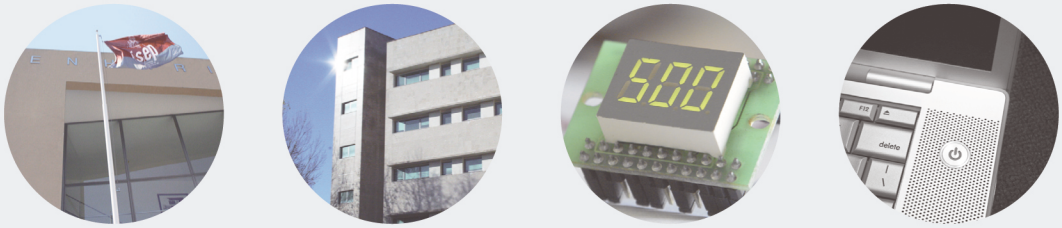
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema de deteção e monitorização de obstáculos para navegação autónoma marítima

DANIEL DA SILVA FREIRE
Outubro de 2019

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Sistema de deteção e monitorização de obstáculos para navegação autónoma marítima

Daniel da Silva Freire

Mestrado em Engenharia Electrotécnica e de Computadores

Orientador: Alfredo Manuel Oliveira Martins

Coorientador: José Miguel Soares de Almeida

9 de Outubro de 2019

Abstract

Autonomous Surface Vehicles (ASVs), operating near ship harbors or relatively close to shorelines must be able to steer away from incoming vessels and other possible obstacles, be they dynamic or not. To do this, one must implement some type of multi-target tracking and obstacle avoidance algorithms that lets the vehicle dodge obstacles.

This thesis presents a radar-based multi-target tracking system developed for obstacle detection and monitoring. The proposed architecture system can use different types of sensors to improve the quality of the data. This work is focused in the radar sensor. The system was designed for ROAZ II ASV belonging to INESC TEC/ISEP and implemented in Robot Operating System (ROS) for easier integration with the already existing software.

The developed aggregation, classification and tracking algorithms are presented, as well as the algorithm for estimation of possible collisions between vessel's. Aggregation and classification algorithms were tested with real data and the results are presented in this work. A simulation environment could prove the correct behavior of tracking and estimation of possible collisions algorithms.

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Agradecimentos

Desejo exprimir os meus agradecimentos a todos aqueles que, de alguma forma, permitiram que esta tese se concretizasse.

À minha namorada, Raquel Carneiro Rodrigues, por me suportar nos momentos de angustia, me aturar no dias menos bons e por nunca me deixar desistir de querer alcançar os meus objectivos.

Aos engenheiros Alfredo Manuel Oliveira Martins e José Miguel Soares de Almeida pela orientação, apoio, pelas opiniões e na total colaboração para resolução de problemas que surgiram ao longo do desenvolvimento deste trabalho.

Aos meus colegas e amigos do Laboratório de Sistemas Autónomos que estiveram sempre ao meu lado para me suportar nos momentos de falhas e permitiram que este trabalho fosse finalizado.

Por ultimo e não menos importante, tendo consciência de que sozinho nada disto teria sido possível quero agradecer aos meus pais e irmã pelo apoio incondicional, incentivo e paciência demonstrados ao longo destes três anos de mestrado.

Daniel Freire

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Conteúdo

Lista de Figuras

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Lista de Tabelas

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Lista de abreviaturas

AIS	**A**utomatic **I**dentification **S**ystem
ANN	**A**rtificial **N**eural **N**etwork
APF	**A**rtifical **P**otencial **F**ields
ASV	***A**utonomous **S**urface **V**ehicles*
ATR	**A**utomatic **T**arget **R**ecognition
CPA	**C**losest **P**oint **A**pproach
DVL	***D**oppler **V**elocity **L**og*
ECEF	**E**arth-**C**entered, **E**arth-**F**ixed
EKF	**E**xtended **K**alman **F**ilter
EMI	**E**lectro**m**agnetic **I**nterference
ENU	**E**ast-**N**orth-**U**p
FMCW	**F**requency **M**odulated **C**ontinuous **W**ave
GPS	**G**lobal **P**ositioning **S**ystem
INS	**I**nertial **N**avigation **S**ystem
ISEP	**I**nstituto **S**uperior de **E**ngenharia do **P**orto
LSA	**L**aboratório de **S**istema **A**utónomos
NIS	**N**ormalised **I**nnovation **S**quared test
ROS	**R**obot **O**perating **S**ystem
ROSM	***R**obotic **O**il **S**pill **M**itigation*
SBP	***S**ub-**B**ottom **P**rofiler*
USV	***U**nmanned **S**urface **V**ehicles*
VFH	**V**ector **F**ield **H**istogram

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Capítulo 1

Introdução

O oceano ocupa dois terços do planeta terra[1], porém apenas uma pequena porção deste foi explorado[2]. Na comunidade científica têm sido feitas muitas pesquisas e estudos no espaço tomando o lugar do estudo dos oceanos. O fundo dos oceanos é abundante em recursos minerais que ainda não foram explorados, devido ao lento crescimento tecnológico na área[1]. A Oceanografia é a ciência que estuda todos os fenómenos dos oceanos, desde a biologia marinha, poluição marinha até ao comportamento das correntes e marés. A Oceanografia pode ser dividida em quatro grandes áreas: física, química, biológica e geológica.

Os *Autonomous Surface Vehicles* (ASV) ou muitas vezes conhecidos como *Unmanned Surface Vehicles* (USV) são veículos não tripulados que operam à superfície da água com um conjunto de aplicações bem distintas. Do ponto de vista cientifico, os ASV podem prestar apoio no mapeamento e reconstrução do solo do oceano, estabelecimento de comunicação entre diferentes veículos e aquisição de dados para estudos de eco-sistemas. Os ASV podem também prestar apoio na exploração e manutenção de oleodutos, plataformas petrolíferas e mineração sub-aquática. Existe um vasto conjunto de aplicações para as quais os ASV são indicados dado a sua versatilidade. É fundamental para um ASV ser capaz de detectar o ambiente que o rodeia de modo a evitar colisões.

Com o objectivo de aplicar os conhecimentos adquiridos ao longo do mestrado de Engenharia Electrotécnica e de Computadores efectuado no Laboratório de Sistema Autónomos (LSA) do Instituto Superior de Engenharia do Porto (ISEP), foi proposto o desenvolvimento de um sistema capaz de efectuar detecção de monitorização de obstáculos para o ASV ROAZ II focando na tecnologia radar.

O ROAZ II é um ASV que tem sido desenvolvido pelo LSA e INESC TEC e que é dotado de um conjunto de sensores que o permitem operar autonomamente em ambiente aquático como: sistema de *Global Positioning System* (GPS), *Inertial navigation system* (INS), Radar, LIDAR, Câmera termográfica entre outros. Este veículo tem a capacidade de executar batimetria, recolha de dados para oceanografia, patrulha de uma área, busca

e salvamento e estabelecer a comunicação entre diferentes veículos sub-aquáticos[3]. De forma a poder navegar de modo seguro pelo oceano, o ROAZ II necessita de um sistema capaz de detectar e monitorizar obstáculos nas proximidades do mesmo.

O projecto *Robotic Oil Spill Mitigation* (ROSM) tem como principal objectivo combater o derrame de petróleo com recurso a sistemas multi robóticos e é levado a cabo, em parte, pelo INESC TEC[4]. Este projecto faz uso de dois veículos autónomos que combatem o derrame: um veículo aéreo e um na superfície da água (ROAZ II). O primeiro veículo é responsável por mapear a área afectada e combater o interior da área afectada com uma bactéria em pó. O ROAZ II, após obter a definição da área afectada, fica responsável por contornar a área lançando a bactéria misturada com água. A configuração de dois veículos permite actuar de forma rápida e eficaz, reduzindo deste modo os custos ambientais e económicos. A importância de um sistema de detecção e monitorização de obstáculos para este cenário em particular é deveras importante visto que possibilita que a missão decorra da forma mais segura possível.

1.1 Objectivos

O objectivo principal do projecto, como mencionado anteriormente (capítulo 1), é desenvolver um sistema capaz de fazer *target tracking* a embarcações bem como extrair a fronteira que separa o mar da terra. Para tal, as seguintes tarefas devem ser cumpridas:

- Desenvolvimento de *software* para comunicação com sensores.
- Estudo e implementação de algoritmos de agregação de dados.
- Desenvolvimento de *software* capaz de extrair características que descrevam um objecto.
- Desenvolvimento e validação de *software* para efectuar monitorização de múltiplos alvos.
- Desenvolvimento e validação de *software* capaz de extrair linha da costa para criação de mapa de navegação.
- Desenvolvimento de um sistema capaz de prever possíveis colisões.
- Teste e validação do projecto em cenários reais.

1.2 Estrutura da Tese

No capítulo 2, é feito um levantamento dos sensores usados na navegação náutica, métodos de *automatic target recognition*, de desvio de obstáculos e de desvio de obstáculos tendo em consideração as regras de navegação náuticas.

No capítulo 3, é apresentado o ASV ROAZ II de forma detalhada bem como a arquitectura do sistema implementado.

O capítulo 4 pretende expor o funcionamento básico de um radar, em concreto do radar presente no ROAZ II, e apresentar os módulos de *software* desenvolvidos para estabelecer comunicação com o radar presente no ROAZ II.

No capítulo 5 são apresentados diferentes métodos de agregação de pontos, o método implementado para funcionamento em tempo real e por ultimo o módulo de classificação/descrição de cada objecto que é resultado da agregação de pontos.

Por ultimo, o capítulo 6 faz uma breve exposição teórica do filtro de Kalman linear, é apresentado o sistema implementado para efectuar *tracking* a múltiplos alvos bem como o simulador que permitiu validar o funcionamento do filtro. De forma adicional foi implementado um sistema capaz prever possíveis colisões com embarcações usando os resultados do filtro implementado.

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Capítulo 2

Estado da Arte

Numa primeira parte são apresentadas e comparadas as tecnologias que permitem observar o mundo seguido de soluções existentes para *ATR*. Por fim são apresentados diferentes algoritmos de *obstacle avoindance* genérico e tendo em consideração as regras de navegação náuticas, *colregs*.

2.1 Sensores de navegação náutica

Quando se fala em navegação autónoma, ou mesmo navegação náutica, é muito importante a escolha do sensor ou sensores que poderão ser usados para este efeito. Os radares são dos sensores mais usados para navegação dado o seu alcance, imunidade a condições ambientais e capacidade de detecção[5]. Contudo os radares são muito susceptíveis a interferência electromagnética (do inglês *electromagnetic interference* (EMI)), que poderão ser provenientes de comunicações nas proximidades do radar. O radar é um sensor que está desenhado para conseguir observar objectos de tamanho elevado e a grandes distâncias. Em situações em que se pretende detectar pequenas embarcações ou até mesmo detritos na água, o radar não é o sensor indicado para tal.

Outro sensor normalmente usado para navegação náutica é um receptor de *Automatic Identification System* (AIS)[5]. Este tipo de sensor permite receber informações como:

- Localização
- Velocidade
- Nome da Embarcação
- *Heading*

de embarcações que se encontrem nas proximidades e que possuam um transmissor de AIS. Porém, para poder efectuar navegação autónoma, não é possível usar apenas um receptor de AIS, visto que a legislação não obriga que todas as embarcações possuam

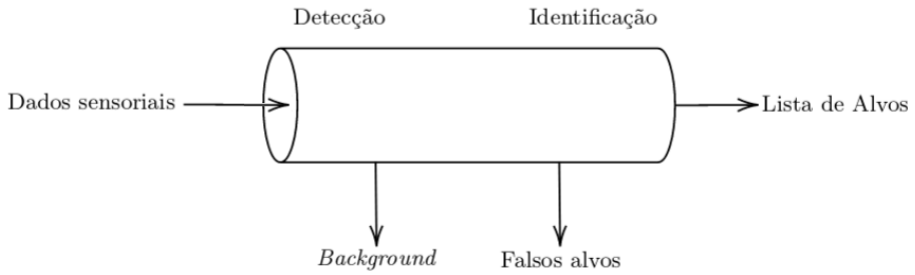


Figura 2.1: *Pipeline* do processo de ATR

um transmissor de AIS. Apenas embarcações maiores que 300 toneladas brutas ou de passageiros, sem restrições de tamanho, é que estão obrigadas a ter a bordo um transmissor de AIS[5]. O receptor de AIS sofre do mesmo problema do radar, como as embarcações pequenas não possuem um transmissor de AIS, não é possível detectar essas embarcações ou objectos pequenos presentes na água.

Existem outros sensores que poderão ser usados para navegação e ajudar a colmatar as limitações do radar como por exemplo câmeras e LIDAR. As câmeras convencionais são sensores a 2D. Contudo utilizando uma configuração de duas câmeras é possível obter uma imagem a 3D conseguindo deste modo ter o sentido de profundidade. Estas podem ser usadas para detectar pequenos objectos que poderão passar despercebidos a outros sensores. Contudo, câmeras são sensores que não funcionam em ambientes escuros ou com limitações de visibilidade frequentes em meios marinhos como por exemplo o nevoeiro. Por outro lado, o LIDAR é um sensor que permite descrever melhor os objectos a curta distância e que pode ser usado para navegação em locais aonde existem objectos em redor, como por exemplo num porto[6].

2.2 Automatic Target Recognition

O problema de *Automatic Target Recognition* (ATR) pode ser dividido em aquisição, detecção, identificação e seguimento (*tracking*) de um determinado alvo com recurso a sensores imperfeitos, com ruído [7]. A figura 2.1 representa o *pipeline* de um sistema de ATR. Idealmente todos os alvos desejados presentes na leitura dos sensores, são vistos na lista de alvos para poder ser efectuado *tracking* destes.

Mecocci et al. [8] apresentaram um método de ATR para controlo de tráfego de embarcações no acesso a um porto marítimo. O sistema proposto faz uso de um radar que gera uma imagem, em *grey-level*, por cada rotação do mesmo. A cada imagem gerada pelo radar, é aplicado um processo de segmentação da imagem, que consiste em criar um histograma da imagem de modo a observar se existem alvos na imagem. No caso de existir algum alvo na imagem, o *threshold* é guardado de forma a binarizar a imagem. Aplicando este método é possível extrair o *background* e falsos alvos da imagem. Após obter uma

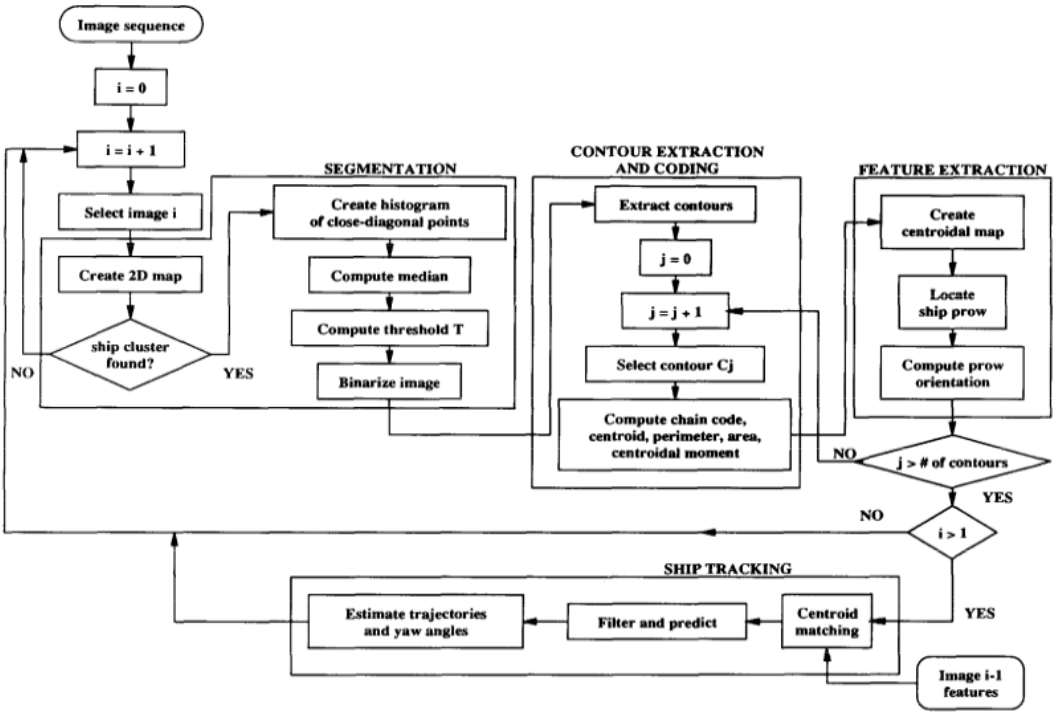


Figura 2.2: Diagrama proposto por *Mecocci et al.* [8]

imagem binária, é extraído o contorno e um conjunto de *features* que permitam descrever o alvo da melhor forma tais como centroide, perímetro, área, posição da proa e orientação.

De um conjunto de imagens sequenciais são extraídas as *features* e é aplicado o algoritmo de *tracking* de alvos. O algoritmo consiste em comparar *features* de forma iterativa da *frame* actual com os da *frame* anterior. Na situação de existirem *features* comuns entre *frames* é aplicado um filtro $\alpha\beta$. O filtro $\alpha\beta$ é um derivação do filtro de Kalman linear de fácil implementação e que possui uma grande flexibilidade. Contudo o filtro $\alpha\beta$ é menos eficaz do que o filtro de Kalman linear em ambientes não estacionários e nas inicializações do filtro[8]. As equações usadas no filtro são dadas por:

$$\begin{aligned} i(k) &= x_m(k) - x_p(k) \\ x_s(k) &= x_p(k) + \alpha i(k) \\ v_s(k) &= v_s(k-1) + \tfrac{\beta}{T} i(k) \\ x_p(k+1) &= x_s(k) + T v_s(k) \end{aligned} \tag{2.1}$$

Onde a primeira equação é usada no acto de actualização e as restantes no acto de previsão, $x_m(k)$, $x_p(k)$, $x_s(k)$ são respectivamente posições observadas, previstas e suavizadas, $v_s(k)$ é a velocidade estimada, $i(k)$ a inovação, T o intervalo entre observações e α e β são os parâmetros configuráveis do filtro. Na situação de existirem *features* na *frame* actual que não correspondam a nenhum alvo previamente visto, é inicializado um novo filtro $\alpha\beta$ para efectuar *tracking* ao alvo. A figura 2.2 é o diagrama do processo proposto

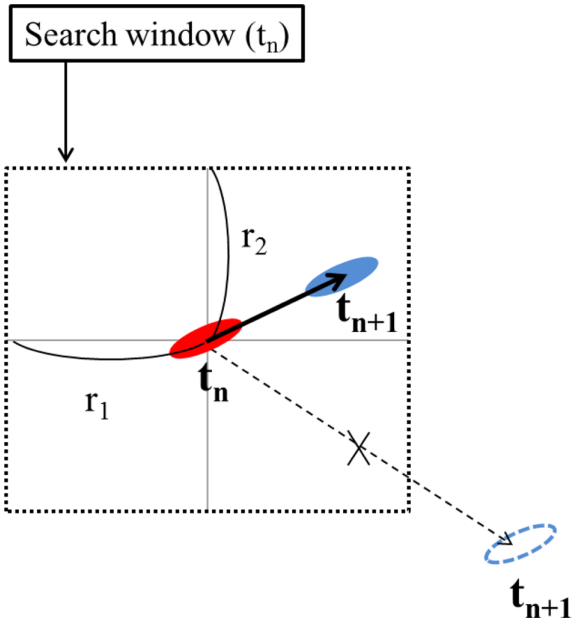


Figura 2.3: Algoritmo de janela de procura apresentado por *Hong et al.* [9]

por *Mecocci et al.*[8].

Hong et al. [9] apresentam um algoritmo para efectuar detecção e *tracking* de embarcações com recurso a um radar de Frequência Modulada e Onda Contínua, do inglês *Frequency Modulated Continuous Wave* (FMCW), para controlo de embarcações na entrada do porto de *Pyeogtaek*, Coreia. O algoritmo apresentado começa por adquirir dados de um radar sobre a forma de uma imagem, onde regiões escuras representam o vazio e regiões claras representam objectos e ruído. Numa primeira fase são removidos os locais que representam terra e ruído. A imagem obtida é comparada com a imagem da *frame* seguinte de modo a eliminar possíveis falsos alvos. Após conseguir isolar as embarcações do *background* e ruído, os pixeis da imagem são convertidos para coordenadas geográficas e é calculado o centroide de cada embarcação.

De modo a conseguir efectuar *tracking* de embarcações, foi usado um algoritmo de janela de procura. Este algoritmo consiste em criar uma janela à volta da *frame* actual de tamanho r_1 por r_2, em metros. Esta janela representa a possível posição da embarcação na *frame* seguinte. O tamanho da janela é calculado tendo em consideração uma velocidade máxima da embarcação de $19kts$ e tempo entre *frames* de 5 segundos. Após ter uma janela de possibilidade da *frame* anterior para cada alvo, são comparadas as embarcações observadas pelo radar, da *frame* actual, com as vária janelas. Se existir uma embarcação observada que esteja dentro da janela de procura, assume-se que a embarcação observada é o mesmo alvo da janela. Como é possível ver na figura 2.3, na *frame* t_n é calculada a janela de procura. Na *frame* t_{n+1} são observadas duas embarcações, uma dentro da janela de procura e outra fora. A embarcação que se encontra fora da janela de procura

é descartada e a embarcação que se encontra dentro da janela é reconhecida como a nova posição da embarcação. É criada uma janela na *frame* t_{n+1} e aplicado o algoritmo até deixar de observar medidas do radar.

A vantagem do algoritmo apresentado é ser de fácil implementação e de não necessitar de grande poder computacional. A grande desvantagem deste algoritmo de *tracking* é funcionar bem apenas quando não existem sobreposições de embarcações. Na situação de existir sobreposição, o algoritmo não está preparado para resolver esse tipo de conflito. Os resultados do *tracking* foram comparados com os resultados provenientes de um receptor de AIS, tendo estes demonstrado uma ligeira diferença[9].

Lokukalage et al. [10], apresentam um sistema capaz detectar e efectuar *tracking* de embarcações bem como prever a trajectória das mesmas. O sistema é composto três módulos principais: um módulo que detecta e efectua *tracking*, um que estima a trajectória e um que estabelece a comunicação entre as diferentes embarcações como é possível ver na figura 2.4. O módulo de detecção é responsável por ler informações provenientes de sensores como radar e/ou LIDAR, e com recurso a uma rede neuronal artificial (do inglês *Artificial Neural Network* (ANN)), proceder à detecção, identificação e classificação de múltiplos alvos. O segundo módulo é responsável por estimar a trajectória e os estados (posição, velocidade e aceleração) das diferentes embarcações. Este módulo é alimentado pelos resultados provenientes da ANN e consiste num *Extended Kalman Filter* (EKF). O terceiro módulo é responsável por enviar para todas as embarcações as informações das restantes embarcações.

O módulo de detecção e *tracking* está sub-dividido em quatro diferentes passos[11]: leitura de dados, agregação de dados, classificação de objectos e *tracking* dos mesmos. O processo de leitura de dados consiste em utilizar sensores como radar e/ou LIDAR, de modo a observar um mundo que pode ser estático ou dinâmico. O processo de agregação de dados consiste em utilizar os dados dos sensores e separar em diferentes objectos. O processo de classificação consiste em descrever um objecto da melhor forma possível para que possa ser usado para efectuar *tracking*.

O módulo de estimação de estados e previsão da trajectória é dividido em três secções [12]: desenvolvimento de um modelo de movimento, observação do mundo e *tracking* da trajectória e estimação de estados.

É utilizado um modelo cinemático a duas dimensões que consiga descrever da melhor forma o comportamento de uma embarcação e o tipo de manobras que esta pode efectuar. Geralmente a dinâmica das embarcações é lenta em comparação com veículos terrestres ou aéreos, podendo-se aproximar a uma parábola lenta [10]. É considerado que a embarcação é apenas um ponto no espaço não tendo em conta as dimensões da embarcação. Na figura 2.5, a posição da embarcação é representada por $x(t)$ e $y(t)$, a velocidade por $V_a(t)$ sendo que esta é decomposta em $v_x(t)$ e $v_y(t)$, orientação por $X_a(t)$, aceleração tangencial por $a_t(t)$ e aceleração normal por $a_n(t)$. Um modelo de movimento curvilíneo em tempo

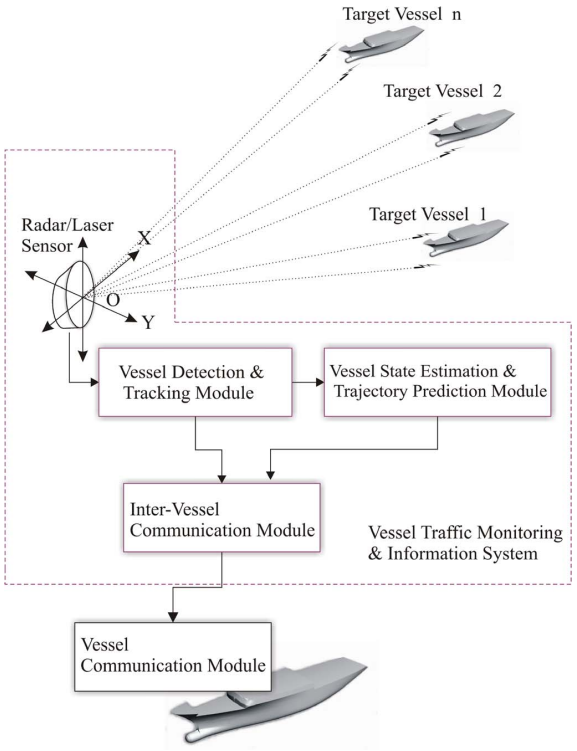


Figura 2.4: Sistema de monitorização de embarcações proposto por *Lokukalage et al.* [10]

contínuo pode ser aproximado por[10]:

$$\begin{aligned} \dot{X}_a(t) &= \tfrac{a_n(t)}{V_a(t)}; & \dot{V}_a(t) &= a_t(t) \\ v_x(t) &= V_a(t) \cdot \sin X_a(t); & v_y(t) &= V_a(t) \cdot \cos X_a(t) \end{aligned} \tag{2.2}$$

O modelo de movimento curvilíneo pode ser aproximado a um modelo de movimento não linear[10]:

$$\dot{x}_x(t) = f(x_x(t)) + w_x(t) \tag{2.3}$$

onde $w_x(t)$ é o ruído branco com valor médio de 0, $x_x(t)$ é o vector de estados e $f(x_x(t))$ a função do sistema e são dados por:

$$x_x(t) = \begin{bmatrix} x(t) \\ v_x(t) \\ y(t) \\ v_y(t) \\ a_t(t) \\ a_n(t) \end{bmatrix}, \quad f(x_x(t)) = \begin{bmatrix} v_x(t) \\ a_t(t) f^{vx} + a_n(t) f^{vy} \\ v_y(t) \\ a_t(t) f^{vy} - a_n(t) f^{vx} \\ 0 \\ 0 \end{bmatrix} \tag{2.4}$$

As componentes f^{vx} e f^{vy} são dadas por:

$$f^{vx} = \frac{v_x(t)}{\sqrt{{v_x}^2(t) + v_y^2(t)}}, \quad f^{vy} = \frac{v_y(t)}{\sqrt{{v_x}^2(t) + v_y^2(t)}} \tag{2.5}$$

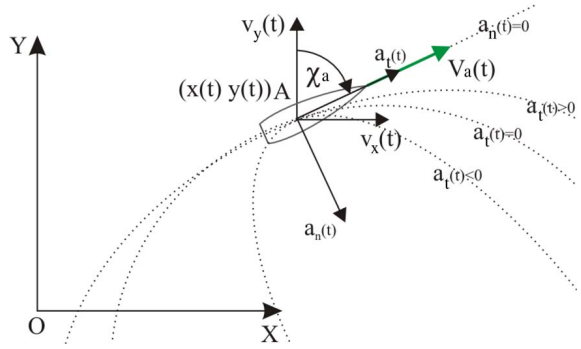


Figura 2.5: Modelo de movimento curvilíneo [10]

Como é considerado que a embarcação é um ponto no espaço, os dados provenientes dos sensores não permitem estimar directamente a orientação da embarcação, não sendo assim um dos estados do sistema [10].

O modelo de observação é formulado em tempo discreto visto que as observações, dos sensores, de embarcações acontecem casualmente. O modelo de observação discreto é dado por:

$$z_z(k) = h(x_x(k)) + w_z(k) \tag{2.6}$$

onde $z_z(k)$ são os estados do sistema, $h(x_x(k))$ é a função de observação e são dados por:

$$\begin{aligned} z_z(k) &= \begin{bmatrix} z_x(k) \\ z_y(k) \end{bmatrix} \\ h(x_x(k)) &= \begin{bmatrix} z_x(k) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & z_y(k) & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{2.7}$$

onde $z_x(k)$ e $z_y(k)$ são os pontos observados das embarcações, $w_z(k)$ ruído branco com valor médio nulo e covariância de $R(k)$ dada por:

$$R(k) = diag[R_x(k) \quad R_y(k)] \tag{2.8}$$

Com um modelo de movimento e de observação é possível aplicar um EKF de modo a estimar o vector de estados e efectuar *tracking* de embarcações. Os resultados apresentados por *Lokukalage et al.* [10], foram obtidos com recurso a simulações de múltiplas embarcações em *Matlab*, tendo obtido resultados positivos.

2.3 Desvio de obstáculos

Uma parte importante quando se fala em navegação autónoma é conseguir efectuar *obstacle avoidance*. Este processo consiste em desviar de possíveis embarcações ou objectos sejam eles dinâmicos ou não, que se encontrem na rota de navegação. Para poder efectuar desvio de obstáculos é necessário conhecer o local, as embarcações ou objectos que nele existam. Para tal, é necessário existir algum tipo de ATR, como exposto na secção 2.2, que permita estimar a posição das embarcações e eventualmente prever a sua trajectória.

Kammer et al. [13] apresentam um método de desvio de obstáculos baseado em detecção de arestas. O algoritmo apresentado consiste em usar um sonar para criar um mapa, procurar arestas de objectos, estimar qual a aresta que leva o robô para o objectivo e manter uma distância de segurança dessa aresta. Este ciclo é repetido até o robô chegar ao seu objectivo. O algoritmo funciona bem na presença de objecto estáticos, contudo com objecto dinâmicos surge a necessidade de criar uma lista de regras. Se o objecto se estiver a mover-se na direcção do robô é verificado se existe alguma rota livre para este recorrer. No caso de um objecto estar a afastar-se do robô é necessário esperar que o objecto saía de rota do robô e depois retomar o seu destino original.

Ao contrário do método apresentado por *Kammer et al.* [13], *Borenstein et al.* [14] apresentam um algoritmo, *Vector Field Histogram* (VFH), baseado em histograma de detecção radial. Este algoritmo consiste, numa primeira fase, em observar o mundo com recurso a sensores e criar uma mapa dos objectos observados. Como é possível ver na figura 2.6, o robô encontra-se presente no meio de um conjunto de objectos. Na figura do meio, está presente um mapa local criado pelo robô, mapa este que se trata de um grelha de tamanho fixo, onde rectângulos pretos representam espaço ocupado e rectângulos brancos espaço livre. É criado um histograma polar em volta da posição do robô com recurso ao mapa anterior.

Na figura 2.6 mais à direita, encontra-se o histograma criado para a posição do robô.

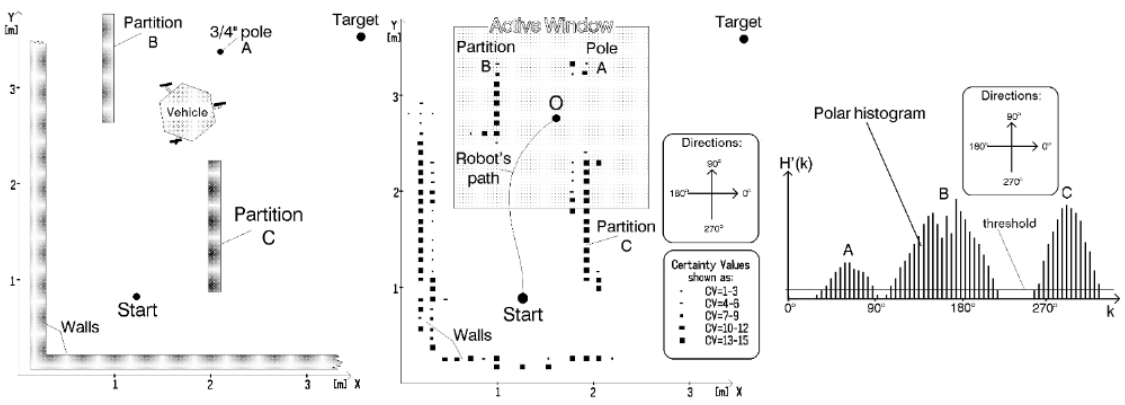


Figura 2.6: Algoritmo VFH apresentado por *Borenstein et al.* [14]

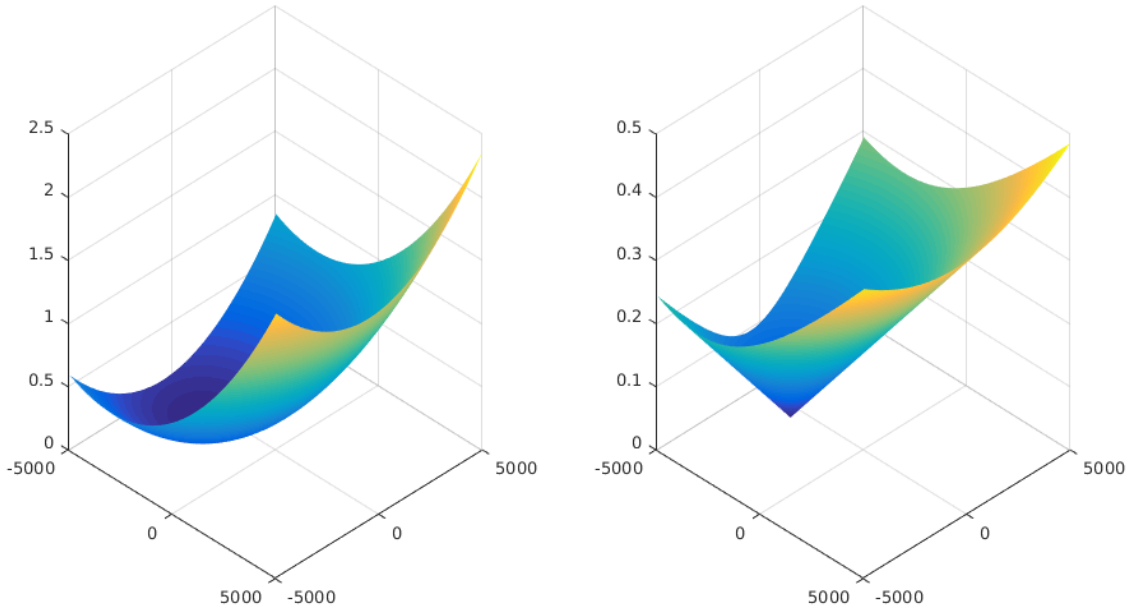


Figura 2.7: Exemplos de mapas de atracção

É possível observar que o histograma é composto por vales que representam possíveis caminhos livres e por montanhas que representam objectos. O critério de selecção para a escolha do caminho a seguir é dado por um *threshold* que é aplicado no histograma. Os vales que ficarem abaixo do *threshold* são as diferentes rotas que o robô poderá seguir. Ao contrário de outro métodos de *obstacle avoidance* que fazem recurso de um *threshold*, onde a escolha do *threshold* é uma decisão importante, no algoritmo VFH não existe a necessidade de escolher um *threshold* perfeito, visto que se o valor de *threshold* mudar um pouco apenas vai aumentar ou diminuir o tamanho do cone de ângulos possíveis. Como o ângulo a seguir é o valor médio dos ângulos presentes no cone, mudar o *threshold* não irá impedir que o robô não consiga seguir o caminho certo[14].

Os *Artifical Potencial Fields* (APF) são um conjunto de algoritmos mais recentes que os apresentados anteriormente[15][16]. Estes métodos consistem, numa primeira fase, em criar uma grelha 3D que é a representação de um campo de potencial de uma dada função de atracção (utilizada para enviar o veículo para o objectivo pretendido). Como é possível ver na figura 2.7, é criado um mapa 3D onde cores mais escuras representam o valor de potencial mais baixo e cores mais claras o valor de potencial mas alto. O mapa é criado em função de um ponto inicial. Após ter o mapa é aplicado um algoritmo que procura o valor de potencial mais baixo nas células adjacentes à célula actual, como por exemplo o algoritmo de *gradient descent*. Na figura 2.7 estão presentes dois tipos de potencias, à esquerda uma função de potencial paraboloidal e à direita cónico. O potencial de atracão

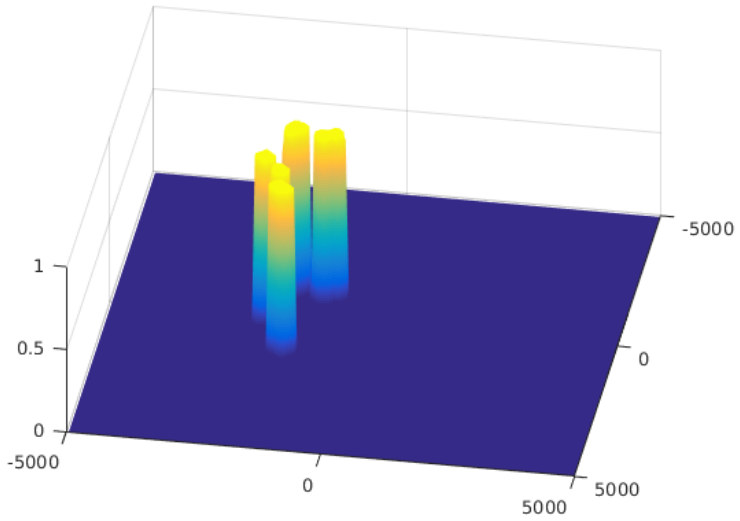


Figura 2.8: Exemplo mapa de repulsão

paraboloidal e cónico são calculados respeitadamente por:

$$\begin{aligned} U_{att}(p) &= \lambda|d-p|^2 \\ U_{att}(p) &= \lambda|d-p| \end{aligned} \tag{2.9}$$

onde d é o um vector $[x\ y]$ com local de destino, p é um vector $[x\ y]$ de cada posição da grelha do mapa e λ é um factor de ganho positivo e constante.

Numa segunda fase é necessário gerar um novo mapa de potencial que representa os obstáculos. A figura 2.8 é o resultado de um mapa de repulsão. Como é possível observar, pontos claros representam objectos, potencial elevado, e pontos escuros representam áreas livres, potencial baixo. O mapa de repulsão é criado com recurso a sensores. O valor do potencial em cada célula da grelha é dado por:

$$U_{rep}(p) = \begin{cases} k(\frac{1}{|p-o|} - \frac{1}{r})^2 & \textbf{se} |p-o| \leq r \\ 0 & \textbf{se} |p-o| > r \end{cases} \tag{2.10}$$

onde k é um factor de ganho positivo e constante, p é uma célula da grelha, o é a posição de um obstáculo e r é o alcance máximo de potencial que um obstáculo pode gerar.

De modo a conseguir executar *obstacle avoidance* é necessário juntar o mapa de atracão e o mapa de repulsão num só. Com um novo mapa criado o algoritmo aplicado consiste em procurar qual a célula adjacente à célula actual que tem o valor de potencial mais baixo, seguir para a nova célula e procurar novamente o valor de potencial mais baixo. O algoritmo acaba quando existir um mínimo local ou global, ou seja, quando o potencial nas células adjacentes for maior do que o potencial na célula actual [15][16]. A grande desvantagem dos APF é na situação do algoritmo encontra um mínimo local, sendo necessário aplicar alguma estratégia para conseguir sair do mínimo local.

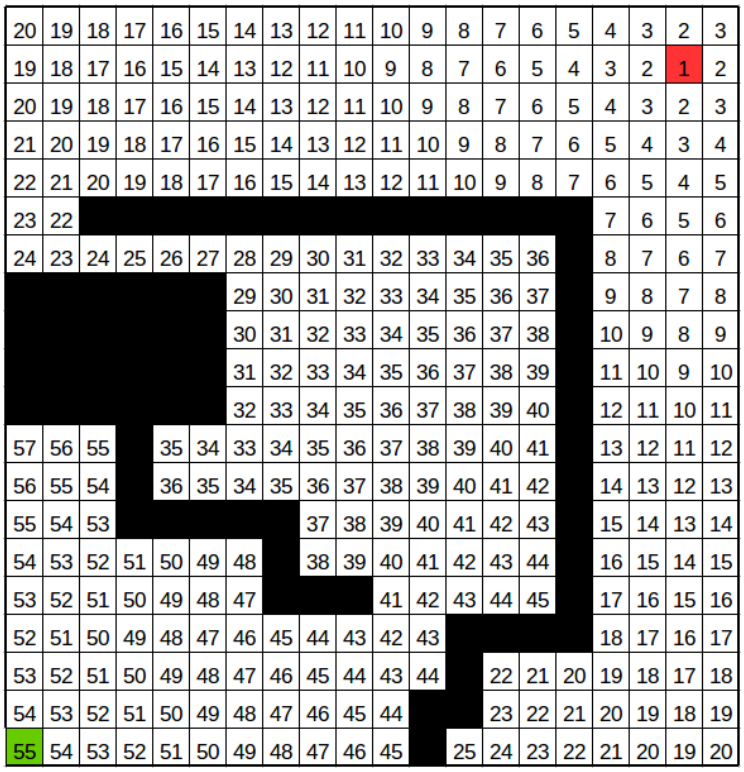


Figura 2.9: Exemplo de um mapa de *Wave Front Expansion*

O *Wave Front Expansion* é um algoritmo similar com os APF e que veio colmatar a desvantagem dos mínimos locais[17]. Este algoritmo consiste em criar uma mapa, do tipo grelha, local onde células pretas representam locais ocupados e células brancas representam espaço liberto, como é possível ver na figura 2.9. Ao contrário dos APF, o *Wave Front Expansion* cria uma mapa de potencial que retrato o custo do trajecto de cada célula até ao alvo. Na figura 2.9, é possível observar que a célula final, a vermelho, possui o valor de potencial mais baixo e a célula inicial, a verde, possui um valor de potencial de 55. Isto significa que, para ir da célula inicial até à final o robô terá de passar por 55 células. O algoritmo de procura da próxima célula consiste em procurar nas quatro células adjacentes à célula actual, a que possui o valor mais baixo e seguir para essa.

A grande vantagem do *Wave Front Expansion* é não ter problemas com mínimos locais. Quando o algoritmo chegar a um valor mínimo de potencial, é garantido que o algoritmo chegou ao valor mínimo global, isto é, chegou ao destino.

Os métodos apresentados anteriormente, APF e *Wave Front Expansion*, estão concebidos para funcionarem num ambiente aonde todos os objectos estão estáticos, isto é, os objectos presentes no mundo têm velocidade nula e posição constante. Quando um robô pretende executar navegação num mundo real, este tem de ter em conta que poderão existir objectos estáticos bem como objectos dinâmicos e ter a capacidade de dinamicamente evitar colisões[18].

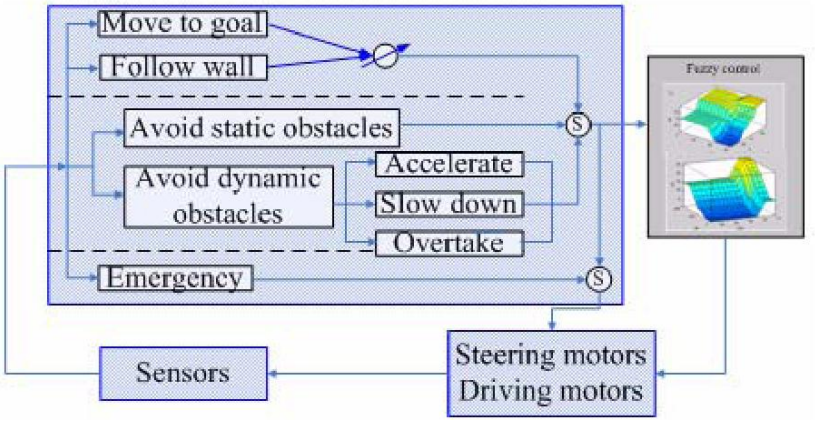


Figura 2.10: Controlador *Fuzzy Logic* para *Obstacle avoidance* apresentado por *Li et al.* [19]

Li et al. [19] apresentam um método de executar *obstacle avoidance* num ambiente com objectos estáticos e dinâmicos com recurso a um controlador *Fuzzy Logic*. A figura 2.10 é o sistema apresentado para *obstacle avoidance*. O sistema tem 5 modos de funcionamento: ir para o objectivo, seguir uma parede, evitar obstáculos estáticos, evitar obstáculos dinâmicos e emergência. De acordo com os dados obtidos pelos sensores um dos modos de funcionamento é seleccionado. O modo de emergência apresenta maior prioridade em relação aos restantes.

Num ambiente estático o controlador *Fuzzy Logic* consiste em duas camadas, uma primeira que, em função das distâncias aos obstáculos à direita, frente e esquerda e a orientação do alvo calcula o ângulo de direcção. Numa segunda camada, em função do ângulo de direcção, da distância ao alvo e da distância ao obstáculo mais próximo é calculada a velocidade a ser aplicada aos motores do robô. Na situação de o robô estar num mundo estático este poderá seguir a parede, ir para o objectivo ou desviar de obstáculos estáticos. Num cenário aonde existam obstáculos dinâmicos, é considerado que os obstáculos têm um movimento linear. Um módulo de processamento prévio, estima a velocidade, orientação e distância aos diferentes obstáculos. Na situação de existir apenas um obstáculo dinâmico que se move na direcção do robô, é considerado que existe a possibilidade de colisão e é estimado o ponto de intersecção da trajectória do obstáculo com o robô, como mostra na figura 2.11. P é o ponto de intersecção das duas trajectórias, d_{rp} a distância do robô ao ponto de intersecção, θ_o o ângulo de direcção do obstáculo, d_{ro} a distância entre o obstáculo e o robô e θ_{ro} o ângulo entre as trajectórias do robô e obstáculo .

O desvio de obstáculos de alvos dinâmicos pode ser separado em três diferentes categorias: o robô passa primeiro o ponto de intersecção, robô espera que o obstáculo passe o ponto de intersecção e robô aplica uma manobra de ultrapassagem ao obstáculo[19]. Na situação em que o robô decide passar primeiro no ponto de intersecção, é necessário que este aumente a sua velocidade e mantenha o ângulo de direcção. Na situação de o

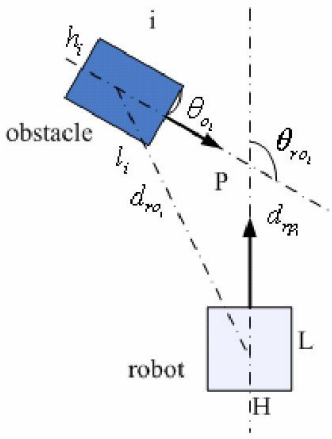


Figura 2.11: Exemplo de colisão de robô com obstáculo[19]

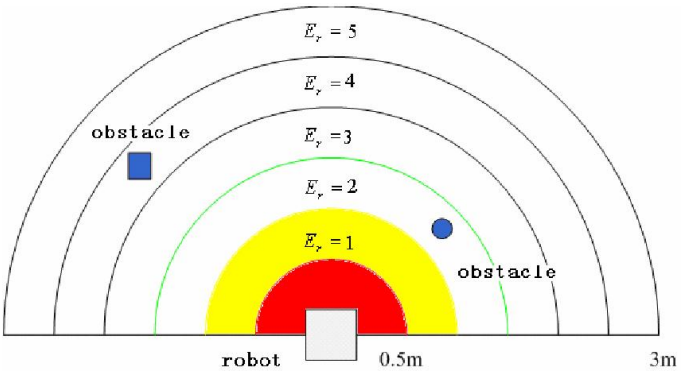


Figura 2.12: Grau de prioridade do robô para obstáculos dinâmicos[19]

obstáculo passar primeiro no ponto de intersecção, é necessário que o robô diminua a sua velocidade mantendo o ângulo de direcção. No último caso tanto a velocidade como o ângulo de direcção vão sofrer modificações ao longo da trajectória. A escolha entre as três diferentes situações é dependente das velocidades do robô e do obstáculo[19].

Na situação de existir ocorrência de mais do que um obstáculo dinâmico, surge a necessidade de estabelecer uma lista de prioridades. Esta lista é construída em função da distância dos obstáculos ao robô tendo em consideração as suas velocidades. A figura 2.12 é um exemplo da distribuição de prioridades para os diferentes obstáculos. O obstáculo da direita tem uma prioridade maior do que o obstáculo que se encontra do lado esquerdo por estar mais próximo do robô. Após o robô calcular qual o obstáculo que tem prioridade maior, este irá escolher entre aumentar a velocidade, diminuir a velocidade ou ultrapassar o obstáculo. Obstáculos com menor prioridade serão atendidos após.

2.4 Navegação através de *colregs*

De modo a poder navegar no mundo de modo seguro é necessário respeitar algumas regras de navegação. Tal como no trânsito terrestre, os condutores sabem que têm prioridade se estiverem à direita de um outro condutor, no mar acontece a mesma coisa[20]. Foi criado um conjunto de regras de navegação de modo a conseguir evitar qualquer tipo de colisão e poder navegar de forma segura no mar. O conjunto de regras de navegação é chamado de *Colreg* que é acrónimo para *International Regulations for Preventing Collisions at Sea*. Existem aproximadamente 40 regras de navegação, das quais metade são para sons e luzes[21]. Das 40 regras de navegação destacam-se as regras 8 e 13-17. Estas são as mais importantes para um sistema de navegação autónomo[20][22][23][24].

A regra 8 rege acções para evitar colisões (*Action to avoid collision*). As principais acções estão abaixo apresentadas:

- Qualquer mudança de orientação ou de velocidade para evitar colisão deve ser grande o suficiente para que as outras embarcações possam visualizar a alteração.
- Manobras de evitar colisão devem ser efectuadas com uma distância de segurança.
- Se necessário, para evitar colisão uma embarcação deve parar ou reverter a direcção de rotação do motor.

A regra 13 rege a ultrapassagem de embarcações (*Overtaking*).

- Para efectuar uma ultrapassagem, a embarcação que o deseja fazer deve mudar a sua orientação em pelo menos 22.5 graus.
- Durante a noite as luzes de sinalização devem ser vistas durante a manobra.
- Na incerteza de uma embarcação achar que está a ser ultrapassada por uma outra, esta deve sempre considerar que está e respeitar as regras necessárias.

A regra 14 rege a situação de duas embarcações estarem a mover-se uma em direcção à outra (*Head-on situation*).

- Se duas embarcações estão a mover-se uma em direcção à outra, as duas devem mudar a sua orientação para estibordo.
- Na incerteza de ter um confronto frente a frente, a embarcação deve sempre considerar que o têm e respeitar as regras necessárias.

A regra 15 rege a situação de duas embarcações terem um ponto na sua rota em que pode existir colisão (*Crossing section*)

- Na situação de cruzamento de duas embarcações, aquela que possui a outra a estibordo deve permitir a passagem desta em primeiro lugar pelo local de cruzamento.

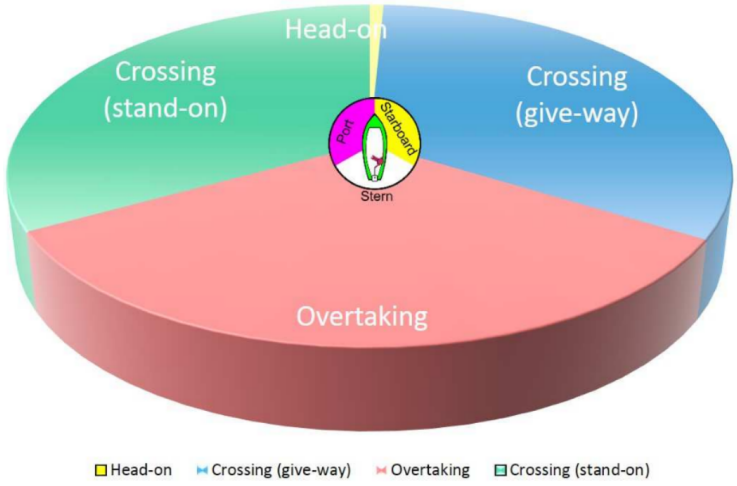


Figura 2.13: Diferentes regiões para *Colregs*[22]

A regra 16 rege a situação de uma embarcação ter de permitir que outra tenha prioridade (*Action by give-way vessel*)

- Uma embarcação que tenha de ceder prioridade deve tomar a medida correcta com tempo suficiente para não provocar situações preocupantes.

A regra 17 rege a situação de uma embarcação ter prioridade no cruzamento (*Action by stand-on vessel*).

- No caso de uma embarcação ter prioridade no cruzamento, esta deve manter a sua orientação e velocidade.
- No caso de uma embarcação ter prioridade no cruzamento de duas embarcações e se encontrar demasiado próxima da outra embarcação, esta deve tomar medidas para evitar colisões.

Hu et al. [22] apresentam um sistema de planeamento de trajectória para um ASV com compatibilidade com *Colregs*. O processo de planeamento de trajectória apresentado é composto por dois módulos: planeador global e um local. O módulo de planeamento global é responsável por criar um conjunto de *waypoints*, conjunto de pontos a seguir, que levam o ASV do ponto inicial até ao ponto final. Este processo acontece uma vez para cada missão sendo assim considerado um processo *offline*. O módulo de planeamento local é um módulo que apenas funciona quando um obstáculo aparece entre dois *waypoints*. Este módulo é composto por 3 processos diferentes: avaliação de risco, tomada de decisão e escolha de regra e planeamento da nova trajectória.

De forma a avaliar o risco de colisão é utilizado o método de *Closest Point Approach* (CPA). Este método consiste em comparar o tempo e distância para o alvo com dois parâmetros que são dependentes do tipo de embarcação e do local aonde se encontram

(t_{max}(TCPA) d_{min}(DCPA))[22]:.

$$0 \leqslant TCPA \leqslant t_{max} \quad e \quad 0 \leqslant DCPA \leqslant d_{min} \tag{2.11}$$

Se as condições anteriores forem verificadas, então a variável*risk* é colocada a 1. Isto significa que existe um obstáculo no caminho da embarcação. Uma segunda condição é criada, sendo esta o limite de segurança da embarcação.

$$0 \leqslant TCPA \leqslant t_{safe} \quad e \quad 0 \leqslant DCPA \leqslant d_{safe} \tag{2.12}$$

Em situações normais, $t_{safe} < t_{max}$ e $d_{safe} < d_{min}$. Se as condições presentes na equação anterior se cumprirem, então a variável *risk* é colocada a 2. Quanto maior o valor da variável, maior o risco de colisão. Em função da TCPA e DCPA é verificado se o obstáculo está a cumprir as *Colregs*.

Após conseguir verificar se existe risco de colisão é necessário determinar qual a *Colreg* a seguir. A figura 2.13 é o mapa das diferentes zonas das *Colregs*. Se a decisão tomada pelo segundo módulo for *Head-on*, então o ASV vai manter a sua rota de navegação. Caso a decisão seja *give-way* surge a necessidade de recalcular um nova trajectória.

Uma trajectória para evitar colisão pode ser gerada com um ou mais *waypoints*. O terceiro módulo é responsável por gerar trajectórias de forma segura e para tal tem de respeitar 3 regras[21]:

- A mudança mínima de orientação é de 15 °;
- A mudança máxima de orientação não deve ser superior a 60 °;
- Manobras para estibordo são preferenciais do que a bombordo.

Nem todas as condições acima apresentadas são implicadas pelas *Colregs*. A mudança máxima de orientação ser 60 ° é devido a eficiência. Alterações de rota superiores tornam a navegação ineficiente.

A figura 2.14 é o fluxo de decisão apresentado por *Hu et al.* [22] para efectuar planeamento global e local.

Pinto *et al.* [23] apresentam um sistema de navegação autónoma. Este sistema é composto por duas partes: uma primeira responsável por gerar uma trajectória tendo em consideração a posição inicial e posição e orientação finais e uma segunda responsável por evitar colisões tendo em conta as regras de navegação *Colregs*. Em situações normais, apenas a primeira parte acontece. Na iminência de colisão a segunda parte actua sobre o ASV.

O trabalho apresentado foi efectuado como recurso a dois ASV com forma aproximadamente rectangular, ZARCO e GAMA, que possuem comunicação entre eles, possibilitando assim um ASV saber o estado (posição, velocidade e orientação) do outro ASV. A figura 2.15 é a arquitectura de navegação apresentada. Os dois módulos principais de controlo

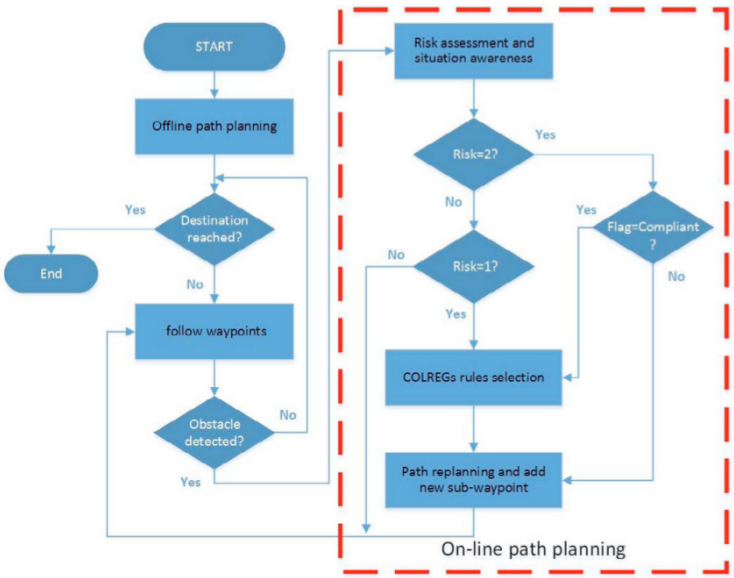


Figura 2.14: Fluxograma do sistema de planeamento de trajectória[22]

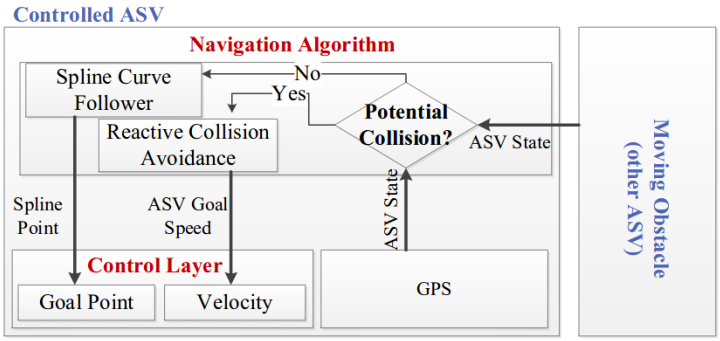


Figura 2.15: Arquitectura apresentada por Pinto *et al.* [23]

encontra-se presentes dentro da secção *Navigation Algorithm* (*Spline Curve Follower* e *Reactive Collision Avoindace*). Cada um destes módulos actua na camada de controlo do ASV de modo diferente. O módulo de *Spline Curve Follower* é responsável por gerar um conjunto de *waypoints*, actuando assim na posição desejada para o ASV na camada de controlo. O módulo de *Reactive Collision Avoidance* é o módulo que entra em funcionamento na iminência de colisão, tendo a necessidade de ter acesso directo à velocidade do barco. Este módulo tem em consideração as regras de navegação *Colregs* (8 e 14) bem como a velocidade relativa ao obstáculo e a possível rota de fuga.

O primeiro passo do módulo *Reactive Collision Avoidance* é definir um área que limite a navegação. Para tal é criada área de colisão, local onde existe uma alta probabilidade de colisão, que depende do tamanho do obstáculo. Esta área é criada com 3 vezes a diagonal maior. Na figura 2.16, esta área está representada a preto, *Collision Area - CA*. O grande objectivo é impedir que o centro geométrico do ASV entre nesta área. Em cada iteração

do sistema, o ASV sabe o estado do obstáculo:

$$Estado = [x_{MO} \quad y_{MO} \quad \Phi_{MO} \quad vx_{MO} \quad vy_{MO}] \tag{2.13}$$

Tendo o estado do obstáculo e o estado do ASV é possível calcular um vector de velocidade e um cone de movimento(*Moving Obstacle Cone*). Um segundo cone é criado com um factor de incerteza como mostra a figura 2.16. A ideia do módulo é colocar o vector de direcção fora do cone de movimento com incerteza. Para tal é criada uma rota de fuga, ET, com velocidade frontal e rotacional máximas.

$$W_{ET} = -W_{max}, V_{ET} = V_{max} \tag{2.14}$$

A velocidade de rotação é máxima, porém negativa. Deste modo o ASV irá rodar para o seu estibordo respeitando assim as regras de navegação. O raio de rota de fuga é dado por:

$$r_{ET} = \frac{V_{ET}}{W_{ET}} + \frac{diag}{2} \tag{2.15}$$

onde *diag* é a maior diagonal do obstáculo[23].

Em função da velocidade relativa e da rota de fuga é possível criar um rota de fuga tendo em conta a velocidade relativa. Na figura 2.17 estão representadas as duas rotas de fuga. É possível definir duas condições para potencial colisão: o vector velocidade estar dentro do cone de movimento com incerteza(MOCU) e rota de colisão relativa(RET) coincidir com a área de colisão do obstáculo(CA).

A implementação apresentada permitiu concluir que os ASV conseguem seguir o controlo *spline* e quando surge algum obstáculo trocam de modo de funcionamento para *Reactive Collision Avoidance*. O trabalho apresentado foi testado com dois ASV num ambiente controlador e utilizando apenas as regras de navegação 8 e 14. Num mundo real, todas as regras mencionadas anteriormente devem ser incluídas num sistema de navegação autónomo.

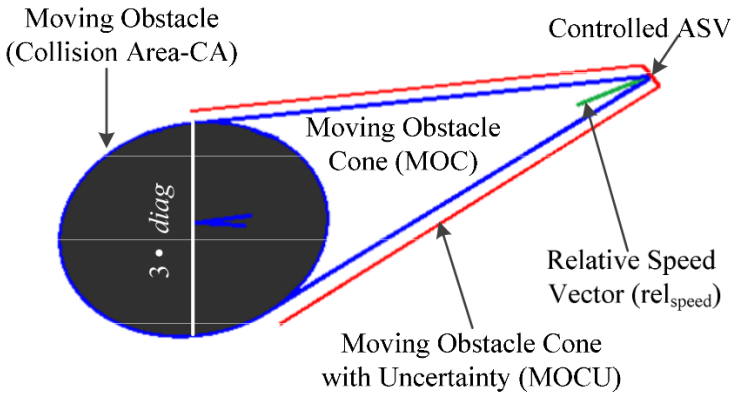


Figura 2.16: Área de colisão [23]

Benjamin et al. [24] apresentam uma solução em tudo semelhante à anterior. O sistema de planeamento consiste em dois módulos: seguir *waypoints* e evitar obstáculos. O primeiro módulo consiste em utilizar as coordenadas GPS para saber a posição exacta do ASV e seguir um conjunto de *waypoints*. A geração de *waypoints* pode ser efectuada por um planeador global de trajectória (*offline*) ou pelo utilizador. O módulo que permite ao ASV evitar obstáculos depende da posição, orientação e velocidade do próprio ASV bem como do obstáculo em causa. O primeiro passo no módulo de evitar obstáculos é conseguir estimar o CPA. Para tal a seguinte equação é usada[24]:

$$CPA(\theta, v, t) = \sqrt{k_2 * t_{min}^2 + k_1 * t_{min} + k_0} \tag{2.16}$$

com:

$$\begin{aligned} k_2 &= \cos\theta^2 v^2 - 2\cos\theta v \cos\theta_b v_b + \cos\theta_b{}^2 v_b^2 \\ &\quad + \sin\theta^2 v^2 - 2\sin\theta v \sin\theta_b v_b + \sin\theta_b{}^2 v_b^2 \end{aligned} \tag{2.17}$$

$$\begin{aligned} k_1 &= 2\cos\theta v y - 2\cos\theta v y_b - 2y\cos\theta_b v_b \\ &\quad + 2\cos\theta_b v_b y_b + 2\sin\theta v x - 2\sin\theta v x_b \\ &\quad - 2x\sin\theta_b v_b + 2\sin\theta_b v_b x_b \end{aligned} \tag{2.18}$$

$$k_0 = y^2 - 2yy_b + y_b^2 + x^2 - 2xx_b + x_b^2 \tag{2.19}$$

onde (x, y, θ, v) é o vector de estados do ASV $(x_b, y_b, \theta_b, v_b)$ é o vector de estados do obstáculo em causa. t_{min} é o tempo quando acontece a menor distância entre as duas embarcações. O valor deste tempo pode ser inferior a 0, isto significa que as embarcações estão a aumentar a distância entre si. Após averiguar se a embarcação apresenta perigo, é necessário perceber qual a regra de navegação a utilizar. Se o ângulo entre o ASV e a embarcação for inferior a 15° é considerado que se trata de uma situação de *Head-on* (regra 14) e são favorecidas manobras efectuadas a estibordo. Se o ângulo for superior a 15° e inferior a 90°, é considerado que se encontram na situação de *Crossing* (regra 15). Nesta situação é utilizada a função de CPA para verificar quem, mantendo a trajectória,

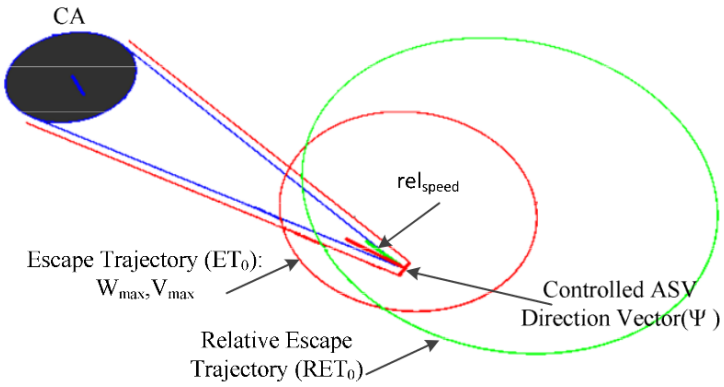


Figura 2.17: Rota de fuga [23]

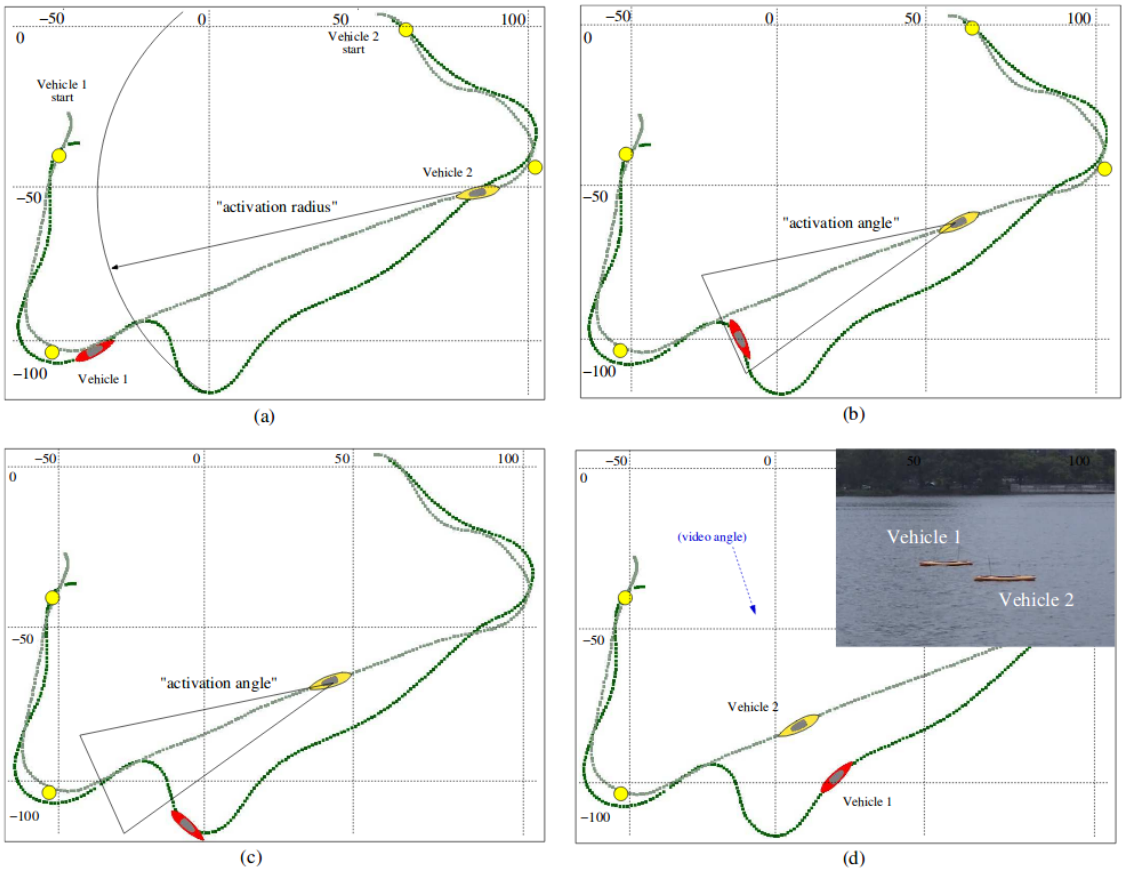


Figura 2.18: Situação de *Head-On*[24]

passa primeiro no cruzamento das duas trajectórias. São penalizadas manobras onde o ASV passa primeiro no cruzamento.

Foram efectuadas testes com recurso a dois ASV equipados com sistema de localização e comunicação. Em qualquer instante de tempo qualquer ASV sabe da posição do outro ASV. Foram testadas as duas situações acima descritas, *Head-On* e *Crossing*. A figura 2.18 é o resultado da situação de *Head-On*. Foi programado um conjunto de *waypoints* para os dois ASV. A vermelho encontra-se o ASV que possuí os dois módulos de funcionamento activados, seguir *waypoints* e evitar obstáculos com recurso à regra de navegação 14. O ASV a amarelo apenas possuí o módulo de seguidor de *waypoints* activo. Na figura a), os dois ASV encontra-se na situação de *Head-On*. Na figura b) o veículo a vermelho percebe que está numa rota de colisão e aplicar a regra de virar a estibordo para evitar colisão. Como o ASV a amarelo apenas segue *waypoints*, este continua na sua trajectória. Na figura c), o veículo a vermelho sai da área de perigo de colisão (15°) do veículo amarelo. A partir deste momento, a navegação segundo a regra 14 torna-se inactiva. Na figura d), o ASV a vermelho tenta volta à trajectória inicialmente proposta.

A situação de *crossing* foi também validade e está apresentada na figura 2.19. Nesta

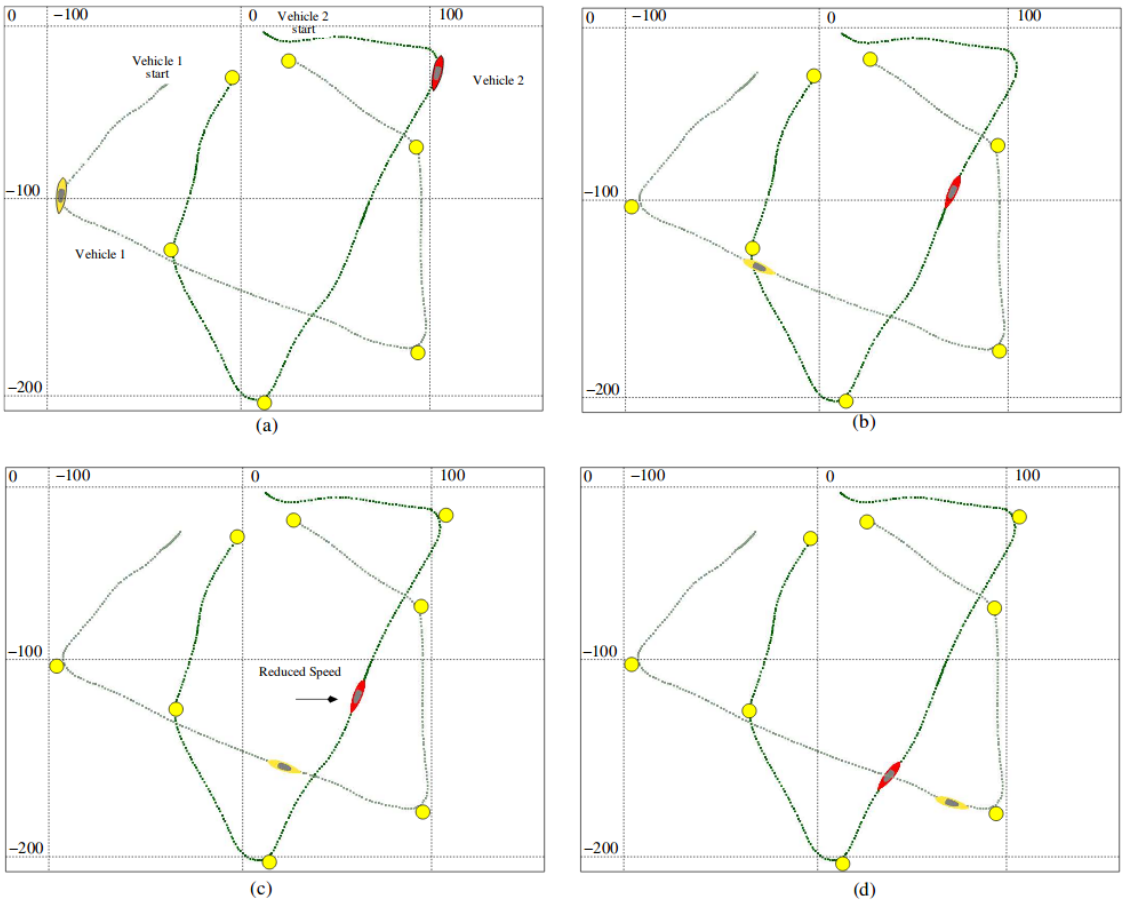


Figura 2.19: Situação de *Crossing*[24]

situação, o ASV a amarelo apenas está a usar o módulo de seguir *waypoints*, enquanto que o ASV a vermelho tem os dois modulos de funcionamento activos. Na figura a), os dois ASV seguem os *waypoints* de forma normal. Na figura b) o ASV a vermelho detecta que tem o outro ASV a um ângulo superior a 15° e inferior a 90°. Este usa a regra de navegação 15 para tomar uma decisão. Como tal a manobra a ser efectuada é reduzir a velocidade para permitir o ASV que está a seu estibordo passar primeiro. A figura c) mostra o ASV a reduzir a velocidade para ceder prioridade no cruzamento. O ponto onde as duas embarcações se cruzam é dado pelo CPA. Na figua d), o ASV a amarelo já passou o local de cruzamento, isto é, já saiu da área de possível colisão e por isso o ASV a vermelho aumenta a sua velocidade para retornar a sua trajectória.

Quando surge a necessidade de implementar um sistema de navegação autónomo com recurso às regras de navegação num mundo real, muitas das vezes não existe apenas uma embarcação para evitar colisão. Numa situação real, poderão existir múltiplas embarcações que demonstram perigo de colisão. A escolhe da regra mais adequada para uma situação deste tipo é um problema de difícil resolução. Quando é uma embarcação manobrada por um ser humano, este consegue tomar decisões que façam sentido segundo as regas de

navegação. Para um computador avaliar o perigo e escolher a melhor rota de fuga é um problema bastante complexo.

Capítulo 3

Arquitectura do sistema

Neste capítulo é apresentado o veiculo de navegação autónomo ROAZ II descrevendo quais os sensores existentes no mesmo. Numa segunda parte é feita a apresentação da arquitectura da solução proposta para o problema de desvio de obstáculos.

3.1 ASV ROAZ II

Como mencionado anteriormente, capítulo 1, o projecto desenvolvido permite que a embarcação autónoma ROAZ II consiga estimar a posição de outras embarcações ou objectos e consequentemente se desvie de objectos estáticos ou dinâmicos que se encontrem no caminho do mesmo. O ROAZ II, figura 3.1, é uma embarcação que possui dois cascos classificando-se assim como um catamarã. É dotado de um conjunto de sensores que permitem efectuar uma diversidade de missões. Como é possível observar na figura 3.1, o ROAZ II possui duas caixas, nas laterais da embarcação, onde estão localizadas as baterias com uma autonomia estimada de 10 horas conseguindo assim fazer entre 60 a 100 Km. Como meio de propulsão, o ROAZ II faz uso de dois motores eléctricos conseguindo atingir uma velocidade máxima de $5m/s$. Dado a natureza da embarcação, os flutuadores permitem ter um carga extra máxima de $300Kg$ com a possibilidade de conexão com o ROAZ II pelas interfaces Ethernet, RS232/RS485 e CAN Bus.

Localizado no meio do veículo, encontra-se presente uma terceira caixa com o computadores central do ROAZ II, interfaces de comunicação e controladores. Nesta caixa estão conectados os sensores necessários para cada missão. Na plataforma superior encontram-se presentes os sensores Radar FMCW, Câmera Termográfica, antena de GPS e LIDAR como é possível ver na figura 3.1. O ROAZ II possui um conjunto de modos de operação: manual, teleoperado, seguidor de pontos e missões autónomas adaptativas. No modo de operação manual, um piloto com experiência vai a bordo da embarcação. Nos restantes modos de operação, a embarcação é controlada via estação terrestre com recurso a uma estação de trabalho. A comunicação entre ROAZ II e estação de trabalho é feita por

Figura 3.1: ASV ROAZ II

Wi-Fi.

3.2 Sistema proposto

3.2.1 Sistema de detecção de obstáculos multi sensorial

A solução abstracta para o sistema de navegação para um ASV faz uso de todos os sensores possíveis de modo a aproveitar as vantagens de cada um dos sensores. A figura 3.2 é um esquema do *pipeline* ideal para navegação autónoma. Este faz uso do radar, como sensor principal, que é dos sensores mais usados para navegação. Como mencionado no capítulo 1, este possuí a desvantagem de não conseguir ver objectos pequenos. De modo a conseguir ultrapassar esta falha, seria usado uma Câmera e um LIDAR. De modo a complementar as medições do radar e obter uma medida exacta da posição das embarcações próximas, poderia ser usado um receptor AIS. Por último, para obter a posição exacta do ROAZ II seria usado um GPS e um IMU.

Como é possível ver na figura 3.2, os sensores Radar, LIDAR e Câmera, possuem um módulo de processamento. Este módulo consiste numa primeira fase em estabelecer a comunicação com o respectivo sensor, colocar os dados dos sensores num referencial global e por fim aplicar algum algoritmo de agregação de dados, isto é, conseguir separar os dados lidos dos sensores em diferentes objectos. A agregação em diferentes objectos é aplicada aos dados do radar e LIDAR, a cada rotação dos mesmos e no caso da camêra a uma *frame*. O resultado dos módulos de processamento é uma lista de objectos no referencial global, onde cada objecto é um conjunto de pontos.

O módulo de processamento do receptor AIS é responsável por estabelecer a comunicação com o mesmo. Como os dados do receptor AIS já se encontram no referencial

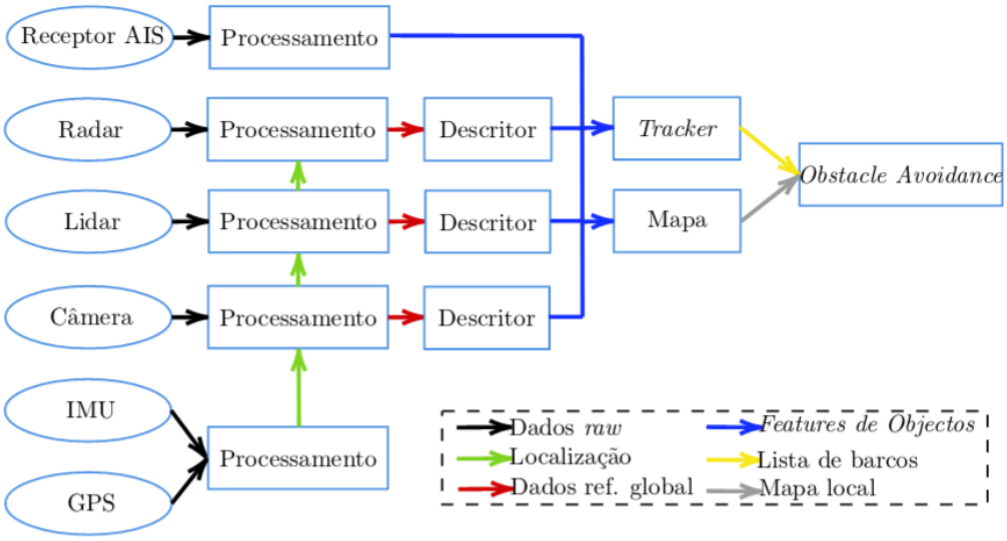


Figura 3.2: *Pipeline* ideal para sistema de navegação autónomo

global não existe a necessidade de aplicar uma mudança de referencial. O módulo de processamento do IMU e GPS consiste num filtro de *Kalman*. O resultado deste módulo é a posição e orientação exactas do ROAZ II.

Um segundo módulo, descritor, é aplicado aos resultados obtidos dos módulos de processamento dos sensores radar, LIDAR e Câmera. Este módulo recebe uma lista de diferentes objectos, em que cada objecto é um conjunto de pontos. O objectivo principal deste módulo é conseguir descrever um objecto de forma compacta e agrupando as suas características mais relevantes, isto é, com um conjunto de pontos é possível extrair algumas informações tais como:

- Centroide
- *Bounding Box*
- Orientação
- Elipse

O resultado final deste módulo é uma lista de características que define um determinado alvo. Estas características são usadas no módulo *tracker* para efectuar *tracking* de diferentes alvos. Este módulo é responsável por receber uma lista de características e estimar os estados do sistema bem como prever a rota de cada objecto. O módulo mapa fica responsável por receber uma lista de características de um conjunto de objectos e tentar perceber quais poderão ser terra. Este módulo poderá fazer uso das cartas de navegação náuticas correlacionando com os dados recebidos permitindo assim eliminar falsos positivos. Fazendo uma pequena análise, são descartadas as embarcações e é construído

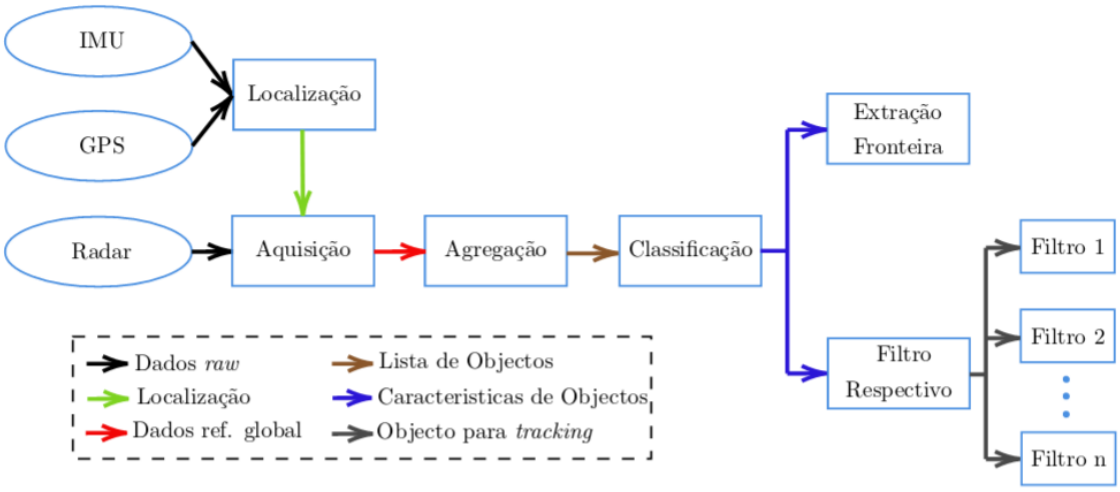


Figura 3.3: *Pipeline* do sistema implementado

um mapa, mapa este que é o limite de navegação do ROAZ II, isto é, é criada a linha de fronteira entre terra e mar.

De modo a que o ROAZ II consiga desviar-se de obstáculos estáticos ou dinâmicos existe a necessidade de ter um módulo que aplique algum tipo de *obstacle avoidance*, que receba informações das posições de outras embarcações nas proximidades bem como a fronteira de navegação.

3.2.2 Solução Implementada

Devido às limitações temporais para o trabalho realizado, restrições de ordem técnica e operacional (como por exemplo disponibilidade de sensores e recursos computacionais) não foi implementada na totalidade a solução apresentada.

A solução implementada consiste em usar apenas o radar como sensor para detecção de objectos, IMU e GPS para localização do ROAZ II e numa fase inicial o receptor de AIS como meio de comparação dos dados obtidos do radar. A figura 3.3 apresenta o *pipeline* de funcionamento proposto para este trabalho. Numa primeira camada existem dois módulos que estabelecem a comunicação com os sensores, um que faz a comunicação e aquisição dos dados do radar e um segundo que adquire os dados do IMU e GPS de modo a obter uma localização e orientação exactas do ROAZ II. O resultado do módulo de aquisição de dados do radar é um conjunto de pontos no referencial global. Após aquisição é necessário agregar os pontos em diferentes objectos para posterior classificação dos mesmos. Dos objectos classificados, surge a necessidade de separar objectos em duas categorias: objectos para efectuar *tracking* e objectos que pertencem a linha costeira. Estes segundos, são usados para construir um mapa que é o limite de navegação do ROAZ II. O módulo de Filtro Respectivo é responsável por receber a descrição de um determinado

objecto e perceber se as características do objecto correspondem a algum dos objectos anteriormente observados. No caso do objecto observado corresponder, este irá para o filtro respectivo. No caso de não corresponder a nenhum filtro, é criado um novo filtro e inicializado com as características do objecto observado. O filtro é responsável por estimar os estados do sistema bem como uma possível trajectória para as diferentes embarcações. Os estados do sistema são:

- Localização
- Velocidade
- Orientação
- Dimensão

De modo a poder efectuar *obstacle avoidance*, é necessário saber os estados de todos os objectos nas proximidades bem como o limite de navegação do ROAZ II.

O *software* presente no ROAZ II faz uso da *framework Robot Operating System* (ROS)[25]. O ROS apesar do nome, não se trata de um sistema operativo. O ROS é uma *framework* que tem uma abstracção de *hardware*, permite ter controlo de dispositivos de baixo nível e estabelecer comunicação entre processos através de mensagens. O sistema apresentado anteriormente é desenvolvido em ROS para uma maior facilidade de integração com os sistemas já existentes no ROAZ II.

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Capítulo 4

Aquisição de Dados do Radar

Neste capítulo é feita uma análise ao funcionamento da tecnologia radar, é descrito o nó de ROS que estabelece a comunicação com o radar e por último são apresentados resultados práticos da comunicação com o radar em ROS.

4.1 Funcionamento da tecnologia Radar

Durante a 2ª Guerra Mundial a Marinha Norte Americana utilizou a definição Radar como acrónimo para *Radio Detection and Ranging*. O Radar é um equipamento que permite detectar a distância, velocidade, coordenadas angulares e o grau de reflectividade de objectos[26].

Genericamente, o radar possui uma antena que está conectada a um duplexer, equipamento que permite usar a antena para transmitir e receber ondas electromagnéticas. Numa primeira fase, a antena é usada para transmissão, é gerada uma onda electromagnética que é propagada para a atmosfera à velocidade da luz (aproximadamente 3 x 10^8 m/s [27]), onda esta que é direccional. Após emissão, o duplexer é colocado no modo de recepção.

Se existir um objecto na direcção da onda electromagnética, esta irá ser dispersada para todas as direcções e uma pequena parte irá voltar na direcção do radar, sendo este fenómeno conhecido com *backscattering*. O tempo entre transmissão e recepção da onda electromagnética é contado e é possível calcular a distância ao objecto pela relação de

$$T = \frac{distância}{velocidade} = \frac{2 \cdot R}{c} \tag{4.1}$$

onde c é a velocidade da luz e R a distância ao objecto. A figura 4.1 é um diagrama de blocos representativo do funcionamento do radar e dos diferentes módulos existentes.

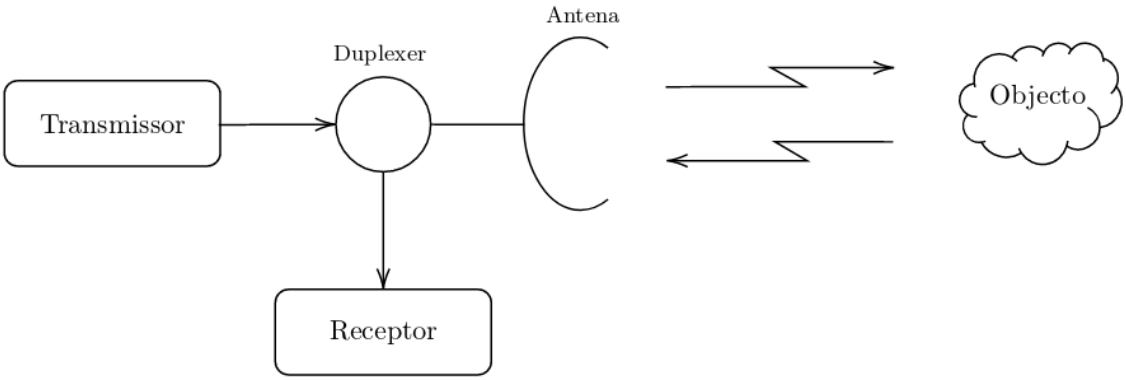


Figura 4.1: Principio de Funcionamento do radar

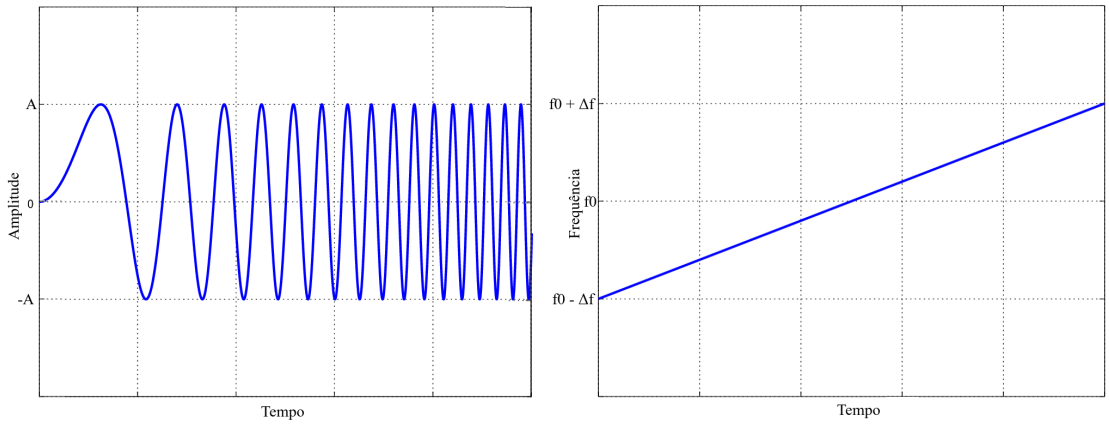


Figura 4.2: Sinal *chirp* ascendente

Existe um vasto conjunto de configurações possíveis de radar como por exemplo[26]:

- Radar de Onda Contínua (CW);
- Radar de efeito de Doppler;
- Radar de Frequência Modulada e Onda Contínua;
- Radar de Abertura Sintética (SAR).

Nem todas as configurações de radar usam apenas uma antena e um duplexer. O radar FMCW é um radar que usa uma antena para transmissão da onda electromagnética para o meio e usa uma segunda para recepção das ondas de *backscattering*. Os radares FMCW transmitem uma onda de frequência variável ao longo do tempo e observam a onda recebida. Efectuando a correlação de frequências entre a onda transmitida e recebida é possível estimar a distância ao objecto.

Num radar FMCW existem diferentes tipos de modulações possíveis, como modulação linear, modulação sinusoidal entre outros. Num radar com modulação linear, a frequência

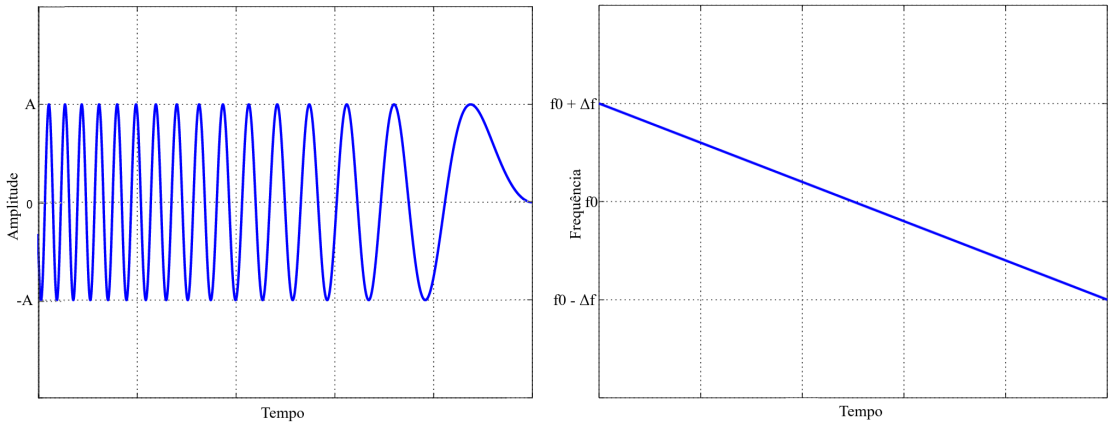


Figura 4.3: Sinal *chirp* descendente

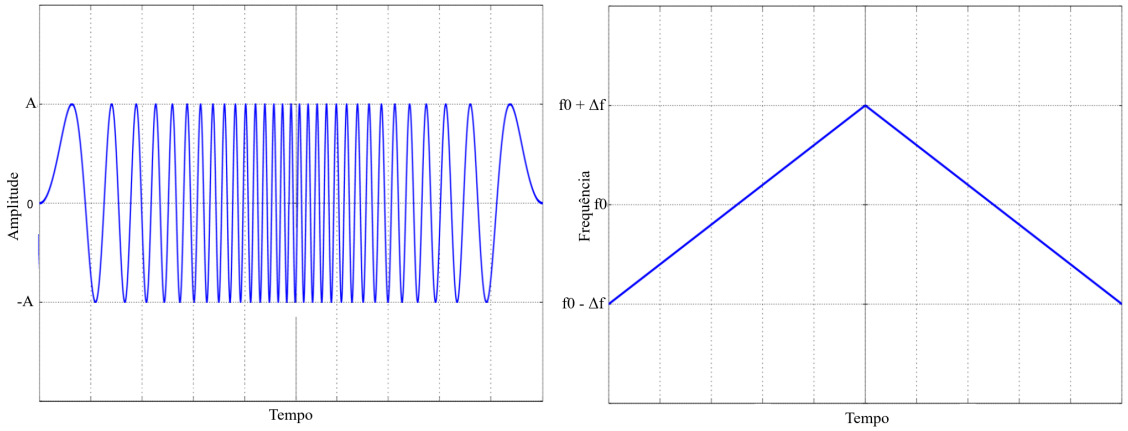


Figura 4.4: Sinal *chirp* usado para modulação linear de frequência

de transmissão muda linearmente com o tempo. A modulação linear pode ser aproximada a um sinal *chirp*. A figura 4.2 é um sinal *chirp* com um valor médio de frequência $f0$ e variação máxima de Δf. O sinal começa com uma valor de frequência de $f0 - \Delta f$ e vai até $f0 + \Delta f$, tratando-se assim de um *chirp* ascendente. A figura 4.3 é um sinal *chirp* com as mesmas condições do sinal anterior porém é um *chirp* descendente. Os radares de FMCW com modulação linear fazem uso dos dois sinais anteriores. Metade de tempo usam um *chirp* em que a frequência vai aumentado ao longo do tempo e na outra metade do tempo usam o *chirp* em que a frequência diminui ao longo do tempo.

A figura 4.4 é o sinal obtido com os dois *chirps*. O sinal obtido possui um valor médio de frequência de $f0$ e um desvio máximo de frequência de Δf. Os radares de FMCW de modulação linear transmitem uma onda semelhante à da figura 4.4.

Fazendo a correlação das frequências do sinal transmitido e recebido é possível extrair a distância ao objecto bem com a sua velocidade radial[26]. A figura 4.5 mostra três sinais, a azul o sinal transmitido pelo radar, a vermelho o sinal recebido e a verde a diferença dos dois sinais. Comparando os sinais transmitido e recebido, é possível obter dois valores:

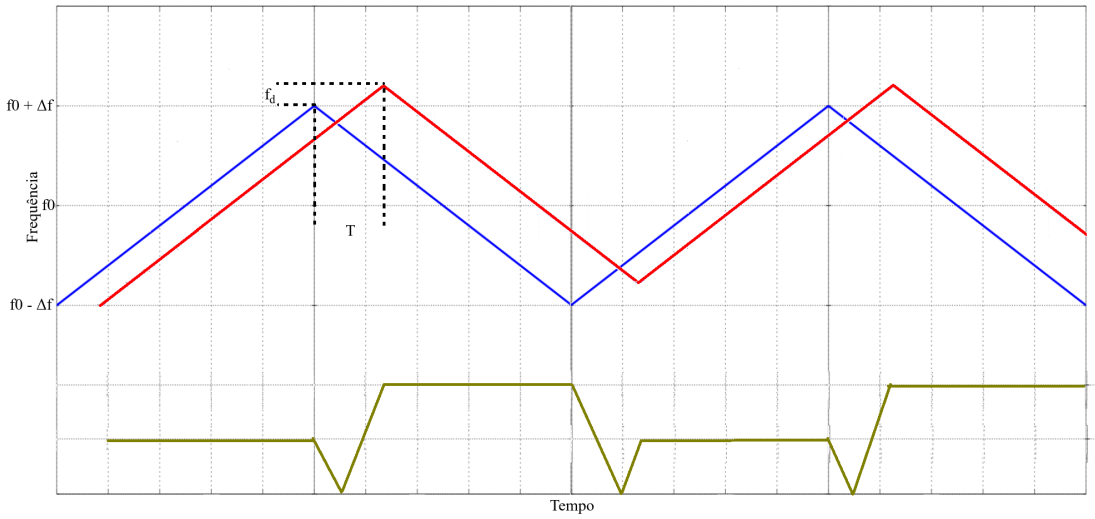


Figura 4.5: Diferença de frequência entre sinal transmitido e recebido

um que é a diferença temporal entre picos dos sinais, T, e um segundo que é um *offset* de frequência no sinal recebido, f_d. Este segundo é resultado do efeito de *Doppler* no objecto em causa. O efeito de *Doppler* deve-se objecto estar em movimento, criando assim uma pequena variação da frequência no sinal que sofre *backscattering*[27].

A diferença instantânea de frequências quando os dois sinais têm inclinação positiva é dado por[26]:

$$f_b^+ = \frac{8 \cdot \Delta f \cdot f_m}{c} - f_d \tag{4.2}$$

onde Δf é a variação máxima de frequência, R a distância ao objecto, f_d a frequência induzida pelo efeito de *Doppler*, c a velocidade da luz e f_m a taxa de variação de frequência.

Quando os dois sinais têm uma inclinação negativa, a diferença instantânea de frequência é dado por:

$$f_b^- = \frac{8 \cdot \Delta f \cdot f_m}{c} + f_d \tag{4.3}$$

Assim a distância ao objecto é dada por:

$$R = \frac{c}{8 \cdot \Delta f \cdot f_m} < f_b > \tag{4.4}$$

onde $< f_b >$ é o valor médio da diferença de frequência. A velocidade radial do alvo é dado por:

$$v = \frac{\lambda}{4}(f_b^- - f_b^+) \tag{4.5}$$

De modo a calcular a distância e velocidade radial do objecto é necessário apenas determinar a diferença instantânea de frequências entre o sinal transmitido e o sinal recebido. Na situação do objecto estar estático, não irá surgir efeito de *Doppler* e o sinal recebido não vai ter o *offset* f_d. Nesse caso, pelas equações 4.2 e 4.3, o valor de f_b^- e f_b^+ serão iguais e consequentemente pela equação 4.5 a velocidade radial irá ser 0.

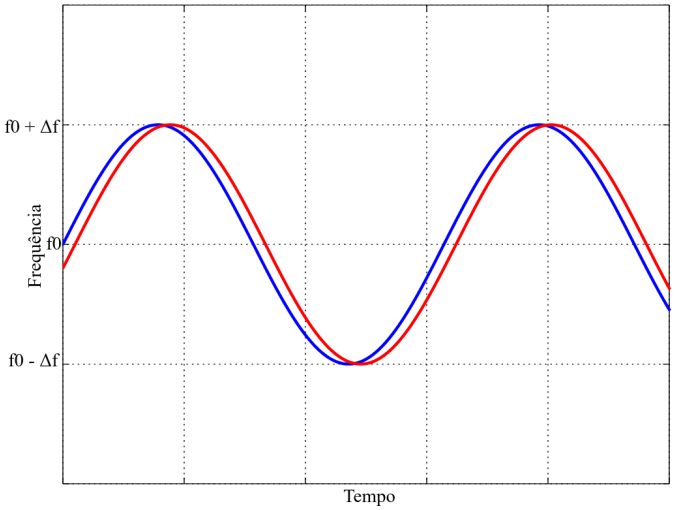


Figura 4.6: Modulação sinusoidal

Um dos métodos com maior facilidade de implementação do ponto de vista prático é a modulação sinusoidal de frequência. Na figura 4.6 estão presentes dois sinais, a azul a frequência do sinal transmitido e a vermelho a frequência do sinal recebido para um alvo estático. A frequência do sinal transmitido é dada por:

$$f(t) = f_0 + \Delta f \cdot cos(2\pi f_m t) \tag{4.6}$$

O sinal transmitido é dado por:

$$s(t) = A_1 \cdot sin(\phi(t)) \tag{4.7}$$

onde

$$\phi(t) = 2\pi \int f(t)dt = 2\pi f_0 t + \frac{\Delta f}{f_m} \cdot sin(2\pi f_m t) \tag{4.8}$$

reescrevendo a equação 4.7

$$s(t) = A_1 \cdot sin\left[2\pi f_0 t + \frac{\Delta f}{f_m} \cdot sin(2\pi f_m t)\right] \tag{4.9}$$

O sinal recebido de um alvo estático vai ser uma réplica do sinal transmitido com um pequeno desvio, T, no tempo que é dado pela equação 4.1. O sinal recebido pode ser escrito como:

$$\begin{aligned} r(t) &= A_2 \cdot sin[\phi(t-T)] \\ &= A_2 \cdot sin\left[2\pi f_0(t-T) + \tfrac{\Delta f}{f_m} \cdot sin[2\pi f_m(t-T)]\right] \end{aligned} \tag{4.10}$$

Após comparação e filtragem dos sinais transmitido e recebido, o resultado pode ser aproximado por:

$$\begin{aligned} n(t) &= A_3 \cdot sin[\phi(t) - \phi(t-T)] \\ &= A_3 \cdot sin\left(2\pi f_0 T + \tfrac{2\Delta f}{f_m} \cdot sin(\pi f_m T) cos\left[2\pi f_m\left(t - \tfrac{T}{2}\right)\right]\right) \end{aligned} \tag{4.11}$$

Se for assumido que $T << 1/f_m$, a frequência f_b do sinal acima pode ser calculada por[26]:

$$< f_b >= \frac{8 \cdot R \cdot f_m \cdot \Delta f}{c} \tag{4.12}$$

resolvendo a equação anterior em ordem à distância R temos que:

$$R = \frac{c}{8 \cdot \Delta f \cdot f_m} < f_b > \tag{4.13}$$

que é o resultado obtido em 4.4 para a modulação linear.

4.2 Radar do ASV ROAZ II

O radar presente no ROAZ II é um radar do tipo FMCW (*Lowrance Broadband 3G*) que é possível comprar com uma consola que permite visualizar os dados do radar. Sendo uma antena rotativa, esta vai emitindo nas diferentes direcções observando os ecos e construindo uma imagem de reflexão circular que é apresentada ao fim de uma volta na consola do radar. Esta consola permite também fazer seguimento até 10 embarcações ao mesmo tempo.

Contudo a consola não consegue fornecer informações sobre as embarcações para o exterior da mesma. A consola é um instrumento meramente visual incapaz de comunicar com o computador de bordo do ROAZ II. Surge a necessidade de conseguir usar os dados do radar (ecos) para implementar um sistema de seguimento de embarcações que permita fornecer informações úteis ao ROAZ II.

Este radar (*Lowrance Broadband 3G*) possui um alcance máximo de aproximadamente 45 Km, uma velocidade rotacional mínima de 24 rpm e máxima de 36 rpm. A comunicação entre o radar e computador é estabelecida via *ethernet*. Algumas das características do radar *Lowrance Broadband 3G* são [28]:

- Alcance máximo configurável (45 Km);
- Dupla velocidade rotacional;
- Alcance mínimo de 50 m;
- Consumo em operação de $18W$;
- Consumo em *standby* de $2W$;
- Grau de protecção IPX6;
- Tecnologia FMCW;
- Conectividade *ethernet*;
- Começo de funcionamento com comando.

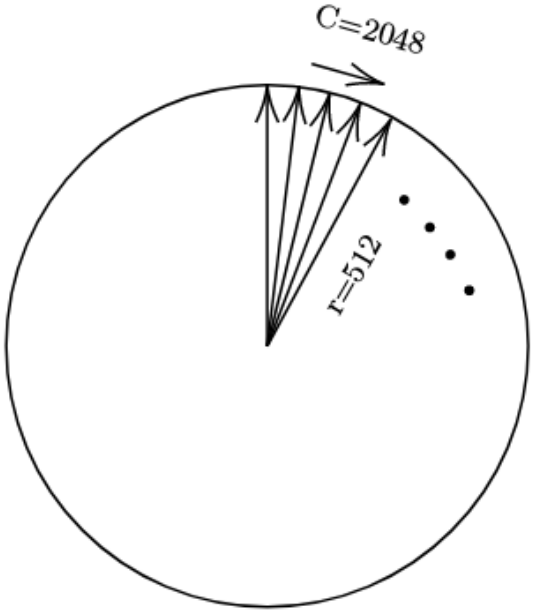


Figura 4.7: Raio de funcionamento do *Lowrance 3G*

A figura 4.7 representa o raio de funcionamento do radar. Como é possível ver, o radar possui 2048 *scanlines* diferentes, onde cada *scanline* é composta por 512 possíveis segmentos. O tamanho de cada segmento é dependente do alcance máximo configurado e é dado por:

$$l = \frac{D_{max}}{r} \tag{4.14}$$

onde r é o número máximo de segmentos numa *scanline* (512), l a distância entre segmentos e D_{max} o alcance máximo configurado do radar.

Usualmente as empresas que vendem radares para navegação náutica, vendem um solução completa, isto é, vendem um sistema que engloba o sensor, uma caixa que estabelece a comunicação com o sensor e um visualizador. *Dabrowski et al.* [29] tentaram fazer engenharia reversa à comunicação do radar *Lowrance BR24* com a caixa de interface. Para tal foi usado o *software Wireshark* para interceptar pacotes transmitidos por *ethernet*. Concluíram que o radar envia a trama de comunicação presente na figura 4.8.

Onde *ns* é o número de *scanlines* por pacote (32), *ss* é o tamanho de cada *scanline* (512), *l* é o tamanho do cabeçalho *scanline* (24), *st* é desconhecido, *rc* é o contador da *scanline* (0...4095), *a* é o ângulo da *scanline* (0...4095 = 0...360°), *scale* é raio máximo de leitura e u_x é desconhecido. Como é possível ver na figura, esta é composta por um cabeçalho inicial de tamanho 8, por um segundo cabeçalho que é respectivo a cada *scanline* e por fim a respectiva *scanline*. Cada pacote enviado é composto por um cabeçalho e 32 *scanlines*. Cada *scanline* é composta por um, segundo, cabeçalho com informações relativas ao ângulo e os respectivos dados observados.

É possível que o radar salte uma ou mais *scanlines* quando o módulo interno de rejeição

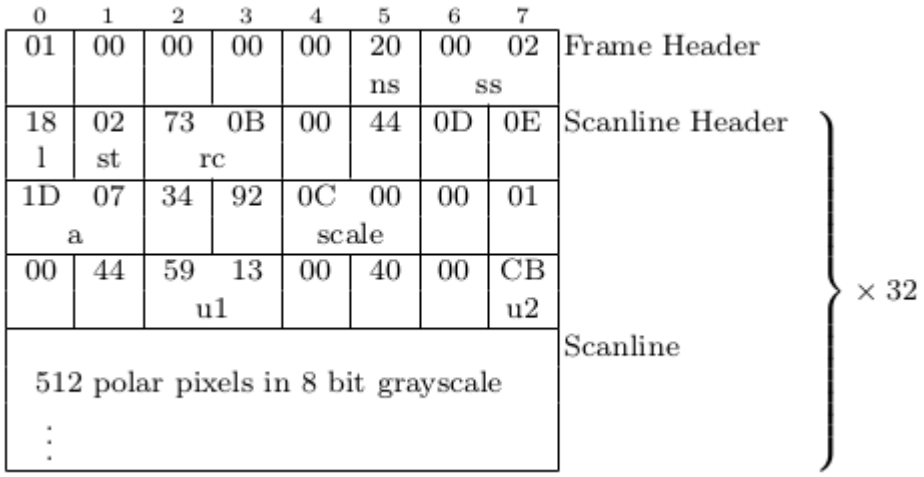


Figura 4.8: Pacote enviando pelo radar[29]

Tabela 4.1: Exemplo de envio de comandos para radar

0x03	0xC1	0x10	0x27	00	00
reg	cmd	data			

de frequência descarta medidas[29]. Em situações normais, o ângulo entre duas *scanlines* consecutivas irá ser um incremento em duas unidades. Existe um conjunto de registos no radar que permitem configurar filtros, velocidade de rotação, alcance máximo e início de leitura. O envio de comandos para o radar é feito segundo a tabela 4.1. A mensagem enviada para o radar é composta por 3 campos, um primeiro que é o registo onde vai ser escrito, *reg*, um segundo que é para indicar se é escrita ou leitura, *cmd*, o último é o valor que irá para ser escrito no registo seleccionado *data*. Ao total existem nove registos no radar que podem ser configurados. Os dois primeiros registos são o 0x01 e 0x02 que quando é escrito o valor 1 neles o radar começa a enviar dados e quando é escrito um 0 neles o radar para de enviar dados. Estes registos permitem controlar quando o radar irá dar início e término de envio de dados. O registo 0x03 é o registo responsável por estabelecer o alcance máximo de leitura de dados. *Dabrowski et al.* [29] sugere o uso da tabela 4.2 para diferentes alcances máximos e seus respectivos dados a serem enviados. Observando a tabela 4.1, exemplo de comando enviado, o registo 0x03 é o registo relativo ao alcance máximo. O segundo termo *cmd* está com 0xC1 que significa escrita. O último termo é o valor a ser escrito no registo 0x03 que é 0x10,0x27,0,0. Comparando este valor com a tabela 4.2 é possível concluir que o alcance máximo irá ser configurado para *1 Km*.

O registo 0x06 oferece acesso a uma variedade de filtros e pré-processamento como *Sea Clutter Compensation, Rain Clutter* e controlo do ganho do radar. O registo 0x08 é destinado à rejeição de interferência. Este tem quatro opções possíveis: 0 = *desligado*, 1 = *baixo*, 2 = *medio* e 3 = *alto*. O registo 0x0A é de *target boost* com três opções possiveis:

Tabela 4.2: Registo para diferentes alcances máximos[29]

Alcance	Dados (Hex)	Alcance	Dados (Hex)
50 m	0xf4,0x01,0,0	2 km	0x20,0x4e,0,0
75 m	0xee,0x02,0,0	3 km	0x30,0x75,0,0
100 m	0xee,0x03,0,0	4 km	0x40,0x9c,0,0
250 m	0xc4,0x09,0,0	6 km	0x60,0xea,0,0
500 m	0x88,0x13,0,0	8 km	0x80,0x38,1,0
750 m	0x4c,0x1d,0,0	12 km	0xc0,0xd4,1,0
1 km	0x10,0x27,0,0	16 km	0x00,0x71,2,0
1.5 km	0x98,0x3a,0,0	24 km	0x80,0xa9,3,0

Tabela 4.3: Resumo dos registos do radar *Lowrance Broadband 3G*

Registo(hex)	Função	Valores
0x01/0x02	Início de leitura de dados	0 ou 1
0x03	Alcance máximo	tabela 4.2
0x06	Filtro de Pré-Processamento	00/02/04
0x08	Rejeição de interferência	0 - 3
0x0A	*Target Boost*	0 - 2
0x0E	Filtro de interferência local	0 - 3
0x0F	Velocidade de rotação	0 ou 1
0xA0	*Keep Alive*	2

0 = *desligado*, 1 = *baixo* e 2 = *alto*. O registo 0x0E é de um filtro de interferência local. Este poderá ter os mesmos quatro valores que o registo 0x08, rejeição de interferência. O registo 0x0F é para controlo da velocidade de rotação do radar. Este registo poderá ter dois valores, 1 significa colocar a velocidade de rotação a *36 rpm* e 0 a *24 rpm*. Por ultimo, existe um registo que se chama *Keep Alive* com código 0xA0. O radar desliga-se automaticamente após 20 a 60 segundos sem ter recebido nada neste registo. De modo a prevenir este facto, periodicamente deve ser escrito neste registo o valor 2[29]. A tabela 4.3 é um resumo dos comandos possíveis a serem enviados para o radar.

O radar usado por *Dabrowski et al.* [29], *Lowrance BR24*, é muito semelhante ao radar que o ROAZ II possuí, *Lowrance Broadband 3G*. O trabalho feito por *Dabrowski et al.* for de enorme relevância na implementação de um nó ROS para comunicação com o radar.

Como mencionado anteriormente no capítulo 3.2, o sistema apresentado é desenvolvido em ROS para permitir uma maior facilidade de integração com os sistemas já existentes no ROAZ II. A figura 4.9 apresenta a arquitectura implementada em ROS para comunicação, configuração e visualização dos dados provenientes do radar. O nó *radar_comms* é responsável por estabelecer a comunicação com o radar via *ethernet*. Periodicamente,

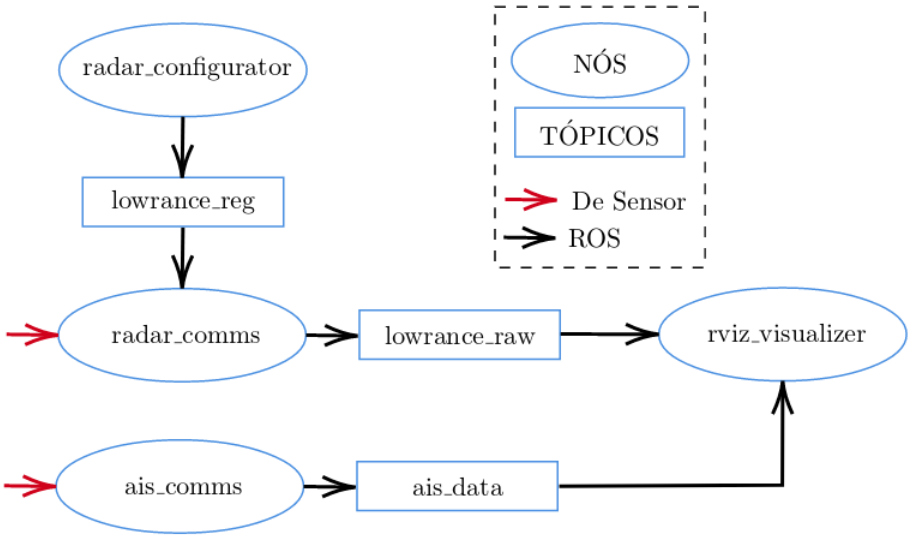


Figura 4.9: Nós e tópicos de ROS para aquisição de dados de radar

Tabela 4.4: Parte da trama de informação

0x00	0x00	0x07	0x1a	0x23	0x1b	0x00	0x00	0x0c	0x12	0x1a	0x1b
start	index=7		eco 7	eco 8	eco 9	start	index=13		eco 12	eco 13	eco 14

de dois em dois segundos, este nó envia uma mensagem *keep alive* para o radar para que este se mantenha ligado. Este nó permite também o envio de comandos para configuração do radar. Sempre que o radar mandar informação para o nó, este trata de converter a informação do formato da figura 4.8 para um formato mais conveniente para posterior processamento e interpretação.

A mensagem enviada no tópico trata-se de um vector de *uint8_t*. Esta é composta por um cabeçalho de 14 bytes e um corpo de N bytes:

- 8 bytes para representação de segundos (*Unix time*);
- 2 bytes para representação de milisegundos(*Unix time*);
- 2 bytes para representação de ângulo da *scanline* (0 - 4095);
- 2 bytes para representação do alcance máximo;
- N bytes para representação dos ecos recebidos.

A figura 4.10 é um exemplo de detecção do radar para um determinado ângulo. Neste exemplo apenas estão representados 14 segmentos porém o radar possui 512 por *scanline*. A azul e verde são duas embarcações distintas. Como é possível ver, apenas os segmentos {7,8,9} e {12,13,14} têm alguma informação. Os N bytes que servem para representar os dados observados pelo radar aparecem na forma da tabela 4.4. Analisando a trama de

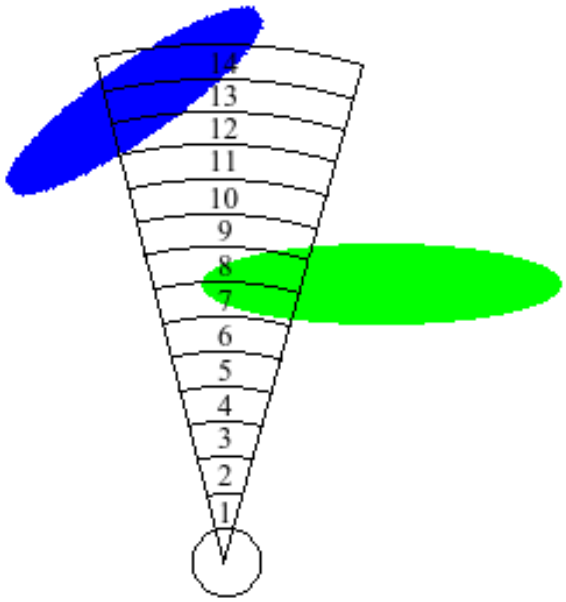


Figura 4.10: Exemplo de detecção do radar

comunicação da esquerda para a direita, é possível observar que um conjunto de pontos começa com uma condição de inicio, *start*, ocupando um byte. Após inicio de trama surgem dois bytes para indicar o *index* do primeiro segmento. Segundo o exemplo anterior, o primeiro segmento a aparecer é no *index* 7. As posições seguintes na trama, até acontecer novamente uma condição de inicio, são os valores das diferentes intensidades dos ecos sendo que cada byte é a representação da intensidade o eco. No caso da trama anterior a intensidade dos ecos **7**, **8** e **9** são respectivamente: **0x1a** ou **26**, **0x23** ou **35** e **0x1b** ou **27**. Na trama exemplo, a posição 6 é novamente uma condição de inicio. Isto significa que objecto anterior não possui mais segmentos e vai começar um novo objecto. As duas posições seguintes definem o *index* do primeiro segmento. Na situação de existir 512 segmentos diferentes por cada *scanline*, a trama de comunicação irá ser tão grande quanto o número de objectos detectados. No tópico *lowrance_raw* irá ser publicado sempre que o radar detectar algum objecto.

Foi desenvolvido um nó de ROS adicional de modo a poder enviar comandos para o radar, de modo a modificar os registos. Este nó é o *radar_configurator* e comunica com o nó anteriormente descrito pelo tópico *lowrance_reg*. Este nó periodicamente verifica se o utilizador pretende alterar algum dos registos presentes na tabela 4.3, usando para tal o ROS *parameter server*.

De modo a validar os dados obtidos do radar, foi usado um receptor AIS. Como mencionado no capítulo 2.1, este tipo de sensor permite receber informações de outras embarcações que estejam na proximidade do receptor. Foi utilizado um nó que estabelece a comunicação com o receptor de AIS desenvolvido por elementos do LSA. O nó utilizado consiste em receber informações do receptor de AIS e publicar num tópico de ROS,

Figura 4.11: *Setup* utilizado na praia de Matosinhos

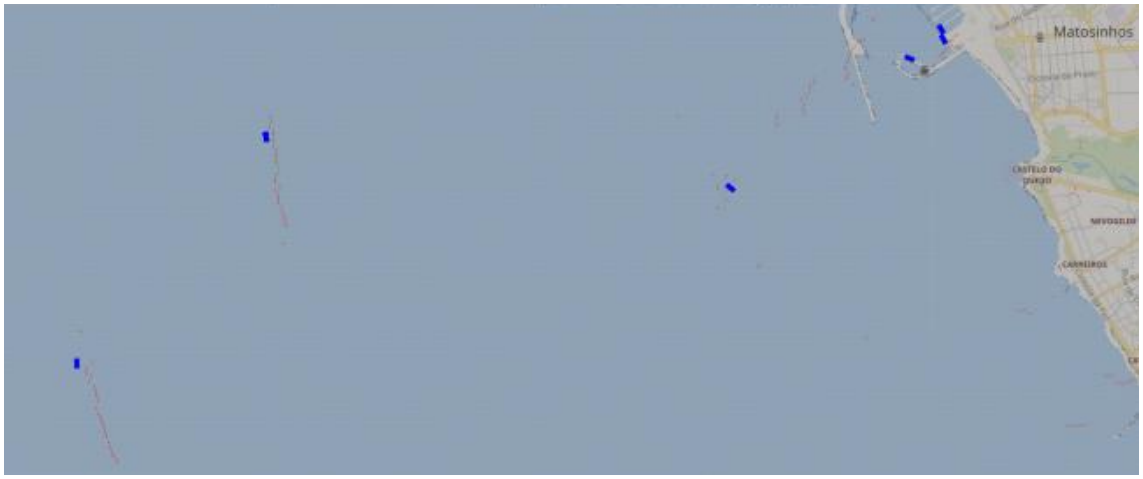


Figura 4.12: Dados do radar no visualizador *rviz* com sobreposição de AIS

ais_data, que é usado, posteriormente, para comparação com radar.

Para visualizar os dados observados do radar, foi desenvolvido um nó de ROS que fica responsável por converter os dados presentes no tópico *lowrance_raw* para pixeis que serão usados no visualizador de ROS *rviz*. Este nó, é também responsável por converter os dados do tópico *ais_data* para pixeis e enviar para o visualizador *rviz*.

4.3 Resultados de validação de integração do radar

Numa primeira fase, foram efectuados testes para validar o *software* desenvolvido para o radar num referencial estático, isto é, o radar encontrava-se imóvel durante o período de testes.

O *setup* experimental presente na figura 4.11 é composto pelo radar do ROAZ II, uma fonte de alimentação para fornecer energia ao radar, um receptor de AIS e um computador para guardar informação.

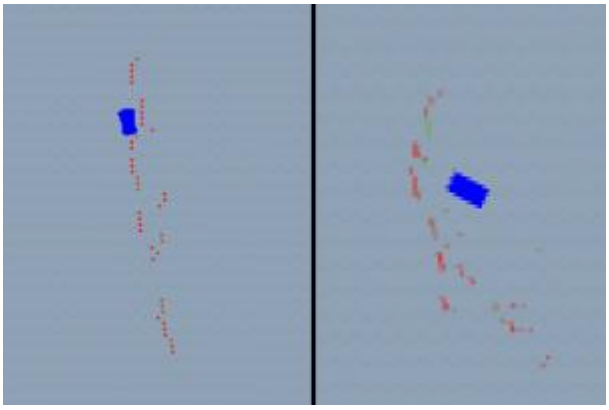


Figura 4.13: Detecção de barcos no visualizador *rviz*

Após primeiros testes, foi observado que a posição retornada do AIS não coincidia com as posições observadas pelo radar. Tendo como base o trabalho desenvolvido por *Dabrowski et al.* [29] foi efectuado alguma engenharia reversa ao radar *Lowrance BR24*. O radar utilizado neste trabalho é em tudo semelhante ao de [29], porém foi constatado que o valor do alcance máximo recebido do radar vinha dividido por 4, isto é se o radar fosse configurado, em função da tabela 4.2, para para um alcance máximo de 10 Km, o valor do alcance máximo que o radar retornava era 2,5 Km . O valor correcto a escrever no registo 0x03, alcance máximo, é dado por:

$$4 * alcance_{desejado(metros)} * \sqrt{2} \tag{4.15}$$

A figura 4.12 é o mapa do local aonde estava localizado o radar. A azul está presente a posição real das embarcações proveniente do receptor de AIS, sendo que a orientação dos rectângulos não é representação da orientação dos barcos. A vermelho e verde são os diferentes segmentos observados pelo radar. As cores dos segmentos são dadas em função da intensidade do eco, quanto maior o valor do eco mais vermelho é o segmento e quanto menor mais verde. A cor dos segmentos é dada por:

$$\begin{aligned} red &= \frac{eco}{255} \\ green &= 1 - \frac{eco}{255} \\ blue &= 0 \end{aligned} \tag{4.16}$$

Observando com atenção a figura 4.13, ampliação da figura 4.12, é possível concluir que os dados do radar encontram-se no local correcto. Com este *setup* foi testado um alcance máximo de *50 Km* comprovando o seu funcionamento, porém não foi possível comprovar os dados do radar com o receptor de AIS. O limite máximo de recepção de dados do receptor de AIS é aproximadamente de *10 Km*. No cenário acima, o radar estava configurado para um alcance máximo de *25 Km*. Usando a equação 4.14, a distância entre segmentos é de aproximadamente 48 metros. O facto de só existir um segmento em cada ângulo deve-se

a este mesmo facto. As funcionalidades dos filtros, rejeição de interferência e velocidade de rotação foram também testadas e foi comprovado o seu funcionamento conforme [29].

Capítulo 5

Agregação e Classificação

Neste capítulo, numa primeira fase, são apresentados um conjunto de técnicas de agregação de pontos e apresentada a solução implementada em ROS. Posteriormente é descrito o funcionamento de um nó de ROS que faz classificação de objectos retornados pela agregação de pontos. Por ultimo é apresentado um algoritmo que permite extrair a linha de fronteira terra mar.

5.1 Métodos clássicos de agregação

A agregação, ou também conhecido como *clustering*, é uma técnica muito importante na análise de dados. Esta pode ser definida como o problema de separar um conjunto de dados em diferentes pequenos conjuntos, *clusters*. Elementos dentro do mesmo *cluster* possuem características similares e elementos de *clusters* diferentes possuem um discrepância nas suas características[30]. No caso particular deste trabalho, o problema é com um conjunto de pontos obtidos do radar, conseguir separar em diferentes objectos para posterior análise.

Este *clustering* pode ser dividido em três grandes grupos:

- Baseado na Conectividade
- Baseado no Centroide
- Baseado na Densidade

O primeiro consiste em utilizar um conjunto de dados e considerar que cada ponto, inicialmente, é *cluster* diferente. A figura 5.1, é um pequeno exemplo para demonstrar o principio de *clustering* baseado em conectividade. Na primeira iteração, os *cluters* B e C vão se juntar num só criando um novo *cluster* BC. Os *cluters* D e E vão se juntar num novo *cluster* DE como é possível ver na figura 5.2. Na segunda iteração, os *clusters* DE e F juntam-se num novo formando o *cluster* DEF. Este processo acontece até só existir um

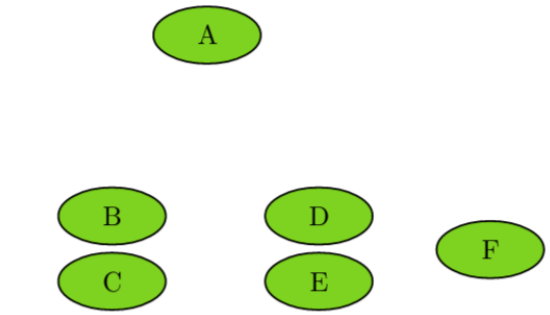


Figura 5.1: Exemplo do *clustering* baseado em conectividade

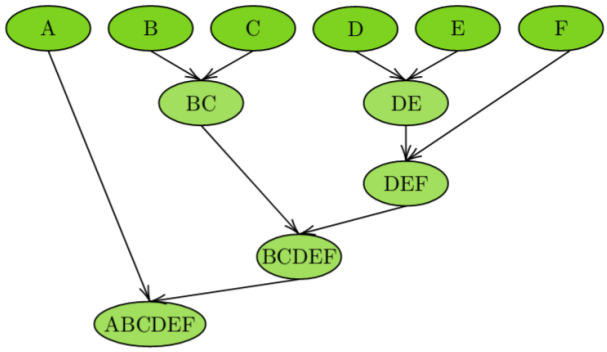


Figura 5.2: Resultdo do *clustering* baseado em conectividade

cluster. Este processo pode ser comparado a uma hierarquia, onde quanto mais baixo na figura 5.2 maior é o cluster.

O *clustering* baseado em centroide consiste em utilizar um conjunto de dados e encontrar vários *clusters* em função de uma lista de centroides iniciais. O método mais comum deste tipo de *clustering* é o *K-means*[31]. Este tipo de algoritmo consiste, numa primeira fase, em escolher o número de objectos desejados. Por cada objeto escolhido é necessário de escolher um centroide inicial.

Algoritmo 1 K-means

```
1: Selecionar K pontos com centroides iniciais
2: while Novo centroide diferente centroide anterior do
3:     Associar cada ponto ao centroide mais próximo
4:     Recalcular o novo centroide para cada cluster
5: end while
```

Como mostra o algoritmo 1, este algoritmo depende muito da escolha inicial dos diferentes centroides. O ciclo presente no algoritmo vai efetuar o número de vezes necessário

Figura 5.3: Algoritmo *K-means* com dados do radar

até conseguir minimizar a função custo:

$$Custo(C) = \sum_{i=1}^{k} \sum_{x \in C_i} dist(x, c) \tag{5.1}$$

onde C são os diferentes *clusters*, k o número de *clusters*, x é um ponto e c o centroide do *cluster*. O algoritmo *K-means* é um algoritmo que depende muito da escolha inicial dos centroides. Se os centroides iniciais estiverem perto do centroide correcto, o tempo de processamento vai ser baixo. Caso os centroides iniciais difiram muito dos centroides correctos, o algoritmo vai demorar muito tempo a convergir. Este algoritmo não possui uma só solução possível. Com o mesmo conjunto de dados iniciais e é possível obter diferentes configurações de centroides finais. Este algoritmo considera que todos os pontos existentes pertencem a algum *cluster*. Na situação de existirem pontos isolados, como por exemplo ruído, o algoritmo não está preparado para os descartar.

Este algoritmo foi testado para a agregação de pontos do radar. Numa primeira fase foi feita a implementação do algoritmo recorrendo ao *software* Matlab para uma fácil validação. Durante a fase de testes do algoritmo, foi possível observar que o algoritmo possuí uma característica muito importante, o número total de centroides. No cenário de fazer agregação de pontos do radar, não é possível saber o número de objectos que existem. Para contornar este facto, foi decidido inicializar o algoritmo com um número elevado de objectos desejados, mais do que o número de objectos esperados no mundo. A figura 5.3 é o resultado do algoritmo de *K-means* para número inicial de centroides de 100. Na figura diferentes cores correspondem a diferentes *clusters*. Um optimizador para o algoritmo *K-means* foi desenvolvido. Este optimizador consiste em verificar se a distancia entre dois *clusters* é inferior a um determinado *threshold*. Caso positivo, os dois *clusters* são juntos num novo *cluster*. O processo repete-se até não existirem *clusters* para serem juntos. A figura 5.4 é o resultado o optimizador para o algoritmo *K-means* da figura 5.3.

O conjunto de dados inicial é composto por mais de 8000 pontos que correspondem a uma volta completa do radar. Noutras situações foi observado que uma volta do radar

Figura 5.4: Optimizador do algoritmo *K-means* com dados do radar

poderia retornar mais de 20000 pontos. Para executar o algoritmo *K-means* e o optimizador demorou aproximadamente 2.01 segundos. Se for considerado que o radar se encontra a funcionar na velocidade mais baixa, 2 segundos para dar uma volta, o tempo de processamento para agregar os pontos com o algoritmo *k-means* é superior ao tempo de uma volta. Esta solução não é viável para o contexto deste trabalho.

Um dos algoritmos mais comuns no *clustering* baseado na densidade é o *DBSCAN*[32]. Este algoritmo consiste em isolar ruído de pontos que pertençam a diferentes objectos.

Algoritmo 2 DBSCAN

```
1: considerar todos os pontos como não classificados
2: while existir pontos como não classificados ou ruído do
3:    seleccionar um ponto não classificado
4:    if expandir cluster no ponto seleccionado then
5:       adicionar pontos ao cluster
6:       considerar os pontos com classificados
7:    end if
8: end while
```

O algoritmo 2 é o algoritmo do DBSCAN. Os pontos, inicialmente, começam todos como não classificados. É seleccionado um ponto do conjunto inicial e feita a expansão deste ponto. Este processo consiste em verificar quantos pontos se encontram a uma distância inferior de um determinado *threshold*, *Eps*. Se o número de pontos que cumpram a condição anterior for maior que o numero mínimo de pontos, *minPts*, o conjunto de pontos é adicionado ao *cluster* e ficam classificados como já classificados. Se o número de pontos não atingir o numero mínimo de pontos, estes são classificados como ruído. Este algoritmo permite diferenciar ruído de *clusters*, estando dependente de apenas dois parâmetros *Eps* e *minPts*[32]. A figura 5.5 é o resultado da implementação do algoritmo DBSCAN para um conjunto de pontos do radar. Tal como no *K-means*, o conjunto de dados é referente a uma rotação completa do radar.

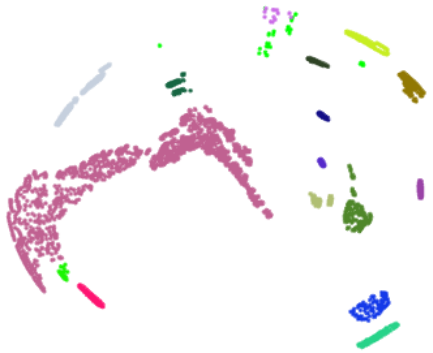


Figura 5.5: Algoritmo DBSCAN com dados do radar

Comparativamente ao *K-means* sem optimizador, este algoritmo apresenta melhores resultados devido ao facto de conseguir isolar o ruído dos *clusters*. O algoritmo implementado em *Matlab* demorou aproximadamente 2.15 segundos para processar 8000 pontos. O DBSCAN é um algoritmo determinístico, isto é, se aplicar o algoritmo mais que vez ao mesmo conjunto de pontos com as mesmas condições de *minPts* e *Eps* o resultado irá ser o mesmo. O algoritmo *K-means* é um algoritmo não determinístico que depende da escolha inicial dos centroides.

5.2 Solução proposta para agregação

A solução pretendida para agregação de pontos neste cenário é conseguir efectuar em tempo real, isto é, supondo que o radar é configurado para 30 *rpm* e que todos os ângulos possuem alguma informação útil, o tempo entre cada ângulo é aproximadamente 0.977 ms. Efectuar agregação em tempo real significa que o tempo de processamento para cada ângulo terá que ser inferior a 0.977 ms. O tempo anterior é na situação em que o computador que processa o nó de ROS apenas tem esta tarefa para executar. Numa situação real, o computador cede uma fatia do tempo para cada processo que está a ser executado. A solução de agregação deverá ser capaz de distinguir ruído dos restantes pontos e ser um sistema determinístico.

5.2.1 Mudança de referencial

No capítulo 4.3 os testes efectuados foram na situação de ter o radar estático. Num cenário real, o radar estaria montado no ROAZ II e este estaria em movimento. De modo a poder usar os pontos do radar com o ROAZ II em movimento é necessária efectuar uma mudança de referencial dos pontos, visto que os pontos que o radar retorna estão referenciados ao radar. Foi desenvolvido um nó de ROS para efectuar a interligação entre o nó de comunicação e um nó de classificação. Este nó recebe um vector de mensagens

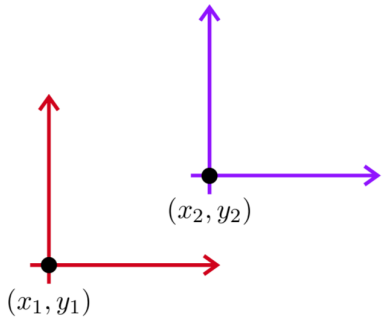


Figura 5.6: Exemplo de translação de referencial

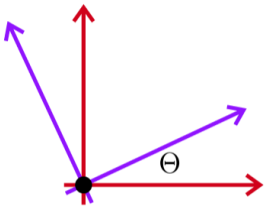


Figura 5.7: Exemplo de rotação de referencial

do tipo *uint8_t* com o formato apresentado na secção 4.2. A trama recebida é composta por um cabeçalho com a descrição do ângulo da *scanline*, alcance máximo configurado e tempo actual. A segunda parte da trama recebida é semelhante à tabela 4.4. De modo a conseguir ter os pontos no referencial cartesiano, ao invés do referencial polar, as seguintes equações são usadas:

$$\begin{aligned} \theta &= angulo * 360/4096 \\ L_{seg} &= alcance_{max}/512 \\ P_x &= index * L_{seg} * cos(\theta) \\ P_y &= index * L_{seg} * sin(\theta) \end{aligned} \tag{5.2}$$

As equações acima representam uma translação e uma rotação no plano e é possível ter os pontos referenciados ao radar no sistema cartesiano de coordenadas. Contudo é necessário mudar os pontos do referencial local, radar, para um referencial global. Esta mudança de referencial pode implicar um translação e/ou uma rotação.

A figura 5.6 é um exemplo de uma translação. Para efectuar a mudança do referencial a vermelho para o referencial a azul as seguintes equações são usadas:

$$\begin{bmatrix} X_2 \\ Y_2 \end{bmatrix} = \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{5.3}$$

onde dx é a diferença entre X_2 e X_1 e dy é a diferença entre Y_2 e Y_1. A equação anterior permite transladar um ponto entre os dois referenciais. Outro processo muito importante é a necessidade de rotação de referencial. A figura 5.7 é um exemplo de rotação de

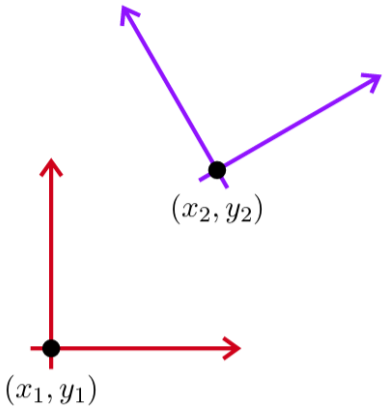


Figura 5.8: Exemplo de rotação e translação de referencial

referencial. Para poder efectuar a rotação de referencial a seguinte equação é usada[33]:

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{5.4}$$

onde θ é o ângulo entre os dois referenciais. Para efectuar a mudança de um ponto entre dois referenciais com uma rotação entre eles a seguinte equação pode ser usada:

$$\begin{bmatrix} X_2 \\ Y_2 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} * \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} \tag{5.5}$$

A figura 5.8 é um exemplo de uma mudança de referencial que implica uma rotação e translação. Este mudança é dada por:

$$\begin{bmatrix} X_2 \\ Y_2 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} * \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{5.6}$$

A equação anterior é usada quando surge a necessidade de mudar de referencial a um ponto. Na situação do radar, surge a necessidade de mudar de referencial porque o ROAZ II encontra-se em movimento e os pontos do radar estão referenciados a este. Para efectuar a mudança de referencial é necessário estabelecer um novo referencial estático. Usado como exemplo a figura 5.8, assumindo que o referencial a vermelho é referencial do radar e a azul o novo referencial estático, para poder mudar de referencial os pontos observados pelo radar é necessário usar as equações 5.2 e 5.6.

De forma a poder efectuar esta mudança é necessário estimar dois parâmetros: localização e orientação do radar em relação ao referencial estático. Para estimar a localização e orientação do ROAZ II é necessário usar o GPS e IMU presentes no ROAZ II. Para tal foi desenvolvido um nó de ROS que recebe mensagens do GPS e IMU de forma assíncrona. O objectivo deste nó é enviar a localização e orientação do ROAZ II para outros nós de ROS a uma frequência fixa(*100Hz*). Na figura 3.3 este nó é o módulo de localização. Os

nós de comunicação e aquisição de dados do GPS e IMU foram desenvolvidos por outros elementos do laboratório e já se encontravam implementados no ROAZ II.

Para o referencial estático é também necessário saber e localização e orientação do mesmo. Nos testes efectuados no porto de Leixões, o referencial estático estava localizado no pontão do porto e a sua localização exacta foi adquirida com recurso a um GPS. A equação 5.6 pode ser rescrita para este cenário em especifico como:

$$\begin{bmatrix} X_{global} \\ Y_{global} \end{bmatrix} = \begin{bmatrix} cos(\theta_{roaz}) & -sin(\theta_{roaz}) \\ sin(\theta_{roaz}) & cos(\theta_{roaz}) \end{bmatrix} * \begin{bmatrix} x_{ponto} \\ y_{ponto} \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix} \tag{5.7}$$

onde dx é a diferença de latitudes e dy a diferença de longitudes.

Para poder efectuar o calculo das diferenças acima é necessário que as coordenadas latitude e longitude estejam num sistema de coordenadas cartesiano. O GPS retorna as coordenadas sobre a forma polar. O sistema de coordenadas cartesiano utilizado foi o *Earth-centered*, Earth-fixed (ECEF). Para converter um ponto do GPS para o sistema de coordenadas ECEF as seguintes equações são usadas[34][35]:

$$\begin{aligned} x &= (\eta + h)\cos\theta\cos\lambda \\ y &= (\eta + h)\cos\theta\sin\lambda \\ z &= (\eta(1 - e^2) + h)\sin\theta \end{aligned} \tag{5.8}$$

onde θ é a latitude, λ a longitude, h a altitude e η é dado por:

$$\eta = r_e/\sqrt{1 - e^2\sin\theta^2} \tag{5.9}$$

onde r_e é o semieixo maior da terra e e dado por:

$$e^2 = \frac{r_e^2 - r_p^2}{r_e^2} \tag{5.10}$$

com r_p como semieixo menor da terra. O sistema de coordenadas ECEF é um sistema cartesiano centrado no centro da terro e fixo com a mesma. Para uma maior facilidade de cálculos é possível usar um sistema de coordenadas que permite ter o centro num local definido. Para tal o sistema de coordenadas *East-North-Up*(ENU) local é utilizado. Para converter de ECEF para ENU as seguintes equações foram usadas[35]:

$$\begin{bmatrix} e \\ n \\ u \end{bmatrix} = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\theta\cos\lambda & -\sin\theta\sin\lambda & \cos\theta \\ \cos\theta\cos\lambda & \cos\theta\sin\lambda & \sin\theta \end{bmatrix} \begin{bmatrix} x - x_z \\ y - y_z \\ z - z_z \end{bmatrix} \tag{5.11}$$

onde x_z, y_z e z_z são as coordenadas ECEF do referencial estático localizado no pontão do porto de Leixões. As coordenadas em ENU ficam referenciadas ao pontão, neste cenário em específico. Na equação 5.7 as componentes dx e dy passam a ser as coordenadas ENU obtidas da equação anterior 5.11.

A figura 5.9 é a representação dos três sistemas de coordenadas apresentados anteriormente. As coordenadas de GPS estão representadas como (λ, θ), o referencial ECEF como (x, y, z), ENU como (E, N, U) e o Meridiano de *Greenwich* como PM.

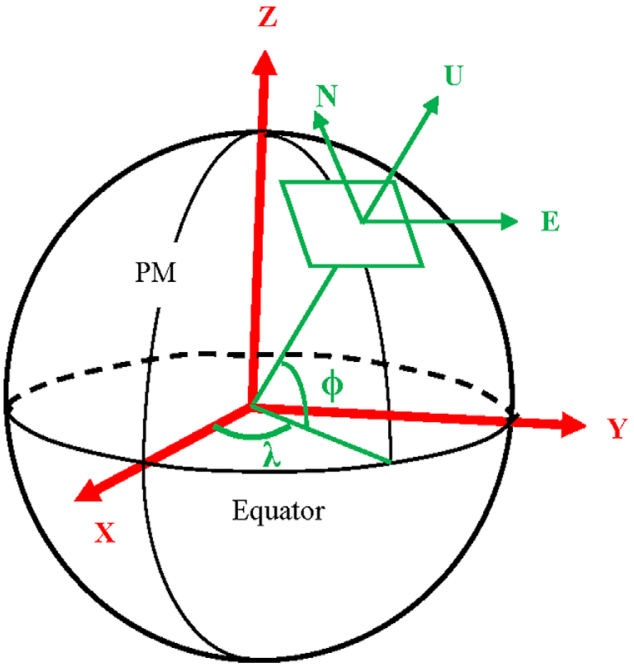


Figura 5.9: Sistema de coordenadas GPS, ECEF e ENU[36]

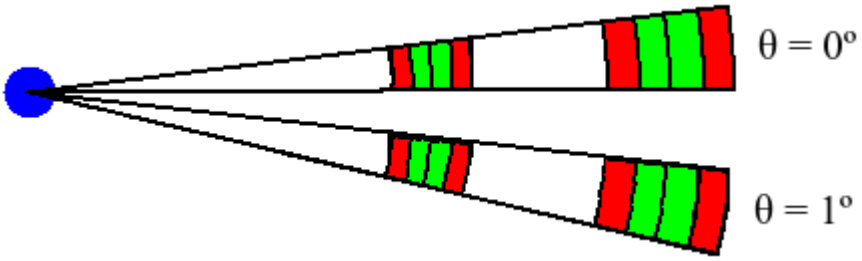


Figura 5.10: Exemplo de conjunto de dados para agregação

5.2.2 Agregação

A figura 5.10 apresenta um exemplo de dois ângulos consecutivos retornados pelo nó de ROS que estabelece a comunicação com o radar. Considerando que o nó de ROS de agregação de pontos recebe em primeiro lugar o ângulo $\theta = 0°$, o algoritmo vai verificar quantos conjuntos de segmentos existem nessa *scanline*, dois no caso do exemplo da figura 5.10. Por cada conjunto de segmentos, o algoritmo vai verificar se já existe algum objecto criado. No caso de não existirem objectos já criados, é criado um novo com o conjunto de segmentos. Na situação de já existirem objectos criados, o algoritmo vai verificar se existem segmentos no conjunto que pertençam a algum dos objectos já criados. Um segmento pertence a um objecto se a diferença entre o segmento e algum dos segmentos

do objecto for inferior a um *threshold*, equação 5.12.

$$\left|\begin{bmatrix} x_{seg} \\ y_{seg} \end{bmatrix} - \begin{bmatrix} x_{obj} \\ y_{obj} \end{bmatrix}\right| < \begin{bmatrix} threshold_x \\ threshold_y \end{bmatrix} \tag{5.12}$$

No caso de um conjunto de segmentos pertencer a apenas um objecto, este conjunto de segmentos é adicionado ao objecto. Na situação de o conjunto de segmentos não pertencer a nenhum dos objectos, é criado um novo objecto com o conjunto de segmentos. Na situação de um conjunto de segmentos pertencer a mais do que um objecto, é criado um novo objecto com segmentos dos objectos ao qual o conjunto de segmentos pertence. Na situação da

Algoritmo 3 Agregação de pontos radar

```
1: if número de objectos = 0 then
2:    Cria novo objecto com conjunto de segmentos
3: else
4:    for objecto em objectos do
5:       if conjunto de segmentos pertencer a objecto then
6:          Adicionar conjunto de segmentos ao objecto
7:       end if
8:    end for
9:    if Conjunto de segmentos pertencer mais que um objecto then
10:      Juntar objectos num novo
11:   end if
12: end if
```

figura 5.10, o primeiro conjunto de segmentos a ser processado é o mais próximo do radar e com ângulo $\theta = 0°$. Nessa situação ainda nenhum objecto foi criado, consequentemente é inicializado um novo objecto, **obj1**. O segundo conjunto de segmentos a ser processado é o mais distante do radar com ângulo $\theta = 0°$. Se a diferença entre o conjunto de segmentos e os segmentos do **obj1** for inferior ao *threshold*, o conjunto de segmentos é adicionado ao **obj1**. Caso contrário, é criado um novo objecto, **obj2**. Quando surge uma nova *scanline*, ângulo $\theta = 1°$, o primeiro conjunto de segmentos a ser processado é o mais próximo do radar. É verificado se este conjunto pertence aos objectos **obj1** e **obj2**. Considerando que pertence ao **obj1**, o conjunto de segmentos é adicionado ao objecto tal como mostra o algoritmo 3(linha6).

A figura 5.11, é o resultado do algoritmo de agregação para o exemplo da figura 5.10. O algoritmo de agregação acontece sempre que receber um nova *scanline*.

Surge a necessidade de perceber quando é que um objecto deixou de receber segmentos/pontos. Para este efeito, sempre que um ponto é adicionado a um objecto é gravado o instante de tempo de ROS. Periodicamente é feita a verificação do ultimo conjunto de pontos recebido num determinado objecto. Se a diferença entre o tempo actual e o tempo

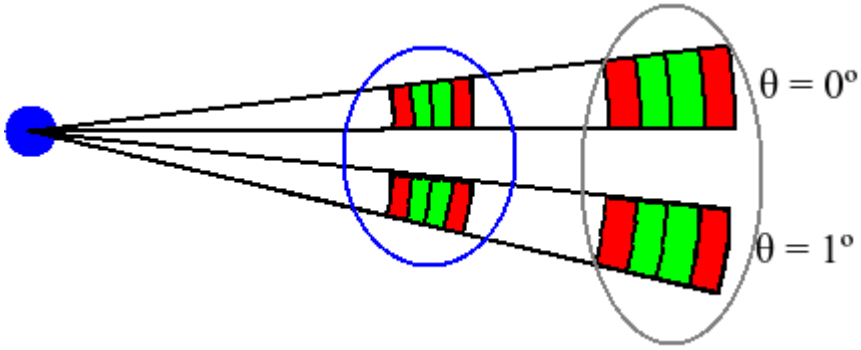


Figura 5.11: Exemplo resultado do algoritmo agregação implementado

do ultimo conjunto de pontos de um determinado objecto for superior a um *threshold* é considerado que o objecto já não vai receber mais pontos na rotação actual do radar. Por exemplo, se o ultimo conjunto de pontos de um determinado objecto tiver acontecido à mais de 1 segundo, pode-se assumir que não vão existir mais pontos desse mesmo objecto.

Uma segunda verificação foi adicionada de modo a contornar a situação de o barco encontrar-se dentro de um objecto, isto é, se o radar detectar e agregar pontos num objecto e se objecto irá acontecer em todas as *scanlines*. Nessa situação, a primeira condição nunca irá falhar e o objecto irá acumular pontos ao longo do tempo levando à falha do *software* por excesso de informação. A condição usada para contornar este facto é guardar o tempo de ROS no momento em que é criado o objecto. Se o objecto existir à mais de, por exemplo, 2 segundos, o objecto é automaticamente publicado no tópico de ROS e eliminado da lista de objectos.

Se um objecto respeitar as condições, existe a necessidade de avaliar o objecto. Esta avaliação consiste em verificar a quantidade de pontos existentes. Se o número de pontos existentes for muito pequena, este objecto é descartado e eliminado da lista de objectos. Caso contrário, este objecto é publicado num tópico de ROS e eliminado da lista de objectos. Deste modo é possível eliminar pequenos conjuntos de pontos isolados no mundo da informação útil.

5.3 Resultados da solução proposta para agregação

Como mencionado anteriormente, os testes foram efectuados, com o radar no ROAZ II em movimento, na marina do porto de Leixões. A figura 5.12 é o resultado do nó de agregação de pontos de uma volta completa do radar. Na figura, o radar é o ponto azul e estava configurado para um alcance máximo de 500 metros. Pontos vermelhos representam ecos fortes e pontos verdes ecos fracos (ver equação 4.16). Como é possível ver, foram criados 11 objectos com o algoritmo de agregação de pontos apresentado anteriormente. Alguns pontos que se encontram soltos, ruído do radar, foram descartados e não são

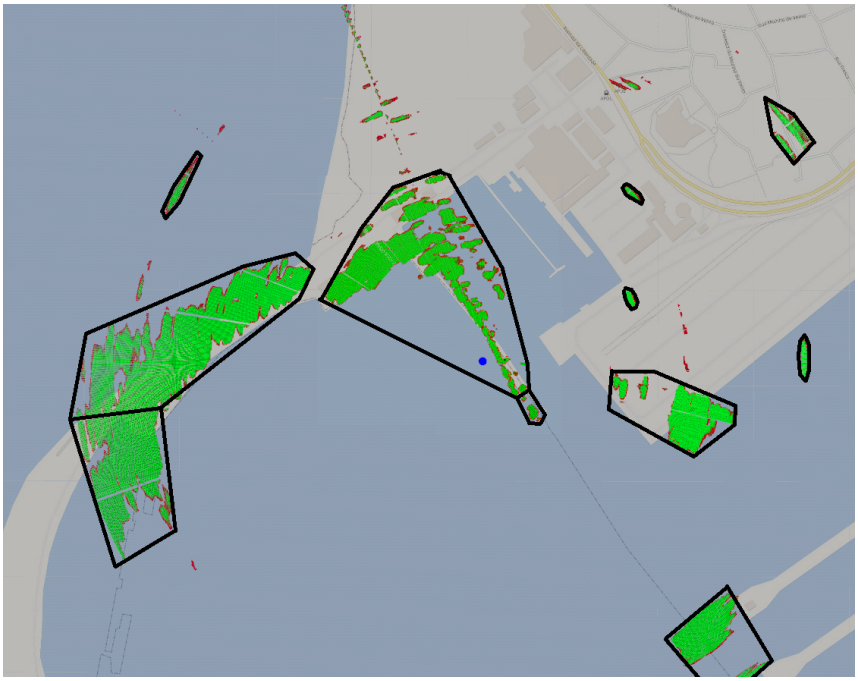


Figura 5.12: Resultado da agregação de pontos do nó de ROS

considerados objectos.

Foram efectuados vários testes ao algoritmo ao longo do tempo e foi verificado que o algoritmo de agregação de pontos do radar funciona segundo o esperado. Foi averiguado que em diversas voltas do radar, o resultado da agregação de pontos era em todas semelhante, podendo assim provar que o algoritmo funciona em tempo real e que não existe perda de informação, dados do radar. Na figura 5.12 o facto de os dois objectos mais à esquerda serem dois ao invés de um deve-se à segunda condição de verificação de um objecto.

De forma a poder validar o algoritmo de agregação de pontos em tempo real, foram guardados os pontos de todos os objectos detectados ao longo do tempo. A figura 5.13 é o resultado de um determinado objecto que foi detectado um diferentes instantes temporais. O objecto presente na figura **a** foi detectado na primeira frame, **b** na segunda frame e assim sucessivamente.

Estes dados foram obtidos com o barco ROAZ II em movimento. Na última frame, é possível obter uma falta de pontos no inicio e no fim do objecto. Após análise dos dados obtidos foi possível concluir que esta falha de dados foi oriunda do próprio radar e não do nó de agregação de pontos. Contudo, estes objectos apresentam muitas similaridades em todas as *frames* podendo assim validar o correcto funcionamento do nó de agregação de pontos em tempo real(é de notar que as escalas dos eixos são diferentes nós quatro gráficos).

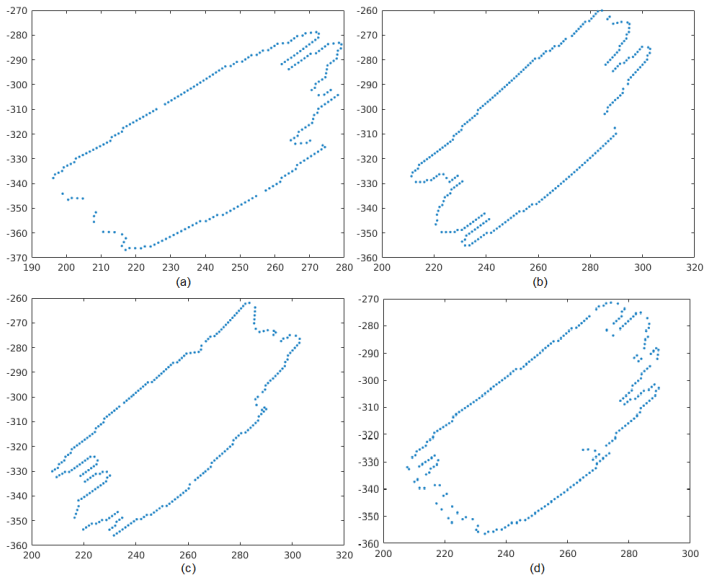


Figura 5.13: Objecto I detectado em rotações consecutivas

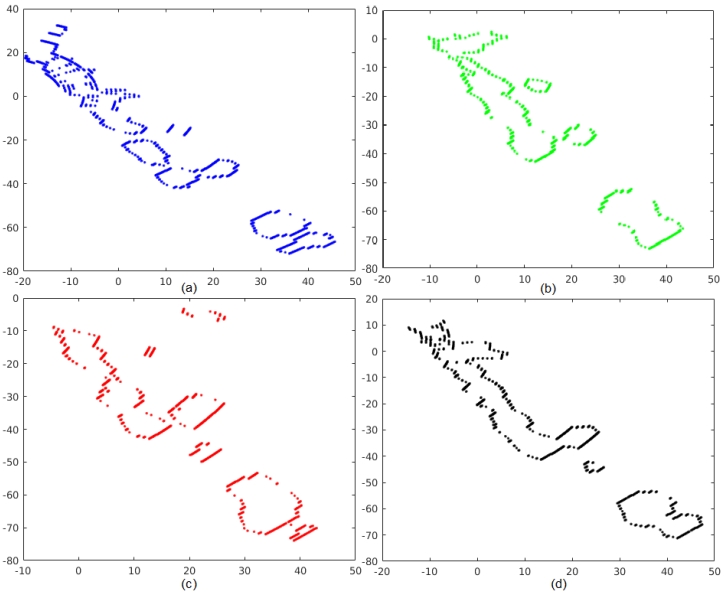


Figura 5.14: Objecto II detectado em rotações consecutivas

A figura 5.14 é um segundo exemplo de um determinado objecto detectado em *frames* consecutivas. Neste exemplo é possível observar parecenças entre as diferentes *frames* e assumir que o objecto foi detectado correctamente nas diferentes *frames*.

Comparativamente ao algoritmo K-means, este considera que todos os pontos pertencem a um objecto e que não existe ruído, enquanto que o algoritmo apresentado consegue fazer uma separação entre ruído e informação útil. A figura 5.15 direita apresenta o resultado do algoritmo K-means após uma volta do radar. Como é possível verificar, todos os

pontos foram contidos num cluster.

A figura da esquerda representa o resultado da agregação de pontos usando o algoritmo DBSCAN. Com este algoritmo já é possível descartar pontos soltos, ruído. Os resultados obtidos pelo método proposto e apresentado anteriormente (figura 5.12) são semelhantes aos do DBSCAN com um tempo de processamento inferior.

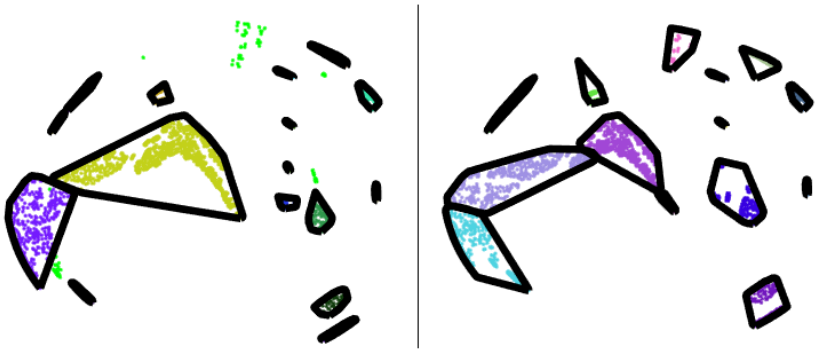


Figura 5.15: Resultado da agregação de pontos. À esquerda DBSCAN e à direita K-means

5.4 Classificação

De modo a poder efectuar *multi-target tracking* surge a necessidade de ter um sistema que classifique/descreva um alvo por forma a poder identificar os alvos ao longo do tempo. Para tal um nó de ROS foi desenvolvido que com base nos pontos e valor das intensidades dos ecos descreve um objecto da melhor maneira.

5.4.1 Centroide do objecto

A primeira parte do descritor consiste em calcular o centroide do objecto recebido. Para tal a seguinte equação é usada:

$$\begin{aligned} c_x &= \sum px_i / n_{pontos} \\ c_y &= \sum py_i / n_{pontos} \end{aligned} \tag{5.13}$$

onde px_i e py_i é um determinado ponto no index i, c_x e c_y é o centroide calculado e n_{pontos} é o numero total de pontos de um determinado objecto. Para complementar o centroide anterior, é calculado também o centro de massa do objecto tendo em conta o valor das intensidades dos ecos recebidos.

$$\begin{aligned} cw_x &= \sum px_i * w_i / \sum w_i \\ cw_y &= \sum py_i * w_i / \sum w_i \end{aligned} \tag{5.14}$$

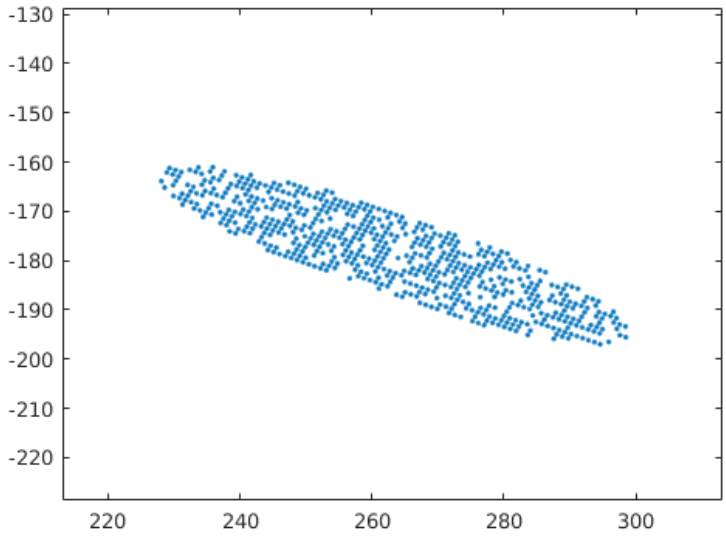


Figura 5.16: Exemplo de detecção de pontos de embarcação

onde w_i é o peso de um ponto no index i e cw_x e cw_y é o centro de massa tendo em conta o valor da intensidade de cada eco. Na situação em que todos os pontos têm um valor de eco igual, as equações 5.13 e 5.14 vão dar exactamente a mesma coisa. A equação 5.14 é uma melhor descrição do centro do objecto do que a equação 5.13.

5.4.2 Aproximação a uma elipse

Usar apenas o centroide para descrever um objecto não é a melhor forma de o descrever uma vez que este apenas exprime a sua posição. Uma aproximação possível para uma embarcação é uma elipse. Uma elipse é a visualização de uma distribuição gaussiana a duas dimensões. Qualquer distribuição gaussiana é caracterizada pelo seu valor médio e pela variância. No caso da distribuição gaussiana a 2D, este é caracterizada por duas médias, uma por cada dimensão, e uma matriz conhecida como matriz da covariância que estabelece a relação entre as diferentes dimensões[37][38]. A figura 5.16 é um exemplo de uma distribuição de pontos normalmente distribuídos para a representação de uma embarcação.

Para calcular a média desta distribuição a seguinte equação pode ser usada.

$$\begin{aligned} \bar{x} &= \tfrac{1}{n} \sum x_i \\ \bar{y} &= \tfrac{1}{n} \sum y_i \end{aligned} \tag{5.15}$$

onde $\bar{x}$ é a média da variável x, $\bar{y}$ é a média da variável y e n o número de pontos. A matriz de covariância a duas dimensões é dada por:

$$cov = \begin{bmatrix} var(x) & cov(xy) \\ cov(yx) & var(y) \end{bmatrix} \tag{5.16}$$

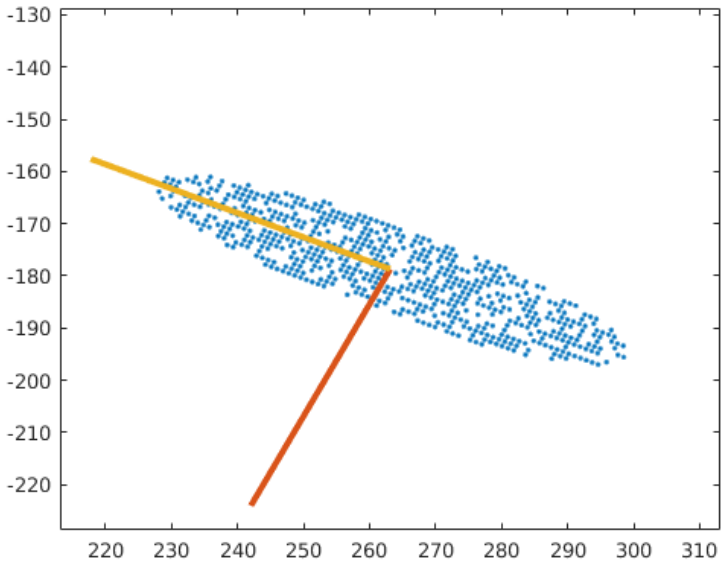


Figura 5.17: Representação de vectores próprios no conjunto de dados

onde $var(x)$ é a variância da variável x, $cov(xy)$ é covariância entre a variável x e y, $var(y)$ é a variância da variável y e $cov(yx)$ é covariância entre a variável y e x. Neste caso a $cov(xy)$ vai ser igual à $cov(yx)$. A variância é uma medida de quanto os valores se afastam do valor médio de uma qualquer variável e é data por:

$$var(x) = \frac{1}{n-1}\sum(x_i - \bar{x})(x_i - \bar{x}) \tag{5.17}$$

A covariância entre duas variáveis é relação entre elas, isto é, se a variável x mudar como é que a variável y vai mudar e é dada por:

$$cov(xy) = \frac{1}{n-1}\sum(x_i - \bar{x})(y_i - \bar{y}) \tag{5.18}$$

A matriz de covariância para o conjunto de dados anterior é:

$$cov = \begin{bmatrix} 314 & -138 \\ -138 & 81 \end{bmatrix} \tag{5.19}$$

Analisando a matriz de covariância anterior, em particular a $cov(xy)$, é possível afirmar que se a variável x aumentar, a variável y vai diminuir devido ao valor ser negativo. Na situação do valor ser positivo, se a variável x aumentar a variável y aumenta também. Analisando os campos $var(x)$ e $var(y)$ é possível afirmar que conjunto de pontos possuí uma maior variação em x do que em y. Como a matriz da covariância é uma matriz quadrada, o número de linhas é o mesmo que o número de colunas, é possível determinar os valores e vectores próprios. Estes são muito importantes porque determinam se existem

padrões no conjunto de dados[37][38]. Os valores próprios da matriz da covariância podem ser obtidos resolvendo a equação seguinte em ordem a λ:

$$det\left(\begin{bmatrix} var(x) & cov(xy) \\ cov(yx) & var(y) \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0 \tag{5.20}$$

obtendo os valores próprios, λ, é possível calcular os vectores próprios resolvendo a equação seguinte em ordem a v:

$$\left(\begin{bmatrix} var(x) & cov(xy) \\ cov(yx) & var(y) \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) v = 0 \tag{5.21}$$

Para cada valor próprio é necessário calcular os seus vectores próprios. Na situação da matriz de covariância a duas dimensões, vão existir dois valores próprios e dois vectores próprios. Os valores próprios, λ, e vectores próprios, v, da matriz de covariância 5.19 são respectivamente:

$$\lambda = \begin{bmatrix} 16 \\ 379 \end{bmatrix}, v = \begin{bmatrix} -0.42 & -0.90 \\ -0.90 & 0.42 \end{bmatrix} \tag{5.22}$$

Na figura 5.17 estão representados os dois vectores próprios obtidos da matriz de covariância. Como é possível ver, o vector a laranja, é o vector que está sobre o semieixo maior da elipse e o vector a vermelho está desfasado 90° e está sobre o semieixo menor. A amplitude do vector não representa a amplitude real do vector, sendo meramente ilustrativa.

O valor próprio maior, $\lambda_M = 379$, origina o vector próprio $v2 = [-0.90, 0.42]$ que se encontra sobre o semieixo maior e o valor próprio menor, $\lambda_m = 16$, origina o vector próprio $v1 = [-0.42, -0.90]$. Com o vector próprio maior, $v2$, é possível calcular o ângulo que este vector têm em relação ao eixo x por:

$$\theta = \arctan(\frac{v2_y}{v2_x}) = \arctan(\frac{0.42}{-0.90}) = 155° \tag{5.23}$$

Este ângulo é o ângulo da elipse em relação ao eixo x. A amplitude dos vectores para a representação da elipse é dado pelos valores próprios[37][38]. Deste modo a amplitude do maior vector é dada por:

$$a = \sqrt{\lambda_M} * \sqrt{\chi^2} \tag{5.24}$$

e a amplitude do menor vector é dada por:

$$b = \sqrt{\lambda_m} * \sqrt{\chi^2} \tag{5.25}$$

onde χ^2 é o valor da distribuição qui-quadrado[37][38]. Para um valor com 10% de incerteza e dois graus de liberdade o valor do χ^2 é de *4.605* segundo o anexo A. O valor calculado de a é de aproximadamente 41.8 metros e o valor de b é de 8.7 metros. A distribuição de pontos do radar não é uma distribuição gaussiana, pode ser aproximada a uma distribuição

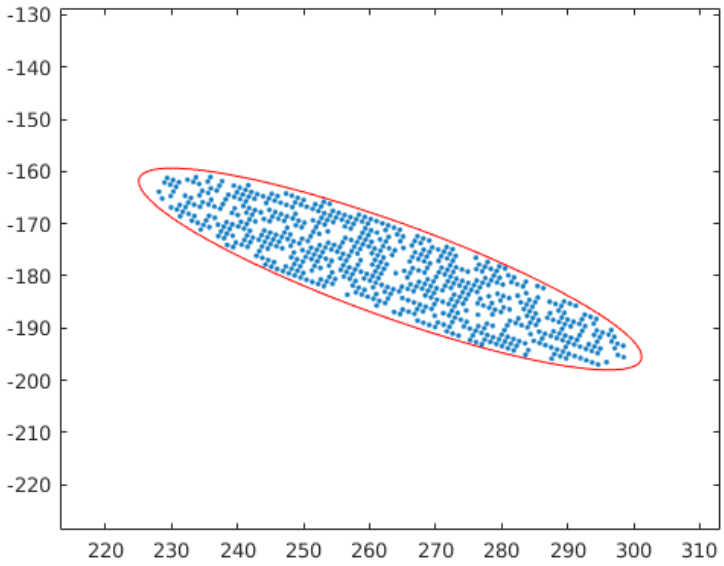


Figura 5.18: Resultado da aproximação a uma elipse

normal. Por este método é possível obter o centroide da elipse, o tamanho dos dois semieixos e o ângulo que a elipse tem sobre o eixo x. A figura 5.18 é o resultado do algoritmo da aproximação a uma elipse. A vermelho encontra-se a elipse estimada ($a = 41.8$, $b = 8.7$, $\theta = 155°$, $cx = 263.1$ e $cy = -178.7$).

5.4.3 Menor rectângulo

De modo a complementar a descrição da elipse, é utilizado um algoritmo que permite estimar o menor rectângulo possível que envolve o objecto.

Algoritmo 4 Menor rectângulo possível

1: *segmentos* ← Aplicar algoritmo de *convex hull*
2: $min_{area} \leftarrow \infty$
3: **for each** *segmento* em *segmentos* **do**
4: Calcular ângulo do *segmento* com eixo x
5: Rodar pontos sobre ângulo calculado
6: Descobrir ponto mais à esquerda, direita, cima e baixo
7: Calcular as coordenadas dos cantos e área
8: **if** área $< min_{area}$ **then**
9: $min_{area} \leftarrow$ área
10: Guardar coordenadas dos cantos
11: **end if**
12: **end for**

O algoritmo 4 é o algoritmo implementado para estimar qual o menor rectângulo possível que envolve um conjunto de pontos, também conhecido como *Minimum Bounding Box*. O primeiro passo é aplicar um algoritmo de *convex hull*. O algoritmo escolhido para o efeito foi o *Graham scan*[39]. O algoritmo de *Graham scan* permite obter um conjunto de segmentos que define a forma convexa de um determinado objecto.

Algoritmo 5 *Graham scan*

```
1: Escolher o ponto com o valor de y mais baixo
2: Ordenar pontos por ângulo em relação ao eixo x
3: while Pontos para verificar do
4:    Seleccionar 3 pontos consecutivos
5:    if Pontos fazem curva à direita then
6:       Descartar ponto do meio
7:    else
8:       Guardar ponto do meio
9:    end if
10: end while
```

O resultado do algoritmo 5, é uma lista de segmentos que define a forma convexa do objecto. A quantidade de segmentos existentes na lista é dependente da quantidade de pontos. Note-se que um segmento no algoritmo *Graham scan* é definido por dois pontos que é diferente dos segmentos do radar.

O segundo passo do algoritmo para obter o menor rectângulo possível, algoritmo 4, consiste em verificar qual o segmento que melhor forma o menor rectângulo. Para tal é seleccionado um segmento e calculado o ângulo que esse segmento faz com o eixo x (linha 4 do algoritmo 4). Todos os pontos são rodados pela equação 5.5, onde θ é o ângulo calculado anteriormente (linha 5 do algoritmo 4.

Com os pontos rodados, é necessário obter a posição do ponto mais à esquerda, direita, cima e baixo. Estes quatro pontos são os pontos mais afastados e que limitam o menor rectângulo possível nesta configuração. Com este pontos facilmente é possível calcular os 4 cantos do rectângulo e assim calcular a área do mesmo. Após ter os 4 pontos dos quantos, o objecto é novamente rodado para o referencial inicial e são guardados os 4 quantos, área, tamanho e ângulo calculados.

O mesmo processo é efectuado para os restantes segmentos. No final é verificado qual a configuração que apresenta menor área. A melhor configuração é guardada e as restantes descartadas.

A figura 5.19 é o resultado do algoritmo 4 para o conjunto de pontos anteriormente apresentado. Como é possível ver, o rectângulo apresentado é o rectângulo mais pequeno que cobre todos os pontos.

O descritor implementado recebe um conjunto de pontos e produz um conjunto de

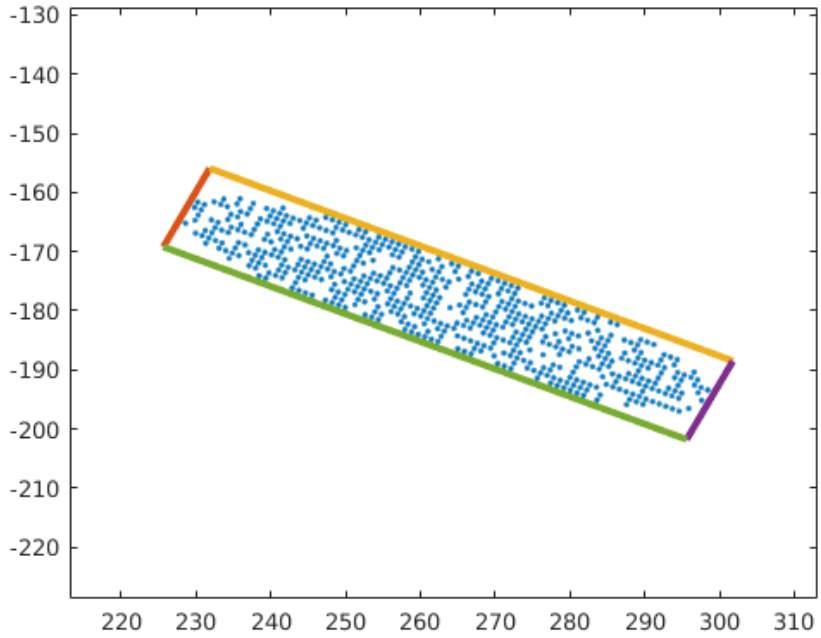


Figura 5.19: Menor Rectângulo Possível

características, *features*, que descrevem o objecto. As *features* de um objecto são:

- Centroide;
- Centroide ponderado;
- Elipse: Centroide, orientação e tamanho dos semieixos
- Rectângulo: Área, tamanho, centroide e orientação

Em função das características anteriores é possível estimar se um determinado objecto é parte da linha costeira ou se é uma embarcação. Para tal são comparadas as características da elipse com as do rectângulo. Numa situação perfeita, uma embarcação que seja descrita por uma elipse é também definida por um rectângulo com as mesmas características, isto é, o rectângulo circunscreve a elipse. Isto implica que o centro da elipse e o centro do rectângulo sejam coincidentes e o semieixo maior e menor sejam iguais aos lados do rectângulo.

Se os parâmetros da elipse e do rectângulo forem suficientemente parecidos é possível classificar o objecto com embarcação e como tal este deve ser seguido com o filtro. Caso exista uma disparidade entre as características da elipse e do rectângulo, é possível que seja um elemento pertencente à linha costeira.

Algoritmo 6 Extracção da linha costeira

```
1: extracção dos objectos mais próximos
2: for each objecto em objectos do
3:    extrair os pontos mais próximos
4: end for
5: criar uma lista de segmentos com os pontos
6: ordenar lista de segmentos por ângulo em relação ao eixo X
7: i = 0
8: while i != tamanho(segmentos) do
9:    if ângulo(segmento_i, segmento_{i+1}) < thr then
10:      segmento_i = [segmento_i(P_1) ; segmento_{i+1}(P_2)]
11:      remover segmento_{i+1}
12:   else
13:      i = i + 1
14:   end if
15: end while
```

5.5 Extracção da linha de costeira

Quando um objecto é classificado como parte da linha costeira é conveniente não descartar estes dados visto que poderão ser usados para criar um mapa que permite uma navegação segura. Foi implementado um nó de ROS descrito na figura 3.3 como "Extracção Fronteira"para conseguir extrair a linha costeira ao longo do tempo.

O algoritmo 6 é o algoritmo implementado para efectuar a extracção desta linha e consiste em criar uma lista de segmentos que definam um objecto, lista esta que se trata do conjunto dos pontos mais próximos do radar (linhas 2 a 5 do algoritmo 6).

Após obter uma lista de segmentos é necessário reduzir o tamanho dessa lista, visto que se um objecto for composto por 20000 pontos irão existir um conjunto elevado de segmentos. Reduzir o conjunto de segmentos pode implicar perder detalhe do objecto, contudo como se trata de uma fronteira de navegação ter detalhe do objecto não é um factor deveras importante.

As linhas 8 a 14 do algoritmo 6 efectuam o processo de minimização da quantidade de segmentos de um determinado objecto. Este processo consiste em verificar o ângulo efectuado entre dois segmentos consecutivos se é inferior a um determinado *threshold*. No caso de o ser, é eliminado o ponto comum aos dois segmentos e é criado um novo segmento com os restantes dois pontos, um do segmento 1 e outro do segmento 2. Um segmento é considerado uma recta que une dois pontos e descrito sobre a forma desses dois pontos.

O resultado do algoritmo proposto é uma lista de segmentos que define um objecto. A figura 5.20 é o resultado do algoritmo 6 aplicado num cenário real no porto de Leixões. A

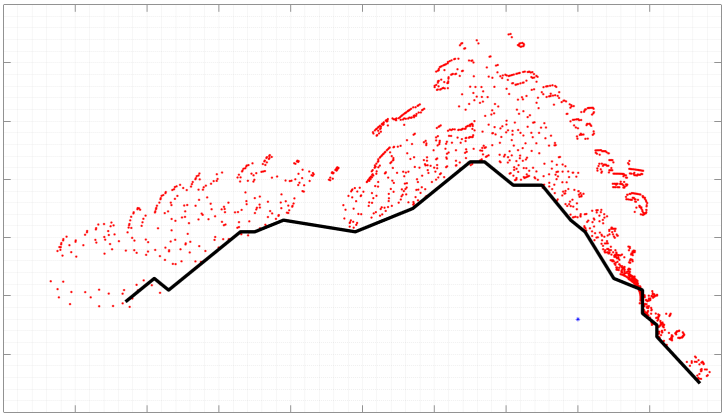


Figura 5.20: Linha costeira para um objecto localizado no porto de Leixões

azul está representada a posição do radar, localizado no ROAZ II, a vermelho os pontos detectados para um determinado objecto e a preto o conjunto de segmentos que definem o objecto.

Como é possível verificar, a linha perdeu algum do detalhe do objecto, porém é possível definir este objecto, inicialmente composto por 2000 pontos, por um conjunto de 19 segmentos que representam uma linha costeira de aproximadamente 350 metros. A linha costeira é apenas estimada no lado que se encontra mais perto do radar, visto ser o único lado aonde existe certeza de estar em água.

Capítulo 6

Monitorização de Obstáculos

Neste capítulo é efectuado um levantamento dos diferentes tipos de filtros existentes, é apresentada a implementação e calibração do filtro escolhido para diferentes alvos e por fim são apresentados resultados para dados simulados em Matlab.

6.1 Estimação de estados

Em 1960, R. E. Kalman apresentou um artigo cientifico intitulado de "*A New Approach to Linear Filtering and Prediction Problems*"[40] com o intuito de ultrapassar as limitações do filtro anteriormente apresentado por *Weiner-Holf*[41]. O filtro apresentado ficou conhecido como Filtro de Kalman. O filtro de Kalman trata-se de um algoritmo recursivo de processamento de informação que possibilita a estimação de estados de um sistema com ruído.

O filtro de Kalman é um estimador que permite obter os estados de um determinado sistema dinâmico tendo como base um modelo de previsão e um conjunto de medidas reais.

Existe um vasto conjunto de aplicações para as quais os filtro de Kalman é útil, das quais destacam-se: fusão de multi-sensorial, auto-localização e monitorização de objectos[42].

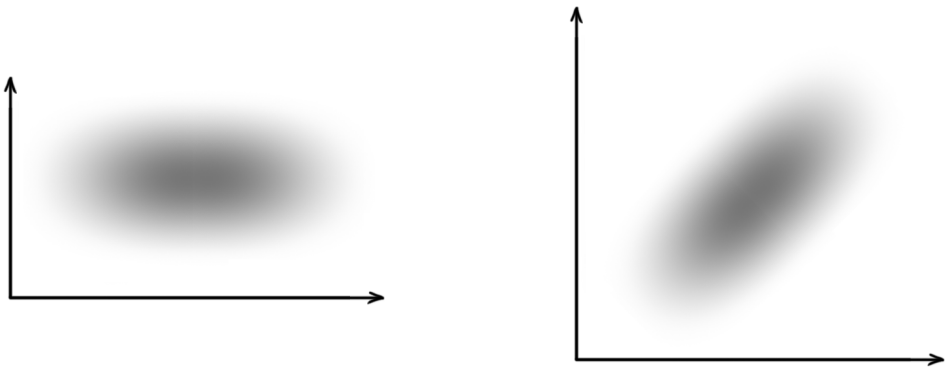


Figura 6.1: Exemplo de uma distribuição gaussiana a 2D

6.1.1 Formulação base do filtro de Kalman

O filtro de Kalman assume que todas as variáveis, estados, nele existentes são aleatórias e gaussianas. Cada variável tem um valor médio μ, que é o centro da distribuição, e uma variância σ^2 que retrata a incerteza. A figura 6.1(esquerda) é uma distribuição gaussiana a duas dimensões, onde cores mais escuras representam o valor mais provável e cores mais claras o menos provável. Na situação da figura 6.1(esquerda) é possível afirmar que as duas variáveis não estão correlacionadas, isto é, o estado de uma variável não influência a outra, como por exemplo posição e tamanho. Na figura 6.1(direita) as duas variáveis estão correlacionadas, isto é, mudanças numa variável afectam directamente a outra variável, como por exemplo posição e velocidade.

O filtro de Kalman linear assume que, tanto o estado como as medidas, são descritas por sistemas lineares, isto é, as relações entre estados podem ser descritas por equações lineares ao longo do tempo. Quando a relação entre diferentes estados dos sistema é não linear é necessário recorrer a outro tipo de filtro. É pressuposto que existe ruído branco tanto nas medidas como no próprio sistema, ruído este que apresenta média nula e uma distribuição gaussiana[40].

O filtro de Kalman original tem como fundamento um sistema de dinâmica linear em espaço de estados em tempo contínuo:

$$\begin{aligned} \dot{x_t} &= A \cdot x_t + B \cdot u_t \\ y_t &= C \cdot x_t \end{aligned} \tag{6.1}$$

onde $x(t)$ é o vector de estados do sistema com dimensão n, $y(t)$ é o vector de saídas do sistema com dimensão q, $u(t)$ é o vector de entradas do sistema com dimensão de p, A é a matriz que relaciona estados e tem dimensão de $n*n$, B é a matriz que relaciona as entradas com estados e tem dimensão de $p*n$ e C é a matriz que relaciona os diferentes estados com as saídas do sistema e tem dimensão de $q*n$. É possível converter o modelo de tempo contínuo apresentado anteriormente para um modelo equivalente em tempo discreto:

$$\begin{aligned} x_{k+1} &= A \cdot x_k + B \cdot u_k \\ y_k &= C \cdot x_k \end{aligned} \tag{6.2}$$

onde k é um número natural e representa o k-ésimo instante de amostragem.

Na versão estocástica, é necessário ter em consideração a incerteza na dinâmica do sistema, $w(k)$, bem com nas medidas, $v(k)$. O modelo estocástico é dado por:

$$\begin{aligned} x_{k+1} &= A \cdot x_k + B \cdot u_k + w_k \\ y_k &= C \cdot x_k + v_k \end{aligned} \tag{6.3}$$

De modo a garantir a convergência do método, w_k e v_k têm que ser processos Gaussianos com média nula e definidos por uma matriz diagonal:

$$R_w(i,k) = \begin{cases} Q(k) & se \quad i = k \\ 0 & se \quad i \neq k \end{cases} \tag{6.4}$$

$$R_v(i,k) = \begin{cases} R(k) & se \quad i = k \\ 0 & se \quad i \neq k \end{cases} \tag{6.5}$$

A matriz Q presente na equação 6.4, representa a confiança na estimativa para o estado adquirida através do modelo do sistema. A matriz R presente na equação 6.5, representa a confiança nas medidas dos sensores do sistema.

6.1.2 Inicialização do filtro de Kalman linear

O filtro de Kalman após inicializado executa acções de previsão e actualização efectuadas alternadamente. O filtro de Kalman linear é inicializado com uma estimativa inicial do estado, x_0, bem como a confiança na estimativa ou covariância inicial, P_0. O tempo de convergência do filtro está dependente da escolha do estado e covariâncias iniciais.

6.1.3 Previsão do filtro de Kalman linear

Na acção de previsão, o filtro de Kalman estima o vector de estados do sistema tendo em consideração o estado anterior e o modelo de dinâmica do sistema:

$$\widehat{x}_{k|k-1} = A_k \cdot \widehat{x}_{k-1|k-1} + B_k \cdot u_k \tag{6.6}$$

A melhor estimativa, após um instante de tempo, feita pelo filtro Kalman é a propagação do ultimo estado pelo modelo de dinâmica do sistema. Como mencionado anteriormente, o vector de estados do sistema tem um valor médio e uma matriz de covariância dada por:

$$P_{k|k-1} = A_k \cdot P_{k-1|k-1} \cdot A_k^T + Q_k \tag{6.7}$$

A primeira componente da equação anterior, $A_k \cdot P_{k-1|k-1} \cdot A_k^T$, é a propagação da incerteza do estado anterior para o estado actual e a segunda componente, Q_k, é o ruído associado ao modelo de propagação. Nas equações anteriores $\widehat{x}_{k|k-1}$ é o estado estimado na previsão, $P_{k|k-1}$ a covariância do estado estimado na previsão, $\widehat{x}_{k-1|k-1}$ estado utilizado na previsão e $P_{k-1|k-1}$ a covariância usada na previsão.

6.1.4 Actualização do filtro de Kalman linear

A acção de actualização do filtro de Kalman usa o estado e covariância estimados na previsão ($\widehat{x}_{k|k-1}$ e $P_{k|k-1}$) para calcular o novo vector de estados e matriz covariância. Esta acção apenas acontece quando existirem medidas provenientes dos sensores. Nem sempre as observações dos sensores correspondem directamente aos estados do sistema, sendo necessário a utilização de uma matriz que relacione as observações com o vector de estados. A matriz H no filtro de Kalman é a matriz que permite passar do espaço de estados para espaço de observações e é usada para o calculo da inovação:

$$\widetilde{y}_k \ = z_k - H_k \cdot \widehat{x}_{k|k-1} \tag{6.8}$$

onde $\widetilde{y}_k$ é a inovação, z_k a observação e $\widehat{x}_{k|k-1}$ o último estado conhecido do sistema. Note-se que $\widehat{x}_{k|k-1}$ usado na equação anterior é calculado na previsão dada pela equação 6.6. A inovação é uma medida que permite relacionar as observações com os últimos estados conhecidos. Com este medida é possível qualificar o filtro. Ao longo do tempo a inovação deverá ter valor médio nulo.

O filtro Kalman considera que as observações possuem ruído associado e que não são perfeitas, sendo necessário que o filtro de Kalman pondere as observações com o ultimo estado conhecido. Para este efeito é necessário calcular o ganho de Kalman:

$$K_k = P_{k|k-1} \cdot H_k^T \cdot S_k^{-1} \tag{6.9}$$

onde K_k é o ganho de Kalman, $P_{k|k-1}$ a covariância calculada na previsão e S_k a covariância da inovação que é dada por:

$$S_k = H_k \cdot P_{k|k-1} \cdot H_k^T + R_k \tag{6.10}$$

com R_k como a matriz de ruído associada ao modelo de observação(sensor e/ou processo). Tendo a inovação, o ganho de Kalman e a covariância da inovação é possível calcular o novo estado do sistema bem como a matriz de covariância:

$$\widehat{x}_{k|k} = \widehat{x}_{k|k-1} + K_k \cdot \widetilde{y}_k \tag{6.11}$$

$$P_{k|k} = (I - K_k \cdot H_k) \cdot P_{k|k-1} \tag{6.12}$$

onde $\widehat{x}_{k|k}$ é o estado estimado na actualização, $P_{k|k}$ a covariância do estado estimado na actualização e I a matriz identidade com o mesmo tamanho do número de estados do sistema.

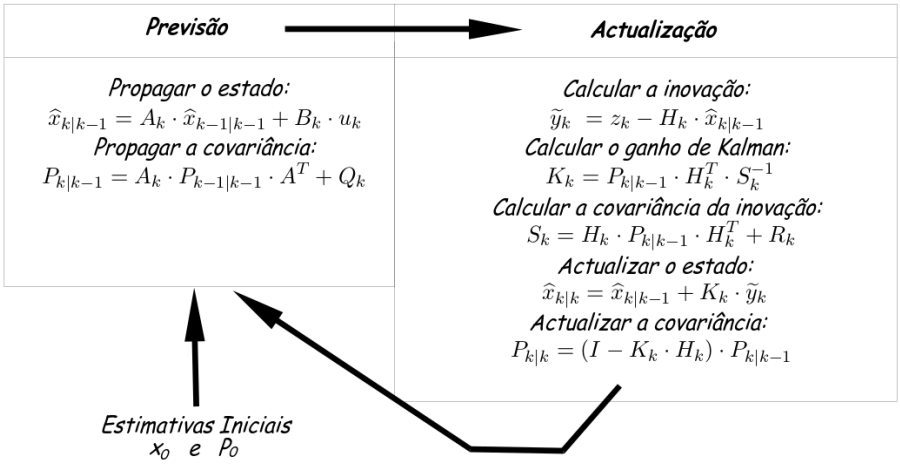


Figura 6.2: Resumo das acções do filtro de Kalman linear

A figura 6.2 é um resumo das diferentes acções a serem utilizadas aquando o uso de um filtro de Kalman linear. A saída da acção de actualização é entrada da acção previsão. Quando não existem medições, apenas a previsão é efectuada.

6.1.5 Filtro de Kalman não linear

Quando o modelo de dinâmica é não linear, o filtro de Kalman linear não pode ser usado. Para modelos não lineares foi criado o filtro de Kalman extendido, tratando-se de uma derivação do filtro de Kalman original. Este filtro tem em consideração que é possível linearizar os modelos de dinâmica e observação em torno do valor estimado, aplicando depois, as equações do filtro de Kalman original, como se de um sistema linear se tratasse. Os modelos da dinâmica e observação não precisam de ser lineares, porém necessitam de ser diferenciáveis.

Tal como no filtro de Kalman linear, a equação 6.6 pode ser usada da mesma forma, contudo para o calculo da covariância do estado é necessário primeiro obter a matriz de derivadas parciais do modelo de dinâmica do sistema (f)[42]:

$$F = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{6.13}$$

sendo f o modelo de dinâmica. A covariância estimada na previsão é dada por:

$$P_{k|k-1} = F_k \cdot P_{k-1|k-1} \cdot F_k^T + Q_k \tag{6.14}$$

Na acção de actualização é necessário calcular a matriz das derivadas parciais do modelo de observação (h):

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_n}{\partial x_1} & \frac{\partial h_n}{\partial x_2} & \cdots & \frac{\partial h_n}{\partial x_n} \end{bmatrix} \tag{6.15}$$

Diferente do filtro de Kalman linear, este usa a matriz acima descrita para o calculo da covariância da inovação, usado no ganho do filtro de Kalman e consequentemente na covariância do estado. O filtro de Kalman extendido faz uso dos modelos de dinâmica e observação para o cálculo do estado do sistema e a matriz de derivadas parciais para a covariância do estado. Existem outras variantes do filtro de Kalman para sistemas não lineares assim como outros tipos de filtros.

6.1.6 Calibração do filtro de Kalman

Um estimador de estados é considerado consistente se os erros associados à estimação de estados satisfazerem as seguintes equações:

$$\begin{aligned} E\left[\widetilde{x}(k|k)\right] &= 0 \\ E\left[\widetilde{x}(k|k)\widetilde{x}(k|k)'\right] &= P(k|k) \end{aligned} \tag{6.16}$$

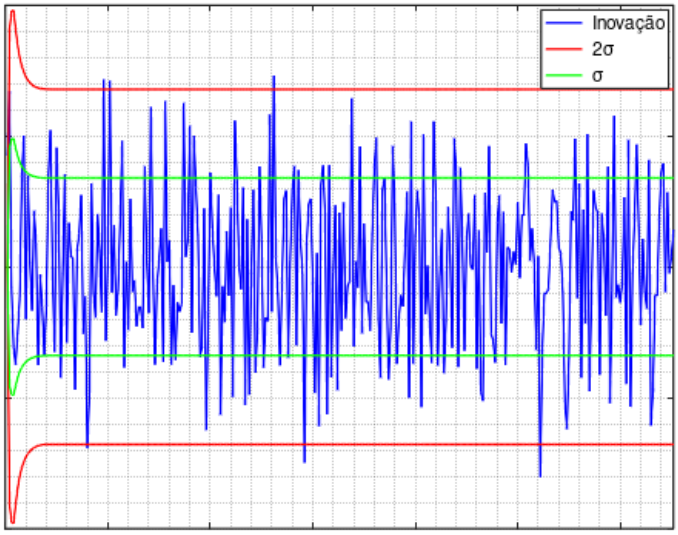


Figura 6.3: Exemplo de sequência da inovação com teste de covariância

É possível definir três critérios de consistência do um filtro[43]:

- O erro no estado deve possuir média nula e a sua magnitude deve ser consistente com a covariância calculada pelo filtro;
- A inovação deve partilhar da propriedade anterior;
- A inovação deve ser aceite como ruído branco.

Em cenários reais, apenas os dois últimos critérios podem ser validados. Para poder validar o primeiro critério é necessário ter o *ground truth* de todos os estados ao longo do tempo. Este tipo de situações é apenas possível alcançar em cenários totalmente controlados, isto é, em simulação. O primeiro critério é o mais importante dos três visto que relaciona os estados estimados com os estados reais.

Cumprir com os testes acima descritos é fundamental para validar a qualidade e convergência do filtro. Estes testes permitem, de forma analítica, calibrar as duas matrizes de ruído existentes no filtro (Q e R). Valores elevados na matriz de ruído do modelo de observação, implicam que existe muita incerteza na medida e consequentemente apenas o modelo de previsão irá ter efeito. Valores muito baixos oferecem total confiança no modelo de observação descartando completamente a parte de previsão. É necessário existir um balanceamento dos valores utilizados nas matrizes.

De modo a validar se a magnitude da inovação é consistente com a sua covariância é necessário verificar se esta se encontra 95% dentro dos limites estabelecidos pela covariância. A figura 6.3 é um exemplo de uma sequência de inovação que se encontra consistente com a sua covariância. A azul encontra-se a magnitude da inovação, a vermelho o limite $\pm 2\sigma$ e a verde o limite $\pm\sigma$.

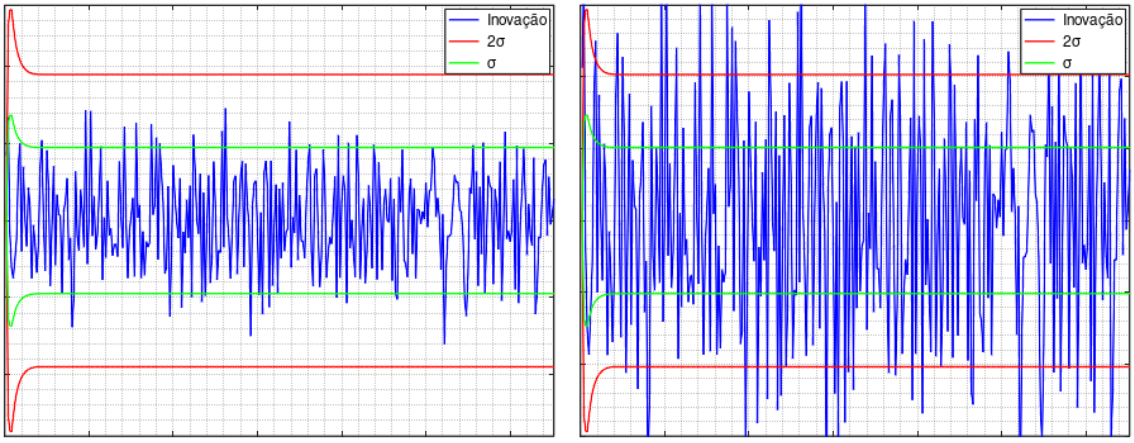


Figura 6.4: Exemplo de sequência da inovação com teste de covariância

A figura 6.4 é um exemplo da inovação ao longo do tempo para um determinada variável. À esquerda é possível observar que a inovação encontra-se bem dentro dos limites desejáveis. À direita encontra-se bastante fora. É desejável que a magnitude da inovação se encontre dentro dos limites $\pm 2\sigma$ em 95% [44]. Se a magnitude estiver totalmente dentro do limite significa que ocorreu um erro muito baixo entre o estado previsto e o estado estimado podendo estar a descartar completamente a medida. Pela análise da figura 6.4 esquerda, é possível concluir que a matriz de ruído associada as observações (R) está sobrestimada.

Na situação da figura da direita, a magnitude da inovação ocorre muitas vezes fora do limite, $\pm 2\sigma$. Isto significa que existe um erro muito grande entre o ultimo estado estimado e a medida. É possível concluir que a matriz de ruído associada as observações (R) está a ser subestimada. Idealmente é desejável que exista um equilibro entre a magnitude da inovação e a covariância.

Sob a hipótese de o filtro ser consistente é possível definir inovação normalizada do inglês *Normalised Innovation Squared Test* (NIS) como:

$$\epsilon = \widetilde{y}' \cdot S^{-1} \cdot \widetilde{y} \tag{6.17}$$

onde $\widetilde{y}_k$ é a inovação, S_k a covariância da inovação e ϵ_k a inovação normalizada que segue uma distribuição qui-quadrado com n_z graus de liberdade onde n_z é a dimensão das medidas do sistema.

Com a inovação normalizada é possível calcular uma média, média esta que é actualizada ao longo do tempo com novas medidas. Este processo é vulgarmente conhecido como media deslizante e é dada por:

$$\overline{\epsilon} = \frac{1}{N} \sum_{i=1}^{N} \epsilon(i) \tag{6.18}$$

onde N é n-ésima medida. A média deslizante é então testada com o teste do qui-quadrado

e é aceite se:

$$\overline{\epsilon} \in [r_1, r_2] \tag{6.19}$$

onde o intervalo de aceitação é definido por:

$$P\left\{N\overline{\epsilon} \in [r_1, r_2] \mid H_0\right\} = 1 - \alpha \tag{6.20}$$

A confiança da distribuição qui-quadrado é dada por $1 - \alpha$. A figura 6.5 é um exemplo do teste da hipótese acima apresentado com $\alpha = 2.5\%$. A azul encontra-se a inovação normalizada, a vermelho a média deslizante, a verde o limite para $\alpha = 0.975$ e a preto o limite para $\alpha = 0.025$. Como é possível ver na figura o valor médio encontra-se dentro do intervalo de aceitação $[0.742, 1.296]$ podendo assim aceitar a hipótese deste cenário.

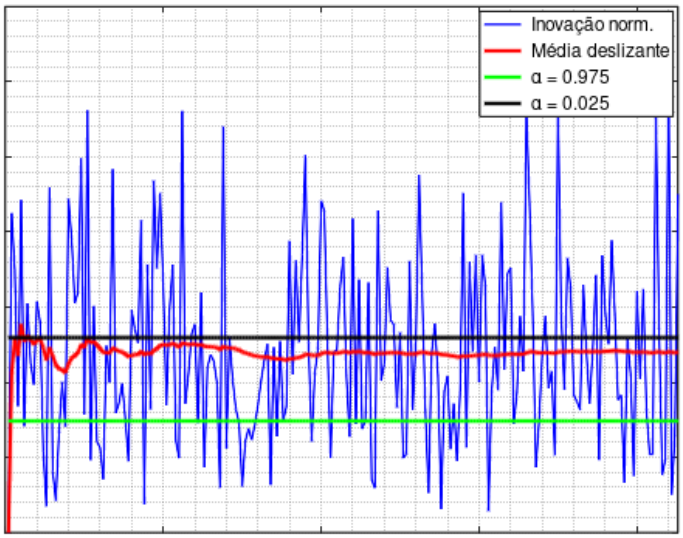


Figura 6.5: Inovação normalizada com teste de qui-quadrado

Para testar se a inovação é ruído branco é possível avaliar a auto correlação da inovação. A figura 6.6 é um exemplo da auto correlação de uma determinada sequência de inovação. Neste teste é desejável que exista apenas um pico, alta amplitude, em $t = 0$ e ser aleatoriamente distribuído em todo o restante sinal.

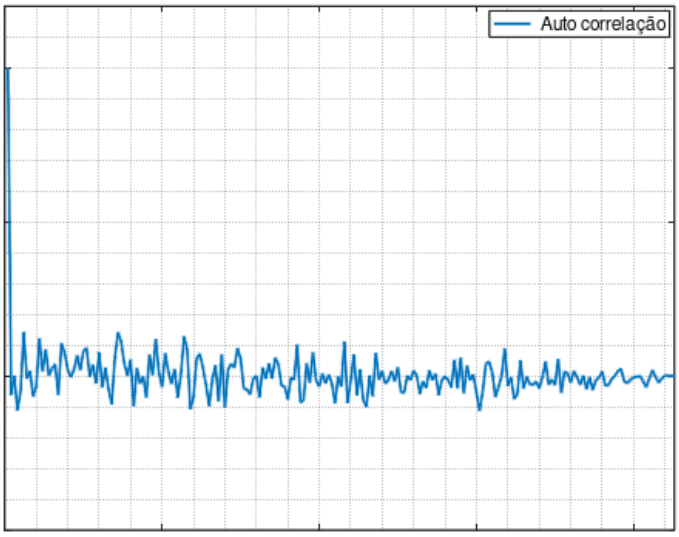


Figura 6.6: Auto correlação da inovação

Com os testes mencionados anteriormente é possível calibrar o filtro de Kalman de forma analítica conseguindo alcançar a convergência do filtro.

De forma a simplificar o processo de calibração usando os testes anteriores, é possível aplicar um método que permite extrair os parâmetros relacionados com o modelo de observação. Os parâmetros presentes na matriz de ruído associada ao modelo de observação consistem num conjunto de desvios padrões que descrevem a incerteza do sensor e do processo.

É possível estimar o desvio padrão de um determinado sensor e/ou processo usando o seguinte procedimento:

- Obter n amostras de um objecto invariante no tempo;
- Calcular o valor médio das n amostras;
- Estimar o desvio padrão para cada amostra;
- Calcular o desvio padrão médio das n amostras.

Para o procedimento anterior poder retratar com confiança o desvio padrão de um determinado sensor e/ou processo é necessário ter um número considerável de amostras de um determinado objecto.

A combinação dos processos apresentados em [43][44] com o procedimento anterior permitem calibrar o filtro de Kalman de uma forma relativamente fácil. Numa possível situação de não ser possível calibrar a matriz de ruído do modelo de observação do filtro por incapacidade de obtenção das n amostras desejadas, é ainda assim possível calibrar o filtro de Kalman através dos métodos estatísticos apresentados inicialmente. *Reid* [44] apresenta um manual de calibração do filtro de Kalman que em função dos resultados dos testes estatísticos apresentados sugere que mudanças devem ser efectuadas.

6.2 Implementação do filtro de Kalman

6.2.1 Formulação do filtro

O tipo de embarcações a serem seguidas apresentam uma dinâmica lenta, isto é, a mudança da velocidade e de orientação das embarcações não é abrupta. Tendo em consideração a dinâmica das embarcações é possível aproximar o seu movimento ao longo do tempo por equações lineares:

$$\begin{aligned} x_k &= x_{k-1} + vx \cdot dt \\ y_k &= y_{k-1} + vy \cdot dt \end{aligned} \tag{6.21}$$

onde $[x_k \quad y_k]$ é a posição da embarcação no instante k, $[x_{k-1} \quad y_{k-1}]$ a posição da embarcação no instante $k-1$, $[vx \quad vy]$ a velocidade no instante k e dt o tempo decorrido entre o instante de tempo k e $k-1$. Da equação anterior podem ser extraídos quatro parâmetros que necessitam de ser monitorizados pelo filtro de Kalman. A orientação e o tamanho da embarcação são também parâmetros relevantes que podem ser estimados pelo filtro, visto que o classificador anteriormente apresentado consegue os estimar de um conjunto de pontos. Assim sendo, o vector de estados é definido por:

$$\widehat{x} = \begin{bmatrix} x & y & vx & vy & \theta & sa & sb \end{bmatrix} \tag{6.22}$$

com $\widehat{x}$ sendo vector de estados, θ a orientação, sa o tamanho da embarcação sobre o eixo maior e sb o tamanho da embarcação sobre o eixo menor. No sistema de equações seguinte, estão presentes as equações permitem relacionar o vector de estados na *frame* anterior com o novo vector de estados, *frame* actual, no processo de previsão.

$$\begin{cases} x_k &= x_{k-1} + vx_{k-1} \cdot dt \\ y_k &= y_{k-1} + vy_{k-1} \cdot dt \\ vx_k &= vx_{k-1} \\ vy_k &= vy_{k-1} \end{cases} \quad \begin{cases} \theta_k &= \theta_{k-1} \\ sa_k &= sa_{k-1} \\ sb_k &= sb_{k-1} \end{cases} \tag{6.23}$$

Para o processo de previsão do filtro de Kalman linear, é necessário definir a matriz A que se encontra na equação 6.6. Este matriz é responsável por propagar o estado anterior para o estado actual do sistema e é definida segundo a equação 6.23:

$$A = \begin{bmatrix} 1 & 0 & dt & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & dt & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.24}$$

Na equação 6.6, encontra-se também presente a matriz B. Esta matriz relaciona as entradas de dados no processo de previsão com o vector de estados. Esta matriz é omitida na implementação por não existir a necessidade de dar *inputs* ao filtro no processo da previsão. A matriz Q usada para o calculo da covariância no passo de previsão é dada por:

$$Q_k = \begin{bmatrix} \sigma_x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_y & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{vx} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{vy} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\theta & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{sa} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{sb} \end{bmatrix} \tag{6.25}$$

onde cada elemento na diagonal principal da matriz corresponde ao respectivo estado. A escolha de Q é deveras importante para a convergência e qualidade do filtro.

O passo de actualização faz uso dos resultados obtidos pelo classificador de objectos. Este classificador descreve um objecto da melhor maneira através do centroide, orientação e tamanho. Na equação 6.8 a componente z_k é a observação feita pelos sensores e é definida por:

$$z_k = \begin{bmatrix} cw_x & cw_y & \theta & a & b \end{bmatrix} \tag{6.26}$$

onde cw_x e cw_y é o centroide, θ a orientação e s_x e s_y o tamanho do semieixo maior e menor, respectivamente. O vector de observações, z_k, não possuí todos os estados presentes no vector de estados, visto que a velocidade é um estado não observável directamente. A matriz H, ou modelo de observação, presente na equação 6.8, é a matriz que permite mapear o vector de estados do sistema num vector de observação e é definida por:

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.27}$$

O modelo de observação do sensor possuí uma matriz que define o ruído associado ao sensor e processo:

$$R_k = \begin{bmatrix} \sigma_{cw_x} & 0 & 0 & 0 & 0 \\ 0 & \sigma_{cw_y} & 0 & 0 & 0 \\ 0 & 0 & \sigma_\theta & 0 & 0 \\ 0 & 0 & 0 & \sigma_a & 0 \\ 0 & 0 & 0 & 0 & \sigma_b \end{bmatrix} \tag{6.28}$$

Esta matriz é usada para o calculo da covariância no passo da actualização e retrata a incerteza do sensor e/ou processo.

6.2.2 Filtro multi-alvo

O primeiro passo na implementação de um filtro multi-alvo é conseguir ter um algoritmo capaz de corresponder uma observação com uma lista de alvos já conhecidos. Foi desenvolvido um nó de ROS que sempre que receber um alvo do classificador tem a função de o corresponder com a lista de alvos já conhecida. Tendo em conta que o filtro de Kalman considera que todos os estados são variáveis Gaussianas, é possível usar o valor médio e a incerteza de cada variável para verificar se um determinado alvo pertence ao filtro.

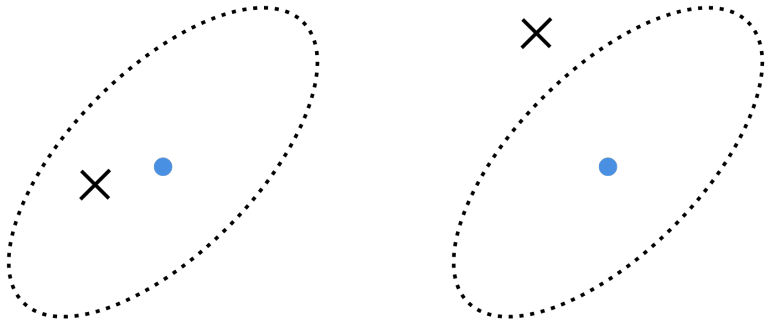


Figura 6.7: Exemplo do algoritmo para correspondência do filtro

A figura 6.7 é um exemplo da representação de duas variáveis gaussianas de um determinado filtro, aproximação a uma elipse. A azul encontra-se o valor médio, a tracejado a incerteza e a cruz preta representa a observação. Na figura da esquerda, a observação ocorreu dentro da área de incerteza das variáveis. Nesta situação, é considerado que a observação corresponde ao filtro. Na figura da direita, a observação acontece fora da área de incerteza das variáveis e é considerado que a observação não corresponde ao filtro. A algoritmo 7 é o algoritmo implementado para efectuar correspondência de um alvo com uma lista de filtros. Apenas é apresentada a verificação de dois estados para efeitos de simplificação. Na implementação é efectuada a verificação para todos os estados observados.

No algoritmo, μ e σ são o valor médio e a incerteza da variável, correspondentemente, do último estado conhecido do filtro. É efectuada a verificação para todos os filtros já conhecidos e na situação do alvo não corresponder a nenhum filtro, é criado um novo filtro e inicializado com o alvo.

Numa primeira fase, a implementação do filtro multi-alvo foi feita com recurso ao *software* Matlab. De forma a facilitar a implementação do filtro em C++ com integração com ROS, o filtro multi-alvo foi implementado como uma classe. Para cada novo alvo é criado um objecto, instância da classe, e inicializado com as características conhecidas do alvo. Como se trata de um objecto, é possível definir diferentes matriz de ruído para diferentes alvos, pelo simples motivo de existir independência entre diferentes objectos. Um alvo de tamanho considerável, como por exemplo um porta-contentores, possuí uma

Algoritmo 7 Algoritmo de correspondência de filtro

```
1: for each filtro em filtros do
2:    if μ_x − σ_x < cw_x < μ_x + σ_x then
3:       if μ_y − σ_y < cw_y < μ_y + σ_y then
4:          [...]
5:          alvo pertence a filtro
6:          [...]
7:       end if
8:    end if
9: end for
10: if alvo não pertence a nenhum filtro em filtros then
11:    criar novo filtro com alvo
12: end if
```

dinâmica diferente de uma lancha, sendo possível definir diferentes matrizes de ruído associadas ao modelo de propagação, Q, para diferentes alvos.

Na situação de um determinado alvo não apresentar perigo para o ROAZ II, isto é, encontra-se longe o suficiente para não existir perigo de colisão, é possível eliminar o objecto da lista de filtros a serem monitorizados. O processo de averiguar a existência de possibilidade de colisão é apresentado no fim do capitulo 6.

6.3 Simulador

De forma a poder validar o filtro multi-alvo, foi desenvolvido um simulador recorrendo ao *software* Matlab. Este simulador possibilita a criação de diversos cenários, para múltiplos alvos, permitindo que utilizador defina os perfis de velocidade e orientação ao longo do tempo. Em função dos perfis utilizados é calculada uma rota bem como as observações, aproximadas, que seriam retornadas pelo radar.

O simulador desenvolvido consiste em três processos: gerar perfis de velocidade e orientação, calcular trajectória em função dos perfis e gerar observações de acordo com o radar *Lowrance 3G*. Na primeira fase, são gerados o perfil de velocidade e o perfil de orientação ao longo do tempo, perfis este que poderão ser aleatórios ou configurados.

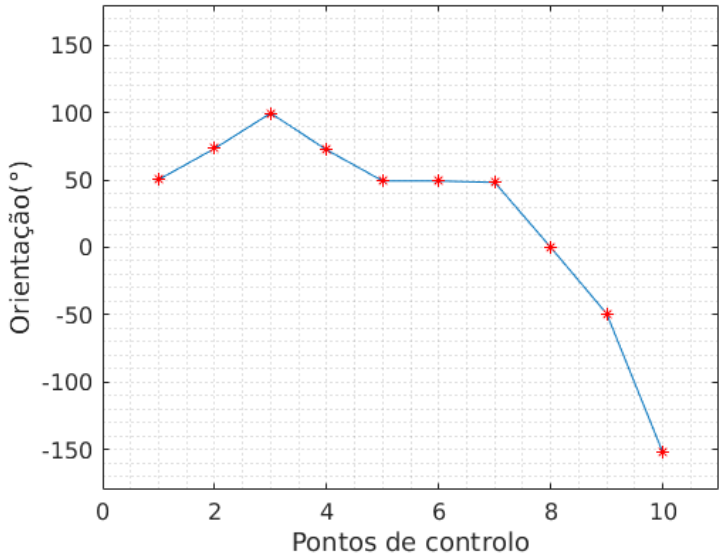


Figura 6.8: Exemplo de perfil de orientação para uma embarcação

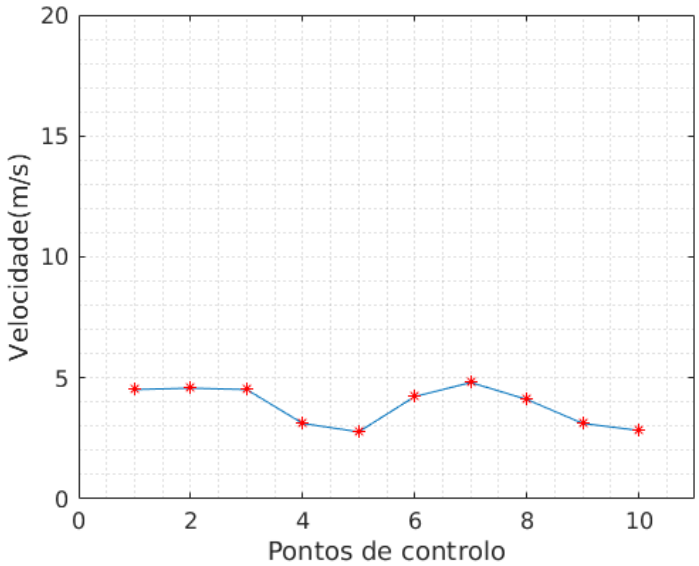


Figura 6.9: Exemplo de perfil de velocidade para uma embarcação

As figuras 6.8 e 6.9 são um exemplo dos perfis de orientação e velocidade, respectivamente, que poderão ser gerados com o simulador. É possível gerar uma variedade imensa de perfis conseguindo deste modo testar o filtro em diferentes cenários. No caso dos perfis apresentados anteriormente, estes possuem 10 pontos de controlo (pontos representados pelo asterisco vermelho). A quantidade de pontos de controlo é um parâmetro configurável no simulador.

Após criação dos perfis de velocidade e orientação é possível efectuar o segundo passo

do processo de simulação, geração de uma trajectória. Para tal é possível definir o número total de pontos que a trajectória possuí, sendo este parâmetro configurável. Dado que o número total de pontos é bastante maior do que o número de pontos de controlo, é necessário usar algum tipo de mecanismo que permita descrever os perfis de velocidade e orientação, definidos com poucos pontos, com um elevado conjunto de pontos. O processo usado para tal é conhecido com interpolação linear. Este processo permite gerar n pontos igualmente espaçados através de dois pontos de controlo.

Os novos perfis de velocidade e orientação são então guardados para posterior análise, sendo estes considerados como *ground truth*. Com os perfis anteriores é possível gerar a rota do alvo. Para tal a seguinte equação é usada:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} + step \cdot v_{i-1} \cdot \begin{bmatrix} cos(\theta_{i-1}) \\ sin(\theta_{i-1}) \end{bmatrix} \tag{6.29}$$

onde x_i e y_i é o nova posição, x_{i-1} e y_{i-1} a última posição, $step$ o tempo decorrido entre duas posições consecutivas, v_{i-1} a velocidade e θ_{i-1} a orientação no instante anterior. O ponto inicial da trajectória pode ser definido ou gerado aleatoriamente. Esta trajectória é guardada para posterior análise.

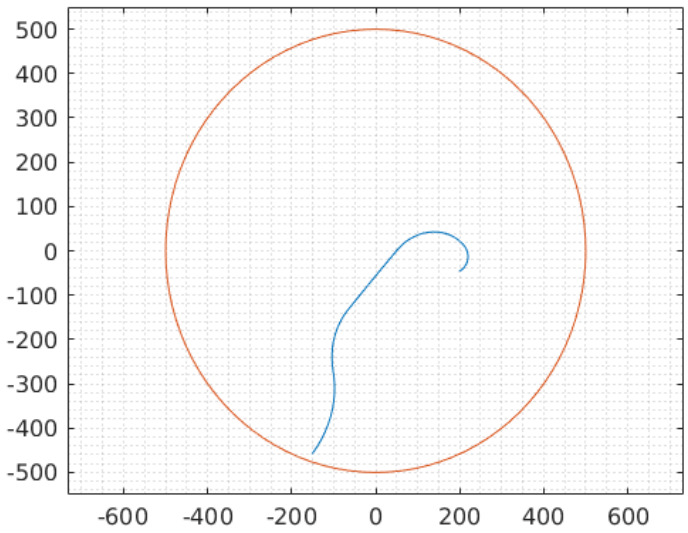


Figura 6.10: Exemplo de trajectória gerada em função dos perfis anteriores

A figura 6.10 é o resultado de uma trajectória gerada em função dos perfis de velocidade e orientação anteriormente apresentados, onde o ponto inicial é $\begin{bmatrix} -150 & -450 \end{bmatrix}$. Na figura, a trajectória é representada pela linha azul e o círculo vermelho define o alcance máximo configurado do radar de 500 metros. Como é possível observar, a trajectória apresentada vai de acordo com os perfis de velocidade e orientação inicialmente escolhidos.

Para cada posição guardada existe uma probabilidade, parâmetro configurável, de serem geradas observações de forma a introduzir incerteza para o filtro tornando assim os dados mais próximos da realidade. Na situação da trajectória gerada sair fora do alcance máximo do radar, são eliminados todos os pontos que ficam fora do alcance máximo.

O último passo do simulador é gerar um conjunto de observações para cada posição definida anteriormente. Como as embarcações a serem seguidas pelo filtro aproximam-se a elipses, é possível gerar um conjunto de observações que se aproximem a tal. De modo a aproximar as observações simuladas às observações reais, é necessário estimar o tamanho dos segmentos reais do radar. Cada segmento pode ser aproximadamente definido por um rectângulo com tamanho definido por:

$$\begin{aligned} h &= max_{range}/512 \\ l &= 2 \cdot \pi \cdot r/2048 \end{aligned} \tag{6.30}$$

onde h é a altura do segmento, l a largura do segmento, max_{range} o alcance máximo configurado, 512 o número total de segmentos numa *scanline*, 2048 o número máximo de ângulos possíveis e r a distância do radar ao segmento em questão e é dada por:

$$r = \sqrt{dx^2 + dy^2} \tag{6.31}$$

Através da equação 6.30 é possível aproximar um segmento de radar a um rectângulo. Tendo em consideração o tamanho aproximando de um segmento real, é possível gerar um conjunto de segmentos que se aproximam ás observações do radar *Lowrance 3G*. A intensidade do eco é definida em função da distância de cada ponto ao centro do objecto, isto é, um ponto que esteja muito próximo do centro do objecto possuí uma intensidade baixa e um ponto na fronteira do objecto possuí uma intensidade alta. Os valores de intensidade não estão de acordo com os valores de intensidade retornados pelo radar.

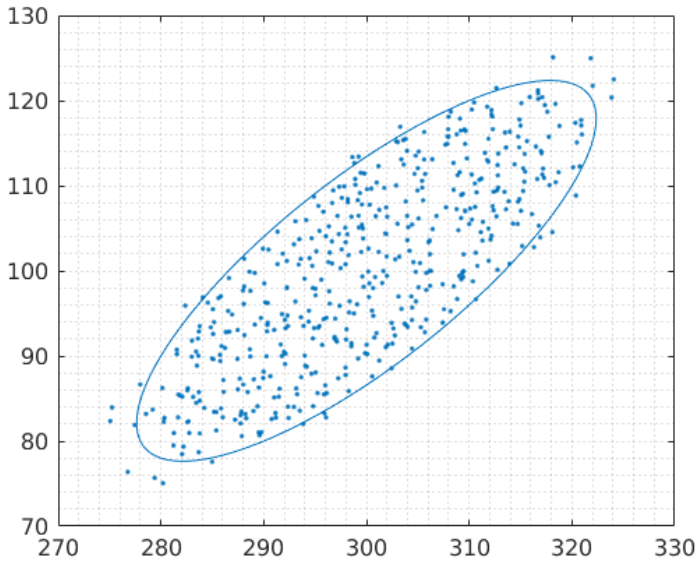


Figura 6.11: Exemplo de observação simulada

Na figura 6.11 os pontos azuis representam observações com ruído adicionado e a azul a elipse que representa a embarcação. O tamanho dos pontos apresentados não representa o tamanho real do segmento. A quantidade total de pontos está totalmente dependente do tamanho da embarcação bem como da sua distância ao radar. É possível configurar a probabilidade de um determinado ponto ser observado ou não e a percentagem de ruído máximo adicionado (normalmente distribuído).

A tabela seguinte é o resumo de todos os parâmetros que é possível configurar no simulador:

Tabela 6.1: Parâmetros configuráveis no simulador

Parâmetros	Unidades
número total de alvos	–
número de pontos de controlo	–
número total de pontos	–
step da simulação	s
posição inicial (x, y)	m
alcance máximo	m
orientação	$^{\circ}$
velocidade (v_x, v_y)	m/s
probabilidade de falhas de observação	–
probabilidade de ponto existir	–
percentagem de ruído máximo adicionado	–

6.4 Resultados da implementação do filtro de Kalman

As simulações efectuadas geram uma trajectória ao longo do tempo bem como as respectivas observações. As trajectórias geradas para a validação do filtro são criadas em função dos perfis de velocidade e orientação facultados pelo utilizador e é adicionado ruído nos elementos gerados na trajectória: posição, orientação e tamanho da embarcação. Os erros máximos adicionados são dados por:

- Erro máximo na posição (x e y) - 5 m;
- Erro máximo na orientação - 3 °;
- Erro máximo no tamanho da embarcação (eixo maior) - 5 m;
- Erro máximo no tamanho da embarcação (eixo menor) - 2 m.

O ruído adicionado é ruído normalmente distribuído com valores máximos como os descritos anteriormente. Com este ruído associado é possível obter uma trajectória aceitável para a validação do filtro de Kalman. Foram geradas diferentes trajectórias para diferentes cenários.

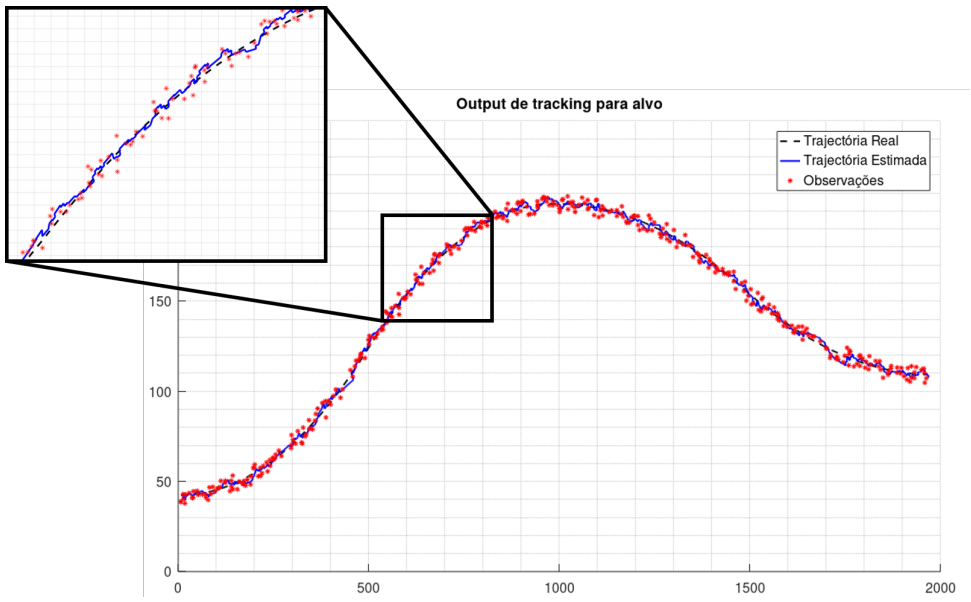


Figura 6.12: Exemplo de output do filtro de Kalman para uma embarcação - cenário I

A figura 6.12 é um exemplo de uma trajectória simulada. Ao total foram simulados aproximadamente 17 minutos de trajectória. Como é possível averiguar na figura anterior, a estimação da posição ao longo do tempo do filtro de Kalman consegue se aproximar na trajectória real efectuada para embarcação. Na ampliação da figura 6.12 é possível verificar que as observações do radar encontra-se com bastante ruído e que o filtro de Kalman consegue não dar demasiada confiança a estas. O erro da posição estimado em

relação à trajectória real é de aproximadamente 4.11 m tendo este sido calculado pela distância da posição estimada em relação à respectiva posição real. A inovação para posição possuí valor médio nulo o que indica que os testes demonstrados na secção 6.1.6 estão de acordo com os resultados obtidos.

A estimação da posição e do tamanho da embarcação foram também bem sucedidos para este cenário. O erro médio da orientação estimada ronda os 1.12 ° e do tamanho da embarcação os 1.66 m. Ambos os estados estão de acordo com os testes para calibração do filtro.

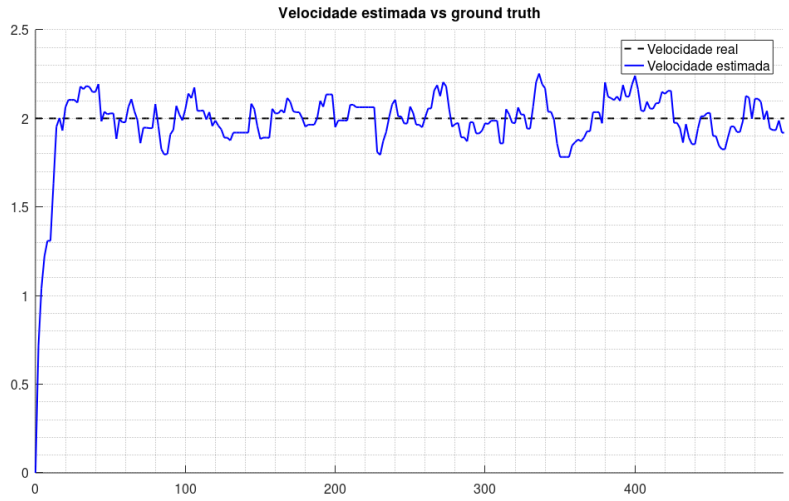


Figura 6.13: Velocidade estimada pelo filtro de Kalman para o cenário da figura 6.12

A figura 6.13 é o resultado da velocidade estimada pelo filtro de Kalman para o cenário da figura 6.12. A velocidade simulada para este cenário é constante, aceleração nula. Como é possível verificar na figura 6.13 a velocidade estimada pelo filtro possuí um erro baixo aproximando-se do valor simulado. A velocidade é o único estado do filtro que não é possível observar sendo que a diferença entre a velocidade estimada e a real ronda os 0.14 m/s. Para este cenário foi considerado que o radar falha observações em 30% das vezes.

Foi também testado o cenário do radar falhar 50% das observações. A figura 6.14 é o exemplo de uma simulação aonde ocorreu falhas do radar em 50% das vezes e com velocidade variável ao longo do tempo.

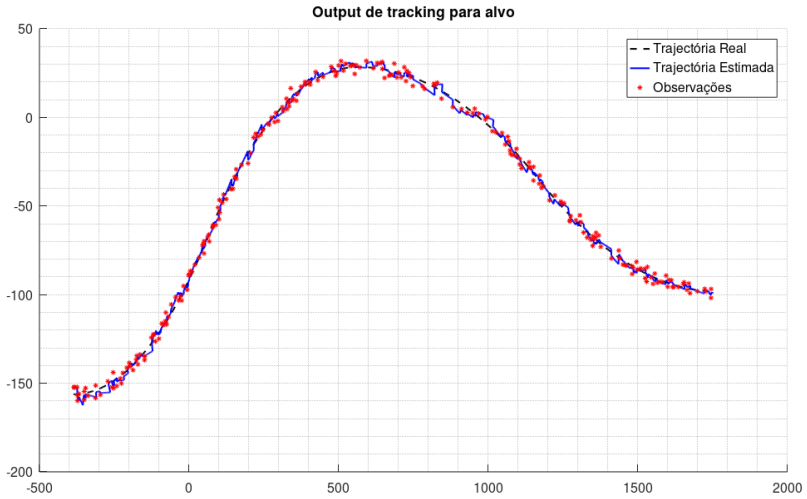


Figura 6.14: Exemplo de output do filtro de Kalman para uma embarcação - cenário II

Como é possível ver na figura 6.14 a estimação da trajectória ao longo do tempo continua a fornecer resultados aceitáveis. Neste cenário em a diferença entre a trajectória estimada e a trajectória real tem um erro médio de 4.93 m sendo um pouco superior ao erro da simulação da figura 6.12 mas ainda assim aceitável.

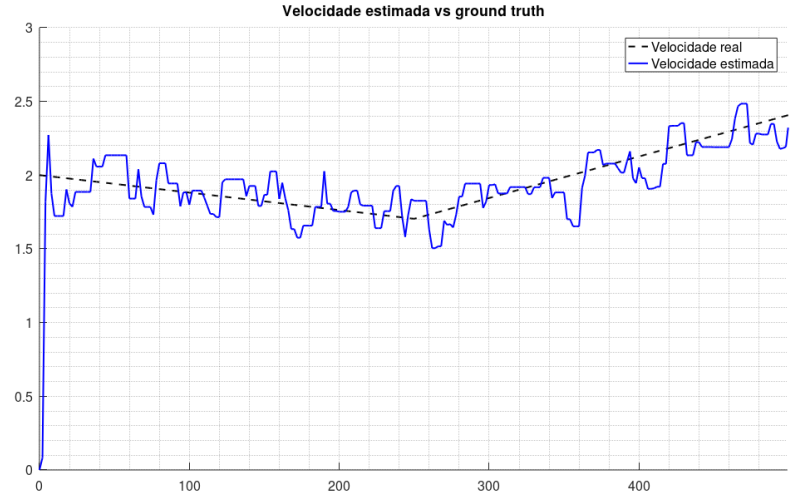


Figura 6.15: Velocidade estimada pelo filtro de Kalman para o cenário da figura 6.14

A figura 6.15 é o resultado da estimação da velocidade para o cenário da figura 6.14. Neste caso existe uma variação da velocidade ao longo do tempo e o valor médio da diferença da velocidade estimada pelo filtro e velocidade real é de 0.16 m/s. Apesar de existir muito menos ciclos de actualização do filtro de Kalman, este consegue à mesma estimar a velocidade não existindo uma grande variação da velocidade estimada para a

velocidade real ao longo do tempo.

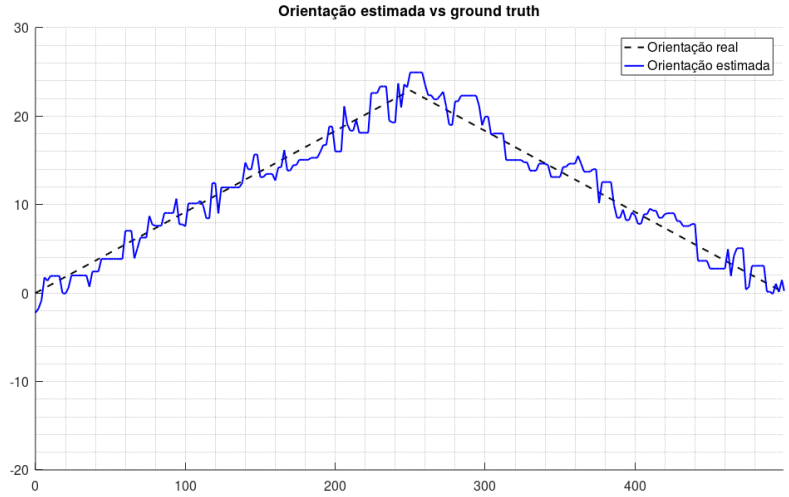


Figura 6.16: Orientação estimada pelo filtro de Kalman para o cenário da figura 6.14

Na figura 6.16 estão representadas a orientação estimada pelo filtro de Kalman bem com a orientação real. Com uma simples análise é possível verificar que a orientação foi bem estimada apesar do alto número de falhas nas observações do radar. O valor médio da diferença entre valor estimado e valor real é aproximadamente de 1.23 °.

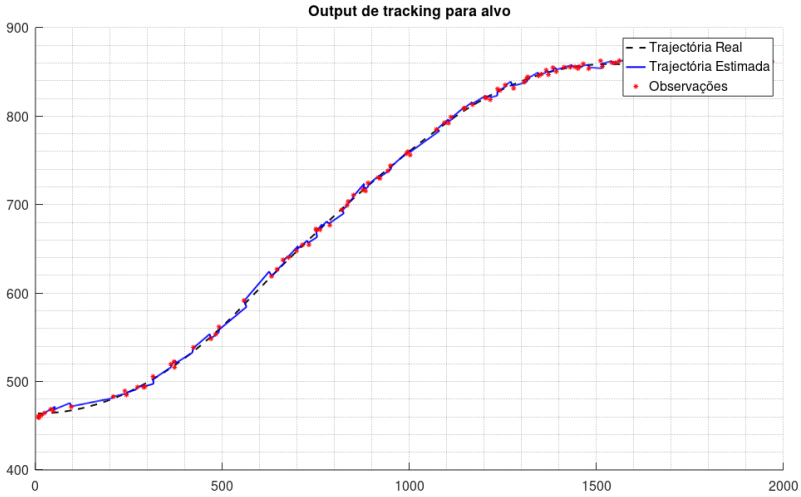


Figura 6.17: Exemplo de output do filtro de Kalman para uma embarcação - cenário III

O filtro foi também testado num cenário aonde a probabilidade de falhas de observação do radar é muito alto. Isto implica que maior parte do tempo apenas o ciclo de previsão irá acontecer e quando surge uma nova medida é desejável que o filtro não dê demasiado pelo à nova medida. A figura 6.17 é o resultado de uma simulação gerada com probabilidade de o

radar falhar observações de 20% e com variação de velocidade ao longo do tempo. Como é possível ver pela figura muito poucas observações ocorrem durante o período de simulação de aproximadamente 17 minutos. Como esperado, a estimação da posição possuí um erro de 6.52 m estando um pouco acima dos valores das simulações anteriores.

É espectável que os resultados apresentados no último cenário não sejam os melhores, visto que apenas existem observações em 20% do tempo. Contudo, apesar do número baixo de actualizações do filtro de Kalman este consegue à mesma seguir o alvo. A velocidade estimada difere em valor médio para a velocidade real de 0.163 m/s. A orientação e o tamanho estimados pelo filtro apresentam relutados semelhantes aos cenários anteriores.

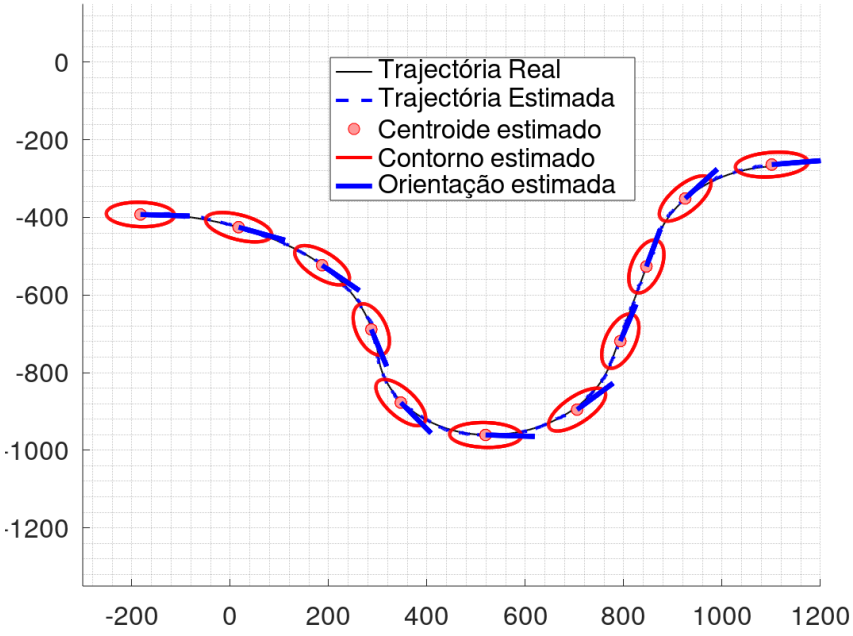


Figura 6.18: Representação da orientação e tamanho estimados para uma embarcação - cenário IV

A figura 6.18 é uma representação da estimação da orientação e tamanho de um dada embarcação. Na figura, elipse vermelha representa o tamanho estimado da embarcação pelo filtro de Kalman. Esta embarcação tem a seguinte dimensão: $150m$ no eixo maior e $75m$ no eixo menor, sendo que o erro médio na estimação do tamanho da embarcação é de aproximadamente $1,62m$. A estimação da orientação está representada na figura pela recta azul e possui um erro médio de $1,22°$

A tabela 6.2 é uma tabela resumo das erros médios da estimação de estados para os três cenários apresentados anteriormente. No cenário III o erro referente à estimação da posição é significativamente maior do que nos outros mas ainda aceitável. Apesar de neste cenário existirem observações apenas 20% das vezes o filtro consegue estimar bem a posição do alvo, como e possível verificar pela figura 6.17.

Tabela 6.2: Resumo de erros das simulações de embarcações

	Posição (m)	Velocidade (m/s)	Orientação (°)	Tamanho (m)
Cenário I (30%)	4,11	0,14	1,12	1,66
Cenário II (50%)	4,93	0,16	1,23	1,71
Cenário III (80%)	6,52	0,17	1,27	1,73
Cenário IV (30%)	3,93	0,12	1,22	1,62

De forma a poder validar o filtro para múltiplos alvos e validar o algoritmo 7 é necessário gerar um conjunto de simulações que possuem duas ou mais embarcações observadas ao longo do tempo.

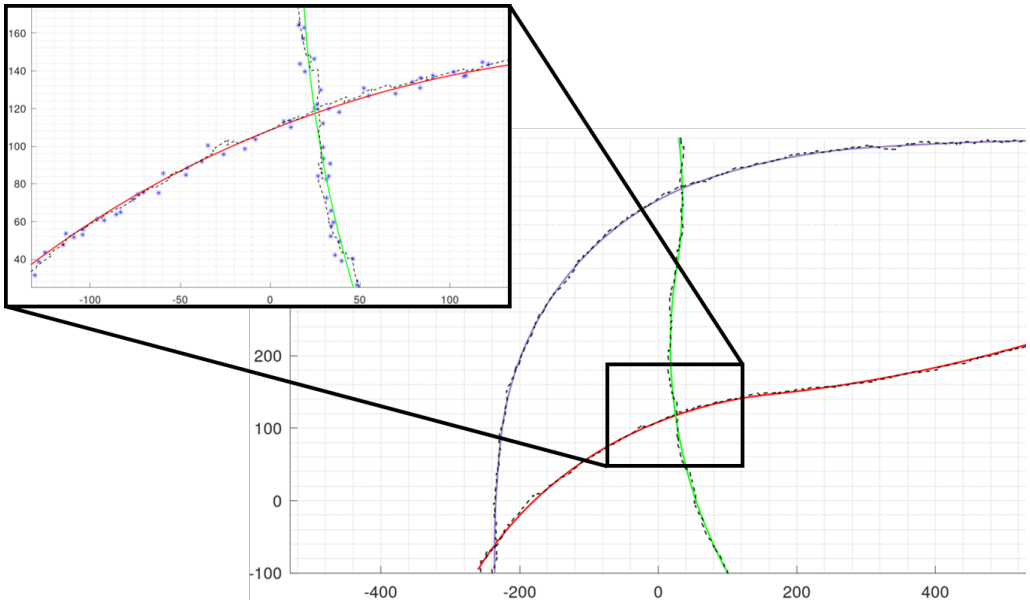


Figura 6.19: Exemplo de output do filtro de Kalman para múltiplas embarcações

A figura 6.19 é o resultado do filtro seguindo múltiplos alvos simulados. As cores vermelho, verde e azul representam as trajectórias reais para as diferentes embarcações. A tracejado é possível observar as trajectórias estimadas pelo filtro de Kalman. Na aproximação feita no cruzamento das trajectórias verde e vermelha, é possível visualizar as observações de posição feitas pelo radar (asteriscos azuis). O erros médios na estimação dos estados foram em tudo semelhantes aos erros apresentados na tabela 6.2 para o cenário IV.

Deste modo foi possível validar o correcto funcionamento do módulo **filtro respectivo**, representado na figura 3.3 e apresentado na secção 6.2.2, bem como a implementação do filtro de Kalman linear em Matlab. Os testes de calibração do filtro de Kalman apresentados na secção 6.1.6 foram efectuados de forma a validar a convergência do filtro. Nos cenários apresentados, os testes efectuados confirmaram o perfeito funcionamento do filtro

podendo afirmar que o filtro encontra-se bem calibrado para a dinâmica das embarcações simuladas.

Num cenário real seria necessário calibrar calibrar a matriz de ruído associada ao modelo de observação em função das características reais do radar. Dado que o tamanho dos segmentos do radar Lowrance 3G dependem inteiramente do alcance máximo configurado, é necessário obter uma matriz de ruído de observação para cada alcance possível de configurar de forma a poder escolher o valor mais apropriado.

Para a matriz de ruído associada ao modelo de propagação no filtro de Kalman é necessário ajustar esta em função do tipo de alvo a seguir visto que esta matriz está directamente relacionada com a dinâmica do alvo a seguir. A matriz que é usada para seguir uma embarcação grande é totalmente diferente da matriz para seguir lanchas.

6.5 Detecção de Colisões

O filtro de Kalman apresentado anteriormente consegue monitorizar e estimar o estado de que cada embarcação nas proximidades do ROAZ II, podendo fornecer informações importantes de modo a evitar colisões entre embarcações. O filtro de Kalman consegue estimar a posição, velocidade e orientação de um determinado alvo, sendo possível com estas características prever uma possível trajectória.

Foi feita a implementação de um sistema capaz de prever possíveis colisões utilizado como base o ultimo estado conhecido do filtro de Kalman. Este sistema é divido em duas partes distintas: verificar se existe intercepção de trajectórias e estimar a possível área de colisão.

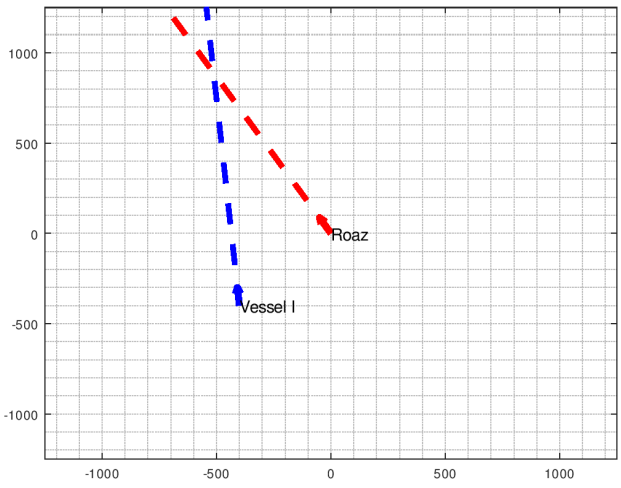


Figura 6.20: Exemplo de estimação de trajectórias para ROAZ e embarcação

A figura 6.20 é um exemplo da previsão de trajectória para o ROAZ II e uma em-

barcação. Como é possível verificar existe intercepção das duas trajectórias.

O primeiro processo para verificar se existe perigo de colisão é calcular o ponto de colisão[45].

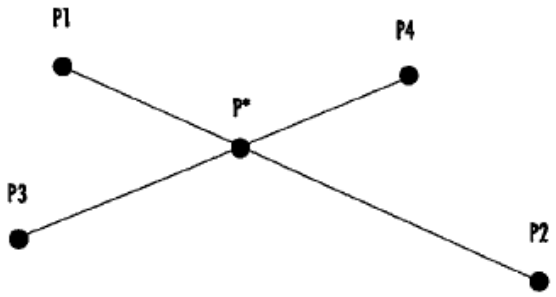


Figura 6.21: Intercepção de linhas apresentada por *Franklin Antonio* [45]

A figura 6.21 é um exemplo da intercepção de duas linhas usado na detecção do ponto de colisão. Cada linha é definida por um conjunto de dois pontos. O algoritmo apresentado por *Franklin Antonio* [45] tem como objectivo calcular o ponto de intercepção $P*$.

O ponto de intercepção pode ser calculado por:

$$P* = \alpha P1 + (1 - \alpha)P2 \tag{6.32}$$

com α como um escalar. A equação anterior pode ser rescrita de um maneira mais conveniente:

$$P* = P1 + \alpha(P2 - P1) \tag{6.33}$$

É possível escrever a mesma equação para a segunda linha:

$$P* = P3 + \beta(P4 - P3) \tag{6.34}$$

subtraindo as equações anteriores obtém-se que:

$$0 = (P1 - P3) + \alpha(P2 - P1) + \beta(P3 - P4) \tag{6.35}$$

A solução da equação anterior é dada por:

$$\begin{aligned} \alpha &= \frac{B_y C_x - B_x C_y}{A_y B_x - A_x B_y} \\ \beta &= \frac{A_x C_y - A_y C_x}{A_y B_x - A_x B_y} \end{aligned} \tag{6.36}$$

com $A = P2 - P1$, $B = P3 - P4$ e $C = P1 - P3$. Se as duas linhas se interceptarem α e β têm de estar compreendidos entre $[0, 1]$. Na situação de o ponto de intersecção não se encontrar definido entre os pontos representados na figura, o valor de α tem de estar compreendido entre $[0, \infty]$ e β entre $[-\infty, 0]$. O ponto de intersecção é calculado resolvendo a equação 6.33 ou 6.34.

No cenário apresentado na figura 6.20, as linhas são definidas pela posição actual da embarcação e uma posição no futuro, isto é, é necessário propagar a posição actual para calcular um ponto auxiliar. Com a posição actual e o ponto auxiliar é possível verificar se as duas linhas se interceptam e no caso de existir intercepção, calcular o ponto.

Tendo o ponto de intercepção das duas trajectórias é necessário verificar se o ROAZ II e a embarcação vão atravessar no mesmo instante de tempo. Para tal é necessário estimar o tempo que o ROAZ II e a embarcação demoram para chegar ao ponto. O tempo pode ser facilmente calculado pelo quociente da distância pela velocidade. É considerado que tanto o ROAZ II e a embarcação navegam a uma velocidade constante e que se irá manter até ao ponto.

Para o exemplo da figura 6.20 a posição actual do ROAZ é $[0, 0]$ com um vector de velocidades de $[-1, 1.7]$. A posição da embarcação é $[-400, -400]$ com um vector de velocidades de $[-0.2, 2.3]$. O ponto de intercepção estimado é $[513, 888]$. O tempo que o ROAZ II necessita para chegar ao ponto de intercepção é $8m : 33s$ e a embarcação de $9m : 22s$. Existe uma diferença temporal da chegada ao ponto de $49s$. É possível obter conclusões em função deste tempo, contudo deste modo não estamos a utilizar todas as informações fornecidas pelo filtro de Kalman.

A solução implementada para detectar colisões, passa por propagar a matriz de covariância da posição actual para o ponto de intercepção do o ROAZ e da embarcação. Esta propagação faz uso das mesmas equações utilizadas no processo de previsão do filtro de Kalman (equação 6.7).

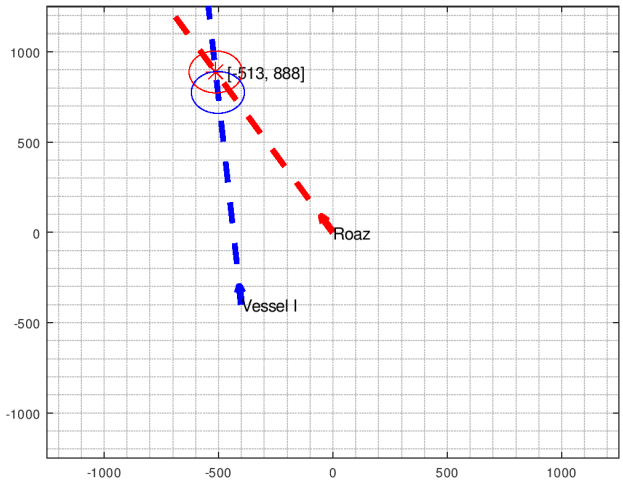


Figura 6.22: Exemplo de propagação de covariância para verificar de colisão

O modelo de previsão utilizado depende da diferença temporal entre a nova posição e a ultima posição conhecida. Esta diferença temporal é o tempo que o ROAZ II demora para chegar desde a posição actual até ao ponto de intercepção.

A figura 6.22 é o resultado da propagação das matrizes de covariância do ROAZ II, a vermelho, e da embarcação, a azul. Como é possível verificar, existe uma sobreposição das duas elipses, isto significa que existe uma área com uma alta probabilidade de colisão.

O segundo passo do sistema implementado consiste em aproximar a área de intercepção das duas elipses a uma nova elipse de forma a descrever a possível área de perigo. O processo começa por calcular o ângulo efectuado entre, o segmento de linha formando entre os dois centroides das elipses e o eixo do x. Na figura 6.23 o segmento de linha entre os dois centroides é representado pela linha preta tracejada.

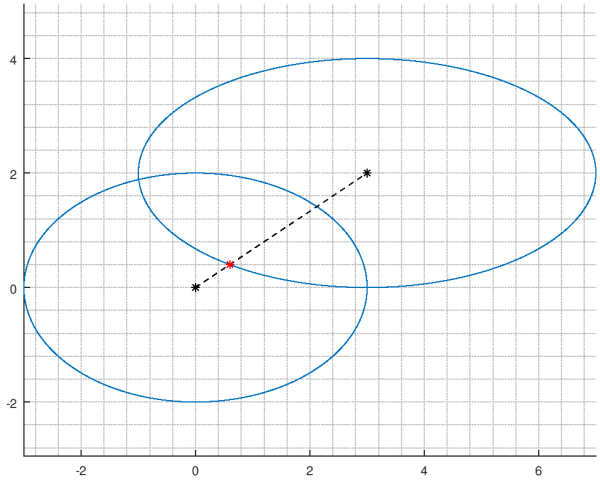


Figura 6.23: Exemplo de sobreposição de elipses

Após calculo do ângulo é necessário calcular o ponto representado na figura a vermelho, ponto este que é definido pelo seu ângulo e pela amplitude. A amplitude do ponto em relação ao centroide da elipse mais à direita é dada por:

$$r = \sqrt{\frac{a_1^2 * b_1^2}{b_1^2 * \cos\theta^2 + a_1^2 * \sin\theta^2}} \tag{6.37}$$

onde a_1 e b_1 são a amplitude da elipse mais à direita do semieixo maior e menor respectivamente. Com o ângulo e a amplitude é possível passar do sistema polar de coordenadas para o sistema cartesiano, obtendo assim o ponto da elipse da direita que se encontra mais próxima da elipse da esquerda. Este ponto é calculado usando:

$$\begin{aligned} P_x &= r * \cos\theta + x_1 \\ P_y &= r * \sin\theta + y_1 \end{aligned} \tag{6.38}$$

com x_1 e y_1 como centroide da elipse da direita. É necessário verificar se o ponto calculado, ponto a vermelho na figura 6.23, se pertence à elipse da esquerda. Para tal apenas é

necessário substituir o ponto obtido pela equação genérica da elipse mais à esquerda.

$$c = \frac{(P_x - x_0)^2}{a_0^2} + \frac{(P_y - y_0)^2}{b_0^2} \tag{6.39}$$

O valor obtido da equação anterior é então averiguado. Se este for igual ou inferior a 1, então o ponto pertence à elipse e é considerado que existe sobreposição das elipses. Caso contrário o ponto está fora da elipse não ocorrendo sobreposição das duas elipses.

Na situação de ser considerado que existe sobreposição das duas elipses é necessário estimar uma nova elipse que represente a possível área de colisão, isto pode ser alcançado por[46]:

$$\begin{aligned} \widehat{x}_n &= P_n \cdot P_0^{-1} \cdot \widehat{x}_0 + P_n \cdot P_1^{-1} \cdot \widehat{x}_1 \\ P_n &= P_0^{-1} + P_1^{-1} \end{aligned} \tag{6.40}$$

onde P_n é a nova covariância, P_0 e $\widehat{x}_0$ a covariância e estado da elipse 1, P_1 e $\widehat{x}_1$ a covariância e estado da elipse 2 e $\widehat{x}_n$ o estado referente à nova elipse. A figura 6.24 é o resultado de intersecção das elipses relativas ao ROAZ II e embarcação. A elipse resultante está representada a cor preta sendo esta definida por um valor médio ($\widehat{x}_n$) e uma matriz de covariância (P_n).

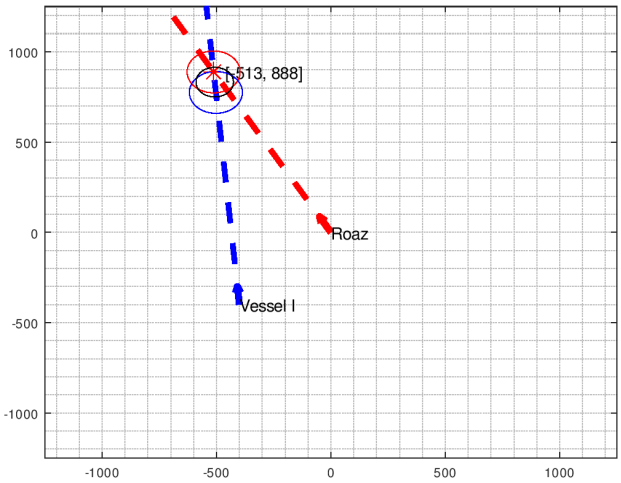


Figura 6.24: Exemplo de uma possível área de colisão

A figura 6.25 é o resultado do sistema implementado para múltiplas embarcações. Como é possível verificar as embarcações **I**, **II** e **IV** estão em rota de colisão com a rota do ROAZ II. A embarcação **IV** apesar de ter uma rota que intercepta a rota do ROAZ II não oferece perigo pois existe uma diferença temporal de $1m : 49s$ e não ocorre sobreposição das elipses. Na situação das embarcações **I** e **II** é possível verificar que existe uma sobreposição de rotas e sobreposição das elipses, podendo concluir que existe um grande probabilidade

de colisão. A embarcação **III** não oferece qualquer risco para o ROAZ II dado que as rotas não se interceptam em momento algum.

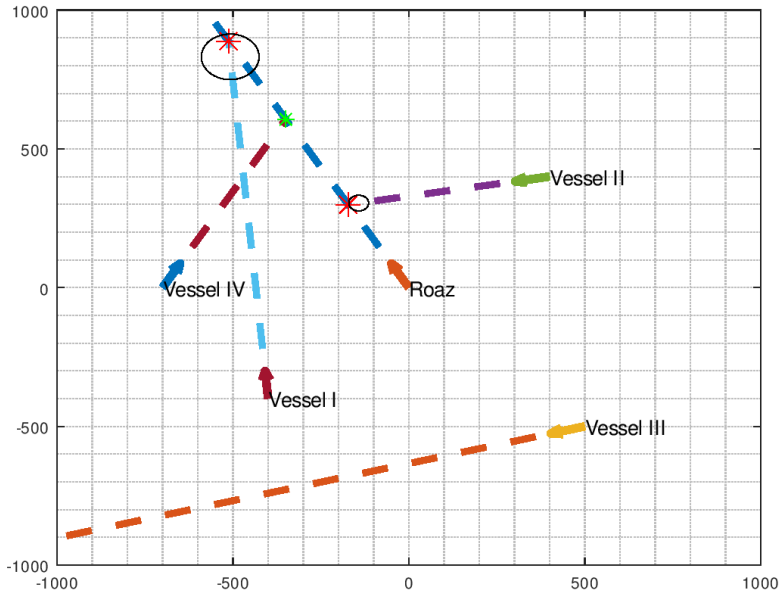


Figura 6.25: Exemplo de múltiplas embarcações em rota de colisão

As elipses representadas a cor preta na figura 6.25 retratam a possível área de colisão na situação de todos os intervenientes manterem a mesma rota com a mesma velocidade. Deste modo é possível averiguar se existe probabilidade de colisão do ROAZ II com outras embarcações nas redondezas. No caso de existirem possíveis áreas de colisão, é necessário tomar as devidas medidas, isto é, é necessário tomar decisões em função das regras de navegação apresentadas no secção 2.4.

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Capítulo 7

Conclusões

7.1 Conclusões

O trabalho apresentado neste documento tem como objecto o desenvolvimento de um sistema de detecção e monitorização de obstáculos para o ASV ROAZ II pertencente ao LSA e INESC TEC. A solução apresentada possibilita o uso de múltiplos sensores com o objectivo de conseguir aproveitar as vantagens de cada um dos sensores sendo que este trabalho é focado na tecnologia radar.

No decorrer deste trabalho foram desenvolvidos vários módulos de *software* com o intuito de efectuar aquisição, agregação e descrição de dados para posterior monitorização de obstáculos. O trabalho desenvolvido foi implementado com recurso à *framework* de *software* ROS por forma a facilitar a integração com os sistemas já existentes no ASV ROAZ II.

O módulo de aquisição de dados do radar tem como base o trabalho desenvolvido por *Dabrowski et al.* [29] e é exposto no subcapítulo 4.2. Este módulo faz a interface entre o *hardware* (radar Lowrance 3G) e a *framework* de *software* ROS, fornecendo os dados recebidos do radar num tópico de ROS. Este módulo também é responsável por configurar os diferentes registos existentes no radar, abordado no subcapítulo 4.2. Os resultados do nó de ROS foram comparados com dados obtidos de um receptor de AIS de forma a conseguir validar o correcto funcionamento deste nó e podem ser observados no subcapítulo 4.3.

Foram analisados e testados diferentes algoritmos de agregação de pontos tais como DBSCAN e K-means. Os algoritmos foram confrontados com dados provenientes do radar de forma a avaliar a viabilidade dos mesmos para o cenário em causa, expostos no subcapítulo 5.1. O algoritmo de agregação de dados implementado tem como base os fundamentos dos algoritmos anteriores e está projectado para conseguir funcionar em tempo real para os dados do radar, abordado no subcapítulo 5.2. Este módulo foi testado num cenário real, porto de Leixões, e os dados obtidos foram comparados com o mapa do local do teste de modo a validar o correcto funcionamento. Os resultados da agregação de pon-

tos são apresentados no subcapítulo 5.3 e são comparados com os algoritmos tradicionais de agregação de pontos.

Os dados obtidos da agregação de pontos estão disponíveis para serem usados no módulo de descrição/classificação. Este módulo é responsável por descrever um determinado objecto com um conjunto de *features*. O algoritmo de extracção de *features* de um objecto é apresentado no subcapítulo 5.4 e descreve um objecto pelo seu centroide, pela aproximação a uma elipse e pelo menor rectângulo que circunscreve todos os pontos. O objecto é então avaliado com possível obstáculo para ser monitorizado ou como parte da linha costeira. Foi desenvolvido um algoritmo para extracção da linha costeira, linha esta que é representação da fronteira de navegação, algoritmo implementado exposto no subcapítulo 5.5, sendo que não foi possível validar o algoritmo na sua totalidade.

Foi implementado um filtro de Kalman para estimar a posição, velocidade, orientação e tamanho de possíveis embarcações. Este filtro é exposto nos subcapítulos 6.1 e 6.2 bem como os modelos de dinâmica e observação do radar. A solução de implementada é capaz de fazer monitorização para múltiplas embarcações ao mesmo tempo. Para tal é criado um filtro para cada embarcação a ser seguida e um algoritmo para decidir a qual filtro pertence um determinado objecto é usado, exposto no subcapítulo 6.2.

De forma a conseguir validar o correcto funcionamento do filtro de Kalman para múltiplas embarcações, foi desenvolvido um ambiente de simulação que permite gerar uma variedade de cenários, apresentado no subcapítulo 6.3. Este ambiente de simulação possibilitou a validação dos módulos de filtro de Kalman e classificação de objecto verificando o seu correcto funcionamento, subcapítulo 6.4.

Por fim, foi desenvolvido um módulo que permite estimar a possíveis colisões entre embarcações, apresentado no subcapítulo 6.5. Este módulo faz uso dos estados estimados pelo filtro de Kalman para cada embarcação, de modo a conseguir prever possíveis colisões. Os resultados obtidos são promissores contudo testes mais profundos deverão ser efectuados para validar o correcto funcionamento deste módulo.

O módulo de previsão de colisões e o módulo de extracção da fronteira de navegação poderão fornecer informações úteis para um trabalho futuro. Este trabalho serve de base para possíveis trabalhos relacionados com o desvio de obstáculos de forma dinâmica tendo em consideração as regras de navegação (COLREGS) apresentadas no subcapítulo 2.4

7.2 Trabalho Futuro

Considerando que a implementação dos módulos de aquisição, agregação e classificação encontram-se estáveis e totalmente validados, desenvolvimentos futuros deverão incidir em:

- Introdução de outros sensores, como AIS, com o intuito de tornar o filtro de Kalman para a estimação de estados mais robusto.

- O algoritmo de extracção da linha costeira deverá usar informação passada para gerar um mapa ao longo do tempo que poderá ser usado para navegação.
- Informações das embarcações nas proximidades obtidas pelo filtro bem como um mapa de navegação poderão ser usados na implementação de algoritmos de *obstacle avoidance*.
- Um sistema de *obstacle avoidance* usando as *colregs* deverá ser desenvolvido com o objectivo de actuar sempre que existir a probabilidade de colisão com outras embarcações.
- De forma a melhorar a informação do mapa de navegação dentro de portos marítimos, dados provenientes do LIDAR poderão ser usados.
- Testes de integração deverão ser efectuados para garantir o perfeito funcionamento de todo o sistema como um só bem como com os sistemas já existentes no ROAZ II.
- Estudo da possibilidade de utilização de um filtro de Kalman extendido para monitorização de embarcações com dinâmica elevada.
- Análise do impacto das correntes marítimas no sistema e estimação das mesmas com o filtro de Kalman.

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Anexo A

Tabela distribuição *chi-square*

Chi-square Distribution Table

d.f.	.995	.99	.975	.95	.9	.1	.05	.025	.01
1	0.00	0.00	0.00	0.00	0.02	2.71	3.84	5.02	6.63
2	0.01	0.02	0.05	0.10	0.21	4.61	5.99	7.38	9.21
3	0.07	0.11	0.22	0.35	0.58	6.25	7.81	9.35	11.34
4	0.21	0.30	0.48	0.71	1.06	7.78	9.49	11.14	13.28
5	0.41	0.55	0.83	1.15	1.61	9.24	11.07	12.83	15.09
6	0.68	0.87	1.24	1.64	2.20	10.64	12.59	14.45	16.81
7	0.99	1.24	1.69	2.17	2.83	12.02	14.07	16.01	18.48
8	1.34	1.65	2.18	2.73	3.49	13.36	15.51	17.53	20.09
9	1.73	2.09	2.70	3.33	4.17	14.68	16.92	19.02	21.67
10	2.16	2.56	3.25	3.94	4.87	15.99	18.31	20.48	23.21
11	2.60	3.05	3.82	4.57	5.58	17.28	19.68	21.92	24.72
12	3.07	3.57	4.40	5.23	6.30	18.55	21.03	23.34	26.22
13	3.57	4.11	5.01	5.89	7.04	19.81	22.36	24.74	27.69
14	4.07	4.66	5.63	6.57	7.79	21.06	23.68	26.12	29.14
15	4.60	5.23	6.26	7.26	8.55	22.31	25.00	27.49	30.58
16	5.14	5.81	6.91	7.96	9.31	23.54	26.30	28.85	32.00
17	5.70	6.41	7.56	8.67	10.09	24.77	27.59	30.19	33.41
18	6.26	7.01	8.23	9.39	10.86	25.99	28.87	31.53	34.81
19	6.84	7.63	8.91	10.12	11.65	27.20	30.14	32.85	36.19
20	7.43	8.26	9.59	10.85	12.44	28.41	31.41	34.17	37.57
22	8.64	9.54	10.98	12.34	14.04	30.81	33.92	36.78	40.29
24	9.89	10.86	12.40	13.85	15.66	33.20	36.42	39.36	42.98
26	11.16	12.20	13.84	15.38	17.29	35.56	38.89	41.92	45.64
28	12.46	13.56	15.31	16.93	18.94	37.92	41.34	44.46	48.28
30	13.79	14.95	16.79	18.49	20.60	40.26	43.77	46.98	50.89
32	15.13	16.36	18.29	20.07	22.27	42.58	46.19	49.48	53.49
34	16.50	17.79	19.81	21.66	23.95	44.90	48.60	51.97	56.06
38	19.29	20.69	22.88	24.88	27.34	49.51	53.38	56.90	61.16
42	22.14	23.65	26.00	28.14	30.77	54.09	58.12	61.78	66.21
46	25.04	26.66	29.16	31.44	34.22	58.64	62.83	66.62	71.20
50	27.99	29.71	32.36	34.76	37.69	63.17	67.50	71.42	76.15
55	31.73	33.57	36.40	38.96	42.06	68.80	73.31	77.38	82.29
60	35.53	37.48	40.48	43.19	46.46	74.40	79.08	83.30	88.38
65	39.38	41.44	44.60	47.45	50.88	79.97	84.82	89.18	94.42
70	43.28	45.44	48.76	51.74	55.33	85.53	90.53	95.02	100.43
75	47.21	49.48	52.94	56.05	59.79	91.06	96.22	100.84	106.39
80	51.17	53.54	57.15	60.39	64.28	96.58	101.88	106.63	112.33
85	55.17	57.63	61.39	64.75	68.78	102.08	107.52	112.39	118.24
90	59.20	61.75	65.65	69.13	73.29	107.57	113.15	118.14	124.12
95	63.25	65.90	69.92	73.52	77.82	113.04	118.75	123.86	129.97
100	67.33	70.06	74.22	77.93	82.36	118.50	124.34	129.56	135.81

Figura A.1: Distribuição *Chi-square*[47]

PÁGINA
INTENCIONALMENTE
DEIXADA
EM BRANCO

Bibliografia

[1] J. Yuh, Giacomo Marani e D. Richard Blidberg. ≪Applications of marine robotic vehicles≫. Em: *Intelligent Service Robotics* 4.4 (jul. de 2011), p. 221. ISSN: 1861-2784. DOI: 10.1007/s11370-011-0096-5. URL: https://doi.org/10.1007/s11370-011-0096-5.

[2] Camilo Mora *e outros*. ≪How Many Species Are There on Earth and in the Ocean?≫ Em: *PLOS Biology* 9.8 (ago. de 2011), pp. 1–8. DOI: 10.1371/journal.pbio.1001127. URL: https://doi.org/10.1371/journal.pbio.1001127.

[3] H. Ferreira *e outros*. ≪Autonomous bathymetry for risk assessment with ROAZ robotic surface vehicle≫. Em: (mai. de 2009), pp. 1–6. DOI: 10.1109/OCEANSE.2009.5278235.

[4] D. Pedrosa *e outros*. ≪Control-Law for Oil Spill Mitigation with an Autonomous Surface Vehicle≫. Em: *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*. Mai. de 2018, pp. 1–6. DOI: 10.1109/OCEANSKOBE.2018.8559397.

[5] Andy Norris *e outros*. *Radar and ARPA Manual : Radar, AIS and Target Tracking for Marine Radar Users*. 3rd edition. Elsvier Science & Technology, 2014, p. 552. ISBN: 9780080977522.

[6] J. Lee, J. Woo e N. Kim. ≪Vision and 2D LiDAR based autonomous surface vehicle docking for identify symbols and dock task in 2016 Maritime RobotX Challenge≫. Em: (fev. de 2017), pp. 1–5. DOI: 10.1109/UT.2017.7890273.

[7] D E. Dudgeon e Richard Lacoss. ≪An overview of automatic target recognition≫. Em: *The Lincoln Laboratory Journal* 6 (jan. de 1993).

[8] A. Mecocci *e outros*. ≪Radar image processing for ship-traffic control≫. Em: *Image and Vision Computing* 13.2 (1995), pp. 119–128. ISSN: 02628856. DOI: 10.1016/0262-8856(95)93153-J.

[9] Danbee Hong e Chan-Su Yang. ≪Algorithm Implementation for Detection and Tracking of Ships Using FMCW Radar≫. Em: *Journal of the Korean Society for Marine Environment and Energy* 16 (fev. de 2013). DOI: 10.7846/JKOSMEE.2013.16.1.1.

[10] L. P. Perera, P. Oliveira e C. Guedes Soares. ≪Maritime Traffic Monitoring Based on Vessel Detection, Tracking, State Estimation, and Trajectory Prediction≫. Em: *IEEE Transactions on Intelligent Transportation Systems* 13.3 (out. de 2012), pp. 1188–1200. ISSN: 1524-9050. DOI: 10.1109/TITS.2012.2187282.

[11] A. Mendes, L. C. Bento e U. Nunes. ≪Multi-target detection and tracking with a laser scanner≫. Em: (jun. de 2004), pp. 796–801. DOI: 10.1109/IVS.2004.1336486.

[12] X Rong Li e Vesselin P. Jilkov. ≪A Survey of Maneuvering Target Tracking—Part IV: Decision-Based Methods≫. Em: *Proceedings of SPIE - The International Society for Optical Engineering* 4728 (ago. de 2002). DOI: 10.1117/12.478535.

[13] D. W. Kammer C. R. Weisbin G. de Saussure. ≪A Real-Time Expert System For Control Of An Autonomous Mobile Robot Including Diagnosis Of Unexpected Occurrences≫. Em: *Proc.SPIE* 0729 (1987), p. 49. DOI: 10.1117/12.964857. URL: https://doi.org/10.1117/12.964857.

[14] J. Borenstein e Y. Koren. ≪The vector field histogram-fast obstacle avoidance for mobile robots≫. Em: *IEEE Transactions on Robotics and Automation* 7.3 (jun. de 1991), pp. 278–288. ISSN: 1042-296X. DOI: 10.1109/70.88137.

[15] Y. Koren e J. Borenstein. ≪Potential field methods and their inherent limitations for mobile robot navigation≫. Em: *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. Abr. de 1991, 1398–1404 vol.2. DOI: 10.1109/ROBOT.1991.131810.

[16] B. Zhang, W. Chen e M. Fei. ≪An Optimized Method for Path Planning Based on Artificial Potential Field≫. Em: *Sixth International Conference on Intelligent Systems Design and Applications*. Vol. 3. Out. de 2006, pp. 35–39. DOI: 10.1109/ISDA.2006.11.

[17] Jean-Claude Latombe. ≪Robot motion planning≫. Em: (1991). DOI: 10.1007/978-1-4615-4022-9. arXiv: arXiv:1011.1669v3.

[18] F. Xu, R. Moreas e H. Van Brussel. ≪A new dynamic obstacle avoidance method for mobile robots.≫ Em: *Proceeding of the Third European Advanced Robotics Systems Masterclass and Conference Robotics*. Abr. de 2000, pp. 12–14.

[19] H. Li *e outros*. ≪Avoiding Static and Dynamic Objects in Navigation≫. Em: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Out. de 2006, pp. 639–644. DOI: 10.1109/IROS.2006.282547.

[20] K. L. Woerner *e outros*. ≪Collision avoidance road test for COLREGS-constrained autonomous vehicles≫. Em: *OCEANS 2016 MTS/IEEE Monterey*. Set. de 2016, pp. 1–6. DOI: 10.1109/OCEANS.2016.7761413.

[21] A N Cockcroft e J N F Lameijer. *A Guide to the Collision Avoidance Rules International Regulations for Preventing Collisions at Sea*. 7th. ELSEVIER SCIENCE & TECHNOLOGY, 2007, p. 259.

[22] Liang Hu *e outros*. «COLREGs-Compliant Path Planning for Autonomous Surface Vehicles: A Multiobjective Optimization Approach**The authors should like to thank Innovate UK, grant reference, TSB 102308, for the funding of this project». Em: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 13662–13667. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.2525. URL: http://www.sciencedirect.com/science/article/pii/S2405896317334468.

[23] M. Pinto *e outros*. «Spline navigation and reactive collision avoidance with COLREGs for ASVs». Em: *2013 OCEANS - San Diego*. Set. de 2013, pp. 1–9. DOI: 10.23919/OCEANS.2013.6741341.

[24] Michael Benjamin *e outros*. «A method for protocol-based collision avoidance between autonomous marine surface craft». Em: *J. Field Robotics* 23 (mai. de 2006), pp. 333–346. DOI: 10.1002/rob.20121.

[25] *Documentation - ROS Wiki*. URL: http://wiki.ros.org/ (acedido em 19 de dez. de 2017).

[26] Jerry L Eaves e Edward K Reedy. *Principles of Modern Radar*. 1987, p. 708. DOI: 10.1049/SBRA503E. arXiv: arXiv:1011.1669v3.

[27] Raimond A. Serway e John W. Jewett. *Physics for Scientists and Engineers*. eighth edi. 2010, p. 1558. ISBN: 9781439048443.

[28] Lowrance. *Broadband 3G/4G^{TM} Radar Installation Guide*. Rel. téc., p. 36. URL: www.lowrance.com.

[29] Adrian Dabrowski, Sebastian Busch e Roland Stelzer. «A Digital Interface for Imagery and Control of a Navico/Lowrance Broadband Radar». Em: jan. de 2011, pp. 169–181. DOI: 10.1007/978-3-642-22836-0_12.

[30] Aristides Gionis, Heikki Mannila e Panayiotis Tsaparas. «Clustering aggregation». Em: *TKDD* 1 (mar. de 2007). DOI: 10.1109/ICDE.2005.34.

[31] D Pham, Stefan Dimov e Cuong Nguyen. «Selection of K in K -means clustering». Em: *Proceedings of The Institution of Mechanical Engineers Part C-journal of Mechanical Engineering Science - PROC INST MECH ENG C-J MECH E* 219 (jan. de 2005), pp. 103–119. DOI: 10.1243/095440605X8298.

[32] Martin Ester *e outros*. «A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise». Em: *Proceedings of the Second International Conference on Knowledge Dis-*

covery and Data Mining. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231. URL: http://dl.acm.org/citation.cfm?id=3001460.3001507.

[33] Matt Pharr e Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004. ISBN: 012553180X.

[34] T. Grydeland e Y. Larsen. «Beyond Plane Sailing: Solving the Range-Doppler equations in a reduced geometry». Em: *EUSAR 2018; 12th European Conference on Synthetic Aperture Radar*. Jun. de 2018, pp. 1–4.

[35] Samuel Drake. «Converting GPS coordinates [phi, lambda, h] to navigation coordinates (ENU)». Em: *DSTO* DSTO-TN (abr. de 2002).

[36] Yansen Wang, Giap Huynh e Chatt Williamson. «Integration of Google Maps/Earth with microscale meteorology models and data visualization». Em: *Computers and Geosciences* 61 (dez. de 2013), pp. 23–31. DOI: 10.1016/j.cageo.2013.07.016.

[37] Lindsay I Smith. *A tutorial on Principal Components Analysis*. Rel. téc. New Zealand: Department of Computer Science, University of Otago, 2002, p. 28.

[38] Antti Knowles, Horng-Tzer Yau e Jun Yin. «On the principal components of sample covariance matrices». Em: *Probability Theory and Related Fields* 164 (abr. de 2014). DOI: 10.1007/s00440-015-0616-x.

[39] Ronald L. Graham. «An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set». Em: *Inf. Process. Lett.* 1 (1972), pp. 132–133.

[40] Rudolf Kalman. «A New Approach to Linear Filtering and Prediction Problems». Em: *Transactions of the ASME - Journal of basic Engineering* 82 (jan. de 1960), pp. 35–45.

[41] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964. ISBN: 0262730057.

[42] Fernando Pedro Vieira Freitas Pinto. «The Cleaning Robot Project Aplicação do Filtro de Kalman na Auto-Localização de um Sistema Robótico Autónomo». Tese de Mestrado. Faculdade de Engenharia da Universidade do Porto, 2008, p. 127.

[43] Yaakov Bar-Shalom e Xiao-Rong Li. *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN: 047141655X.

[44] Ian Reid. *Estimation II, Lecture Notes 2*. 2001. URL: http://www.robots.ox.ac.uk/~ian/Teaching/Estimation/LectureNotes2.pdf.

[45] Franklin Antonio. «Faster line segment intersection». Em: *Graphics Gems III* (dez. de 1992), pp. 199–202. DOI: 10.1016/B978-0-08-050755-2.50045-2.

[46] D. Franken e A. Hupper. ≪Improved fast covariance intersection for distributed data fusion≫. Em: *2005 7th International Conference on Information Fusion*. Vol. 1. Jul. de 2005, p. 7. DOI: 10.1109/ICIF.2005.1591849.

[47] Yoni Nazarathy. *Chi-square Distribution Table*. URL: https://people.smp.uq.edu.au/YoniNazarathy/stat_models_B_course_spring_07/distributions/chisqtab.pdf.