Nuno Miguel Cavalheiro Marques
Prof. Auxiliar, Universidade Nova de Lisboa

Coorientadores: João Marcelino
Investigador Principal, Laboratório Nacional de Engenharia Civil

João Manso
Investigador Auxiliar, Laboratório Nacional de Engenharia Civil

Júri

Presidente: Carmen Morgado
Professora Auxiliar, FCT-NOVA

Arguente: Luís Cavique
Professor Auxiliar, Universidade Aberta

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Dezembro, 2022

Deteção de impacto de embarcações na atracação a um porto

Para os meus pais, irmã e namorada,

Agradecimentos

Gostaria primeiramente de agradecer ao meu orientador, o Prof. Nuno Marques por me ter ter dado a oportunidade de trabalhar nesta dissertação, juntamente com os engenheiros do LNEC, Eng. João Marcelino e Eng. João Manso, pois sem eles esta dissertação não existiria. Agradeço também à Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa por todo o conhecimento transmitido durante este curso.

Gostaria de agradecer à minha família e em especial aos meus pais e à minha irmã, que sempre acreditaram em mim e me apoiaram. E também aos meus avós, tanto os que ainda cá estão e os que já não, por terem ajudado a criar-me e contribuido para quem sou, e aos meus padrinhos e aos meus primos que sempre me apoiaram.

Gostaria também de agradecer à minha namorada e a todos os meus amigos, em especial ao Bruno, Bárbara, Sara, José, Raúl, mas sem nunca esquecer todos os outros que me apoiaram durante o curso, a tese e a vida. Muito obrigado.

"Thought is the wind, knowledge the sail, and mankind the Vessel." (August Hare)

Resumo

As estruturas portuárias, nomeadamente as estruturas de acostagem de navios de carga ou de passageiros são fortemente solicitadas pelas acções dinâmicas que resultam do processo de acostagem das embarcações e também em resultado da agitação que as embarcações sofrem fruto da acção do vento, agitação marítima e marés. No âmbito desta dissertação é de especial interesse estudar o risco de dano no momento em que a embarcação se encontra a aproximar do cais para a sua amarração. Analisa-se uma metodologia e são estudas e testadas ferramentas de software que, partindo de um filme da manobra de amarração, tentam determinar a força de impacto da embarcação no cais durante essa manobra. Assim, esta dissertação tem como objectivo o estudo, validação e teste inicial da aplicação criada recorrendo a diversas tecnologias já existentes, principalmente o K-nearest neighbor (KNN) BackgroundSubtractor do OpenCV. Para tal foi desenvolvida uma aplicação que implementa mecanismos que estimem o impacto provocado pela embarcação. A avaliação de eficácia da metodologia escolhida será feita com recurso a modelos à escala que representem a acostagem de uma embarcação a um cais, comparando os valores medidos com os valores de impacto estimados pela aplicação.

Palavras-chave: amarração, embarcações, porto, análise de dano, processamento imagem, OpenCV

Abstract

Port structures, namely the berthing structures of cargo or passenger ships, are heavily requested by the dynamic actions that result from the process of berthing of vessels and also as a result of the agitation that vessels suffer as a result of the action of wind, maritime agitation and tides. Within the scope of this dissertation, it is of special interest to study the risk of damage to the pier when the vessel is approaching for mooring. As such, this dissertation has as main interest the application of existing technologies, mainly KNN BackgroundSubtractor from OpenCV. To this end, an application was developed that implements mechanisms that estimate the impact caused by the vessel using the video feed from the surveillance system of the seaport. The evaluation of the effectiveness of this methodology will be made using scale models that represent the mooring of a vessel to a pier, comparing the measured values with the estimated impact values from the application.

Keywords: mooring, vessel, port, damage assessment, image processing, OpenCV

Índice

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Glossário

BackgroundSubtractor É um algoritmo que é usado para separar os objetos de primeiro plano da imagem de fundo na sequência de vídeo contínua. Ao alimentar os quadros do vídeo como entrada, a máscara de primeiro plano binária (ou ternária, se incluir regiões de sombra) e uma estimativa da imagem de fundo estática (se ativada) serão produzidas vii, viii, 30, 31, 53

defensas Objectos moles colocados ao longo do cais para o proteger de colisões com embarcações atracadas. xvii, 1

Fenómeno de Runge Em matemática, em particular no campo específico da análise numérica, o fenômeno de Runge é um problema de oscilação nas bordas de um intervalo, que ocorre quando se usa interpolação polinomial com polinómios de ordem elevada. xvii

frame Cada uma das imagens fixas constituintes de um vídeo 33

frame rate A quantidade de frames que ocorrem por cada unidade de tempo, normalmente por segundo (fps). 33

Geometria Projectiva A geometria projectiva é o ramo da matemática que estuda a relação entre figuras geométricas e a imagem resultante da sua projecção noutra superfície. 11

IMU Inertial Measurement Unit - É um dispositivo eletrónico que mede e relata a força específica de um corpo, a taxa angular e, às vezes, a orientação do corpo, usando uma combinação de acelerómetros, giroscópios e, às vezes, magnetómetros. 2, 24, 27

kernel Semelhante a uma função de similaridade, para cada dois objectos este gera uma pontuação de similaridade. Kernels permitem a computação de produtos escalares em situações nas quais sem a sua utilização seria muito díficil efectuar a computação destes produtos escalares. [35] 9

OpenCV Biblioteca de funções de programação voltada principalmente para visão computacional em tempo real. Originalmente desenvolvido pela Intel, mais tarde foi suportado pela Willow Garage e depois pela Itseez. A biblioteca é multiplataforma e gratuita para uso sob a licença Apache 2 de código aberto. vii, viii, 3, 4, 5, 8, 11, 15, 17, 18, 28, 30, 31

popa A parte traseira das embarcações. xi, xii, 24, 37, 41

proa A parte dianteira das embarcações. xi, 24, 37, 41

Spline Em matemática, um spline é uma função especial definida por partes por polinómios. Em problemas de interpolação, a interpolação spline é frequentemente preferida à interpolação polinomial porque produz resultados semelhantes, mesmo ao usar polinômios de baixo grau, evitando o Fenómeno de Runge para graus mais altos. 33

tracking Seguimento do movimento de um objecto num plano de filmagem. 11

vigas cais Vigas metálicas colocadas junto ao cais de betão de um porto marítimo com o intuito de reforçar este, bem como suportar as defensas. 1

Siglas

CSRT Channel and Spatial Reliability Tracker 17

HOG Histogram of Oriented Gradients 15

KNN K-nearest neighbor vii, viii, 9, 30

LNEC Laboratório Nacional de Engenharia Civil 1, 2, 3, 15, 24, 33

NTP Network Time Protocol 24, 27

OCR Optical Character Recognition 18
ORB Oriented FAST and Rotated BRIEF 17

RCNN Region-based Convolutional Neural Networks 16

SAM Selective Angle Measurements 18
SIFT Scale-invariant feature transform 17
SSD Single Shot Detector 16
STCN Space Time Correspondence Network 16
STM Space Time Memory 16
SURF Speeded up robust features 17
SVM Support Vector Machines 15

TMA Target Motion Analysis 18

YOLO You Only Look Once 16

1

Introdução

1.1 Problema e Motivação

As estruturas portuárias, nomeadamente as estruturas de acostagem de navios de carga ou de passageiros são fortemente solicitadas pelas acções dinâmicas que resultam do processo de acostagem das embarcações e também em resultado da agitação que as embarcações sofrem fruto da acção do vento, agitação marítima e marés. Os cabeços de amarração e as defensas instaladas nas vigas cais foram, em alguns casos dimensionadas há muitos anos, para embarcações de menor capacidade que as que existem actualmente. Por outro lado, por se encontrarem em ambientes desfavoráveis, estas estruturas têm de ser convenientemente acompanhadas para melhor avaliar os ciclos de investimento associados à sua reparação / manutenção / substituição. Anteriormente, o Laboratório Nacional de Engenharia Civil (LNEC) já avaliou o efeito das correntes marítimas numa embarcação amarrada[39], sendo possível avaliar o nível de dano expectado no cais pelo impacto da embarcação amarrada provocada por movimentos marítimos.

Para complementar a informação de dano expectado pelo impacto da embarcação devido a movimentos marítimos pretende-se também obter o impacto da embarcação no momento em que a embarcação se encontra a deslocar para junto do cais onde a embarcação será amarrada. No âmbito de um projecto em curso que envolve o LNEC, a Universidade do Minho, a Administração do Porto de Leixões e a empresa 3Maps irá ser feita uma instalação experimental que se destina a medir, com sensores adequados o efeito das acções dinâmicas sobre as estruturas de acostagem. Porém a instalação de tais sensores é muito onerosa e tem, nesta fase, um carácter pontual. Isto é, os sensores limitam-se à pequena parcela da infraestrutura portuária onde estão instalados. Nesse contexto surge o interesse em desenvolver uma abordagem mais abrangente, baseada no processamento de imagem vídeo e na extracção dos parâmetros de movimento de embarcações. No decorrer de tal necessidade o DI NOVA FCT está a colaborar com o LNEC na elaboração de uma ferramenta que permita a avaliação de tais efeitos de uma maneira menos onerosa, recorrendo para tal a ferramentas existentes, aplicando-as à monitorização das embarcações.

1.2 Objectivos

Esta dissertação de mestrado tem como principal objectivo desenvolver e estudar a viabilidade de ferramentas de software para monitorização de danos em cais portuários, recorrendo maioritariamente à utilização de vídeos efetuados por câmaras no próprio porto. Para tal, pretende-se analisar e desenvolver software que permita utilizar o feed de vídeo existente do sistema de videovigilância em conjunto com a ferramenta OpenCV[1] de maneira a calcular a velocidade de impacto das embarcações no cais na sua aproximação a este para a sua amarração. Com a velocidade da embarcação calculada será possível obter-se a força de impacto tendo acesso à massa da embarcação.

Para automatizar o cálculo da força pretende-se dar a possibilidade de ser inserida a massa da embarcação que será utilizada para calcular a força resultante do movimento da embarcação.

Pretende-se também desenvolver software para instrumentar modelos de teste, realizados nos tanques hidráulicos do LNEC. Com esta instrumentação será possível obter dados relevantes para efectuar a validação dos resultados obtidos pela aplicação e para futura análise deste problema.

1.3 Abordagem

Inicialmente foi efectuada uma pesquisa bibliográfica relativa à identificação de objectos e o seu acompanhamento de forma a permitir determinar a trajectória das embarcações, a sua velocidade, bem como possibilitar a introdução de pontos de referência para transformação das coordenadas na imagem em coordenadas reais, permitindo a obtenção de unidades de medida aproximadas às reais.

O sistema desenvolvido será aplicado a arquivos videográficos relativos a modelo reduzido desenvolvido no LNEC, para simulação das condições de acostagem de embarcações. Para facilitar a obtenção dos parâmetros do movimento, em particular na acostagem, o modelo será instrumentado com acelerómetros que facilitarão a obtenção dos parâmetros do movimento nessa fase. Estes acelerómetros são parte de um sistema de recolha de dados constituido por um micro-controlador ESP32 e um IMU.

2

Estado da Arte

2.1 Estudos LNEC

Como referido anteriormente o LNEC efectuou uma avaliação dos riscos de embarcações atracadas num porto [39], tendo desenvolvido um sistema de alerta SWAMS ALERT. O SWAMS ALERT prevê e avisa sobre os riscos associados a uma embarcação atracada num porto, em particular os provocados pelas condições atmosféricas. A possibilidade de prever estes riscos permite aumentar a segurança das operações e estruturas portuárias. Para tal o SWAMS ALERT identifica e monitoriza situações de emergência relacionada com embarcações atracadas e de seguida notifica o responsável e activa planos de emergência. Para efectuar esta previsão este sistema recorre a previsões e medições de caracteristícas das ondas marítimas. Com estas é possível obter o movimento vertical das embarcações em movimento e atracadas. Este sistema permitirá assim saber as forças nas linhas de cais, que interessa para o nosso caso de uso, mas não irá permitir saber as forças na linha de cais quando o impacto não se deve ao movimento maritimo, mas sim pelo movimento da embarcação quando se aproxima do cais para ser atracada.

2.2 Ferramenta

Para a implementação da nossa aplicação de tracking e calculo de força de impacto de embarcações é necessário a utilização de ferramentas que efectuem parte destas tarefas de forma a minimizar o desenvolvimento necessário. Verificou-se portanto a existência de ferramentas que permitiam o tracking de objectos.

2.2.1 OpenCV

O OpenCV [1] é uma biblioteca de ferramentas de computação visual open source. Permite reconhecimento facial e de gestos, entre muitas outras funcionalidades. Permite também através dos vários feeds de vídeo utilizar também a percepção de profundidade recorrendo a mais que uma câmara. Permite também a detecção de texto, útil na identificação da embarcação a ser analisada. Todas estas funcionalidades mostram algum potencial,

mas nesta tese focamo-nos na detecção de objectos, bem como o acompanhamento do seu percurso. De forma a reduzir-se o custo e simplificar a abordagem aqui aplicada não se irá recorrer à percepção de profundidade, utilizando-se outro método. Detectar uma embarcação no vídeo fornecido e seguir o percurso desta no momento do embate são funcionalidades do OpenCV que nos serão mais importantes. Outra funcionalidade do OpenCV que será muito útil é a capacidade deste de com base num conjunto de imagens com um xadrez criar uma matriz de calibração que permitirá para cada frame remover a distorção criada pela lente da cãmara que produziu o ficheiro de vídeo.

2.2.2 Alternativas

Existem várias alternativas ao OpenCV, sendo a maioria proprietária. Foram analisadas alternativas encontradas num website de listagem de softwares concorrentes [2] no seu segmento de mercado, sendo verificadas se estas se poderiam aproximar das funcionalidades necessárias para o desenvolvimento desta ferramenta. Na tabela 2.1 podemos observar uma comparação entre estas alternativas e o OpenCV.

Tabela 2.1: Software para Computação visual.

Software	Open source	Preço	Análise
OpenCV [1]	Sim	Grátis	Imagens e vídeo
Microsoft Computer Vision API [3]	Não	Pago	Imagens e vídeo
Amazon Rekognition [4]	Não	Pago	Imagens e vídeo
Google Cloud Vision API [5]	Não	Pago	Imagens
scikit-image [6]	Sim	Grátis	Imagens
SimpleCV [7]	Sim	Grátis	Imagens e vídeo
IBM Watson Visual Recognition [8]	Não	Pago [1]	Imagens

Software	Tracking	Info Extra
OpenCV [1]	Configurável	Amplamente utilizado
Microsoft Computer Vision API [3]	Apenas pessoas	Análise espacial [2]
Amazon Rekognition [4]	Apenas pessoas	
Google Cloud Vision API [5]	Apenas pessoas	
scikit-image [6]	Configurável	Poderoso, mas não permite processamento real-time
SimpleCV [7]	Configurável	OpenCV mais fácil de utilizar [3]
IBM Watson Visual Recognition [8]		

Todas as ferramentas que não processam vídeo poderiam ser usadas se convertêssemos este numa colecção de imagens. Para tal teria de se desenvolver mecanismos de relacionar diversas imagens e obtermos resultados que podemos obter de forma mais eficaz com processamento directo de vídeo. Uma vez que se permite desenvolver uma aplicação com

[1] 1000 análises por mês grátis
[2] Grátis, mas apenas durante a fase de preview
[3] Implementa e facilita a utilização de OpenCV. Não é actualizado há 7 anos [9]

um custo mais reduzido não poderemos considerar as alternativas da Microsoft, Amazon, Google e IBM. Consideremos assim apenas o software gratuito. Como se pode verificar na tabela 2.1 o OpenCV além de ser gratuito, tal como o scikit-image e SimpleCV, também apresenta sobre estes a vantagem de ser capaz de processar video directamente e ser activamente actualizado. Processar directamente o vídeo melhora o fluxo de execução da aplicação e o ser activamente actualizado garante suporte para as funcionalidades aqui descritas, bem como a possibilidade de utilização de futura funcionalidades novas que possam ser interessantes à aplicação. Tendo em conta estes factores escolheu-se a utilização do OpenCV.

2.3 Deteção de objectos

O componente principal da aplicação aqui desenvolvida é a detecção de objectos, mais especificamente a detrecção das embarcações das quais será analisado o seu embate com o cais. De forma a analisarmos apenas as embarcações que se encontram em movimento pretendemos descobrir a diferença entre a embarcação e o plano de fundo. Com tal poderemos proceder ao seguimento da embarcação. Existem vários métodos tendo por base a remoção das partes da imagem que se mantêm fixas.

2.3.1 Absdiff

Este é o método mais básico presente no OpenCV. A avaliação é feita a cada par de frames e o resultado consistirá na diferença entre ambos. Graças ao movimento do objecto existirá mudanças nas imagens suficiente para que seja possível identificar onde este se encontra na imagem e seguir o seu movimento.

Na figura 2.1 encontra-se o resultado dos testes feitos ao método absDiff com base no código Python da listagem 1. Neste foi utilizado uma pequena bola de borracha a atravessar o plano e a colidir com uma parede. Como se pode verificar na figura 2.1, o AbsDiff permite-nos obter apenas s bola, mas com alguma deformação pelo facto de ser a diferença entre dois frames consecutivos. Apresenta assim artefactos de ambos os frames para construir a imagem detectada da bola.

2.3.2 Background Subtractor

Em OpenCV existem dois modelos base de tipo BackgroundSubtractor, que são implementações de modelos diferente descritos por Zivkovic[49]. As diferenças entre estes encontram-se nas secções abaixo dedicadas a cada um deles.

Estes avaliam para todos os pixeis se estes pertencem ao plano de fundo (BG) ou ao primeiro plano (FG), gerando um modelo de background que será comparado ao frame actual e caso não pertença ao fundo será apresentado a branco enquanto o fundo será apresentado a preto.

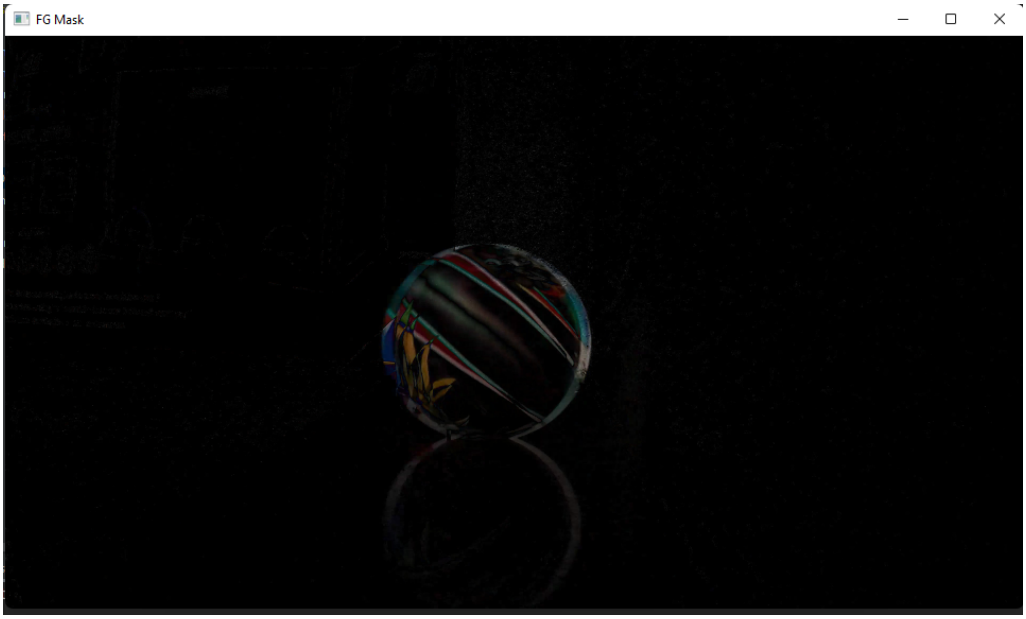

Figura 2.1: Resultado da aplicação do AbsDiff a dois frames consecutivos

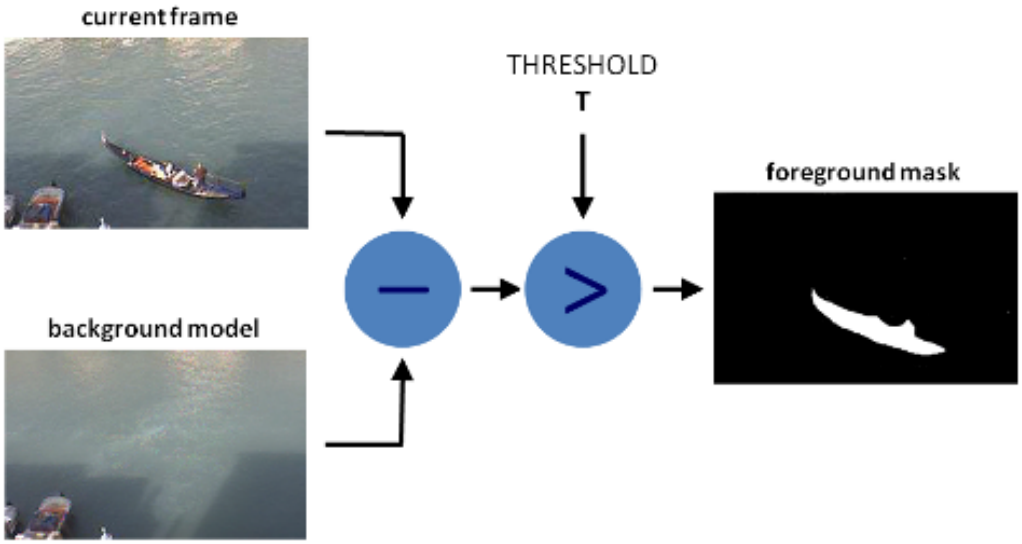

Figura 2.2: Esquema de funcionamento de um background subtractor [10]

```
import cv2

cap = cv2.VideoCapture('video_files/GH010347.MP4')
cap.set(cv2.CAP_PROP_BUFFERSIZE, 40)
cap2.set(cv2.CAP_PROP_BUFFERSIZE, 40)

col_images = []

success, img = cap.read()

while cap.isOpened():
    successNext, imgNext = cap.read()

    if successNext:
        fgMask = cv2.absdiff(img, imgNext)
        fgMaskRs = cv2.resize(fgMask, (960, 540))
        cv2.imshow('FG Mask', fgMaskRs)

        if cv2.waitKey(1) & 0xff == ord('q'):
            break

        img = imgNext

    if not successNext:
        break

cv2.destroyAllWindows()
cap.release()
```

Listing 1: Código implementação AbsDiff.

O modelo de background é estimado por um conjunto de treino. A vantagem da existência deste plano de treino será por exemplo permitir continuar a identificar o plano de fundo ao longo do dia apesar das mudanças de luminosidade. Bem como incorporar mudanças a este, por exemplo adição de novas estruturas. Quando um objecto permanece estático este irá aumentar progressivamente a sua probabilidade de fazer parte do plano de fundo. Ou seja, quanto mais tempo este se encontrar parado mais provável é este fazer parte do plano de fundo. Isto significará que eventualmente este objecto será considerado parte do plano de fundo se continuar estático. As amostras têm o seu peso reduzido de forma exponencial, desta maneira quanto mais tempo a amostra se encontrar estática com mais probabilidade podemos assumir que esta corresponde ao plano de fundo.

Background Subtractor MOG2

O sistema Background Subtractor MOG2 [49] [50] é uma implementação do modelo de mistura Gaussiano. O modelo de mistura Gaussiano é uma categoria do modelo de probabilidades que define todos os pontos de dados gerados são derivados de uma mistura de distribuições Gaussianas finitas. No caso do OpenCV este modela cada pixel de fundo com uma mistura de k distribuições Gaussianas, com k entre 3 e 5. Os valores da mistura são as proporções de tempo em que as cores devem ficar em cena. Quanto mais provável for uma cor pertencer ao fundo mais tempo esta ficará em cena. Portanto as cores que se encontrarem estáticas durante mais tempo pertencerão ao fundo e não farão parte da nossa embarcação.

Na figura 2.3 encontra-se o resultado dos testes feitos ao método BackgroundSubtractorMOG2 do OpenCV com base no código Python da Listagem 2.

```
import cv2

cap = cv2.VideoCapture('video_files/PXL_20210522_093608367.mp4')
cap.set(cv2.CAP_PROP_BUFFERSIZE, 40)
BS_MOG2 = cv2.createBackgroundSubtractorMOG2()

while cap.isOpened():
    success, img = cap.read()

    fgMog = BS_MOG2.apply(img)
    fgMogRs = cv2.resize(fgMog, (960, 540))  # Resize image

    cv2.rectangle(fgMogRs, (10, 2), (140, 20), (255, 255, 255), -1)
    cv2.putText(fgMogRs, str(cap.get(cv2.CAP_PROP_POS_FRAMES)), (15, 15),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))
    cv2.putText(fgMogRs, str(int(cap.get(cv2.CAP_PROP_FPS))), (75, 15),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))
    cv2.imshow("Foreground MOG", fgMogRs)

    if cv2.waitKey(1) & 0xff == ord('q'):
        break

cv2.destroyAllWindows()
cap.release()
```

Listing 2: Código de implementação do BackgroundSubtractor MOG2.

Aplicando o mesmo vídeo com uma pequena bola de borracha a atravessar o plano e a colidir com uma parede. Como se pode verificar este permite-nos obter a bola, mas ao contrário do caso anterior a bola não apresenta deformação devido a esta computação não corresponder a uma relação directa entre frames consecutivos, mas sim com o plano de fundo calculado.

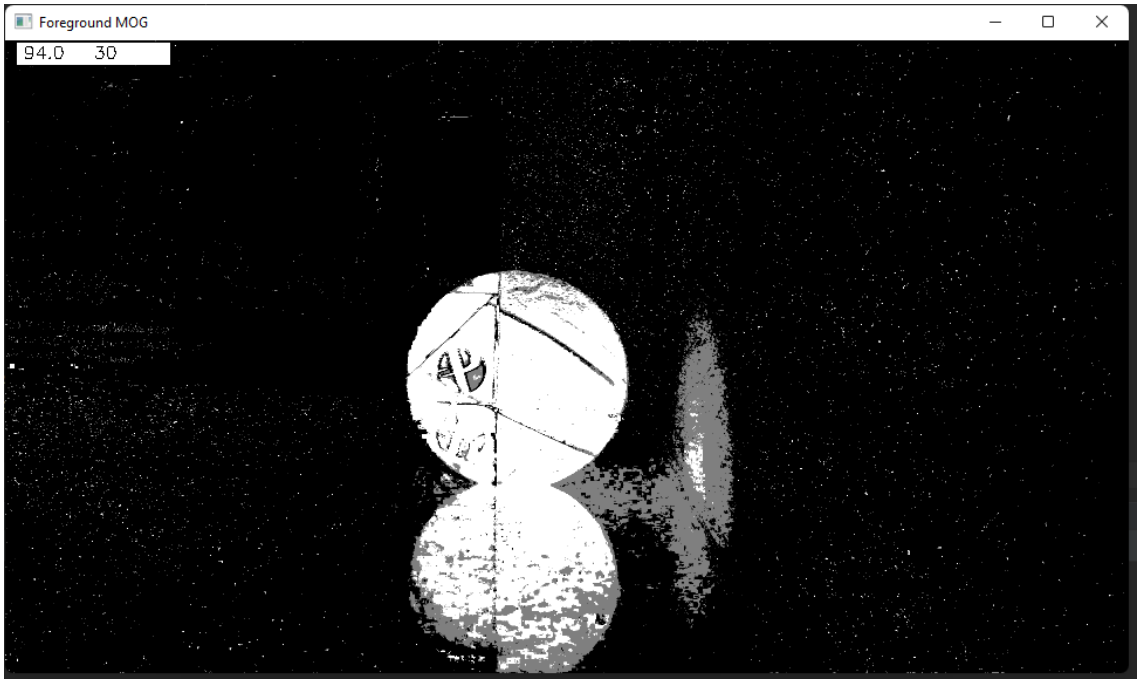

Figura 2.3: Frame de teste MOG2

Background Subtractor KNN

O Background Subtractor KNN é uma implementação do modelo de K-nearest neighbours [49]

A estimativa da densidade deste método é feita com um kernel, subconjunto da amostra, uniforme. Esta é feita contando o número de amostras k do conjunto de dados X_t que se encontram no volume V do kernel. O volume V é a hiperesfera com diâmetro D

Este utiliza um estimador de densidade de kernel variável balão. Ou seja, este adapta o tamanho do kernel a cada estimação. Ao invés de tentar encontrar o valor óptimo de D incrementa-se este valor até um valor fixo de dados k estarem cobertos por este, crescendo como se tratasse de um balão, daí a designação. Isto levará a uma implementação eficiente que seria equivalente a utilizar um kernel fixo uniforme. Quanto maior for o valor de k menor será o ruído gerado no output, mas também aumenta o tamanho da computação.

Na figura 2.4 encontra-se o resultado dos testes feitos ao método KNN com base no código Python da listagem 3

Aplicando o mesmo vídeo com uma pequena bola de borracha a atravessar o plano e a colidir com uma parede. Como se pode verificar na figura 2.4 este permite-nos obter a bola sem deformação, como acontece no caso anterior, mas apresentando muito menos ruído que o caso anterior. Sendo portanto mais provavelmente o que apresentará melhores resultados para o seguimento da bola.

```
import cv2

cap = cv2.VideoCapture('video_files/PXL_20210522_093608367.mp4')
cap.set(cv2.CAP_PROP_BUFFERSIZE, 40)

BS_KNN = cv2.createBackgroundSubtractorKNN()

while cap.isOpened():
    success, img = cap.read()
    fgKnn = BS_KNN.apply(img)
    fgKnnRs = cv2.resize(fgKnn, (960, 540))  # Resize image

    cv2.rectangle(fgKnnRs, (10, 2), (140, 20), (255, 255, 255), -1)
    cv2.putText(fgKnnRs, str(cap.get(cv2.CAP_PROP_POS_FRAMES)), (15, 15),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))
    cv2.putText(fgKnnRs, str(int(cap.get(cv2.CAP_PROP_FPS))), (75, 15),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0))
    cv2.imshow("Foreground KNN", fgKnnRs)

    if cv2.waitKey(1) & 0xff == ord('q'):
        break

cv2.destroyAllWindows()
cap.release()
```

Listing 3: Código de implementação do BackgroundSubtractor KNN.

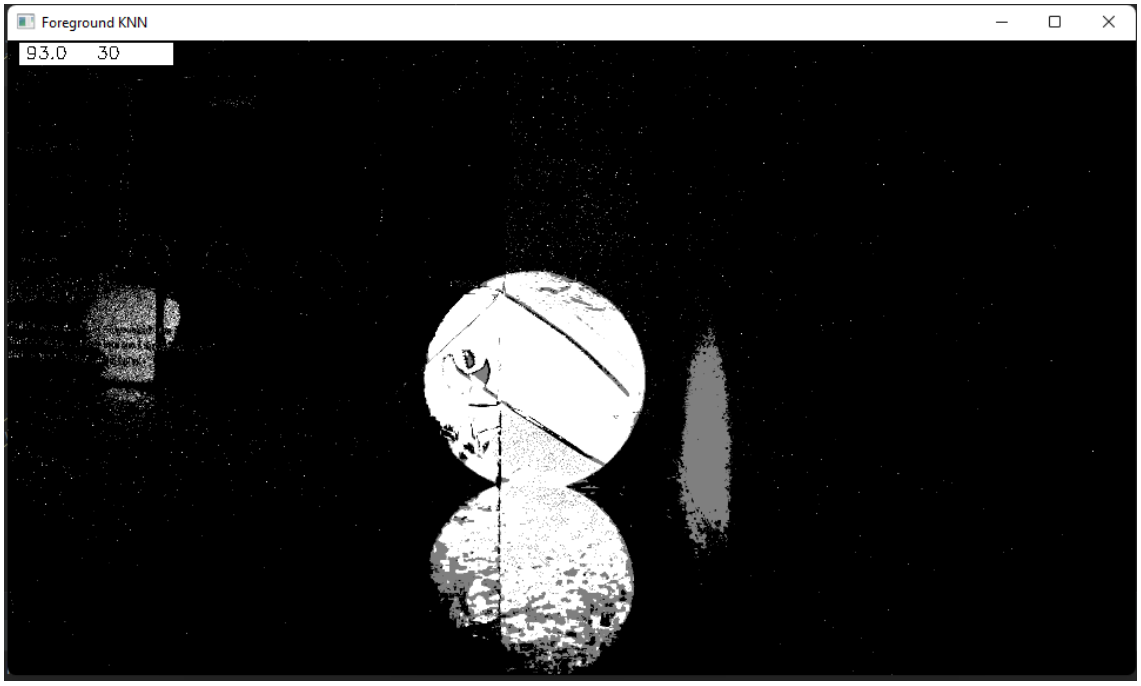

Figura 2.4: Frame de teste KNN

2.4 *Tracking*

O OpenCV tem vários algoritmos de *tracking* de objectos, como se pode confirmar na secção 2.7.3, mas nesta dissertação irá recorrer-se à detecção de contornos presentes numa imagem. Esta função do OpenCV recorre ao algoritmo [43] para encontrar os contornos. Após obtermos estes contornos iremos utilizar a ferramenta centroidTracker[11] para classificar os centroides e os reconhecer como o mesmo objecto entre frames, tal como se poderá observar na secção 4.2.4.

2.5 Escala 3D com base em imagens 2D

Devido à recolha de imagens se basear em câmaras de vigilância estas não têm informação de profundidade, produzindo apenas uma imagem a duas dimensões. Para efectuar os cálculos de velocidade das embarcações é necessário converter a imagem 2D numa imagem 3D. Isto permitirá transformar o movimento em pixeis na imagem em movimento "métrico"no mundo real.

Primeiramente será necessário aplicar uma calibração [12] à câmara utilizada de forma a minimizar a distorção provocada por esta, caso exista. Para fazer a calibração será necessário mostrar um alvo conhecido, como uma grelha de xadrez o que irá permitir calcular a distorção na imagem e com tal obter a matriz intrínseca da câmara. Com a matriz intrínseca poderíamos usar os outros métodos presentes no módulo de calibração e obter uma imagem uniforme. O que mais facilmente nos permitirá uma relação directa entre coordenadas gráficas para coordenadas reais.

Para obtermos uma escala 3D a partir de uma única câmara sem calibração podemos, depois da calibração efectuada, recorrer a uma matriz de transformação com base em pontos de referência de distâncias conhecidas. Tendo as distâncias reais e as distâncias na imagem entre os pontos de referência conhecidas é possível transformar outras medidas no mesmo plano nas imagens da câmara em distâncias reais aproximadas. Chama-se a esta técnica Geometria Projectiva. Com base nos algoritmos desenvolvidos a partir desta área da matemática é possível obtermos a transformação necessária sem a necessidade de se conhecerem os parâmetros internos da câmara utilizada (distância focal, dimensões da imagem, abertura focal, entre outros) ou mesmo os parâmetros externos, como a sua posição e orientação. Para tal será necessário desenvolver este processo ao longo desta dissertação.

A forma de efectuar esta conversão, apesar da investigação dos processos referidos anteriormente será a inicialmente proposta para esta dissertação. Esta transformação entre as coordenadas da fotografia e as coordenadas reais é feita recorrendo a um processo semelhante ao utilizado para as funções de forma usada no método dos elementos definidos [48]. As funções de forma consideradas são polinómios incompletos do segundo grau usados para mapear o espaço em pixeis da fotografia no espaço real, em metros. Para

calcular as constantes dos polinómios é necessário conhecer as coordenadas de 4 pontos que definem um poligno de 4 lados em ambos os espaços.

Dados 2 sistemas de coordenadas S1 e S2 onde são conhecidas as coordenadas de 4 pontos que definem um polígono P em cada referencial, definir a transformação entre os sistemas.

Dados: Sistema 1, pontos conhecidos

$$P_{S_1} = \begin{bmatrix} (x_1, y_1) \\ (x_2, y_2) \\ (x_3, y_3) \\ (x_4, y_4) \end{bmatrix}$$

Sistema 2, pontos conhecidos

$$P_{S_2} = \begin{bmatrix} (u_1, v_1) \\ (u_2, v_2) \\ (u_3, v_3) \\ (u_4, v_4) \end{bmatrix}$$

Pretende-se que $(x_i, y_i) \rightarrow (u_i, v_i)$.

Para converter as coordenadas do referencial 1, para o referencial 2, definem-se funções de interpolação que permitem mapear a forma P(x,y) em P(u,v). Essas funções são do tipo:

$$u_i = N_1(x_i, y_i)x_1 + N_2(x_i, y_i)x_2 + N_3(x_i, y_i)x_3 + N_4(x_i, y_i)x_4$$

e

$$v_i = N_1(x_i, y_i)y_1 + N_2(x_i, y_i)y_2 + N_3(x_i, y_i)y_3 + N_4(x_i, y_i)y_4 \tag{2.1}$$

ou

$$u_i = \sum_{k=1}^{k=4} N_k(x_i, y_i)x_1 + N_2(x_i, y_i)x_2 + N_3(x_i, y_i)x_3 + N_4(x_i, y_i)x_4$$

e

$$v_i = \sum_{k=1}^{k=4} N_k(x_i, y_i)y_1 + N_2(x_i, y_i)y_2 + N_3(x_i, y_i)y_3 + N_4(x_i, y_i)y_4 \tag{2.2}$$

As funções N_i são do tipo polinomial, contínuas, função de x e y, associadas a cada um dos pontos, i.

Para fazer a interpolação considerando 4 pontos com coordenadas conhecidas, os polinómios podem ter 4 parâmetros, sendo, portanto do tipo:

$$N_i = a_i + b_i x + c_i y + d_i x.y \tag{2.3}$$

Que correspondem a polinómios do segundo grau incompletos (faltam os termos em x^2 e y^2)

Assim, para que a expressão anterior seja válida em cada um dos pontos conhecidos, é necessário que se tenha:

$$N_i = \begin{cases} 1 \;\; para \;\; (x,y) = (x_i, y_i) \\ 0 \;\; para \;\; (x,y) = (x_j, y_j); i \neq j \end{cases} \tag{2.4}$$

Assim, para determinar os coeficientes dos polinómios de interpolação, pode-se estabelecer o seguinte sistema de equações,por exemplo para N_1 (o Nó 1)

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a + bx_1 + cy_1 + dx_1y_1 \\ a + bx_2 + cy_2 + dx_2y_2 \\ a + bx_3 + cy_3 + dx_3y_3 \\ a + bx_4 + cy_4 + dx_4y_4 \end{bmatrix}$$

ou

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 + x_1 + y_1 + x_1y_1 \\ 1 + x_2 + y_2 + x_2y_2 \\ 1 + x_3 + y_3 + x_3y_3 \\ 1 + x_4 + y_4 + x_4y_4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{2.5}$$

o que permite calcular os parâmetros de cada função através de:

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1 + x_1 + y_1 + x_1y_1 \\ 1 + x_2 + y_2 + x_2y_2 \\ 1 + x_3 + y_3 + x_3y_3 \\ 1 + x_4 + y_4 + x_4y_4 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.6}$$

Este cálculo é feito para cada ponto do polígono, permitindo assim estabelecer as funções N_1, N_2, N_3 e N_4, necessárias para a interpolação das coordenadas

2.6 Calculo de Forças

Para o calculo da aceleração com base no deslocamento observado recorreu-se a definições de física. Estas definições descritas na presente secção foram consultadas no manual de mecânica [30].

A velocidade instantânea de um objecto é a derivada da sua posição em relação ao tempo, ou seja,

$$v = \frac{dx}{dt} \tag{2.7}$$

> "A aceleração instantânea é o valor-limite da aceleração média, quando o intervalo de tempo Δt se torna muito pequeno. Assim,"

$$a = \lim_{\Delta t \to 0} a_{med} = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t}$$

ou seja, temos que a aceleração corresponde à derivada da velocidade em relação ao tempo:

$$a = \frac{dv}{dt} \tag{2.8}$$

Pela combinação das equações (2.7) e (2.8) é possível obtermos a aceleração a partir do deslocamento. Isto é,

$$a = \frac{dv}{dt} = \frac{d}{dt}\left(\frac{dx}{dt}\right)$$

ou simplificando,

$$a = \frac{d^2v}{dt^2}. \tag{2.9}$$

Tendo a massa da embarcação e a aceleração podemos calcular a força do impacto recorrendo à segunda Lei de Newton [30].

A definição de quantidade de movimento pode ser traduzida matematicamente pela equação

$$F = \frac{d(mv)}{dt}, \tag{2.10}$$

se a massa for constante temos

$$F = m\frac{dv}{dt} ou F = ma. \tag{2.11}$$

Descrevendo a equação 2.11 em palavras, temos:

> Se a massa é constante, a força é igual ao produto da massa pela aceleração

Recorrendo as equações 2.9 e 2.11 obteremos o valor da força do embate da embarcação com o cais durante a atracagem a partir do seu deslocamento.

2.7 Trabalho Relacionado

Para o desenvolvimento do protótipo experimental descrito nesta dissertação foi necessário proceder-se à investigação de outros estudos realizados fora do LNEC que se aplicassem a imagens de uma embarcação a acostar a um cais num porto marítimo. Estes estudos dividem-se em duas categorias. Os estudos directos baseiam-se em tecnologias de tracking e conversão de coordenadas recorrendo a métodos aritméticos simples. Os estudos baseados em aprendizagema automática, tal como o seu nome indica recorrem a aprendizagem automática para detectar um objecto, no presente caso uma embarcação, e seguir o posicionamento deste ao longo de uma sequência de vídeo.

2.7.1 Estudos Directos

Foram analisadas formas de conversão de coordenadas de um referencial 2D, uma imagem, em coordenadas de um referencial 3D, coordenadas reais. Nomeadamente, no caso de uma câmara acima da cena alvo [45]. Com esta abordagem a partir da imagem de uma única câmara é possível obter a localização de um objecto na área de trabalho de um robot uma vez que se sabe no plano as suas coordenadas e a altura do objecto e da câmara são constantes. Tornando-se desta forma um problema a duas dimensões.

Para a detecção de embarcações foi analisado um projecto de detecção de embarcações através de imagens satélite [13], este identifica embarcações em imagens satélite com base num dataset de imagens satélite [14]. Com a utilização de um dataset correspondente a imagens de embarcações de uma perspectiva semelhante às presentes na nossa situação [15], ou seja, nas quais se veja as laterais, parte frontal e traseira, existe a possibilidade de se poder utilizar este projecto para a detecção de embarcações na nossa situação. Este método recorre a Histogram of Oriented Gradients (HOG) como classificador para Support Vector Machines (SVM) para identificar a presença de um navio na imagem.

2.7.2 Aprendizagem automática

Para além dos métodos mais directos já apresentados foram também analisados alguns trabalhos utilizando métodos de aprendizagem automática e, concretamente, redes neuronais para a detecção de objectos na imagem.

Uma das possibilidades no OpenCV é a utilização de ferramentas de aprendizagem automática para a detecção de um objecto à nossa escolha é o Cascade Detection. Fornecendo-lhe imagens exemplo será possível este identificar apenas esse objecto no vídeo fornecido. Este treino poderá ser feito com base na detecção de características Haar [37]. O calculo das caracteristicas Haar consiste na verificação de regiões rectangulares adjacentes numa zona especifica de uma janela de detecção, somar as intensidades dos pixeis de cada região e calcular a diferença entre estas somas. Guardando estas caracteristicas num ficheiro xml que se poderá usar para a rápida identificação destas características no vídeo. Caso o conjunto de treino não seja amplo o suficiente será possível que este não

identifique uma embarcação em cena quando esta se encontra em cena, impedindo assim a avaliação do seu movimento.

Para a detecção da embarcação verificou-se que uma implementação com base em correspondências feitas directamente entre os frames sem se regenerar as *mask features* de cada objecto se obtém um algoritmo de identificação de objectos bastante mais eficaz e fiável [32]. Em Space Time Memory (STM) é criada um banco de memória específico para a afinidade de cada objecto no vídeo. Já com o proposto Space Time Correspondence Network (STCN) é criada apenas uma matriz de afinidade usando relações STCN. Esta diferença torna o processo mais eficiente uma vez que apenas é criada apenas uma matriz de afinidade para o conjunto de imagens ao invés de uma para cada imagem e também mais robusto já que o modelo é forçado a aprender as relações com todos os objectos, não só os identificados.

O artigo [41] utiliza redes neuronais para detectar em tempo real embarcações em espaços confinados. Esta utilização assemelha-se às nossas condições na doca do porto de carga. O trabalho em causa utilizou um *dataset* já existente na internet (Microsoft COCO) para treinar os modelos. Este dataset possui diversos elementos para treino, mas não será o mais adequado para a nossa aplicação uma vez que as embarcações que este identifica são maioritariamente de pequeno porte, sendo que a nossa aplicação se focará mais em embarcações de grande porte.

A eficiência computacional de vários algoritmos de inteligência artificial também é relevante em aplicações de análise de imagem [31]. Por exemplo, relativamente a algoritmos clássicos de detecção de objectos, algoritmos como o You Only Look Once (YOLO) apresentam uma eficiência em termos de tempo superior, mas algoritmos como Faster-Region-based Convolutional Neural Networks (RCNN) e Single Shot Detector (SSD) têm uma precisão superior, sendo portanto os melhores algoritmos que poderíamos usar na nossa aplicação uma vez que não necessitamos de valores em tempo real, mas pretendemos os valores de velocidade mais precisos possível.

Foi também analisada a possibilidade de se utilizar uma outra abordagem usando duas redes neuronais [38]. Uma identificaria o tipo de objecto, o seu tamanho e a sua posição (YOLOv2). A outra, FlowNet, identificaria o fluxo óptico da imagem. Relacionando o tamanho da embarcação com o fluxo permitiria-nos obter a velocidade da embarcação. No trabalho relacionado desta dissertação temos vários artigos que complementam a nossa implementação recorrendo a processamento de imagem.

Como primeira abordagem teremos por base métodos mais directos e consolidados. Visto serem métodos vastamente utilizados permite-nos ter um sistema de análise inicial com relativa precisão apesar de serem métodos mais simples. Métodos mais complexos foram apenas considerados como alternativa, potencialmente a serem utilizados em trabalho futuro como potencial melhoria da proposta actual.

2.7.3 Análise de *tracking*

O OpenCV tem vários algoritmos de tracking de objectos. A performance de alguns destes algoritmos é avaliada no artigo [36]. Dos algoritmos avaliados neste, Speeded up robust features (SURF) e Scale-invariant feature transform (SIFT) são patenteados e portanto é necessário o pagamento de uma taxa pela sua utilização. Sendo Oriented FAST and Rotated BRIEF (ORB) uma alternativa a ambos apesar da necessidade de pagamento de subscrição e este apresenta uma performance bastante próxima, se não melhor que estes dois. Não foram encontradas formas de implementação deste algoritmo para *tracking* da embarcação apenas foram encontradas implementações para *feature matching*, correspondência entre características do conteúdo da imagem, a função principal deste algoritmo. No entanto para outros algoritmos de *tracking* presentes no OpenCV foram encontrados alguns tutoriais [16] [17] onde também estava presente uma breve descrição destes algoritmos e mais importante, algumas métricas relativas de performance. Pela informação dada por este o Channel and Spatial Reliability Tracker (CSRT) será o melhor algoritmo para esta aplicação. Uma vez que apesar de este ser ligeiramente mais lento que os restantes tem maior precisão, o que para a aplicação apresentada é a métrica mais importante.

2.7.4 Estimação de profundidade

Uma vez que pretendemos obter a força de impacto da embarcação a partir de imagens video é necessário encontrar uma forma da converter as unidades do vídeo para unidades de medida reais. Para a obtenção do tempo real decorrido entre cada momento de captura pode-se retirar do ficheiro de vídeo o *framerate* a que este foi gravado. Tendo o *framerate* e sabendo o intervalo de frames analisado é possível saber-se o tempo decorrido aproximadamente. Para as unidades de distância é necessário converter as coordenadas da imagem em distâncias reais.

Foram encontradas diversas outras maneiras de estimar a profundidade dos objectos num plano de imagem [44]. Uma das formas de estimar a profundidade e de tal forma obter as distâncias seria com base em estereo fotométrico. Esta técnica baseia-se em com base em várias fotos do mesmo objecto na mesma posição, mas com iluminação de diferentes ângulos. Uma vez que ao longo do dia o sol muda de posição e cria várias sombras diferentes. Estas diferentes sombras permitem estimar a forma 3D dos objectos poderia ser estimada a profundidade usando esta técnica.

Outra forma de estimar a profundidade seria usar sensores de distância que recorrem à medição do "tempo de voo". Este permitiria estimar a profundidade a que uma embarcação se encontra recorrendo ao tempo que o sinal emitido por estes sensores demora a regressar depois de reflectir num objecto. Uma vez que a velocidade aproximada a que este viaja é conhecida é possível com simples calculos obter-se a distância. Tendo em conta que o objectivo desta tese é utilizar as ferramentas já existentes num porto esta técnica não poderia ser utilizada uma vez que seria necessário instalar estes sensores. No entanto poderá ser uma solução a avaliar em trabalho futuro.

Poderá também ser possível obtermos a estimação de profundidade caso existam mais do que uma câmara a captar a embarcação a atracar. Uma das maneiras será através da correlação entre pixeis das imagens provenientes de duas câmaras [34]. Com esta técnica ao termos a coorelação entre 5 pontos é possível termos uma representação projectiva que será relativa aos 5 pontos de coorelação que definimos.

Poderá também utilizar-se as dimensões da embarcação o que nos permitirá usar as fórmulas de Pumrin e Dailey [40] para obter as medidas reais de deslocamento da embarcação. Estas permitem a passagem das medidas no ecrã, que serão obtidas graças ao método de background subtraction, em medidas do mundo físico real. Utilizando estas fórmulas [40] será possível obter uma estimação da velocidade pelo movimento dos barcos usando as suas medidas como escala na passagem de dimensões reais para dimensões gráficas. Permitindo ainda facilmente escalar caso o movimento seja no sentido da câmara. A desvantagem em relação a uma escala com base em pontos fixos será que não será possível uma configuração inicial que funcionará para todas as embarcações. Para o resultado final esperado não teria grande impacto uma vez que será necessário incluir para cada embarcação o seu afundamento de forma a estimar a massa e assim obter o impacto que esta teve com o cais. Por outro lado não permite a obtenção apenas da velocidade antes do impacto para as embarcações sem estes dados.

Caso existam mais câmaras a apontar para a mesma cena de ângulos diferentes e utilizássemos C++ poderíamos utilizar a funcionalidade presente nas contribuições do OpenCV [18], o mapping3D. Este permite estimar a localização a três dimensões de um ponto recorrendo a uma imagem 3D. Recorrendo a multiplas câmaras permite também estimar a velocidade de um objecto. A ferramenta mapping3D baseia-se num artigo de Kutluyıl Dogançay e Reza Arablouei [33] onde se defende a utilização de Selective Angle Measurements (SAM) na estimativa de ângulos de chegada na analise de movimento de um alvo (TMA). Recorrendo a um Target Motion Analysis (TMA) apenas com ângulos pretende-se estimar a posição, velocidade e aceleração de um alvo recorrendo ao seu azimute e ângulo de elevação. Este prova-se mais eficaz que outros métodos anteriormente utilizados graças à compensação do seu estimador permitindo melhores resultados na presença de ruído.

2.7.5 OCR

As embarcações possuem no casco identificadores únicos, como se tratasse de uma matrícula automóvel. Se pudermos reconhecer este identificador podemos automatizar a parte de converter a altura do calado na massa da embarcação. O OpenCV disponibiliza duas ferramentas de Optical Character Recognition (OCR), o Tesseract [42] e o MMDecoder [29] que permitem o reconhecimento de caracteres numa imagem fornecida. A eficácia do algoritmo de OCR também foi considerada [46] bem como possíveis melhorias na sua implementação de forma a maximizar a sua eficácia.

Este não será usado nesta fase, mas poderá ser útil no futuro para determinar o identificador da embarcação.

3

Arquitectura

Neste capítulo aborda-se a arquitetura da solução desenvolvida no âmbito desta dissertação. Desenvolveu-se uma aplicação-protótipo experimental que foi designada por Vessel Impact Detection. Abaixo encontram-se dois diagramas, um deles ilustra a arquitectura da implementação. A implementação em python irá a partir de um ficheiro de vídeo e ficheiros de configuração devolver o resultado do cálculo da força de embate de uma embarcação no cais de um porto e o outro a relação entre os diferentes componentes da implementação.

3.1 Diagrama de Arquitectura

O diagrama de arquitetura presente na figura 3.1 apresenta o funcionamento do protótipo experimental aqui descrito.

Na figura 3.1 o sistema inicia o processamento recebendo um ficheiro de vídeo. Na notação usada este ficheiro é representado pelo icone de uma folha. De seguida segue-se uma decisão de utilizar ou gerar os ficheiros de calibração. Na figura 3.1 esta decisão é representada pelo losango. Os pictogramas num quadrado tracejado representam um conjunto de imagens, os primeiros serão utilizadas para gerar os ficheiros de calibração caso estes não existam e os segundos serão os frames do vídeo após sofrerem calibração. Na descrição do restante funcionamento do sistema as setas representam a direcção do funcionamento e os rectângulos arredondados representam as diferentes tarefas descritas nesta dissertação. Assim, o algoritmo completo presente no diagrama de arquitectura da figura 3.1 consiste nas seguintes etapas:

1. Escolher um ficheiro de vídeo que contenha o embate que se pretende analisar.

2. Inserir ficheiros de imagem com um xadrez 6 por 9 para criar os ficheiros de calibração para anular o efeito de distorção criado pela lente da câmara que gravou o vídeo. Ou inserir os ficheiros de calibração na pasta correcta.

3. Criar ficheiro de vídeo a partir do conjunto de frames calibrados gerado.

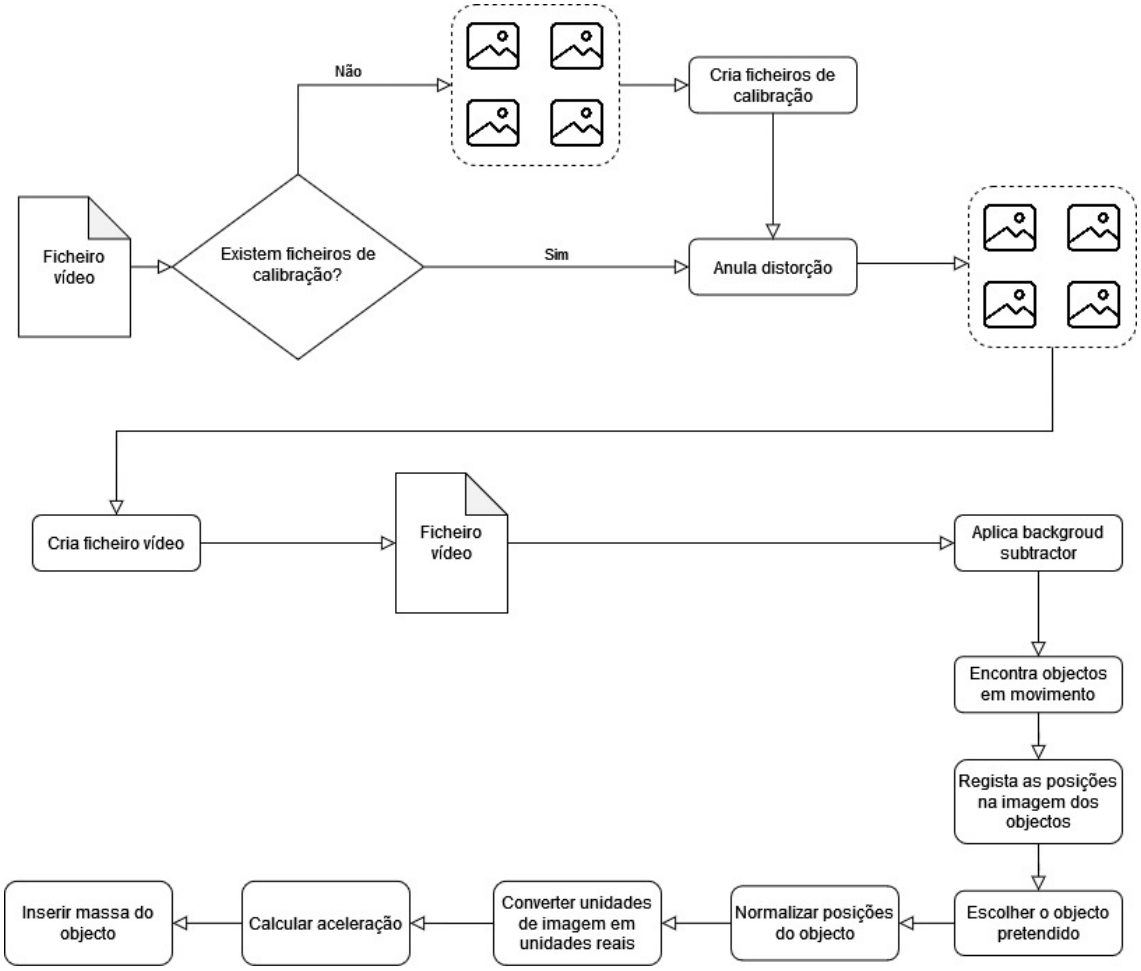

Figura 3.1: Diagrama de funcionamento

4. Inserir o ficheiro de calibração do background após calibração usando o modo de testes. Aplica o background subtractor ao vídeo sem distorção de lente.
5. Encontra os objectos na imagem gerada pelo background subtractor e dá-lhe um id.
6. Regista num ficheiro .csv as posições na imagem dos objectos, bem como o frame em que acontece cada posição e o seu id.
7. Escolher o objecto que se pretende calcular a força de impacto.
8. Normalizar as posições do objecto na imagem de forma a minimizar erros provenientes do tracking.
9. Converter as posições na imagem para posições num referencial real.
10. Calcular a aceleração do objecto.
11. Inserir a massa do objecto.

12. Obter a força de impacto no cais.

3.2 Diagrama de Classes

O diagrama de classes da figura 3.2 apresenta as funções e relações entre as diversas classes utilizadas na nossa implementação. Utiliza uma forma simplificada da notação UML onde, para facilitar a leitura, se omitiu a descrição das relações.

O protótipo experimental desenvolvido no âmbito desta dissertação foi elaborado em Python e de maneira a possibilitar a escolha entre duas formas de utilização. Servidor e terminal, apesar de apenas o modo terminal estar funcional. No entanto foi construído de forma modular de forma a permitir a criação das vertentes servidor e terminal com maior facilidade. Ambas as vertentes necessitam no entanto que antes seja executado o mini-programa *create_calibrated_video* que irá gerar o vídeo sem distorção da imagem provocada pela lente da câmara utilizada. O mini-programa *create_calibrated_video* recorre à classe *Camera Calibration* para criar os ficheiros de calibração, pelo método *calibrate* e anular a distorção da imagem provocada pela lente através do método *undistort*. Após ser gerado o vídeo sem distorção será também sempre necessário uma primeira utilização pelo terminal de forma a configurar-se correctamente o *background subtractor* de maneira a que detecte centroides correctamente. Após esta configuração ambas as versões poderão ser usadas para visualizar o backgroundsubtractor a ser gerado no método *subtractor* e os centroides detectados pela Classe *CentroidTracker* através do método *update*. Este irá avaliar os centroides e registá-los como novos (método *register*) ou removê-los (método *deregister*) caso o objecto tenha saído de cena ou então adicionar a sua localização à lista de posições do centroide. De seguida irão interpolar os centroides com o id seleccionado recorrendo ao método execute da classe *InterpolateCentroids*. Após isto irá converter as unidades de imagem em unidades reais recorrendo ao método execute da classe *ConvertUnits* que recorre ao método execute da classe *Transforma* que implementa as equações de forma mencionadas na secção 2.5. Por fim recorre ao método *execute* da classe *CalculatePhysics*, que com as funções *calc_fist_derivative* e *calc_second_derivative* para calcular e mostrar a força do embate da embarcação no cais.

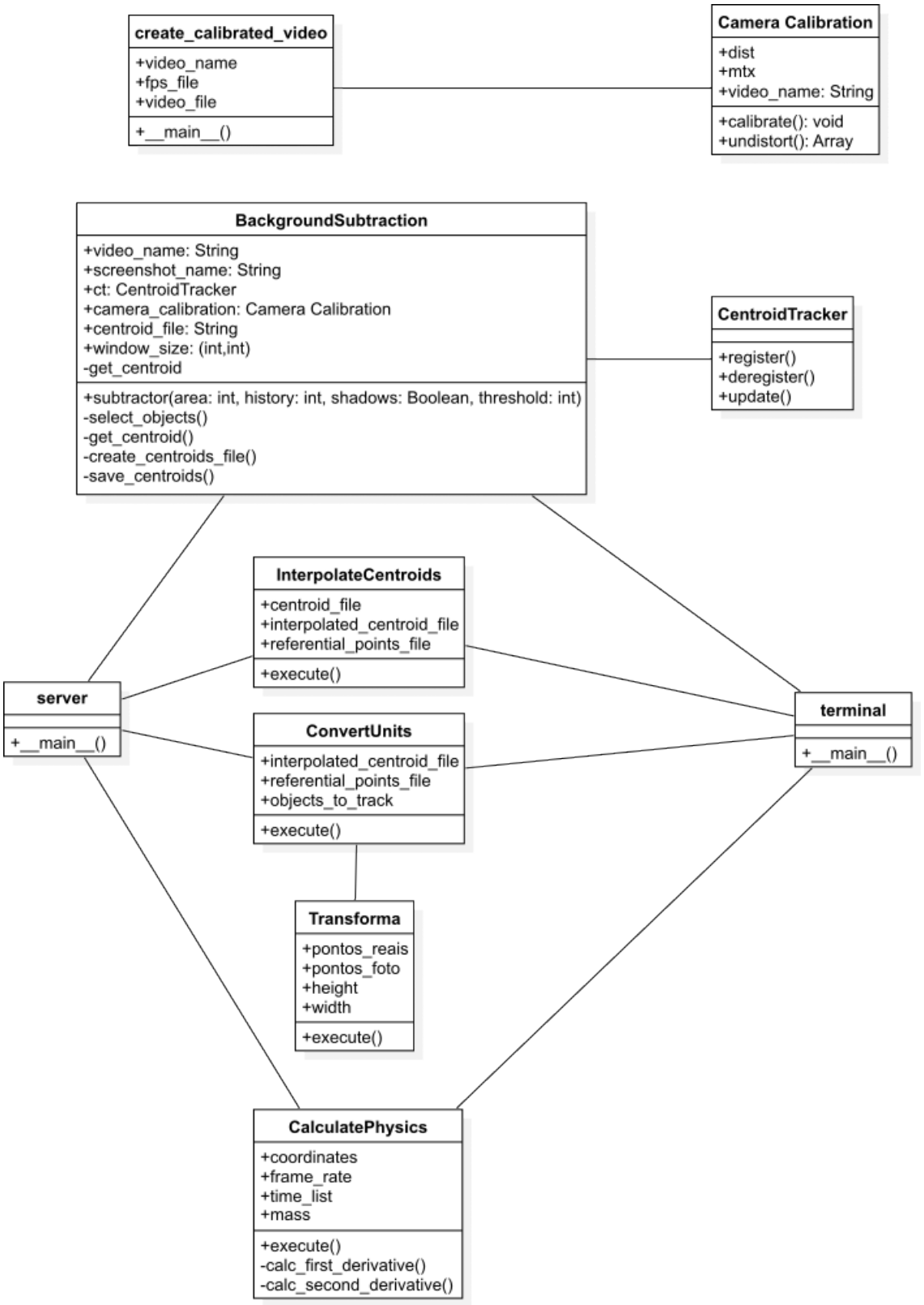

Figura 3.2: Diagrama de Classes

4

Protótipo Experimental

Para a realização de testes iniciais ao protótipo desenvolvido no âmbito da dissertação foi montado um modelo constituído por um dispositivo experimental para simular à escala a atracagem de uma embarcação no cais do porto de Leixões. O modelo foi montado nos tanques hidráulicos do Núcleo de Portos e Estruturas Marítimas do LNEC. Existiram duas versões deste modelo, sendo que a primeira consistia num cais e numa embaracação à escala, ao qual foram adicionados sensores de medição de aceleração na embarcação (um na proa e outro na popa), que permitiram obter os dados reais de aceleração desta embaracação e um sensor de medição de aceleração acoplado ao cais, que permitia obter as acelerações provocadas no cais pelo embate. Ao mesmo tempo também gravámos a colisão com uma câmara a partir da qual obtivémos o ficheiro de vídeo que é processado pela nossa aplicação protótipo. Ao processar o vídeo e compararmos os valores obtidos em relação aos valores reais foi possível sabermos a eficácia da ferramenta bem como a existência de problemas com o modelo. Foram também colocados sensores no cais modelo (sensores de força) que permitiram monitorizar os valores reais de impacto. Com a colocação de duas câmaras de vídeo em diversas posições é possível obtermos várias configurações possíveis de câmaras de vigilância de um porto. Câmaras de acção usarão a definição de lente que deforme menos o ambiente, tornando assim mais fácil a transformação que é feita posteriormente pelo software.

Os sensores de medição de aceleração desenvolvidos para a recolha de dados consistem num microcontrolador ESP32 que processa os dados recolhidos e registados nos outros dois componentes, um IMU que contém um acelerómetro e uma placa de connexão a cartões SD. Uma descrição mais aprofundada deste sistema de sensores está disponível na secção 4.1.

Existiram problemas na recolha de dados devido a falhas no registo de dados no cartão, bem como falta de bateria nos ultimos testes. Em testes anteriores verificou-se que o código inicial não recolhia os dados a alto ritmo verificado nos testes pós-programação. Este problema deveu-se a uma fraca conexão Wi-Fi, que se utilizava para comunicar com um servidor Network Time Protocol (NTP) de forma a obter a hora de cada conjunto de valores recolhido, bem como para comunicação com um servidor local com o objectivo de

se observar em tempo real que a recolha se encontrava a realizar com sucesso.

Devido a este problema foram feitas diversas alterações de forma a aumentar a rapidez de recolha de dados ao máximo possível. A comunicação com o servidor NTP passou a ser feita apenas uma vez, registando nessa altura no cartão SD o valor devolvido por este e o tempo actual do relógio interno do ESP32 (milisegundos desde que este se encontra ligado). Em cada leitura ficará apenas registado o tempo actual interno, mas com base neste registo inicial é possível chegar à data actual de cada um dos registos.

A comunicação com o servidor de diagnóstico foi também reduzida ao mínimo, ocorrendo apenas quando se detecta uma falha na gravação dos valores no ficheiro de dados presente no cartão SD.

Desta forma obteve-se uma versão com maior fluxo de dados enquanto se manteve a possibilidade de diagnosticar ocorrências de erros. Apesar disto ocorreram alguns testes em que não houve registo de qualquer informação pelo acelerómetro, mas também não foi recebido nenhum erro pelo servidor de diagnóstico. Uma vez que no final de cada teste se tinha de recolher os dados presentes no cartão de forma a gerar um novo ficheiro para este teste, foi possível verificar a falha no registo do ficheiro e o teste foi repetido novamente após uma reinstalação do programa de recolha de forma a garantir o seu funcionamento ou ligação à power bank quando se verificou que um dos sensores não tinha sido conectado durante esse teste.

No segundo modelo, que é utilizado ao longo desta dissertação, efectuámos algumas mudanças em relação ao modelo descrito anteriomente. Este modelo encontra-se na figura 4.2e também em mais detalhe no anexo VI e no anexo V. As câmaras foram colocadas a um nível mais baixo, de forma a que à escala se assemelhassem mais ao posicionamento das câmaras de vigilância. O cais tornou-se mais baixo em relação ao nível da água, também para melhorar a visibilidade da embarcação. Esta falta de visibilidade da embaracação impedia a correcta captura desta no momento do acostamento ao cais. Deixou de se utilizar um acelerómetro no cais, visto que a forma como estava implementado prejudicava a visibilidade da embarcação pelas câmaras, como se pode verificar na figura 4.1. De forma a termos testes que se possam repetir recorremos a um sistema de tracção para puxar a embarcação com uma corda com um peso, foi possível assim controlar a força com que movemos a embarcação. Como forma de melhorar a repetibilidade dos testes foi também fixada uma distância de onde a embarcação irá começar o seu movimento. Assim, para o mesmo percurso percorrido pelo peso e a mesma massa a força aplicada na embarcação será teoricamente constante.

Para ambos os modelos recorreu-se a uma embarcação modelo de 13.1Kg de peso seco, com a adição posterior de pesos e instrumentos de medição.

Figura 4.1: Simulação à escala utilizada em testes anteriores

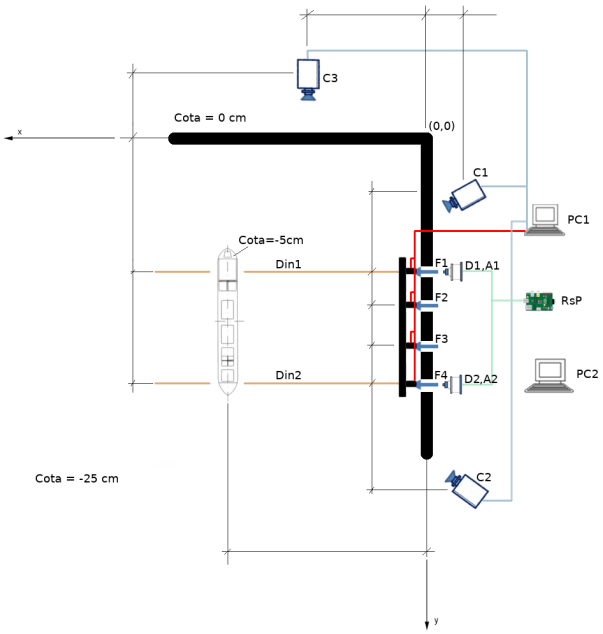

Figura 4.2: Esquema de montagem do modelo utilizado na experiência

4.1 Medição de aceleração

Para a medição da aceleração real da embarcação foi desenvolvido uma ferramenta de recolha de dados com base num microcontrolador ESP32. Este consiste numa unidade de processamento ESP32 ligada a um IMU Grove IMU 9DOF, que contém um acelerómetro, e a um leitor de cartões SD. Para a ligação do ESP32 ao IMU recorreu-se à biblioteca MPU9250 [19] uma vez que pela informação disponível acerca deste módulo este se baseia no IMU MPU9250[20]. Esta biblioteca foi escolhida ao invés de outras uma vez que esta simplicou a implementação da funcionalidade, tendo já um exemplo de calibração muito simples de implementar. A calibração é guardada num ficheiro de configuração gravado no cartão SD[21], o que permite a simplificação do processo de obtenção de dados e eliminar a necessidade de efectuar a calibração a cada recolha de dados. Assim, ao voltar a ligar o controlador este irá ler o ficheiro de configuração e efectuar a calibração com os valores calculados anteriormente[22], podendo assim começar a leitura dos dados imediatamente. De forma a se poder identificar temporalmente quando ocorreu cada uma das entradas no ficheiro de dados foi implementado um cliente NTP[23] de forma a obter a data e hora no momento. Esta biblioteca de implementação de cliente NTP foi escolhida em oposição a outras disponíveis devido à maior precisão dos valores que esta devolve. Outras bibliotecas apenas devolviam o tempo actual até ao segundo, mas uma vez que temos muitas recolhas por segundo foi necessário maior precisão, apresentando esta uma precisão próxima de 1 milisegundo. O código desenvolvido para esta recolha encontra-se disponível online, na conta GitHub do aluno[24]. As leituras de acelerações são escritas para um ficheiro csv gravado no cartão SD[21].

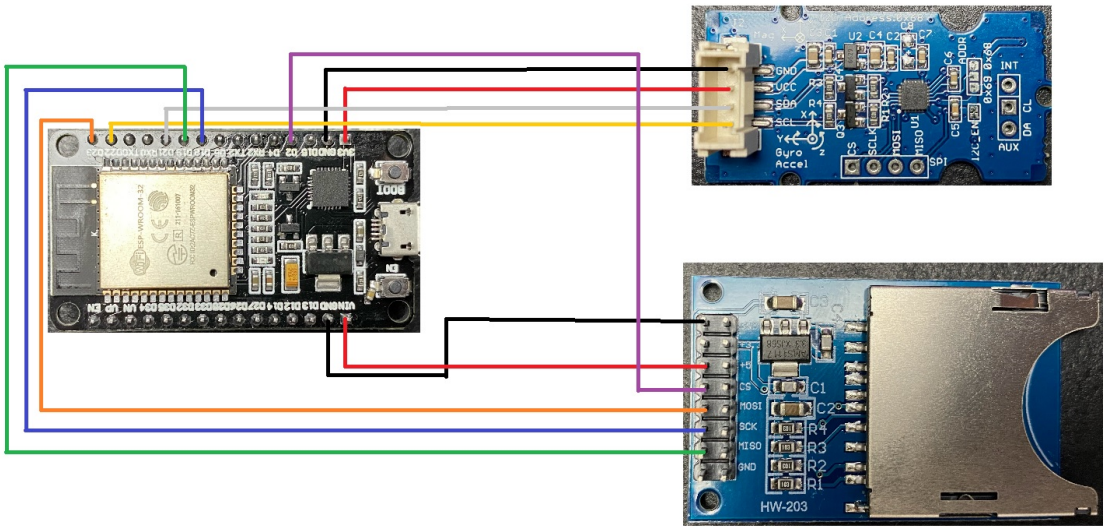

Figura 4.3: Diagrama de ligação do ESP32 aos módulos de acelerómetro e leitor de cartões SD

Para além deste mecanismo criado de raíz também se utilizou o sistema de câmaras

da OptiTrack. O sistema de câmaras da OptiTrack permite a aplicação de um alvo 3D, constituído por várias esferas reflectoras em posições pré-definidas entre elas, num objecto e após calibração do sistema permite efectuar o tracking do objecto com o alvo. O OptiTrack irá devolver as posições do objecto no espaço ao longo do tempo. É portanto uma aproximação ao que pretendemos utilizar, mas sem a complexidade de calibração e utilização de alvos que esta ferramenta implica. No entanto este terá valor na avaliação da precisão da nossa ferramenta.

4.2 Aplicação

4.2.1 Calibração

De forma a anular a deformação das imagens provocada pela lente das câmaras o protótipo experimental descrito nesta dissertação vem preparado para utilizar os coeficientes de distorção e a matriz intrinseca da câmara que tenham sido guardados no seu respectivo ficheiro de configuração ou, caso estes não existam, carregando várias imagens com um tabuleiro de xadrez para efectuar a calibração das imagens. Efectuando esta calibração reduziremos o efeito provocado pela lente utilizada pela câmara de filmar que produziu o vídeo.

Pegando no conjunto de imagens e percorrendo-o iremos utilizar a funcionalidade findChessboardCorners do OpenCV sobre a versão a preto e branco de cada uma das imagens. Este método devolve a lista de cantos encontrados num tabuleiro de xadrez, como se pode observar na figura 4.4. Neste protótipo foi escolhido um xadrez de 6 por 9 quadrados. Para aumentar a precisão das coordenadas dos cantos encontrados recorremos à funcionalidade cornerSubPix que pega na imagem original e nos cantos que encontrámos e procura pela melhor localização para o canto na vizinhança do canto que fornecemos. Com base nos pontos 3D e as suas localizações na imagem corremos o método calibrateCamera[47] do OpenCV. Este permite-nos obter a matriz intrinseca da câmara e os coeficientes de distorção da lente.

Para reverter a distorção da lente só temos de criar uma câmara optima virtual com o método *getOptimalNewCameraMatrix* com o tamanho da imagem a tratar e com a matriz e coeficientes obtidos anteriormente. Tendo esta nova câmara virtual podemos usá-la no método *undistort* do OpenCV de modo a obtermos a imagem sem a distorção da lente da câmara.

Verificou-se que o processo de *undistort* tinha grande impacto no desempenho do resto do protótipo aplicacional, como tal decidiu-se fazer uma separação. Passou a existir, desta forma, uma aplicação *create_calibrate_video* que analisa o vídeo e aplica o *undistort* a cada frame e guarda os frames numa pasta dedicada a este ficheiro de vídeo com todos os frames que foram lidos e transformados em formato imagem. Nesta pasta é também criado um ficheiro de texto que contém o frame rate do vídeo, uma vez que tal informação será necessária posteriormente e não é possível retirar de um conjunto de

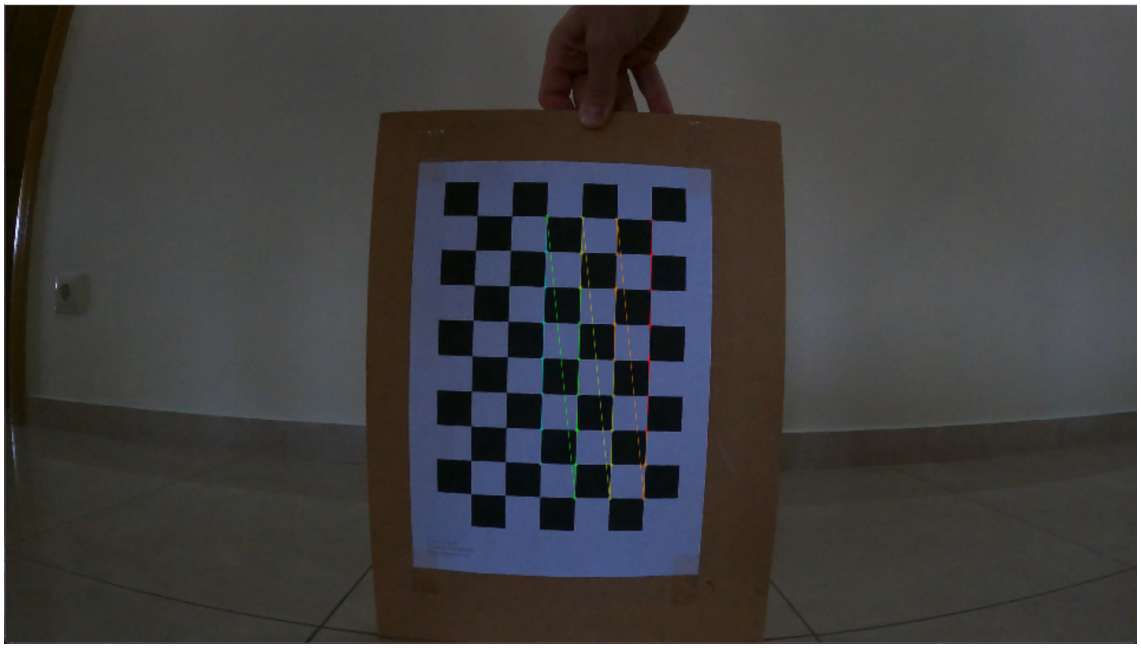

Figura 4.4: Imagem da execução do *findChessboardCorners*

imagens. A utilização deste método ao invés de ser feito directamente a leitura, remoção de distorção e restante algoritmo não tem grande impacto no tempo total da aplicação. Tem no entanto uma grande vantagem na análise do vídeo com a detecção do barco, bem como em situações em que seja necessário ler o vídeo diversas vezes.

Figura 4.5: Imagem retirada directamente da câmara

Como se pode observar na figura 4.5 a linha recta do topo do muro do tanque apresenta-se longe de parecer com uma recta na imagem obtida directamente da câmara.

Figura 4.6: Imagem após se anular a distorção da lente

Aplicando o undistort a esta imagem obtemos a figura 4.6 onde a recta do topo do muro já se assemelha muito mais a uma recta.

4.2.2 Background subtractor

Tal como descrito anteriormente utilizamos o BackgroundSubtractor KNN presente no OpenCV. Para melhorarmos os nossos resultados finais este irá correr com base no video com as imagens sem distorção criada pela lente da câmara pelo método descrito anteriormente.

Os parâmetros para a configuração do BackgroundSubtractor deverão ser introduzidos num ficheiro na pasta "config/"com o nome igual ao nome do ficheiro de vídeo sem a extensão deste seguido de -backgroundsubtractor.txt". Esta configuração é de extrema importância para a eficácia deste programa uma vez que esta permitirá detectar a embarcação e registar a sua posição na imagem, o que será posteriormente utilizado para os nossos calculos de obtenção da força de impacto com que a embarcação embateu no cais.

A estrutura interna do ficheiro deverá ser a seguinte:

History: {valor}
Detect Shadows: {valor (True/False)}
Distance To Threshold: {valor}
Object min area: {valor}

O campo History define a memória do BackgroundSubtractor, ou seja, ao fim de quanto tempo em frames um objecto anteriormente em movimento deve ficar imóvel para ser considerado parte do fundo da cena. O campo Detect Shadows indica se o Background Subtractor deverá considerar as sombras como objectos em movimento ou não. Nos testes efectuados nesta dissertação foi sempre utilizado o valor False. Distance to Threshold define a distância euclidiana entre um pixel e a amostra que decide se o pixel faz parte da amostra ou não. O campo Object min area permite definir o tamanho minimo do objecto para que este seja considerado para tracking, minimizando assim a existência e multiplos centroides devido a ruído no Background Subtractor. Na figura 4.7 podemos observar o resultado do BackgroundSubtractor após a sua configuração.

Figura 4.7: Resultado obtido pela aplicação do background subtractor

4.2.3 Centroid tracking

Com base na imagem criada pelo BackgroundSubtractor utilizaremos o método findContours do OpenCV. Com base nestes contornos iremos verificar se a área de cada um é maior ou menor que o valor que definimos anteriormente. Se for menor discartamos, permitindo assim eliminar objectos detectados no passo anterior, mas que não interessam para a nossa aplicação incluindo pequenos objectos detectados na nossa embarcação.

Também discartamos caso a hierarquia deste seja -1, ou seja, caso não tenha contornos pai ou filho.

Criamos um rectângulo delimitador do contorno (método boundingRect) e enviamos este para o método update do centroidTracker [11] este método verifica se o objecto já foi identificado anteriormente ou não. Caso tenha sido identificado anteriormente associa-o ao Object ID anterior, caso não tenha sido será dado um novo ID. Caso se verifique que um objecto desapareceu de cena este é removido da lista de objectos. Na figura 4.8 pode observar-se um exemplo de um centroid a ser reconhecido, de id 9.

A cada centroide encontrado é adicionado a um ficheiro de centroides para o respectivo vídeo de origem. Neste ficheiro será guardado o framerate do vídeo, o frame actual, o id do objecto, e as coordenadas x e y do objecto na imagem.

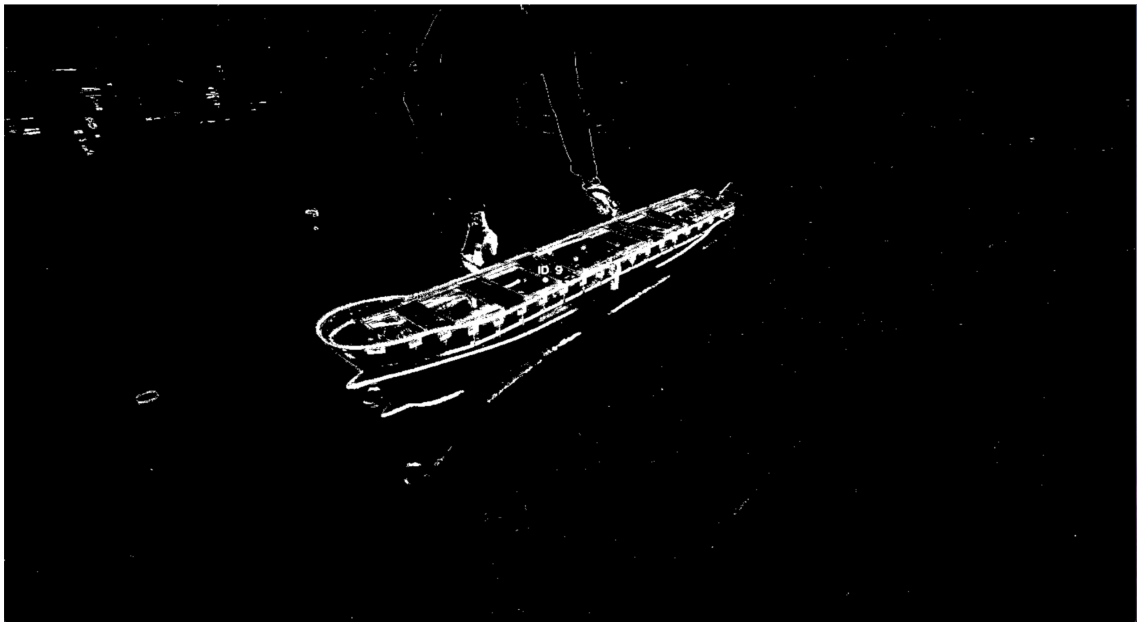

Figura 4.8: Exemplo de tracking de centroid

4.2.4 Centroid processing

No ficheiro de configuração é possível configurar duas rectas com o objectivo de limitar o nosso processamento a uma área mais restrita da nossa gravação, uma vez que se pretende apenas obter o valor da força provocada na colisão da embarcação com o cais portuário. Sendo assim ao configurarmos uma linha na margem do cais e outra um pouco mais para o interior do porto permitirá ignorar movimentos que não contribuem para o âmbito desta dissertação. Esta funcionalidade não foi no entanto utilizada no processamento das experiências realizadas.

4.2.5 Interpolação de centroídes

Por forma a reduzir desvios no percurso do objecto detectado causados por imprecisões no seguimento da embarcação iremos recorrer a interpolação Spline. Para tal irá ler-se o ficheiro de coordenadas gerado no processo anterior. Como este possuí várias entradas para o mesmo frame, uma para cada objecto, iremos agrupar os dados pelo id do objecto, sendo que em cada grupo de objectos este se encontra ordenado do frame inicial do objecto até ao final. Tendo esta ordenação efectuada é possível efectuar a interpolação para cada objecto individualmente. Esta irá receber as coordenadas x e y da posição do objecto na imagem, em duas dimensões, e irá devolve rum novo grupo de coordenadas x e y com redução do ruído referido anteriormente. Após se obter estes valores iremos novamente associá-los ao seu respectivo frame e gerar um ficheiro idêntico ao lido inicialmente, mas com as novas coordenadas. Para o calculo do Spline será utiliazada uma ferramenta externa[25] que já tinha provado as suas capacidades em outras utilizações no LNEC.

4.2.6 Transformação de coordenadas entre dois sistemas planares

Para se efectuar a transformação de coordenadas entre o sistema de coordenadas da imagem e o sistema de coordenadas real utilizam-se as equações descritas na secção 2.5 que se encontram implementadas como descrito no anexo IV.

4.2.7 Calculo de aceleração e forças

Para o cálculo das acelerações sofridas recorreremos às equações referidas em 2.6. Para implementar a segunda derivada recorre-se à operação derivative(n=2) do UnivariateSpline da biblioteca scipy.interpolate.

Uma vez que a unidade temporal que recolhemos é o número do frame é também necessário converter este numa unidade de tempo mais propriada para a situação, neste caso o metro. Sabendo qual o frame actual e o frame rate do vídeo é possível obtermos esta conversão é relativamente directo. Uma vez que o framerate é a quantidade de frames por cada segundo de vídeo temos:

$$framerate = \frac{\Delta frames}{\Delta t}$$

isto permite-nos então ter que:

$$framerate = \frac{\Delta frames}{\Delta t} \tag{4.1}$$

4.3 Utilização da Aplicação

Para a utilização completa deste protótipo experimental é necessário utilizar-se diversas ferramentas presentes no repositório GitHub[26].

A primeira ferramenta a utilizar-se chama-se create_calibrated_video.py. Ao lançarmos este será-nos pedido para inserirmos o nome do vídeo que pretendemos converter.

```
Video file name: <nome do vídeo>
```

Este nome deverá corresponder a um vídeo que tenha sido colocado na pasta video_files presente na raíz do projecto. Apesar do nome desta ferramenta não é criado um ficheiro de vídeo corrigido a partir do ficheiro vídeo original, criando no entanto um conjunto de imagens .png que poderão ser convertidas em vídeo. Estas imagens são criadas anulando a distorção da lente recorrendo às matrizes criadas anteriormente ou que serão criadas com base nas fotografias com o xadrez referido anteriormente.

Para se criar o ficheiro video recorrendo às imagens criadas é necessário aceder à pasta /video_files/converted/⟨nome do vídeo⟩/ e executar o seguinte comando:

```
$ cat *.png | ffmpeg -f image2pipe -r ntsc  -i - -vcodec libx264 filename.mp4
```

Para se executar este comando é necessário ter-se o ffmpeg instalado. No comando deve-se substituir ntsc pela respectiva taxa de fotogramas do vídeo original caso esta não seja 29.97 (30/1.001 na realidade)

Após se executar este comando já se poderá executar a aplicação principal. Ao se iniciar esta aplicação será apresentado a seguinte pergunta:

```
Test (y/n)[default n]?
```

Deverá utilizar-se inicialmente o modo de teste para se configurar correctamente o background subtractor num ficheiro ⟨nome do vídeo⟩-backgroundsubtractor.txt na pasta config/ presente na raíz do projecto, ficando por exemplo:

```
History: 10
Detect Shadows: False
Distance To Threshold: 400
Object min area: 1200
```

Este modo de teste permite também a captura de um frame do vídeo que será necessário para a configuração da conversão de distâncias em pixeis para distâncias reais. Utilizando-se software de visualização/edição, como por exemplo o GIMP, verifica-se a localização dos cantos do quadrado de calibração e quais as coordenadas dos pixeis em que se encontram. Coloca-se as coordenadas dos cantos do quadrado de calibração, bem como a resolução da imagem no ficheiro ⟨nome do vídeo⟩-referential-points.txt na pasta config/ presente na raíz do projecto. Coloca-se também no ficheiro de configuração dos pontos referenciais as coordenadas reais do quadrado centrado, bem como os valores m e b das equações das rectas delimitantes da região de detecção. As rectas delimitantes podem ser calculadas utilizando a ferramenta equation_creation_tool.py inserindo as coordenadas

de dois pontos na imagem, bem como a altura desta imagem. É necessário a altura da imagem uma vez que as coordenadas dadas pelo GIMP têm a origem do referencial no canto superior esquerdo e as coordenadas do OpenCV têm a origem no canto inferior esquerdo, sendo portanto necessário colocar ambas no mesmo referencial. Ficará assim o ficheiro por exemplo:

```
Pontos reais:[[-6.9, -8.8], [6.9, -8.8], [6.9, 8.8],
[-6.9, 8.8]]
Resolucao altura: 934
Resolucao largura: 1469
Pontos foto: [[778, 572], [830, 538], [755, 493],
[703, 524]]
m limite: -0,55
m cais: -0,85
b limite: -837,5
b cais: -1269,1
```

Após esta configuração poderá-se prosseguir fora do modo de teste. A partir deste momento poderá continuar-se utilizando a versão terminal ou a versão web. Para a versão terminal será apresentado o seguinte mensagem

```
Video file name: <nome do vídeo>
```

Inserindo-se o nome do vídeo que se converteu permiritirá prosseguir com o processo. Caso não tenha sido criado um vídeo sem distorção recorrendo ao método explicitado anteriormente será apresentada a seguinte mensagem:

```
Video hasn't been converted yet
```

Caso o vídeo inserido tenha sido processado este começará a ser reproduzido numa janela, bem como uma versão deste vídeo com o background subtractor aplicado. Este permitirá identicar visualmente qual o objecto reconhecido pela aplicação que pretendemos escolher para o calculo da força de embate. Será apresentada a seguinte mensagem à qual deveremos responder com o objecto escolhido:

```
Object id to track:
```

Após enviarmos o id do objecto escolhido as coordenadas no tempo deste objecto serão agrupadas e normalizadas de forma a reduzir o erro provocado por imprecisões no tracking. Após tal este processamento será perguntado ao utilizador qual a massa da embarcação em quilogramas.

```
Vessel mass (in Kg):
```

Tal permitirá calcular a força de embate.Que será apresentanda de seguida.

O código e aspectos complementares desta aplicação poderá ser consultada na sua página de GitHub[26].

5

Avaliação Experimental

5.1 Simulação com modelo à escala

5.1.1 Descrição da experiência

Os ensaios realizados foram designados de acordo com as características. Na tabela 5.1 temos a forma como se faz o particionamento do nome do ensaio e se obtem as características variáveis. Para o ensaio da tabela temos que o primeiro grupo de números representa o a massa do peso, em gramas, agarrado aos fios que puxam a embarcação. O segundo conjunto de números representa a distância, em centimetros, do cais ao local de onde a embarcação partiu. O grupo de números seguinte corresponde à massa, em gramas, dos pesos colocados no interior da embarcação. O conjunto de letras PERI corresponde ao tipo de cais onde se realizou o ensaio, para os ensaios recolhidos para esta dissertação foi sempre PERI, o cais de plataformas com os sensores de força. Estava no entanto também montado um cais feito de blocos de cimento de dimensionamento semelhante. O conjunto de números seguinte representa o ângulo com que o navio inicia o seu movimento e continua até encostar ao cais, no entanto este valor não corresponde a um valor correcto. Não foi efectuada medição do ângulo antes do ínicio dos ensaios, mas sabe-se que a proa inicia o seu movimento a 84,6cm do cais e a popa a 85cm do cais. O último conjunto de números corresponde à altura, em centimetros, da plataforma colocada no caminho percorrido pelo peso com o objectivo de parar a aceleração da embarcação e esta ir desacelerando apenas com o atrito da água e o impacto contra o cais.

Tabela 5.1: Formato designação testes.

Designação do ensaio	**Peso no sistema de tração (g)**	**Distância do navio ao cais (cm)**	**Peso no navio (g)**
GIIP-50-92-450 -PERI-0-53	50	92	450

Plataforma (PERI ou blocks)	**Ângulo de atracagem do navio (º)**	**Altura do balde ao chão (cm)**
PERI	0	53

Foram realizados no total 17 testes (com aproximadamente 10 impactos em cada um), alguns com sucesso completo na recolha de dados e outros apenas parcialmente.

A carga mínima em todos os testes eram os dois acelerómetros e alvos visuais, OptiTrack e distanciómetro. A massa da embarcação modelo nestes era de 13.1238Kg da embarcação, 0.2Kg de cada um dos dois acelerómetros e 0.090Kg dos alvos, perfazendo um total de 13.614Kg Na tabela 5.2 encontram-se as características da embarcação esquematizada na figura 5.1. Nesta figura é também possível observar que a profundidade de água era de 13.8 cm e a altura do cais acima do nível desta era de 5cm.

Tabela 5.2: Dimensões da Embarcação de teste.

Dimensão	**Valor**
Peso vazio da embarcação	13123.8 g
Peso sistema sensorial	2*0.2 Kg
Peso total embarcação + acelerómetros	13523.8 g
Comprimento	172.5 cm
Largura	24.8 cm
Calado	9 cm
Altura total	14 cm

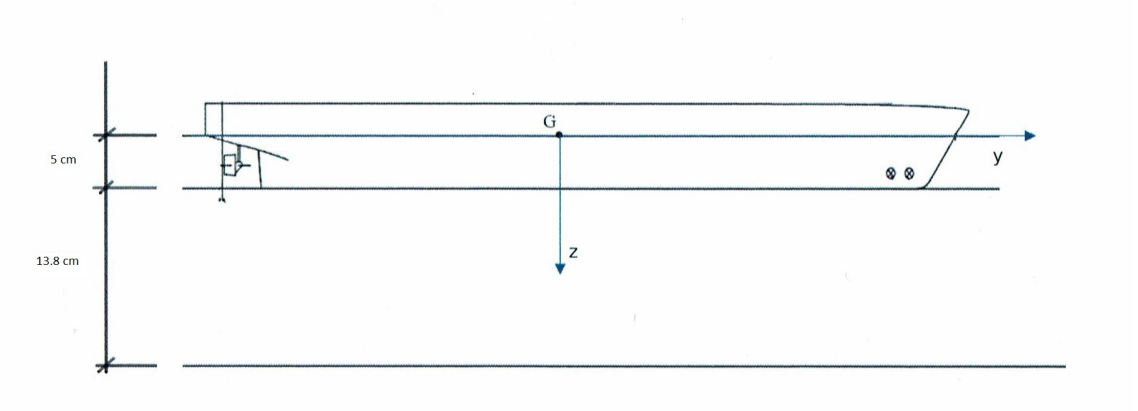

Figura 5.1: Diagrama da embarcação e eixos utilizados

Os ensaios foram efectuados com diferentes níveis de carga que se podem consultar na tabela na coluna carga (g)5.3. A coluna Nº corresponde à ordem pela qual os ensaios foram efectuados. Na coluna id ensaio tempos os identificadores com o formato descrito em 5.1, excepto quando correspondeu a testes não identificados. Na coluna repetições temos a quantidade de impactos com o cais a embarcação efectuou durante esse ensaio. Nas colunas cam 1 e cam 2 temos os nomes dos ficheiros vídeo gravados no decorrer do ensaio.

A totalidade dos dados recolhidos e elementos complementares de análise estão disponíveis para consulta numa pasta partilhada Google Drive.[27]

Tabela 5.3: Características de testes.

Nº	id ensaio	Repetições	Carga (g)	Nome vídeo	
				cam 1	cam 2
1	-	9	-	GX011521	PXL_20221125_114500422
2	-	10	-	GX011522	PXL_20221125_120009148
3	GIIP_50_92_450 _PERI_0_53	13	450	GX011525	PXL_20221125_142413374
4	GIIP_50_92_450 _PERI_0_63	11	450	GX011526	PXL_20221125_143906716
5	-	10	-	GX011527	PXL_20221125_145528449
6	GIIP_50_92_951 _PERI_0_53	11	951	GX011529	PXL_20221125_150948869
7	GIIP_50_92_951 _PERI_0_43	20	951	GX011530	PXL_20221125_152506357
8	-	10	-	GX011531	PXL_20221125_154307567
9	-	13	-	GX011533	PXL_20221125_161420887
10	GIIP_50_92_3200 _PERI_0_43	10	3200	GX011534	PXL_20221125_163502223

5.1.2 Resultado do teste

Nesta secção iremo-nos focar principalmente na análise do primeiro impacto realizado no âmbito do teste GIIP-50-92-450-PERI-0-53, existindo no entanto algumas excepções.

5.1.2.1 Calibração

Como se tinha verificado em testes preliminares que a calibração dos ficheiros de vídeo tinha um impacto muito grande na performance do programa, decidiu-se portanto quantificar este impacto após este ter sido separado da execução do background subtractor. Na tabela 5.4 podemos observar a duração da calibração da imagem dos vídeos em dois hardwares diferentes. Um é um CPU de computador portátil de 2013 (i7-4700HQ) e o outro um CPU desktop de 2019 (Ryzen 7 3700X), sendo a diferença de performance teórica do CPU mais recente muito superior à do mais antigo.

5.1.2.2 Acelerações Reais

Nos gráficos 5.2 e 5.3, captados pelo sistema OptiTrack, é possível observar-se por volta dos miliseguntos 40, 60, 105, 140, 170, 200, 230, 260, 280, 310, 340, 360 e 380 os 13 picos e vales correspondentes aos 13 impactos realizados no teste GIIP-50-92-450-PERI-0-53.

Dos acelerómetros colocados no modelo obteve-se valores precisos da aceleração da embarcação, nas figuras 5.4 e 5.5 podemos observar vários picos e vales na aceleração, correspondentes aos 13 impactos, bem como algum ruído. Os picos na aceleração verifica-se situar-se nas proximidades dos 2 m/s^2 e os vales apresentam valores entre os -1 e -2

Tabela 5.4: Duração da calibração.

Nome vídeo	Duração calibração(s)		Duração vídeo (s)
	Intel i7-4700HQ	AMD Ryzen 7 3700X	
GX011535_impacto1	171.64	160.56	17
GX011535_impacto2	174.25	164.17	17
GX011535_impacto3	161.06	150.33	16
GX011535_impacto4	166.94	156.28	16
GX011535_impacto5	156.19	147.32	15
GX011535_impacto6	156.33	147.67	15
GX011535_impacto7	167.76	155.76	16
GX011535_impacto8	175.30	164.20	17
GX011535_impacto9	156.89	147.96	15
GX011535_impacto10	145.57	153.90	15
Média dos 10 impactos	164.03	153.98	15,9

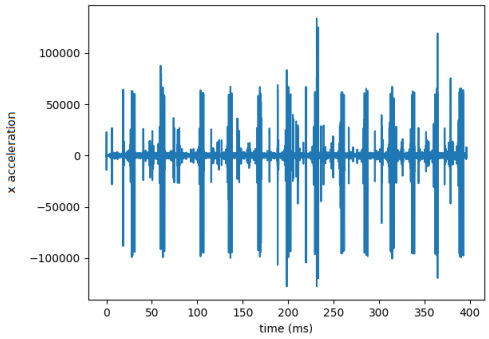

Figura 5.2: Aceleração no eixo dos x do Optitrack durante o teste GIIP-50-92-450-PERI-0-53.

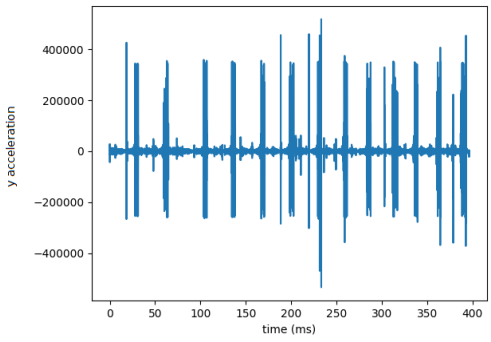

Figura 5.3: Aceleração no eixo dos y do Optitrack durante o teste GIIP-50-92-450-PERI-0-53.

m/s^2.

Nas figuras 5.6 e 5.7 é possível verificar que não se conseguem observar os 13 embates da embarcação com o cais.

5.1.2.3 Tracking

Como se pode verificar nas figuras 5.8 e 5.9 ocorre tracking de um ponto marcado a verde na embarcação no momento da atracação da embarcação. O ponto marcadosegue perfeitamente a embarcação sem efectuar saltos entre frames.

No entanto o tracking por vezes também é errático durante o movimento de aproximação da embarcação, como se pode verificar nas figuras 5.10 e 5.11, marcado a verde. Existe um momento em que o centroide fica parado numa localização e durante 2 frames este não se desloca, mas no frame seguinte o centroide volta para o local correcto no objecto o

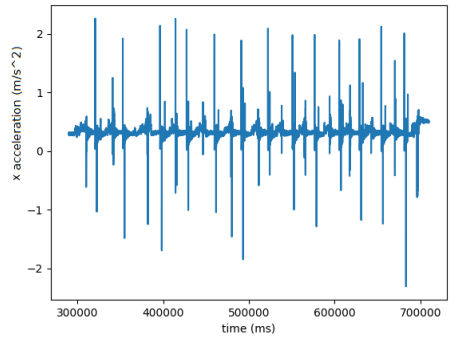

Figura 5.4: Aceleração no eixo dos x do acelerómetro 1 (proa) durante o teste GIIP-50-92-450-PERI-0-53.

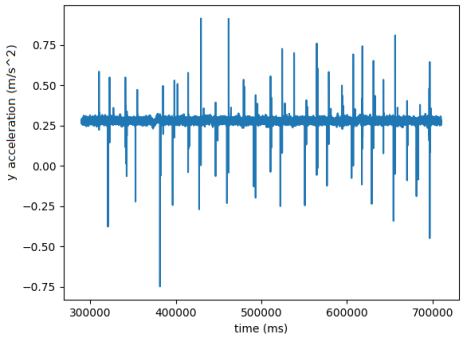

Figura 5.5: Aceleração no eixo dos y do acelerómetro 1 (proa) durante o teste GIIP-50-92-450-PERI-0-53.

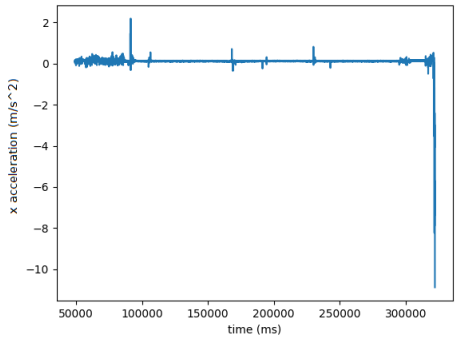

Figura 5.6: Aceleração no eixo dos x do acelerómetro 3 (popa) durante o teste GIIP-50-92-450-PERI-0-53.

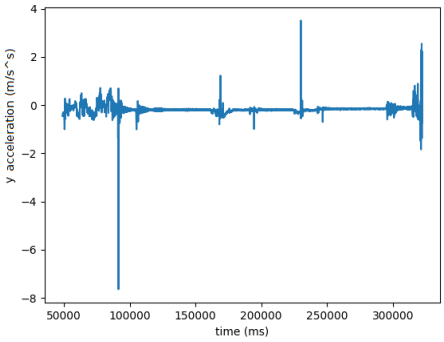

Figura 5.7: Aceleração no eixo dos y do acelerómetro 3 (popa) durante o teste GIIP-50-92-450-PERI-0-53.

que leva a que exista um saltogrande entre as posições do centroide.

5.1.2.4 Interpolação

Uma vez que, como se pode observar na figura 5.12, existem momentos em que o x (coordenada horizontal do pixel da imagem) da posição da embarcação no video corresponde a mais do que um valor de y(coordenada vertical do pixel da imagem), não se pode efectuar a interpolação directa. Como tal analizou-se a possibilidade de se fazer partições dos conjuntos de posições e inverter os referenciais. A complexidade de tal tarefa levou à solução utilizada. A utilização de duas interpolações separadas com os momentos de cada coordenada como o valor de x da interpolação e x e y como valor y da interpolação. Obtemos assim dois gráficos em função de t, um para x 5.14 e outro para y 5.16. Tendo esta separação é possível obtermos facilmente as duas interpolações para x 5.15 e outro para y 5.17. Voltando a unir os resultados de ambas ir levar a um gráfico 5.13 de coordenadas

Figura 5.8: Tracking correcto da embarcação no embate 1 do teste GIIP-50-92-450-PERI-0-53

Figura 5.9: Tracking correcto da embarcação no embate 1 do teste GIIP-50-92-450-PERI-0-53

Figura 5.10: Tracking incorrecto da embarcação no embate 1 do teste GIIP-50-92-450-PERI-0-53

Figura 5.11: Tracking incorrecto da embarcação no embate 1 do teste GIIP-50-92-450-PERI-0-53

interpolado.

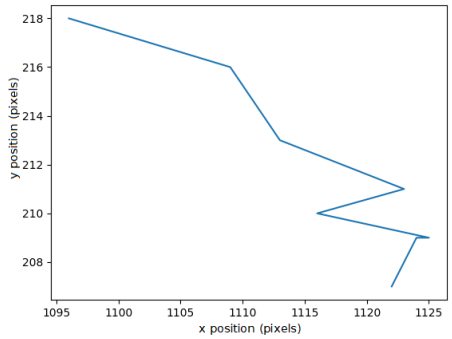

Figura 5.12: Gráfico de posições da embarcação

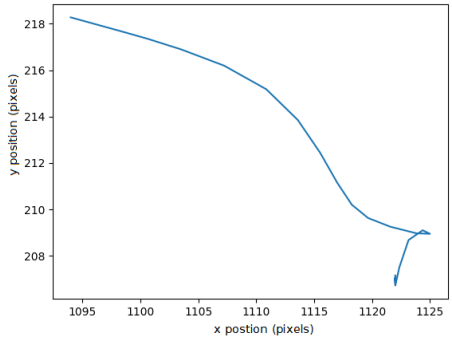

Figura 5.13: Gráfico de posições da embarcação interpolado

Verifica-se que o gráfico da figura 5.12 representa uma trajectória constituída por vários segmentos de recta, mas ao sofrer interpolação se torna uma linha mais curva, tal como se pode verificar em 5.13.

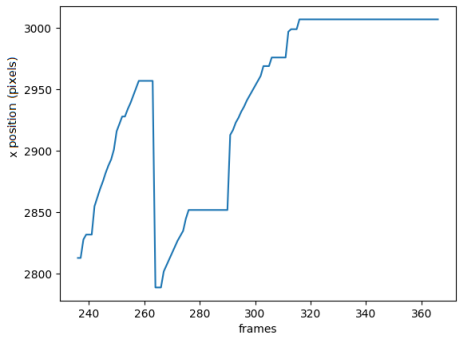

Figura 5.14: Gráfico de posição x em relação ao tempo

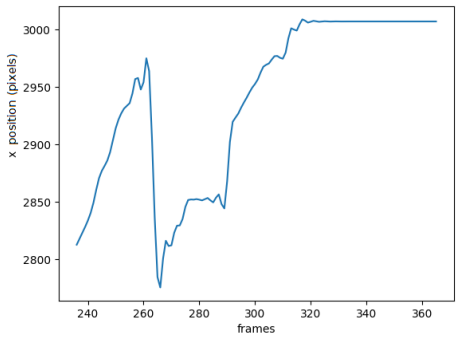

Figura 5.15: Gráfico de posição x em relação ao tempo interpolado

Nos gráficos 5.14 e 5.16 verificamos que entre os 260 frames e os 270 frames ocorre um salto tanto na posição vertical do centroide na imagem como na sua posição vertical na imagem. Este salto corresponderá a um pico de aceleração.

5.1.2.5 Conversão de unidades

Na imagem 5.19 podemos observar os pontos detectados no ficheiro de vídeo marcados a laranja, bem como o rectângulo identificado para a conversão de unidades da imagem em unidades reais a azul. Este ponto corresponde a uma borda da embarcação junto à segunda entrada de carga da embarcação a contar da proa junto ao alvo castanho, que

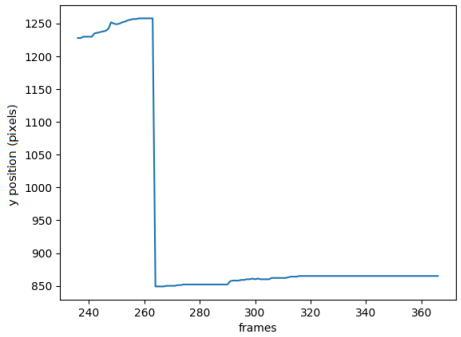

Figura 5.16: Gráfico de posição y em relação ao tempo

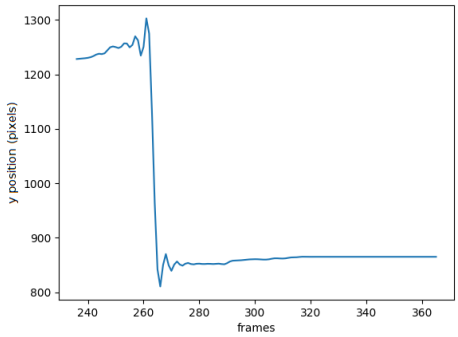

Figura 5.17: Gráfico de posição y em relação ao tempo interpolado

se pode observar na figura 5.18. Este era um dos rectângulos existentes perto da linha de água que nos é permitido visualizar no vídeo gravado e para o qual conhecemos as medidas. Medidas estas essenciais para a conversão descrita na secção 4.2.6.

Na imagem 5.20 podemos observar os pontos detectados bem como o rectângulo usado na conversão, mas agora num referencial correspondente às posições reais.

Figura 5.18: Captura do vídeo da câmara 2 do teste GIIP-50-92-450-PERI-0-53

5.1.2.6 Calculo de acelerações

Nas figuras 5.21 e 5.22 podemos observar a primeira derivada e a segunda derivada do deslocamento, velocidade e aceleração, no eixo dos x e no eixo dos y, respectivamente,

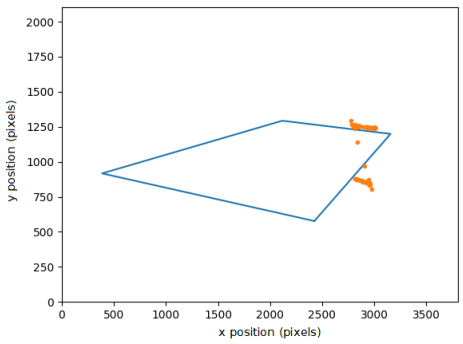

Figura 5.19: Gráfico ilustrativo das posições do ponto da embarcação escolhido na imagem de vídeo

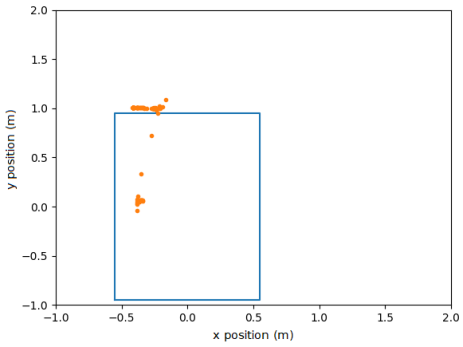

Figura 5.20: Gráfico ilustrativo das posições do ponto da embarcação escolhido em unidades reais (vista superior)

ao longo dos frames da gravação recolhida. Verificam-se nestes que existe uma grande aceleração do centroide na imagem.

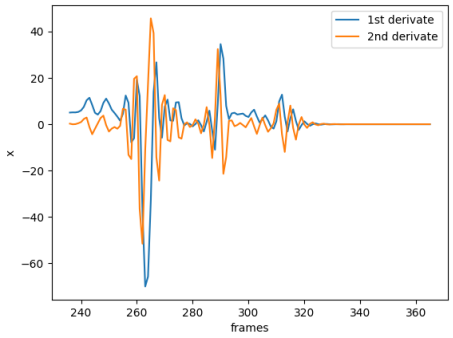

Figura 5.21: Gráfico combinado da primeira e segunda derivada no eixo dos x no movimento na imagem

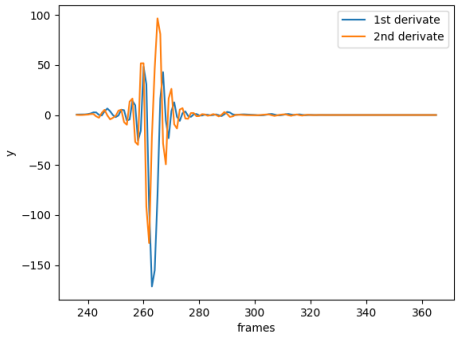

Figura 5.22: Gráfico combinado da primeira e segunda derivada no eixo dos y no movimento na imagem

Nas figuras 5.21 e 5.22 observamos os mesmo valores das duas figuras anteriores, mas agora com ambas as unidades, movimento e tempo, convertidas para unidades reais (metros e segundos ao invés de pixéis e frames). Em ambos os gráficos a velocidade apresenta um gráfico muito mais suave, mas isto não impediu que a aceleração (em m/s^2 ao contrário de m^2/s^2 como se encontra identificado nos gráficos) de apresentar valores muito acentuados.

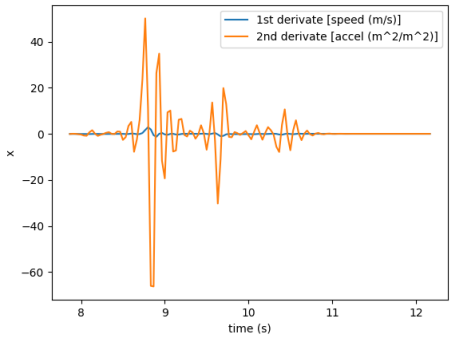

Figura 5.23: Gráfico combinado da primeira e segunda derivada no eixo dos x no movimento na realidade

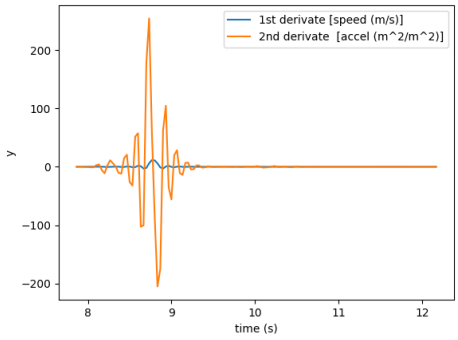

Figura 5.24: Gráfico combinado da primeira e segunda derivada no eixo dos y no movimento na realidade

5.1.2.7 Resultados finais

Na figura 5.25 temos os resultados do teste abreviados. Neste apenas são apresentados os valores pretendidos obter como conclusão da aplicação, a força correspondente à variação máxima de aceleração, correspondendo à desaceleração provocada pelo impacto da embarcação no cais, bem como o resultado da força de impacto com base neste valor e na massa da embarcação.

```
Test (y/n)[default n]?:n
Video file name:PXL_20221125_142413374_impacto1
Object id to track:45
Vessel mass (in Kg):13.6
Max acceleration: 187,12599405361146 m/s^2
Max force: 2544.9135191291157 Kg*m/s^2 (N m)
```

Figura 5.25: Output da consola e dados inseridos para a execução do teste

Nas tabelas 5.5 podemos encontrar os resultados obtidos pelo protótipo experimental aqui apresentado para todos os impactos em ambas as câmaras. Nestas tabelas podemos consultar para cada vídeo o id do centroide que foi seleccionado para seguir e associar à embarcação para a qual queremos saber a força de embate do acostamento. Para cada centroide é possível observar a aceleração máxima calculada a partir da nossa aplicação protótipo, bem como a sua força. A força foi calculada pelo método descrito em 2.6, tendo em conta a massa de 13,Uma versão mais extendida desta tabela, com o resultado para

outros testes encontra-se disponível no anexo III, bem como na pasta partilhada Google Drive [27].

5.1.3 Discussão resultados

Na secção 5.1.2.1 podemos verificar que apesar da grande diferença de performance entre ambos os processadores a duração da calibração continua a ser bastante acentuada. O facto de a calibração demorar aproximadamente 10x a duração do vídeo demostra o porquê da decisão de separar a calibração do vídeo do processamento.

Como se pode verificar, na secção 5.1.2.2, ambos os gráficos temos picos correspondentes aos embates, sendo possível observar nos gráficos relativos ao OptiTrack, gráficos 5.2 e 5.3, que os 13 embates tiveram todos aproximadamente a mesma força. Nos gráficos correspondentes aos acelerómetros instalado na embaracação, 5.4, 5.5, 5.6 e 5.7, não temos o ruído inicial e final que corresponde à instalação e desinstalação deste na embarcação uma vez que foram ignoradas para a criação do gráfico. No acelerómetro da proa é possível observar os picos dos embates, mas também algum ruido.

Quanto ao tracking verificou-se uma grande imprevisibilidade no seu comportamento, em alguns momentos o centroide identificado vai acompanhando o ponto detectado no local correcto, figuras 5.8 e 5.9, e noutros torna-se errático, figuras 5.10 e 5.11 e identifica zonas da imagem longe de onde realmente se encontra ou mesmo identificar o centroide numa localização igual à imediatamente anterior apesar da embarcação se encontrar em movimento. Ao contrário de em testes realizados anteriormente e graças à alteração da montagem do modelo à escala já foi possível seguir a embarcação até ao momento que esta embate no cais. No entanto apesar desta melhoria este método de tracking continua a apresentar uma precisão demasiado baixa nos testes à escala realizados.

Por fim, verifica-se na conversão de referencial de imagem para referencial real que a implementação das funções de forma é capaz de converter entre referenciais com sucesso. Tal pode se verificar nos exemplos das figuras 5.19 e 5.20 da secção 5.1.2.5.

Como se pode verificas pelos dados presentes na tabela 5.5 da secção 5.1.2.7 existe uma grande disparidade nos valores obtidos no tracking em relação aos valores medidos. Esta disparidade ocorre principalmente devido aos saltos que ocorrem durante o tracking, o que provoca a existência de acelerações demasiado grandes onde não deveriam existir. O elevado valor destas acelerações deve-se também à forma como se obtém este. A implementação presente assume que o valor que pretendemos obter é o valor máximo de desaceleração, uma vez que teoricamente o momento do embate corresponderá a este.

Em dados de outros testes foi possível obter-se alguns valores de grandeza semelhante aos recolhidos pelos equipamentos de medição usados para validação, tal como se pode verificar nos anexos III e **??**. No entanto, como se pode verificar nesse mesmo anexo este é um caso raro. Sendo que na sua larga maioria os valores obtidos são excessivamente altos em relação às acelerações reais. Estes valores ocorrem em impactos com baixo nível de posições de tracking. Por tal motivo, reconhece-se que recorrendo a mínimos locais

Tabela 5.5: Resultados dos vários impactos do teste GIIP-50-92-450-PERI-0-53.

Vídeo	**Id centroide**	**Max acceleration** (m/s^2)	**Max force** ($Kg * m/s^2$ **(N m)**)
GX011525_impacto1	25	109,636238150279	1534,90733410391
GX011525_impacto2	24	201,142579016029	2815,99610622441
GX011525_impacto3	19	458,479547155415	6418,71366017581
GX011525_impacto4	19	302,663514585806	4237,28920420128
GX011525_impacto5	21	66,5563801968738	931,789322756233
GX011525_impacto6	18	82,3908996092529	1153,47259452954
GX011525_impacto7	24	34,6040410110913	484,456574155278
GX011525_impacto8	24	-	-
GX011525_impacto9	26	84,7759788050637	1186,86370327089
GX011525_impacto10	17	220,327148370423	3084,58007718592
GX011525_impacto11	19	141,658020635521	1983,21228889729
GX011525_impacto12	30	713,339666299545	9986,75532819363
GX011525_impacto13	12	103,871556146919	1454,20178605687

Vídeo	**Id centroide**	**Max acceleration** (m/s^2)	**Max force** ($Kg * m/s^2$ **(N m)**)
PXL_20221125_1424 13374_impacto1	45	187,125994053611	2619,76391675055
PXL_20221125_1424 13374_impacto2	30	391,231812189352	5477,24537065093
PXL_20221125_1424 13374_impacto3	55	487,735570938326	6828,29799313656
PXL_20221125_1424 13374_impacto4	56	-	-
PXL_20221125_1424 13374_impacto5	19	701,826158155955	9825,56621418337
PXL_20221125_1424 13374_impacto6	64	700,823006134325	9811,52208588055
PXL_20221125_1424 13374_impacto7	64	464,147552880305	6498,06574032427
PXL_20221125_1424 13374_impacto8	-	-	-
PXL_20221125_1424 13374_impacto9	31	428,80512299764	6003,27172196696
PXL_20221125_1424 13374_impacto10	29	472,154488749986	6610,1628424998
PXL_20221125_1424 13374_impacto11	37	37,2117279898309	520,964191857633
PXL_20221125_1424 13374_impacto12	31	498,734567474481	6982,28394464273
PXL_20221125_1424 13374_impacto13	35	262,508145144035	3675,11403201649

poderia melhorar a eficácia do nosso protótipo experimental.

5.2 Testes complementares

Foram também feitos outros testes complementares de forma a analisar a viabilidade de outras ferramentas que mostraram potencial, bem como técnicas para a melhoria da precisão da conversão de coordenadas da fotografia para coordenadas reais.

Figura 5.26: Teste coordenadas GPS com câmara a amarelo e pontos de interesse a azul claro

Na figura 5.26 podemos observar o teste à biblioteca cameratransform[28] que foi avaliado o seu potencial uso para a conversão de posições na imagem em posições reais recorrendo a pontos de interesse globais, os postes, e a informação sobre a câmara. Para fazer esta conversão era necessário fornecer à biblioteca as coordenadas GPS, bem como a altura dos pontos de interesse bem como da câmara. É necessário também fornecer caracteristicas do sensor da câmara (distância focal, tamanho do sensor e resolução da imagem captada). Apesar de se ter feito um teste complementar de forma a avaliar a viabilidade desta ferramenta depressa foi colocada de parte pois verificou-se que não iria ser possível obter as gravações reais de uma embarcação a atracar que inicialmente se esperava poder contar.

Na figura 5.27 temos o teste a outra funcionalidade da biblioteca cameratransform[28], a possibilidade de agrupar duas câmaras. Foi testado a utilização desta biblioteca, que necitaria de se configurar pontos na imagem que sejam comuns a ambas as câmaras de forma a obter uma analogia entre a posição da embarcação em ambos os vídeos e a sua posição num referencial. Este método além das propriedades da câmara referidos anteriormente necessita também dos seus ângulos de rotação lateral e vertical, bem como

Figura 5.27: Teste piscina com duas câmaras

as posições de ambas as câmaras relativamente ao referencial, em distâncias e alturas. Este verificou-se não funcionar no nosso caso, sendo a mais provável causa o facto de a posição em ambas as câmaras dos centroídes não corresponderem exactamente ao mesmo ponto na embarcação uma vez que este ponto identificado é o centro do quadrilátero que contém o objecto detectado pelo background subtractor.

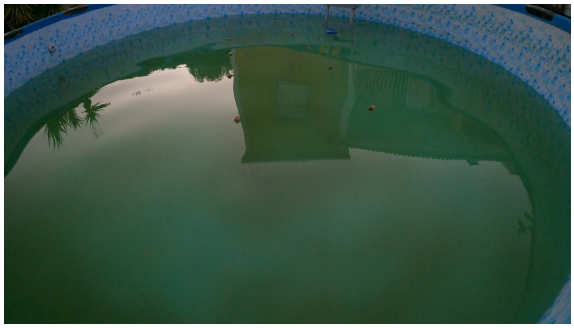

Figura 5.28: Teste piscina com rolhas

Na figura 5.28 temos um teste que foi realizado com o objectivo de se chegar a uma forma de se obter a transformação utilizando as distâncias constantes entres as rolhas à medida que estas se vão afastando da câmara. No entanto acabou-se por se encontrar o método implementado, menos rudimentar que o idealizado para este teste. No entanto a ideia de se ter 4 boias no plano de água a formar um rectângulo poderá ser útil para futuros testes uma vez que permite delimitar um rectângulo maior do que o utilizado nos testes realizados com o modelo à escala.

6

Conclusões e Trabalho Futuro

6.1 Conclusão

Pela secção 5.1.3 podemos obter várias conclusões acerca dos resultados obtidos.

Pelos dados de 5.1.2.2 podemos concluir que a ferramenta de recolha de acelerações desenvolvida e aplicada na embarcação modelo obteve o comportamento esperado, apresentado todos os impactos da embarcação no cais, como se esperava.

Com base nos vídeos dos testes efectuados foi possível, com o *background subtractor*, observar a fase de impacto da embarcação com o cais de atracagem do porto. Sendo assim possível calcular a sua força de impacto. Verifica-se no entanto que existe elevada imprecisão no *tracking* recorrendo a este método o que provoca que o resultado destes calculos nos leve à obtenção de valores altamente imprecisos. No entanto verifica-se a existência de casos em que os valores se aproximam dos obtidos pelos equipamentos de medição no local. Isto poderá dever-se a vários factores presentes na implementação da aplicação protótipo aqui apresentada. As duas possibilidades que mais poderão afectar a nossa medição, dado a imprecisão no *tracking* são a existência de algum *bug* no código desenvolvido, como por exemplo a necessidade de melhoria da ferramenta de identificação e gestão dos centroides, ou o BackgroundSubtractor não ser suficientemente preciso.

O projeto desenvolvido está disponível no GitHub, tanto para a aplicação principal[26] como para os acelerómetros utilizados[24]. É também possível consultar todos os dados recolhidos pela instrumentação presente no teste realizado, bem como os dados processados no âmbito desta dissertação numa pasta partilhada Google Drive[27].

6.2 Trabalho Futuro

Sugerem-se as seguintes melhorias ao projecto já desenvolvido:

- Aprimorar a aplicação actual por exemplo com uma interface gráfica.
- Analisar a possibilidade de aumento de performance.
- Incorporar a criação do vídeo calibrado no código da aplicação.

- Melhorar a parametrização da conversão em unidades reais.
- Melhorar a ferramenta de gestão de centroides(*CentroidTracker*)
- Verificar a utilização de outro método de *tracking*.

Inicialmente pretendia-se que este protótipo tivesse uma interface gráfica que fizesse a ligação à aplicação instalada num servidor que processaria o ficheiro de vídeo. Esta interface deverá permitir a visualização do vídeo de forma a identificar o ponto correcto, bem como possibilitar configurar os parametros de processamento, seja graficamente ou permitindo o *upload* de ficheiros com estas configurações.

Como se verificou anteriormente a calibração do vídeo tem um desempenho extremamente baixo, por tal sugere-se a futura analise de linguagens de programação ou métodos de calibração que apresentem maior performance.

Uma vez que não foi possível incorporar a criação de vídeo calibrado no código em tempo útil aconselha-se investigação mais profunda de tal tarefa.

Apesar de terem sido criados ficheiros de configuração e estes terem sido feitos com os valores que melhores resultados criava para cada vídeo verificou-se que a continuação da sua refinação iria resultar em melhorias cada vez menores com um impacto bastante grande a nível temporal, o que iria resultar em menos tempo para a elaboração desta dissertação.

Uma vez que se verificou alguma instabilidade no tracking feito pelo método analisado será interessante o estudo de outra metodologia. Uma possível opção seria a identificação manual de um ponto na embarcação e seguimento deste mesmo ponto por comparação histográfica.

Bibliografia

[1] 'OpenCV', consultado em 16 de Julho de 2021. URL: https://opencv.org/ (ver pp. 2–4).

[2] 'OpenCV Alternatives', consultado em 16 de Julho de 2021. URL: https://www.g2.com/products/opencv/competitors/alternatives (ver p. 4).

[3] 'Spatial Analysis Overview', consultado em 16 de Julho de 2021. URL: https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/intro-to-spatial-analysis-public-preview (ver p. 4).

[4] 'Amazon Rekognition Video', consultado em 16 de Julho de 2021. URL: https://aws.amazon.com/pt/rekognition/video-features/ (ver p. 4).

[5] 'Google Cloud Vision AI', consultado em 16 de Julho de 2021. URL: https://cloud.google.com/vision (ver p. 4).

[6] 'Scikit-image Handling Video Files', consultado em 16 de Julho de 2021. URL: https://scikit-image.org/docs/dev/user_guide/video.html (ver p. 4).

[7] 'SimpleCV's documentation', consultado em 16 de Julho de 2021. URL: https://simplecv.readthedocs.io/en/latest/ (ver p. 4).

[8] 'Watson Visual Recognition', consultado em 16 de Julho de 2021. URL: https://www.ibm.com/dk-en/cloud/watson-visual-recognition (ver p. 4).

[9] 'SimpleCV's repository', consultado em 16 de Julho de 2021. URL: https://github.com/sightmachine/SimpleCV (ver p. 4).

[10] 'How to Use Background Subtraction Methods ', consultado em 16 de Julho de 2021. URL: https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction. (ver p. 6).

[11] 'Simple object tracking with OpenCV', consultado em 16 de Julho de 2021. URL: https://pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/ (ver pp. 11, 32).

[12] 'Camera Calibration and 3D Reconstruction', consultado em 16 de Julho de 2021. URL: https://docs.opencv.org/4.5.2/d9/d0c/group__calib3d.html (ver p. 11).

[13] 'Ship Detection from Satellite Imagery', consultado em 22 de Fevereiro de 2022. URL: https://github.com/lmzh123/ships_detection (ver p. 15).

[14] 'Game of Deep Learning: Ship datasets', consultado em 22 de Fevereiro de 2022. URL: Ships%20in%20Satellite%20Imagery (ver p. 15).

[15] 'Game of Deep Learning: Ship datasets', consultado em 22 de Fevereiro de 2022. URL: https://www.kaggle.com/arpitjain007/game-of-deep-learning-ship-datasets (ver p. 15).

[16] 'Object Tracking using OpenCV (C++/Python)', consultado em 16 de Julho de 2021. URL: https://learnopencv.com/object-tracking-using-opencv-cpp-python/ (ver p. 17).

[17] 'OpenCV Object Tracking', consultado em 16 de Julho de 2021. URL: https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/ (ver p. 17).

[18] 'Mapping in 3D, helping understand the 3D world from 2D sensors', consultado em 16 de Julho de 2021. URL: https://github.com/Tetragramm/opencv_contrib/tree/master/modules/mapping3d/src (ver p. 18).

[19] 'MPU9250', consultado em 16 de Outubro de 2021. URL: https://github.com/hideakitai/MPU9250 (ver p. 27).

[20] 'Grove - IMU 9DOF v2.0', consultado em 16 de Outubro de 2021. URL: https://wiki.seeedstudio.com/Grove-IMU_9DOF_v2.0/ (ver p. 27).

[21] 'ESP32: Guide for MicroSD Card Module using Arduino IDE', consultado em 16 de Outubro de 2021. URL: https://randomnerdtutorials.com/esp32-microsd-card-arduino/ (ver p. 27).

[22] 'Reading configuration from SD Card', consultado em 16 de Outubro de 2021. URL: https://forum.arduino.cc/t/reading-configuration-from-sd-card/506024 (ver p. 27).

[23] 'ESPNtpClient', consultado em 16 de Outubro de 2021. URL: https://github.com/gmag11/ESPNtpClient (ver p. 27).

[24] Repositório com os desenvolvimentos realizados para a recolha de acelerações da embarcação modelo. URL: https://github.com/RSantos94/esp32-accelerometer-logger (ver pp. 27, 53).

[25] 'Python natural smoothing splines', consultado em 31 de Maio de 2022. URL: https://stackoverflow.com/questions/51321100/python-natural-smoothing-splines (ver p. 33).

[26] Repositório com os desenvolvimentos realizados para a implementação desta proposta. URL: https://github.com/RSantos94/vessel-impact-detection (ver pp. 33, 36, 53).

[27] Pasta Google Drive com todos os dados recolhidos nos testes. URL: https://drive.google.com/drive/folders/1ux7oiYqlMSjuHOWveBTdX-v2UKP1R2S4?usp=sharing (ver pp. 38, 48, 53).

[28] 'Welcome to the CameraTransform Documentation', consultado em 25 de Fevereiro de 2022. URL: https://cameratransform.readthedocs.io/en/latest/ (ver p. 51).

[29] O. E. Agazzi e S.-s. Kuo. "Hidden markov model based optical character recognition in the presence of deterministic transformations". Em: *Pattern Recognition* 26.12 (1993), pp. 1813–1826. ISSN: 0031-3203. DOI: https://doi.org/10.1016/0031-3203(93)90178-Y. URL: https://www.sciencedirect.com/science/article/pii/003132039390178Y (ver p. 18).

[30] M. Alonso e E. J. Finn. *Física: um curso universitário Volume I - Mecânica (Fundamental University Physics*. Ed. por G. Moscati. São Paulo, Brasil: Editora Edgard Blücher Ltda., 1972 (ver p. 14).

[31] G. Chandan et al. "Real Time Object Detection and Tracking Using Deep Learning and OpenCV". Em: *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2018, pp. 1305–1308. DOI: 10.1109/ICIRCA.2018.8597266 (ver p. 16).

[32] H. K. Cheng, Y.-W. Tai e C.-K. Tang. *Rethinking Space-Time Networks with Improved Memory Coverage for Efficient Video Object Segmentation*. 2021. arXiv: 2106.05210 [cs.CV] (ver p. 16).

[33] K. Doğançay e R. Arablouei. "Selective angle measurements for a 3D-AOA instrumental variable TMA algorithm". Em: *2015 23rd European Signal Processing Conference (EUSIPCO)*. 2015, pp. 195–199. DOI: 10.1109/EUSIPCO.2015.7362372 (ver p. 18).

[34] O. D. Faugeras. "What can be seen in three dimensions with an uncalibrated stereo rig". Em: *ECCV*. 1992 (ver p. 18).

[35] Y. Hu e J.-N. Hwang. "Handbook of Neural Network Signal Processing". Em: Boca Raton: CRC Press, 2002, pp. 94–133. ISBN: 9781315220413. DOI: 10.1201/9781315220413 (ver p. xvii).

[36] P. Janku et al. "Comparison of tracking algorithms implemented in OpenCV". Em: *MATEC Web of Conferences* 76 (jan. de 2016), p. 04031. DOI: 10.1051/matecconf/20167604031 (ver p. 17).

[37] R. Lienhart e J. Maydt. "An Extended Set of Haar-Like Features for Rapid Object Detection". Em: *IEEE ICIP 2002*. 2002, pp. 900–903 (ver p. 15).

[38] K. Liu et al. "A Real-Time Method to Estimate Speed of Object Based on Object Detection and Optical Flow Calculation". Em: *Journal of Physics: Conference Series* 1004 (abr. de 2018), p. 012003. DOI: 10.1088/1742-6596/1004/1/012003. URL: https://doi.org/10.1088/1742-6596/1004/1/012003 (ver p. 16).

[39] C. J. E. M. Pinheiro Liliana V.; Fortes e J. A. Santos. "Risk analysis and management of moored ships in ports". Em: *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. Vol. 3. 2018, pp. 1–9. DOI: 10.1109/ICIRCA.2018.8597266 (ver pp. 1, 3).

[40] S. Pumrin e D. Dailey. "Dynamic camera calibration to estimate mean vehicle speed". Em: *Proceedings of SPIE - The International Society for Optical Engineering* (mai. de 2002). DOI: 10.1117/12.468010 (ver p. 18).

[41] S. A. Sánchez, J. Campillo e J. C. Martınez-Santos. "Use of deep learning algorithms for real-time detection of vessels in confined spaces using the Tensorflow framework". Em: *Journal of Physics: Conference Series* 1448 (jan. de 2020), p. 012003. DOI: 10.1088/1742-6596/1448/1/012003. URL: https://doi.org/10.1088/1742-6596/1448/1/012003 (ver p. 16).

[42] R. Smith. "An Overview of the Tesseract OCR Engine". Em: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. 2007, pp. 629–633. DOI: 10.1109/ICDAR.2007.4376991 (ver p. 18).

[43] S. Suzuki e K. be. "Topological structural analysis of digitized binary images by border following". Em: *Computer Vision, Graphics, and Image Processing* 30.1 (1985), pp. 32–46. ISSN: 0734-189X. DOI: https://doi.org/10.1016/0734-189X(85)90016-7. URL: https://www.sciencedirect.com/science/article/pii/0734189X85900167 (ver p. 11).

[44] R. Szeliski. *Computer Vision - Algorithms and Applications, Second Edition*. Texts in Computer Science. Springer, 2022 (ver p. 17).

[45] T. Szkodny. "Calculation of the Location Coordinates of an Object Observed by a Camera". Em: *Man-Machine Interactions 3*. Ed. por D. A. Gruca, T. Czachórski e S. Kozielski. Cham: Springer International Publishing, 2014, pp. 139–151. ISBN: 978-3-319-02309-0 (ver p. 15).

[46] E. Zacharias, M. Teuchler e B. Bernier. "Image Processing Based Scene-Text Detection and Recognition with Tesseract". Em: *CoRR* abs/2004.08079 (2020). arXiv: 2004.08079. URL: https://arxiv.org/abs/2004.08079 (ver p. 18).

[47] Z. Zhang. "A Flexible New Technique for Camera Calibration". Em: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.11 (nov. de 2000), pp. 1330–1334. ISSN: 0162-8828. DOI: 10.1109/34.888718. URL: https://doi.org/10.1109/34.888718 (ver p. 28).

[48] O. C. Zienkiewicz, R. L. Taylor e J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals, Sixth Edition*. 6ª ed. Butterworth-Heinemann, mai. de 2005. ISBN: 0750663200 (ver p. 11).

[49] Z. Zivkovic e F. van der Heijden. "Efficient adaptive density estimation per image pixel for the task of background subtraction". Undefined. Em: *Pattern recognition letters* 27.1636734 (P (jan. de 2006). 10.1016/j.patrec.2005.11.005, pp. 773–780. ISSN: 0167-8655 (ver pp. 5, 8, 9).

[50] Z. Zivkovic. "Improved Adaptive Gaussian Mixture Model for Background Subtraction". Em: *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2 - Volume 02*. ICPR '04. USA: IEEE Computer Society, 2004, pp. 28–31. ISBN: 0769521282 (ver p. 8).

I

Calibração

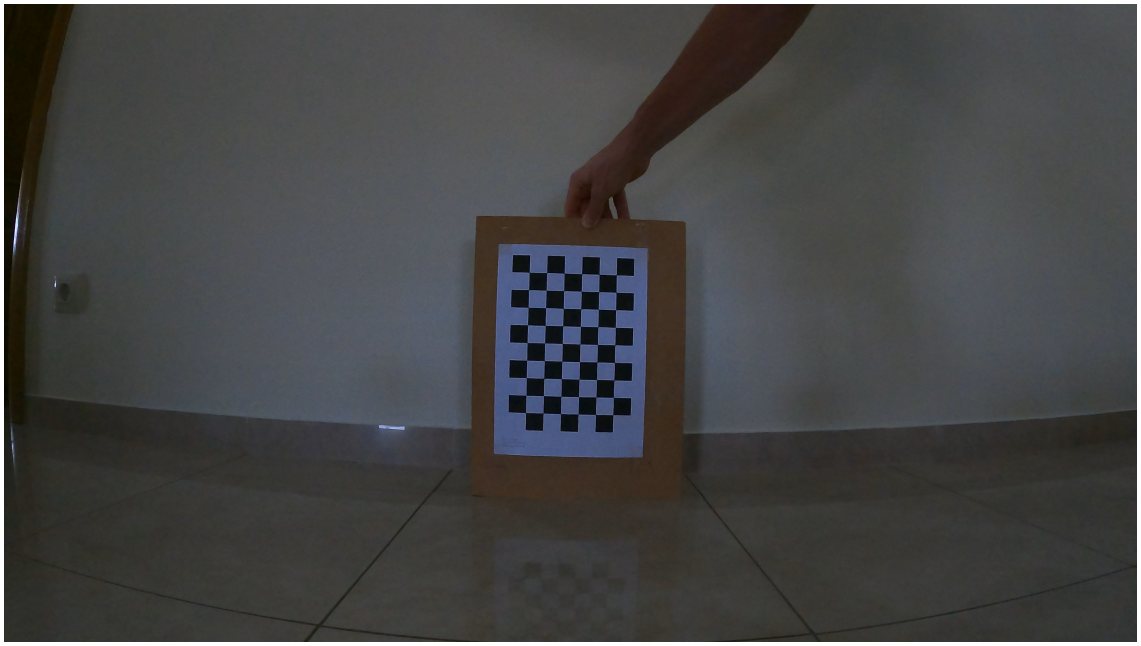

Figura I.1: Imagem retirada directamente da câmara

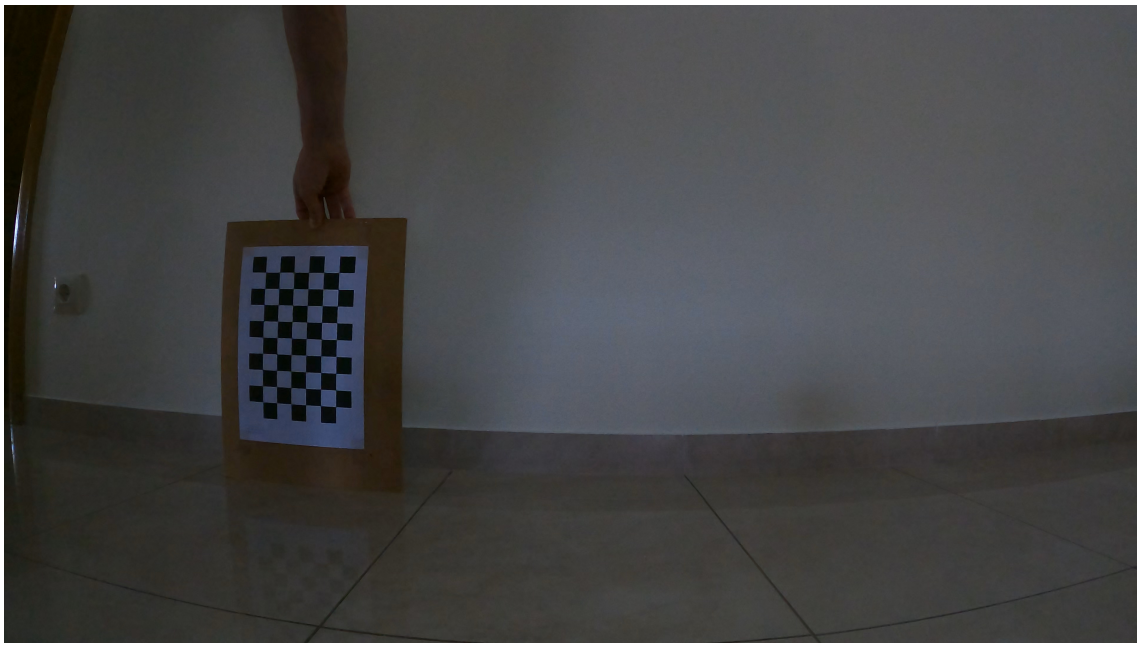

Figura I.2: Imagem retirada directamente da câmara

Figura I.3: Imagem retirada directamente da câmara