ESCOLA NAVAL

talant de bien faire

Hugo Alexander Oliveira Maia da Fonseca

"ICARUS – Busca e Salvamento Utilizando Deteção Sonar"

Sistema de Deteção de Náufragos com Imagens Sonar

Dissertação para obtenção do grau de Mestre em Ciências Militares Navais, na especialidade de Engenheiros Navais – Ramo de Armas e Eletrónica

Alfeite
2015

Hugo Alexander Oliveira Maia da Fonseca

"ICARUS – Busca e Salvamento Utilizando Deteção Sonar"

Sistema de Deteção de Náufragos com Imagens Sonar

Dissertação para obtenção do grau de Mestre em Ciências Militares Navais, na especialidade de Engenheiros Navais - Ramo de Armas e Eletrónica

Orientação de: Professor Doutor Victor José de Almeida e Sousa Lobo
Coorientação de: Capitão-de-fragata Claro Lourenço

O Aluno Mestrando	O Orientador	O Coorientador
[Hugo Maia da Fonseca]	[Victor Sousa Lobo]	[Claro Lourenço]

Alfeite
2015

‘‘Viver não é necessário. Necessário é criar’’

- Fernando Pessoa

Agradecimentos

A todos um muito obrigado pelo acompanhamento, orientação e apoio ao longo da elaboração desta dissertação, que me forma no futuro, ajuda-me a cumprir e superar todos os desafios inerentes ao mesmo.

Ao meu Orientador que apesar de todas as restrições a nível de tempo, sempre demonstrou toda a disponibilidade e vontade de me apoiar e guiar neste processo trabalhoso. O seu conhecimento e interesse foi uma fonte de inspiração para mim, ajudando-me a focar o esforço, dando conselhos de modo a melhorar o meu trabalho e ter um produto final melhor.

Ao Capitão Tenente Claro Lourenço pela sua disponibilidade e cedência de espaço para a realização dos testes na piscina e no tanque de mergulhadores. Apesar dos contratempos e outras obrigações de serviço, fico grato pela sua disponibilidade.

A todos aqueles que me acompanharam ao longo da minha formação militar nesta grande instituição, que me permitiram e ajudaram a crescer tanto a nível pessoal como profissional.

A todos os meus camaradas, irmãos de armas e amigos, pelos grandes momentos proporcionados e experiências vividas.

Por fim, à minha família, pelo apoio incondicional nas minhas escolhas e por fazerem o possível e o impossível para me ajudarem a ser o militar e o cidadão que sou. Pela confiança, amor e dedicação que só família consegue transmitir.

Obrigado.

Resumo

O CINAV no âmbito do projeto ICARUS, está a colaborar no desenvolvimento de veículos não tripulados para busca e salvamento podendo estes ser uma valiosa ferramenta para salvar vidas humanas.

Tendo a Marinha Portuguesa UUV para guerra de minas, surgiu a ideia de usar estes veículos também para na vertente de Busca e Salvamento.

Em situações de naufrágio em alto mar, o corpo humano à deriva fica com cerca de noventa porcento submerso, torna-se pertinente o desenvolvimento de um sistema capaz identificar os náufragos usando imagens recolhidas por sonares, colocados nos UUV, mas apontados para a superfície.

Ao longo dos últimos anos, tem-se assistido ao desenvolvimento de inúmeros métodos associados à deteção de objetos em imagens digitais. Nesta tese é feita uma revisão bibliográfica das várias técnicas utilizadas para classificar objetos em imagens e são apresentados alguns conceitos na área de processamento de imagens, com enfase no reconhecimento de padrões. Nesta tese é descrito o processo de recolha de imagens, efetuado relativamente as posições prováveis de se encontrar um corpo à deriva. São estabelecidos os parâmetros para o Algoritmo aplicado SURF (*Speed-Up Robust Features*) que permitem uma melhor eficácia ao nível da classificação da existência de um náufrago à deriva.

Palavras-chave: deteção de objetos, segmentação de imagens, reconhecimento de padrões, classificação de imagens sonar, deteção de náufragos.

Abstract

The CINAV within the ICARUS project is collaborating in the development of unmanned vehicles for Search and Rescue which could be a valuable tool to save human lives.

Since the Portuguese Navy has in its possession unmanned underwater vehicles (UUV) for use in mine warfare, the idea of using these vehicles also in Search and Rescue was tested.

In the case of high sea shipwrecks, the castaway normally presents approximately ninety percent of he's floating body submerged. Therefore, the development of a system that identifies castaways using images collected by sonar, placed in a UUV, pointing up to the surface is highly relevant.

Over the last few years, we have witnessed the development of numerous methods related with the detection of objects in digital images. A bibliographic review of the various techniques used to classify objects in images was made and some concepts in image processing, with emphasis on pattern recognition are presented. In this thesis is described the process of image acquiring, done in relation to probable positions of finding a body at the surface. The parameters to the SURF algorithm applied are established to give a better efficacy at classifying the existence of a person at drift.

Keywords: Object detection, image segmentation, pattern recognition, image classification, detection shipwrecked.

Índice Geral

Índice de Figuras

Índice de Equações

Índice de Tabelas

Abreviaturas

AUV – Autonomous Unmanned Vehicle

AVI – Audio Video Interleave

CINAV – Centro de Investigação Naval

CUDA - Compute Unified Dispositive Architecture

F – Frames

FP – False Positives

FICI – Falsas imagens com interesse

FISI – Falsas imagens sem interesse

LA – Limiar água

LI – Limiar Intensidade

nP – Number of Positives

OCR – Optical Character Recognition

PNG – Portable Network Graphics

SIFT – Scale Invariant Feature Transform

SURF – Speed-Up Robust Features

TP – True Positives

UUV – Unmaned UnderWater Vehicle

Capítulo 1 - Introdução

1.1 Motivação

O CINAV no âmbito do projeto ICARUS, está a colaborar no desenvolvimento de veículos não tripulados para busca e salvamento podendo estes ser uma valiosa ferramenta para salvar vidas humanas. Projeto este que é centrado na investigação, e no desenvolvimento de componentes que podem ser configurados para busca e salvamento no âmbito da resposta a catástrofes naturais, baseados em veículos robóticos autónomos (ar, mar e terra) equipados com senso de pessoas.

Tendo a Marinha Portuguesa UUV para guerra de minas, surgiu a ideia de usar estes veículos também na vertente de Busca e Salvamento.

Em situações de naufrágio em alto mar, o corpo humano ao estar à deriva fica com cerca de noventa porcento submerso, torna-se então pertinente o desenvolvimento de um sistema capaz de identificar os náufragos usando imagens recolhidas por sonares, colocados nos UUV, mas apontados para a superfície.

Esta ideia é uma mais-valia, pois sabendo como a nossa marinha opera, sabe-se que existe uma grande lacuna na busca de náufragos durante o período noturno, devido à falta de luminosidade. Com este conceito de se ter o sonar direcionado para a superfície conseguira-se otimizar e aumentar o tempo na busca do náufrago.

A visão computacional é o ramo da computação que reúne todas as teorias e tecnologias desenvolvidas com a finalidade de possibilitar que imagens sejam interpretadas por sistemas artificiais implementados em computadores. As técnicas de visão computacional devem ser realizadas através de elementos de *hardware* e *software*. Este ramo que possui inúmeras aplicações engloba controle de processos, navegação de veículos autónomos, sistemas de segurança e outras aplicações importantes.

Na prática, são desenvolvidos algoritmos e técnicas específicas para lidar com os diversos tipos de problemas que surgem neste ramo da ciência, pois não existe uma teoria padronizada ou suficientemente genérica para modelar todos os aspetos da perceção visual. Assim, um determinado problema deve ser tratado através de uma abordagem especificamente desenvolvida para o problema em questão. Segmentação de imagens em

partes, melhoramento do seu contraste, reconstrução de objetos e detetores de bordas são exemplos encontrados neste âmbito.

A deteção e o reconhecimento de objetos são de modo geral, um dos problemas básicos mais comuns em visão computacional. Este problema consiste em determinar se um dado tipo de objeto está presente numa imagem (deteção) e caso algum seja detetado se este corresponde ao que era esperado (reconhecimento). Considerando a minha dissertação de mestrado, que consiste na deteção de náufragos através de imagens sonar. Detetar náufragos significa associar uma posição na imagem sonar ao náufrago ou a um objeto. Reconhecer um náufrago representa a ação de excluir os ecos sonar que não representem náufragos à superfície.

O trabalho produzido ao longo desta dissertação de mestrado, direciona a pesquisa à análise de imagens sonar com o objetivo de criar uma ferramenta computacional que mostre que é viável a deteção de náufragos através de imagens sonar.

1.2 Objetivos da dissertação

Esta tese de mestrado tem como objetivo principal investigar a viabilidade na deteção de náufragos através de imagens sonar, concretamente:

- Conseguir analisar os dados provenientes do sonar, com este direcionado para a superfície;
- Recorrer a técnicas de processamento de imagem em MATLAB por forma a segmentar a imagem, ficando com a área possível de aparecimento do náufrago;
- Aplicar um filtro que consiga diminuir o ruido na imagem não perdendo os detalhes que possam comprometer a deteção de um náufrago;
- A utilização de algoritmo que consiga fazer o reconhecimento automaticamente, ou seja verifique se a imagem em análise tem algo de interesse;
- A criação de uma base de dados que tenha diferentes posições de o náufrago se encontrar à superfície, e que dentro de cada posição se obtenha imagens com diferentes ganhos e frequências, por forma a ter uma base de dados mais robusta;

- Garantir que objetos de diferentes formas e materiais não sejam confundidos como possíveis náufragos;
- Ter uma resposta positiva caso tenha sido identificado um náufrago;
- Fazer deteção em tempo real de náufragos através das imagens provenientes do sonar.

1.3 Organização da dissertação

A presente dissertação encontra-se organizada da seguinte maneira. No Capítulo 2 é feita uma revisão bibliográfica das várias técnicas utilizadas para classificar objetos em imagens, nomeadamente técnicas que são utilizadas em outras áreas, mas cuja complexidade se assemelha ao problema em questão. São apresentados alguns conceitos na área de processamento de imagens, com enfâse no reconhecimento de padrões.

No Capitulo 3 encontra-se a descrição, de forma pormenorizada, do algoritmo e dos métodos propostos no âmbito desta Tese para o reconhecimento de náufragos em imagens sonar. A atenção centrou-se, sobretudo, no desenvolvimento de um método que fosse capaz reconhecer o "objeto" alvo em imagens de baixa qualidade como são as de um sonar devido ao elevado ruido presente nas imagens. Ao longo deste capítulo, o problema será dividido em várias fases até se chegar ao reconhecimento positivo do náufrago.

O Capitulo 4 destina-se:

- Apresentar os testes que foram feitos com o intuito da criação de uma base de dados para aumentar a eficiência do algoritmo;

- Analisar os ecos de objetos de materiais diferentes e verificar que o seu eco sonar é diferente de um corpo;

- Testar a taxa de eficácia do protótipo com ficheiros sonar que contenham o objeto de estudo.

No Capitulo 5 é apresentado um breve sumário sobre o desenvolvimento em cada um dos capítulos da Tese de mestrado. Numa segunda parte, serão desenvolvidas ideias e sugestões para trabalhos futuros, com base na experiência adquirida ao longo da Tese

de mestrado e também com base em trabalho já desenvolvido mas não suficientemente amadurecido para estar presente neste texto.

Capítulo 2 – Revisão Bibliográfica

Por forma a cumprir com os objetivos da presente dissertação de mestrado, toda a pesquisa efetuada tem como base o objetivo de detetar objetos numa determinada imagem. Existem inúmeros métodos para atingir o mesmo objetivo porém, dadas as formas conhecidas do corpo humano, conhecidas as distâncias à superfície em que as imagens são tiradas e a velocidade constante do veículo, é-nos permitido restringir o âmbito da pesquisa e concentrar os esforços em casos de deteção de objetos, por métodos simples e pouco exigentes a nível de processamento computacional.

No entanto, importa salientar o desafio comum a toda a pesquisa bibliográfica efetuada no âmbito da identificação de objetos em ambientes naturais submarinos que é o facto da necessidade de obter um sistema fiável. Sistema esse que garanta a invariância da escala, a rotação, a iluminação e fatores de filtragem introduzidas pela água.

O avanço das técnicas no campo da computação visual nomeadamente na área da deteção de objetos tem a sua explicação na proliferação do conceito de videovigilância e seus sensores associados. No entanto, a especificidade do problema da deteção, localização e identificação de náufragos no fundo do mar tornam escassas as respostas a esta necessidade.

Nos últimos 50 anos de pesquisa, foram obtidos avanços que possibilitaram a evolução da pesquisa em aplicações altamente complexas (JAIN, 2000).

Os autores destacam os seguintes exemplos de aplicações atuais que requerem técnicas eficientes e robustas de reconhecimento de padrões:

a) Bioinformática: análise de sequências do genoma; aplicações e tecnologia de *microarrays*;

b) Diagnóstico médico;

c) Mineração de dados (*Data Mining*): a busca por padrões significativos em espaços multidimensionais, normalmente obtidos de grandes bases de dados e *Data Warehouses*;

d) Classificação de documentos da Internet;

e) Análise de imagens de documentos para reconhecimento de caracteres (*Optical Character Rocognition - OCR*);

f) Inspeção visual para automação industrial;

g) Busca e classificação em base de dados multimídia;

h) Reconhecimento biométrico, incluindo faces, íris ou impressões digitais;

i) Sensores remotos para imagens multiespectrais;

j) Reconhecimento de fala.

Um ponto em comum nestas aplicações são as características disponíveis nos padrões de entrada, tipicamente milhares, que não são diretamente utilizadas. Normalmente utilizam-se características otimizadas extraídas dos padrões de entrada através de procedimentos guiados pelos dados.

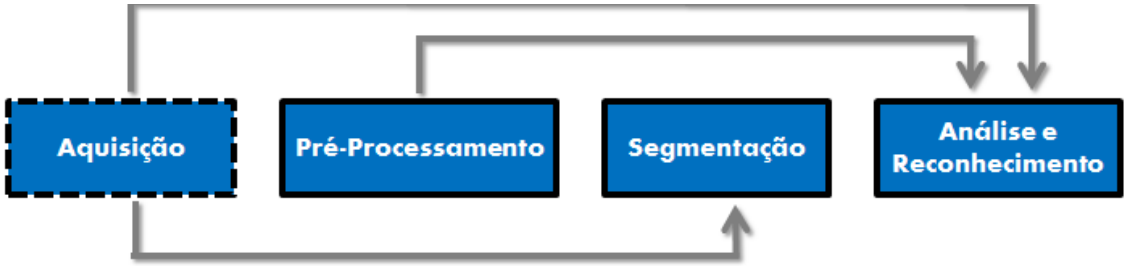
Geralmente em dados reais, os padrões a serem reconhecidos não possuem determinadas características. Nesse fato reside a importância dos algoritmos de extração e seleção de características, pois eles reduzem a dimensionalidade dando prioridade para uma base do espaço de características que não perdem o poder de discriminação dos padrões.

2.1 Principais etapas no processamento de imagens

Para a manipulação de imagens através de ferramentas computacionais, podem ser necessárias algumas etapas de processamento de imagens antes que o problema de reconhecimento de padrões seja abordado. Nas etapas temos: a aquisição, o pré-processamento, a segmentação e a análise de imagens (PEDRINI; SCHWARTZ, 2007). A etapa de aquisição consiste na obtenção das imagens a serem processadas; o pré-processamento visa a melhoria da qualidade da imagem e sua preparação para as fases posteriores; o objetivo da segmentação é identificar regiões similares; enquanto a análise e reconhecimento de imagens é a etapa onde as informações presentes na imagem são interpretadas. Baseando-me neste contexto, todo este capítulo apresenta as etapas associadas ao processamento realizado nas imagens utilizadas na tese.

Na figura 1 é apresentado um esquema com as etapas fundamentais ao processamento de imagens. É de realçar que algumas etapas podem ser transferidas. Essa eliminação de etapas está dependente da aplicação desenvolvida. Nesta tese o processamento de imagens consiste: na aquisição, no pré-processamento e na análise e reconhecimento de imagens. Mesmo não sendo utilizado o método de segmentação, todas as quatro principais etapas do processamento de imagens serão apresentadas no decorrer do capítulo.

Figura 1: Etapas fundamentais no processamento de imagens



2.2 Aquisição

Para a visualização e processamento de uma imagem é necessário que os dados (a imagem) sejam representados de forma apropriada, como por exemplo numa matriz. Cada imagem digital é um conjunto de pontos chamados de *pixels* que de uma forma discreta representam os componentes da região de interesse. A imagem capturada por um sensor é expressa como uma função continua *f(x,y)* de duas coordenadas no plano em uma matriz de *M* linhas e *N* colunas (SONKA; HLAVAC; BOYLE, 1998).

A aquisição de imagens nesta tese consiste em obter imagens do corpo humano. Essa etapa do processamento não é implementada em *software* e necessita de *hardware* especializado. Os dispositivos de captura de imagens (*scanners*) variam conforme o tipo das imagens de interesse.

Normalmente após a captura de imagens, as mesmas poderão ser ou não gravadas em algum equipamento de memória (disco ótico, disquetes, fitas de vídeo, disco rígido, etc.) para posterior estudo.

Dependendo da necessidade da aplicação, as imagens podem ser adquiridas a preto e branco, nível de cinza ou coloridas. Imagens a preto e branco são pobres em detalhes, mas têm a vantagem de requerer pouco espaço de armazenamento sendo que os

procedimentos para trabalhar com as mesmas são geralmente simples e rápidos. Imagens em níveis de cinza têm uma escala que varia de 2 até 256 tonalidades. Quanto maior for a escala de níveis de cinza maior será a riqueza de detalhes da imagem. Porém, é também maior a necessidade de espaço para o armazenamento e os procedimentos para manipulação tendem a ser mais complexos e lentos. Imagens coloridas ainda oferecem algumas restrições, devido ao espaço de armazenamento e à velocidade de processamento.

Embora existam imensos cuidados na aquisição das imagens, nem sempre é possível eliminarmos alguma informação indesejável que sempre aparece. Em imagens sonar, a qualidade dependerá de alguns fatores tais como bolhas de ar, ondulação e vento, sendo que não podem ser controlados pelo operador (ALLAN, 1986). Quanto melhor for a qualidade de imagem adquirida, mais fáceis e céleres serão os processamentos das fases posteriores.

2.3 Pré-processamento

O pré-processamento abrange todas as operações necessárias antes do processamento da imagem, quer seja para melhorá-la ou suprimir informação irrelevante. Nesta etapa, procura-se modificar e preparar os valores dos *pixels* da imagem com o objetivo de facilitar as operações subsequentes e obter melhores resultados (AWCOCK; THOMAS, 1996). As operações realizadas nesta etapa, são dependentes do problema apresentado ou seja são soluções específicas para cada caso. Algumas técnicas que funcionam bem em determinado tipo de imagem podem ser totalmente inadequadas para outros tipos.

O pré-processamento pode ser dividido em dois tipos de abordagens: realce e restauração de imagens (AWCOCK; THOMAS, 1996). Ambos têm o objetivo de melhorar a imagem em algum aspeto. O realce de imagens é utilizado para melhorar a qualidade da imagem ou enfatizar algum aspeto particular na mesma, sendo que os seus resultados podem produzir uma imagem muito diferente da original. Alguns aspetos podem até ser sacrificados de forma a melhorar outros. A restauração é um processo que tenta reconstruir ou recuperar uma imagem que foi degradada usando algum conhecimento do fenómeno que originou a degradação e aplicando o processo inverso para recuperar a imagem original (GONZALEZ; WOODS, 1993).

O realce é a aplicação de procedimentos heurísticos projetados para manipular a imagem, por exemplo:

- O aumento do contraste e manipulação do histograma são essencialmente baseados nos aspetos agradáveis apresentados ao observador;

- Funções de redução de ruído ou suavização são consideradas técnicas de restauro porque envolvem a extração de características de imagem.

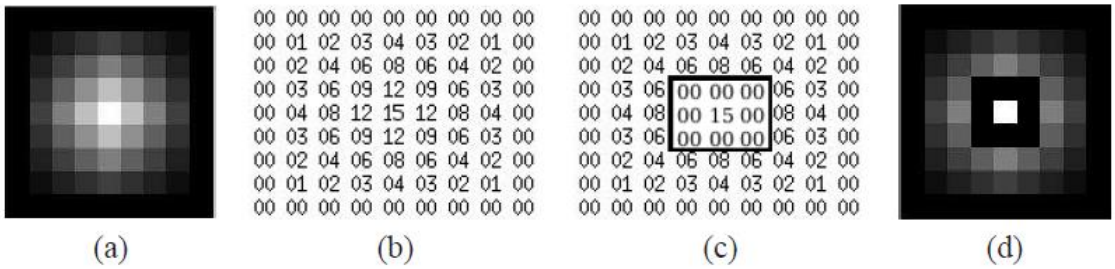
Uma das mais importantes técnicas de realce é a equalização do histograma. O histograma de uma imagem em níveis de cinza representa as frequências relativas de ocorrência de vários níveis de cinza da imagem (JAIN,1989). Pode-se dizer que um histograma informa quantos *pixels* na imagem possuem um determinado nível de cinza, definido entre 0 (preto) a 255 (branco). Histogramas mal distribuídos, ou seja concentrados em um determinado ponto, apresentam imagens más (muito claras ou escuras). A equalização do histograma de uma imagem serve exatamente para melhor distribuir os valores dos níveis de cinza de tal forma que se obtenha uma imagem com melhor qualidade. Nesses histogramas, o eixo *x* indica o nível de cinza, enquanto o eixo *y* apresenta a quantidade de ocorrências de cada nível de cinza.

O ruído é uma característica muito comum em imagens que são provenientes de sonares, o que torna útil as técnicas para suprimir essa degradação. Alguns filtros mais comuns aplicados na redução de ruído são os filtros da média e mediana (SONKA; HLAVAC; BOYLE, 1998). Além desses, podem ser citados os filtros de *crimmis* e o alfa trimmed (BUSSE; CRIMMINS; FIENUP, 1995; BEDNAR; WATT, 1984) por serem mais eficientes para a suavização de ruído.

Dois conceitos importantes para o entendimento das técnicas descritas neste capítulo são a vizinhança e a máscara (FACON, 1993). A vizinhança de um *pixel* consiste nos *pixels* à sua volta. O raio da vizinhança determina o número de *pixels* vizinhos envolvidos no processamento. O processamento de uma imagem baseado em máscara consiste em criar uma nova versão de imagem onde cada *pixel* é o cálculo dos pixels da vizinhança, na mesma posição da imagem original. A máscara pode ser vista como uma matriz de *pixels*, ou ainda uma subimagem usada em processamento local para modificar cada *pixel* na imagem. Na Figura 2 (a) é apresentada uma imagem e na Figura 2 (b) está expresso o seu formato ASCII, na Figura 2 (c) pode-se notar um quadrado no centro da

imagem, esse corresponde a uma máscara 3x3, sendo que a vizinhança do *pixel* central possui raio 1 nessa máscara. Na Figura 2 (d) é exibida a modificação realizada com a máscara em relação a Figura 2 (a). Essa modificação consiste em substituir os valores dos *pixels* da vizinhança por preto (zero).

O filtro da média é obtido calculando a média dos *pixels* da máscara e atribuindo esse valor ao pixel central. Já o filtro da mediana corresponde à atribuição do valor central



da

Figura 2: Exemplos de vizinhança e máscara

vizinhança ao *pixel* central da máscara na imagem original. Este filtro descarta valores extremos, altos ou baixos e é bom para a suavização de ruídos isolados, como o *salt and Pepper* (MALLADI; SETHIAN, 1996).

O filtro *Alfa-trimmed* (SANTANA, 1999) descarta os *outliers*, que são os *pixels* com níveis de cinza mais baixos e mais altos, e realiza o filtro da média com base nos *pixels* restantes. Desta forma a influência de ruídos é evitada, pois o ruído geralmente está nas altas e baixas tonalidades de cinza. O maior problema deste filtro está na escolha do valor *alfa*, o qual define o valor a partir do qual se descarta.

O filtro de *Crimmins* reduz o ruído *Speckle* de uma imagem. Ele usa o algoritmo *Eight Hull*, desenvolvido por Thomas Crimmins. (CRIMMINS, 1985). O objetivo é reduzir a intensidade de ruído *salt and Pepper* de uma imagem. Conforme o número de iterações do algoritmo o nível de redução de ruído aumenta e a suavização dos detalhes na imagem também. O algoritmo consiste em comparar cada *pixel* da imagem com os oito *pixels* da sua vizinhança. O *pixel* é comparado com os quatro pares de vizinhos, onde é verificado se ele é claro ou escuro em relação aos vizinhos. Caso seja claro, a tonalidade de cinza do *pixel* é decrementada para escurecer, caso contrário é incrementada para clarear (FISHER. Et al., 2000).

2.4 Segmentação

A segmentação consiste em juntar partes da imagem que provavelmente pertencem à mesma estrutura (SONKA; HLAVAC; BOYLE, 1998). O objetivo da segmentação é dividir uma imagem em partes ou objetos constituintes. Para Gonzalez e Woods (GONZALEZ; WOODS; EDDINS, 2004) a segmentação automática é uma das tarefas mais difíceis no processamento de imagens digitais, pois um procedimento robusto favorece substancialmente a solução bem-sucedida, mas a segmentação mal realizada quase sempre determina falhas no processamento.

Esta etapa pode ser muito simples se a imagem de interesse possuir poucos objetos bem definidos e com um bom contraste em relação ao fundo. O mesmo não se verifica se for preciso separar diversas regiões com muito ruído e pouco contraste, como acontece normalmente em imagens sonar.

A segmentação de imagens através de níveis de cinza é baseada na descontinuidade ou na similaridade de valores de intensidade da vizinhança do *pixel*. Entre as técnicas tradicionais utilizadas na segmentação estão o crescimento de região por agrupamentos de *pixels*, a limiarização (*threshold*) e o processo de divisão e fusão (*slipt and merge*), que utilizam a similaridade da vizinhança dos *pixels*. O objetivo da limiarização é extrair objetos de interesse da imagem, através da escolha de limiares que sirvam como parâmetros separadores. Pedrini e Schwartz (PEDRINI; SCHWARTZ, 2007), referem que uma forma para melhorar o formato do histograma e consequentemente aumentar as hipóteses de seleção de um bom limiar, é considerar apenas os *pixels* que estejam localizados sobre ou próximos das fronteiras entre os objetos e o fundo da imagem.

As redes neuronais artificiais também são usadas na segmentação com bons resultados onde os métodos tradicionais têm dificuldade em segmentar bordas.

Uma outra técnica para segmentação de imagens é a detecção de borda. “Uma borda é o limite ou fronteira entre duas regiões com propriedades relativamente distintas de nível de cinza” (PEDRINI; SCHWARTZ, 2007). Para segmentar imagens que apresentem distribuição de níveis de cinza suficiente homogênea, a transição entre duas regiões e sua consequente identificação pode ser determinada simplesmente com base na descontinuidade dos níveis de cinza.

2.4.1 Crescimento de Regiões

A técnica de crescimento de regiões, consiste em indicar um ou mais *pixels* que serão usados como ponto inicial do processo para determinar o valor da intensidade que será usada como comparação. A intensidade dos valores dos quatro *pixels* (esquerdo, direito, superior e inferior) da vizinhança é comparada com o inicial. Se a diferença estiver dentro de um limite pré definido, o *pixel* é adicionado à região e seus vizinhos são então avaliados, caso contrário ele é descartado. Este processo continua enquanto existirem *pixels* para serem analisados.

A Figura 3 mostra um exemplo da progressão do algoritmo de crescimento de regiões. Nestas quatro imagens da sequência cada imagem é apresentada em forma de uma grade, onde cada posição representa um *pixel*. Na Figura 3 (a) a letra "S" representa o *pixel* inicial e a letra "V" representa os *pixels* que estão a ser avaliados. A Figura 3 (b) mostra os *pixels* da fase anterior que foram selecionados, marcados com "S" e suas vizinhanças marcadas com "V" para análise. As Figuras 3 (c) e 3 (d) são a sequência do processo até a vizinhança não apresentar nenhuma homogeneidade. Uma forma otimizada desta abordagem é começar a análise com vários *pixels* iniciais colocados em regiões diferentes da imagem. Desta forma, várias regiões são encontradas invés de uma só, sabendo que a imagem pode ter várias regiões.

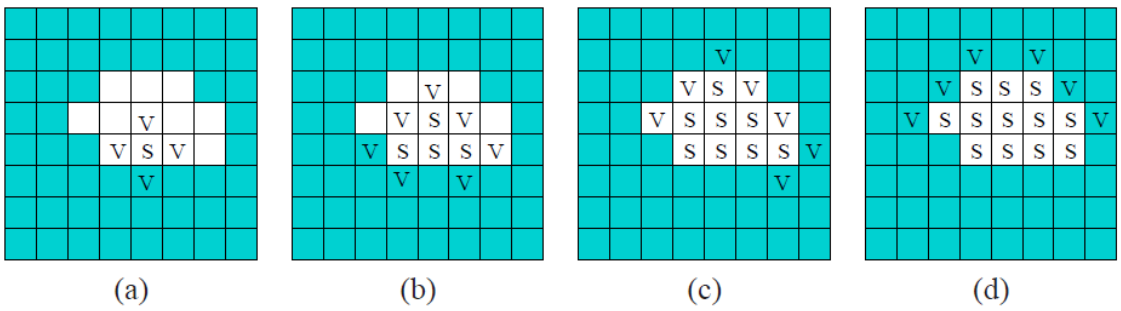


Figura 3: Representação do Algoritmo de crescimento de regiões

2.4.2 Limiarização

A Limiarização é uma técnica que faz a segmentação da imagem utilizando um ou mais valores limiares de níveis de cinza. Estes valores são normalmente obtidos com a análise do histograma da imagem em questão. Aos *pixels* que possuem um valor de intensidade menor que o limiar é atribuído um valor branco e aos restantes, valor preto.

Se dois limiares forem usados, os *pixels* com intensidade menor que o primeiro limiar recebem um valor. Aos de intensidade maior que o segundo limiar recebem outro valor e aqueles cuja intensidade se classifica entre os limiares referidos recebem um terceiro valor. Este método permite a simples separação de objetos do fundo da imagem, ou objetos com valores de intensidade variada. A Figura 4 apresenta dois exemplos de *threshold* diferentes para a mesma imagem. Na Figura 4 (b) foi utilizado o limiar 30 e na Figura 4 (c) o limiar 10. Salienta-se a diferença na área de segmentação de cada uma delas.

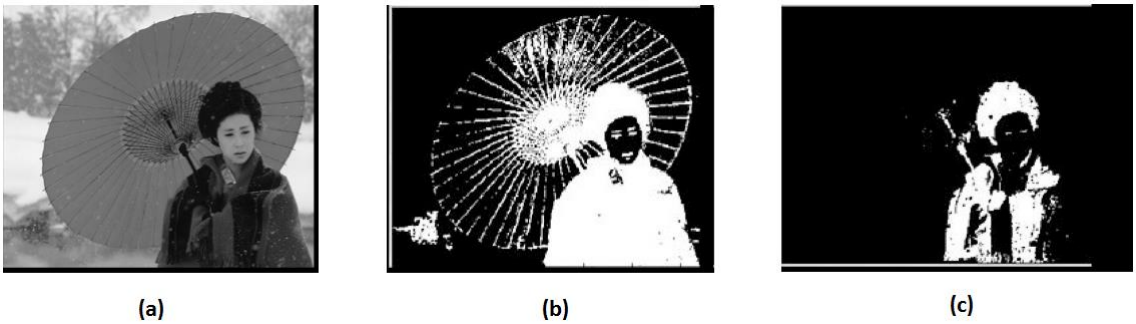


Figura 4: Threshold com dois limiares diferentes

A dificuldade do método é justamente encontrar o limiar ou limiares. Essa tarefa pode ser feita através da análise do histograma da imagem (GRAHAN; BARRETT, 1997).

A utilização da técnica de *threshold* pode não ser útil em muitos casos. A melhoria dos resultados pode ser obtida usando *threshold* multidimensional. Neste método são levadas em consideração outras características da imagem além do valor da intensidade do *pixel*. Mesmo assim, quando as imagens são de baixa qualidade e com bordas pouco definidas, esta técnica é inadequada.

2.4.3 Slip and Merge

A técnica de *split and merge* ou divisão e fusão consiste na procura por homogeneidade na imagem. Uma imagem ou região é homogénea quando todos os *pixels* possuem intensidade semelhante ou igual ao valor médio de intensidade daquela região. O processamento inicia com a imagem como uma região. Caso não seja totalmente homogénea, ela é repartida em quatro regiões menores. Cada região é analisada. Se uma

ou mais sub-regiões não forem homogéneas, serão repartidas novamente. Este processo é repetido enquanto existirem regiões não homogéneas ou até não ser possível separar mais a imagem. Na Figura 5 é apresentado um exemplo do algoritmo de *split and merge* de uma imagem constituída por um copo e uma lâmpada. Começando da esquerda para a direita, temos a imagem original, processo de *splitting*, e por ultimo o processo de *merging*.

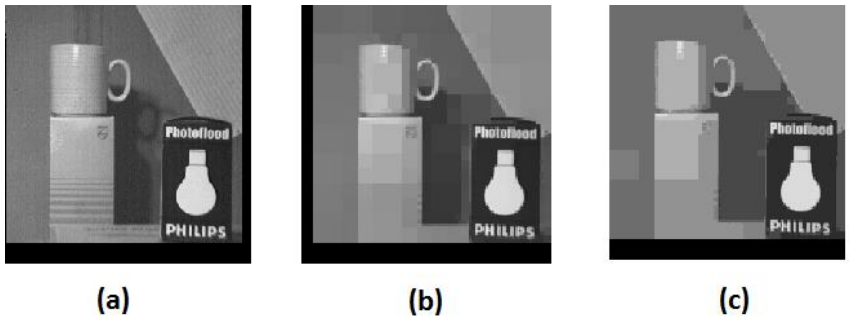


Figura 5: Exemplo de Sequência do algoritmo de split and merge

2.5 Análise e Reconhecimento de imagens

Na etapa de análise e reconhecimento de imagens estão as tarefas relacionadas com a interpretação automática das informações presentes nas imagens. Nesta fase do processamento a inteligência dos algoritmos computacionais é aplicada em busca dos resultados práticos que sirvam aos interesses da aplicação desejada.

2.5.1 Deteção de bordas

Uma borda em uma imagem pode ser definida como uma constante de valores que diferem do fundo ou de outro objeto (JAHNE, 1997). Os métodos para deteção de bordas procuram pela descontinuidade de níveis de cinza, cor, textura ou alguma característica entre *pixels* vizinhos, o que normalmente representa uma fronteira em alguma região da imagem.

A deteção de bordas consiste basicamente em duas etapas. Na primeira todos os *pixels* são avaliados na procura de propriedades de bordas e são listados aqueles que, poderão ser eventualmente pontos pertencentes a bordas. A avaliação é feita através de uma máscara, conhecida como operador que é comparada com os *pixels*. Na segunda etapa, a lista dos *pixels* pré-selecionados é reduzida usando informações extraídas na primeira etapa.

Existem algumas maneiras de implementar a deteção de bordas. Alguns operadores clássicos são: o Roberts, Prewitt e o Sobel (JAHNE, 1997; SONKA; HLAVAC; BOYLE, 1998). Na figura 6 é apresentada as mascaras destes operadores. Na figura 7 são apresentados exemplos de utilização dos três operadores referidos na deteção de bordas numa imagem.

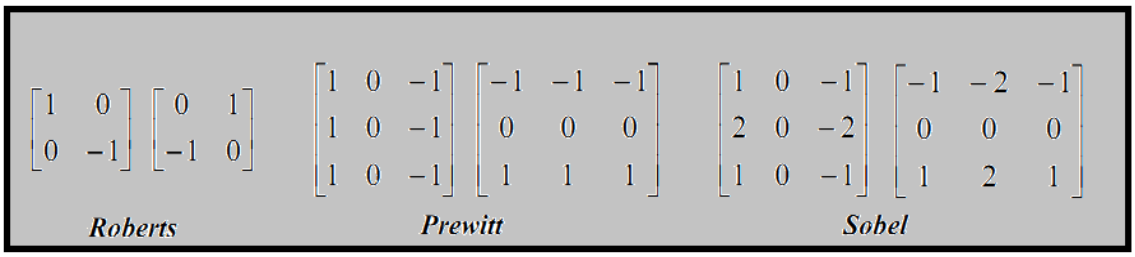


Figura 6: Operadores para deteção de bordas

Figura 7: Exemplo da deteção de bordas. (a) Imagem original, (b) Roberts, (c) Prewitt, (d) Sobel.

Na deteção de bordas temos também modelos de contornos ativos (*Snakes*) que são bastante utilizados na segmentação e reconhecimento de objetos em imagens. Os modelos de contornos ativos utilizam informações locais sobre um contorno e informações sobre a imagem como um todo ou sobre segmentos específicos. Estes são muito utilizados no rastreamento de bordas de objetos, sendo em geral aplicados conjuntamente com técnicas de filtragem usadas na deteção de pontos de bordas. Estes métodos iniciam-se com uma configuração arbitrária, um contorno inicial que evolui até contornar o objeto de interesse. Este comportamento dinâmico faz com que sejam também conhecidos como modelos deformáveis. A evolução dos modelos deformáveis começa

com uma curva inicial e esta é controlada pela ação de forças internas (intrínsecas à geometria da curva) e externas (derivadas da imagem) atingindo o equilíbrio sobre a fronteira do objeto.

Os modelos paramétricos permitem trabalhar com uma curva elástica que pode dinamicamente se adequar às formas dos objetos em resposta à ação de forças internas e externas.

Estes sistemas de forças provem das equações de *Euler-Lagrange* associadas a um funcional de energia definido sobre a curva. Este funcional de energia é constituído por dois termos: um referente à energia interna da curva que incorpora as restrições geométricas estabelecidas à priori para a solução. Outro é referente à energia externa que engloba elementos de interesse na imagem (pontos de borda, texturas, etc.). O propósito destes sistemas de forças é fazer com que a curva (*snake*) se mova em direção a uma borda, ficando em equilíbrio estável sobre a mesma ao atingir o valor mínimo de energia do sistema.

2.5.2 Reconhecimento de padrões em imagens

O Reconhecimento de padrões em imagens abrange uma gama de aplicações úteis. Em geral, o homem não apresenta problemas em reconhecer padrões, o seu sentido de visão é extremamente eficiente, ao contrário do computador.

Segundo Gonzales (GONZALEZ; WOODS, 1993), a dificuldade de entendimento dos problemas de reconhecimento e de como a tarefa pode ser implementada de forma eficiente pela máquina, resulta em sistemas de reconhecimento altamente especializados. Segundo este, um padrão é formado por um conjunto de características, por exemplo, cor, intensidade de cinza e textura.

Algumas das principais abordagens de reconhecimento de padrões podem ser divididas em três linhas de ação (CAETANO, 2000; GONZALEZ; WOODS, 1993; BISHOP, 1995) similaridade de moldes, classificação estatística e redes neuronais artificiais.

Na abordagem da similaridade de moldes, existe um molde padrão que é representado por determinada característica e é comparado com a imagem, tendo como base uma medida de similaridade. Para gerar um molde é necessário conhecimento do

objeto alvo, sendo extremamente importante que este deva tolerar variações de translação, rotação e de escala.

Na abordagem estatística, os padrões são representados por vetores de características. O ideal é utilizar características que reconheçam diferentes classes de padrões. Um fator importante dessa abordagem é a fronteira de decisão entre as características de diferentes classes de padrões. As fronteiras são obtidas através de distribuições de probabilidade envolvendo os padrões pertencentes a cada classe. Para isso são criadas funções de densidade de probabilidade.

A utilização das redes neuronais artificiais no reconhecimento de padrões surge por estas serem adaptáveis a problemas complexos para sistema convencionais (BISHOP, 1995; MURINO, 1998).

2.6 Características utilizadas no processamento de imagens

A análise de uma imagem digital através de ferramentas computacionais é realizada a partir de três abordagens, baseada nos atributos de contexto, espectrais ou espaciais da imagem. Na análise baseada nos atributos de contexto, o processamento considera a vizinhança dos *pixels* da imagem. A análise espectral está relacionada com o brilho isto é, a quantidade de energia refletida pelos alvos que compõem a imagem. Esta é a forma mais utilizada devido a sua simplicidade, utilizando o nível de cinza ou a cor do *pixel*, por exemplo. Os atributos espaciais tratam de informações como a forma dos objetos e textura da imagem, estes relacionam-se com a distribuição espacial dos *pixels* na imagem digital.

Embora o processamento baseado em atributos espectrais seja o mais utilizado, em alguns casos podem ser necessários diferentes atributos, principalmente em imagens de má qualidade, contaminadas de ruídos. Para esses casos convém anexar informação extra. A textura é um atributo espacial da imagem portanto a sua utilização pode trazer melhorias ao processo de classificação de imagens.

2.6.1 A Textura

A textura pode ser caracterizada através de padrões repetidos na imagem, de forma ordenada ou aleatória. É uma informação visual ou táctil percetível facilmente pelo ser humano, que pode ser quantificada através de processamento de imagens.

As características utilizadas nesse trabalho são correspondentes aos atributos visuais comuns de texturas, pois o critério para a seleção destas cinco características foi a influência delas na perceção de diferentes texturas. Estes atributos são baseados em propriedades estatísticas de intensidade do histograma dos níveis de cinza da imagem. Desta forma, média, contraste, aspereza, suavidade, uniformidade e entropia foram estudadas e quantificadas para fins de identificação, diferenciação e classificação. A média é a medida que encontra o valor central dos níveis de cinza para a determinação da região. O contraste corresponde a variações dos tons de cinza presentes na imagem. A suavidade está associada à característica táctil que é percetível visualmente. A uniformidade verifica se a imagem tem poucas transições de níveis de cinza, enquanto a entropia é a medida de desorganização encontrada na imagem (GONZALEZ; WOODS; EDDINS, 2004).

Capitulo 3 – Algoritmo Aplicado

3.1 SURF (Speed-Up Robust Features)

SURF (*Speed-Up Robust Features*) é um detetor robusto utilizado na deteção de regiões de interesse. Este foi primeiramente apresentado por Herbert Bay na Conferência Internacional sobre Visão Computacional realizada na Áustria (BAY, HERBERT, 2008), em maio de 2006, mostrando que pode ser usado em tarefas como o reconhecimento de objetos ou reconstrução 3D. Este foi) inspirado em parte pelo detetor SIFT (*Scale Invariant Feature Transform*). A versão padrão do SURF é muito mais eficaz e mais rápido do que o SIFT e os seus autores alegam ser mais robusto contra diferentes transformações na imagem. O detetor SURF baseia-se em somas de 2D *Haar Wavelet* e faz um uso eficiente de imagens integrais.

Este faz a aproximação de um número inteiro para o determinante de *Hessian Blob Detector,* que pode ser calculado de forma extremamente rápida, com uma imagem integral (3 operações com números inteiros). Para todas as características, este usa a soma da resposta *Haar wavelet* em torno do ponto de interesse. Esta informação é tratada por forma a realizar operações com o intuito de localizar ou reconhecer objetos, pessoas ou faces e de extração de pontos de interesse. Este algoritmo é parte do que a inteligência artificial precisa, pois este é capaz de formar um sistema capaz de interpretar as imagens e determinar o seu conteúdo.

3.1.1 SURF Detector

O detetor SURF centra a sua atenção sobre as estruturas *blob* (regiões de interesse) na imagem. Estas estruturas podem ser encontradas nas esquinas dos objetos, mas também em locais onde a reflexão da luz em superfícies é máxima (manchas claras). Em 1998, Lindeberg notou que o operador *Monge - Ampère* e *Gaussian* que são filtros derivativos, poderiam ser usados para localizar características. Este conseguiu detetar *blobs* ao fazer passar a imagem alvo pelo determinante *Hessiano,* que é constituída por diferentes matrizes 2-D Gaussianas de segunda ordem. Esta métrica é então dividida pela variância, σ^2 Gaussiana, para normalizar a sua resposta temos a seguinte equação:

$$DoH(x, y, \sigma) = \frac{Gxx(x, y, \sigma).Gyy(x, y, \sigma) - Gxy(x, y, \sigma)^2}{\sigma^2}$$

Equação 1: Determinante Hessiano

O máximo da resposta do filtro ocorre em regiões onde ambos G_{XX} e $ᴗ_{yy}$ são fortemente positivos, e onde G_{Xy} é fortemente negativo. Logo, estes extremos ocorrem em regiões da imagem com grandes variações do gradiente de intensidade em várias direções. Visualizando na imagem, significa que se refere a cantos ou manchas.

A razão pela qual muitos sistemas de deteção de regiões contam com filtros de Gauss é com o intuito de conseguir diminuir ou até eliminar o ruído que constitui a imagem. A utilização de filtros (LOWE, DAVID, 1999) por vezes tem como desvantagem a perda de certos detalhes da imagem.

O algoritmo SURF introduz a noção de escala através de oitavas, cada oitava contem amostras de uma pequena região. O tamanho da região é proporcional para as escalas usadas (LINDEBERG, TONY, 1998). Cada oitava adjacente dobra o tamanho da região com o intuito de reduzir a quantidade de pesquisa necessária para escalas maiores. O uso de oitavas pode ter como desvantagem possuir mais erros nos resultados obtidos do que no uso de escalas maiores, devido ao facto dos incrementos serem maiores.

$$sgn\{Gxx(x, y, \sigma) + Gyy(x, y, \sigma)\} = \begin{matrix} +1 & blob\ claro\ sobre\ fundo\ escuro \\ -1 & blob\ escuro\ sobre\ fundo\ claro \end{matrix}$$

O algoritmo SURF resume-se em quatro passos:

1. Forma a resposta que diz qual a escala a ser usada, através da convulsão da imagem alvo com os filtros DoH com σ diferente;

2. Procura máximos locais através dos *pixels* vizinhos e escalas adjacentes usando oitavas;

3. Interpolação para a localização de cada máximo local encontrado;

4. Para cada ponto de interesse tem-se x, y, σ , a magnitude DoH, e o sinal Laplaciano.

3.1.2 SURF Descriptor

Para se conseguir descriminar cada característica, o algoritmo SURF resume as informações de *pixels* dentro da vizinhança. Em primeiro lugar é determinado a

orientação para cada característica, através da convulsão dos *pixels* na vizinhança com os filtros *Wavelet Haar* horizontais e verticais, como é apresentada na Figura 8. Estes filtros podem ser comparados a métodos de calcular derivadas direcionais através da intensidade da imagem. Este descritor usa mudanças de intensidade por forma a caracterizar a orientação da imagem, fazendo com que assim o algoritmo seja mais robusto pois este torna-se independentemente da orientação dos objetos ou dos sensores que captem as imagens alvo de diferentes perspetivas.

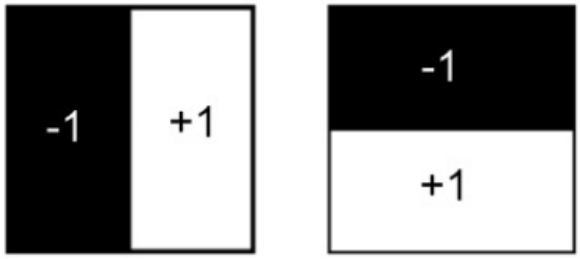


Figura 8: Filtros Wavelet Haar horizontais e verticais

Com a utilização de respostas *Wavelet Haar* para gerar um vetor unitário que represente cada característica e a sua vizinhança, o algoritmo SURF tem propriedades desejáveis, sendo elas a invariância quer em iluminação quer em contraste.

3.1.3 SURF Comparator

Para o reconhecimento de um objeto alvo é necessário ter uma base de dados sólida, onde temos o nosso objeto sobre um fundo liso e escuro preferencialmente a ocupar a imagem inteira. Para melhorar o comparador cada objeto deve ser representado por várias imagens ou seja, capturadas em diferentes pontos de vista, por forma ter imagens que contêm vários vetores característicos.

O comparador tem por base a comparação entre os descritores por forma a procurar pares de semelhança entre eles. O método consiste em obter os pontos característicos da de uma imagem e seu descritor e fazer o mesmo com a outra imagem. Por último comparar as correspondências dos descritores de ambas.

3.2 Sonar Imagenex SportScan

No contexto desta dissertação, o sonar usado é o Imagenex SportScan. A ligação *software* e o comando do sonar é feita via RS-232 para o sonar, com um Baud Rate de

115200 Bps, o sonar transmite um conjunto de impulsos acústicos, recebe-os e posteriormente reencaminha os dados referentes a cada fatia. O comando enviado configura o sonar e permite definir algumas características tais como:

- Alcance do feixe acústico: 15, 30, 60, 90 e 120mts;
- A largura do impulso acústico: 0 a 250 µs;
- Resolução da fatia acústica: Baud Rate de retorno de dados;
- Frequência do feixe acústico: 330kHz ou 800kHz;
- Número de pontos, *Data Points*: 250 ou 500 pontos por lado (Estibordo ou Bombordo);
- *Delay* entre o retorno de dados da fatia anterior, e emissão de nova onda acústica: 0 a 510 ms.

O alcance do feixe acústico, *range scale*, impõe um alcance máximo para o feixe acústico e impõe a resolução de cada *pixel* na imagem acústica.

O número de pontos escolhidos por fatia acústica ou o modo de obtenção da imagem acústica pode ser feita utilizando usando o feixe acústico a estibordo e a bombordo, e ai obtém-se uma imagem de 500 pontos, mas com apenas 250 pontos por lado. Apenas um feixe acústico, em um dos lados do sonar, quer a estibordo ou bombordo, obtém-se uma imagem também de 500 pontos, mas correspondendo apenas a um dos lados do sonar.

O feixe acústico do sonar pode ser configurado em dois modos, 330kHz ou 800kHz. A frequência do feixe acústico está inversamente relacionada com o comprimento de onda e diretamente relacionada com a velocidade de propagação do feixe no meio. Assim, para o modo de funcionamento de 330kHz espera-se um ângulo de incidência superior que para o modo de funcionamento de 800kHz. Como é referido no manual do Imagenex SportScan com a frequência nos 330kHz temos um feixe de 1.8º x 60º e nos 800kHz temos um feixe de 0.7º x 30º.

3.2.1 Software Imagenex

A Imagenex fornece um software "Win881ss.exe" que permite gravar ficheiros provenientes do sonar no formato (filename).81s. Win881ss é um programa para Windows que permite controlar, visualizar e gravar a informação proveniente do sonar.

O programa faz a recepção dos dados provenientes do sonar através de RS-232 COM port, possibilitando a ligação de um recetor GPS que fornece a longitude/latitude, a velocidade e o sentido em que o sonar se movimenta. O sonar SportScan pode ser operado em diferentes alcances, frequências e ganhos. No display do software é apresentada uma imagem com uma resolução de 800 x 600 pixels, onde é possível ver os dados em tempo-real ou visualizar um ficheiro já gravado.

No display do programa é onde são apresentados os dados do sonar. Nele pode-se ver a imagem dos dois bordos em simultâneo caso a transmissão seja feita dos dois bordos do sonar, ou em separado apenas a imagem de um bordo. Em baixo do display encontra-se os comandos que permitem controlar a frequência, o ganho, os bordos de onde vai ser feita a transmissão do feixe acústico e a largura do impulso. Encontra-se também informações adicionais tais como a data/hora, as coordenadas latitude/longitude, a velocidade, o rumo e a profundidade a que se encontra o sonar. Possui também um menu que tem várias opções tais com a cor dos dados que são apresentados no display, o comprimento de determinado objeto de interesse, a medição da altura, a correção da velocidade que pode ser através de GPS ou ser introduzida manualmente e a possibilidade de ampliar a imagem.

Para operar o SportScan é preciso garantir que o conector do cabo de reboque esteja ligado à porta serie RS-232 do computador, alimentar o sonar com 12 VDC e uma corrente de 0,5A, garantindo que a porta esteja selecionada corretamente no menu das portas do *software*.

3.2.2 Ficheiro Sonar

Quando se faz a gravação através do sonar SportScan para um ficheiro .81S, um conjunto de bytes são gravados por feixe acústico emitido, ou seja a cada pingo do sonar. A versão do sonar SportScan presente nesta Tese de mestrado emite 500 pontos de informação (250 pontos por bordo).

A cada pingo do sonar é gravado no ficheiro uma *frame* composta por 640 *Bytes* (0-639 *Bytes*) de informação. Dos 0 aos 99 (100 *Bytes)* temos o *File Header*, dos 100 aos 111 (12 *Bytes*) o *Sonar Return Data Header*, dos 112 aos 611 (500 *Bytes) o Sonar Return Echo Data*, no 612 (1 *Byte)* temos o *Sonar Return Termination Byte, dos 613 ao 637 (25 Bytes) o Zero Fill* e por último dos 638 até aos 639 (2 *Bytes)* o *Pointer to Previus Ping*

que são os dois últimos *Bytes* que contêm um número de 16 *bits* que consistem na soma do número de *Bytes* do feixe acústico anterior. Estes são usados em caso de necessidade de nova reprodução do ficheiro de trás para a frente.

Na *File Header* (0 até 99 Bytes) os três primeiros *Bytes* contêm em código ASCII informação sobre o tipo de ficheiro ou seja no Byte 0 temos em ASCII o número ’8’ no *Byte 1 e 2 temos o numero ‘1’ e a letra ’S’. No* Byte 3 temos o número de Bytes que serão usados para dados, ou seja 500 *Data Bytes*. Nos *Bytes 4 e 5 temos o total de Bytes, isto é o número de Bytes que são gravados no disco a cada feixe acústico, que são 640. Os Bytes* 6 e 7 contêm o número de *Bytes* do *SportScan* que são 513. Do *Byte* 8 ao 28 temos o grupo data hora, onde são gravadas duas *strings*. Na primeira com o seguinte formato “DD-MMM-YYYY” e a segunda “HH:MM:SS”. Dos 29 aos 36 são *Bytes Reserved* e tem o valor 0. O mesmo acontece no *Byte 37 que tem sempre o valor 16.* O *Byte* 38 é relativo ao canal que se está a usar, isto é tem o valor 1 se for o de estibordo, o valor 2 de for o de bombordo e o valor 3 se estiverem ambos a ser usados. O *Byte 39 é referente ao ganho que pode variar entre 0 e 40 dB. Dos bits 40 até ao 46 volta-se a ter bytes Reserved*, que contêm sempre os mesmos valores, dos 40 até 42 o valor de 0, o 43 valor de 5, o 44 valor 9, 45 valor 100 e o 46 valor de 0. Dos *Bytes* 47 até ao 70 temos a informação da posição GPS que se divide em duas partes, os primeiros 12 *Bytes (47-58) a informação é relativa a latitude que é gravada* numa *string* com o seguinte formato “_dd.mm.xxx_N”. Os segundos (59-70) são relativos a longitude que são gravados numa *string* semelhante a da primeira parte “ddd.mm.xxx_E”. Os *Bytes* 71 e 72 definem o tempo entre a repetição do novo feixe acústico. O *Byte* 73 é referente à velocidade a que se está a deslocar o sonar. Os *Bytes* 74 e 75 é onde se grava a informação de proa que provem do GPS. No *Byte 76* é relativo à frequência que se esta a usar, caso seja *Low Frequency* o valor será 0, se for *High Frequency* o valor será 1. No *Byte 77* temos o *Channel Balance* que pode variar entre dois valores, no valor 0 caso tenhamos 0 dB em ambos os bordos, ou no valor 1 caso o ganho seja diferente de 0. O *Byte 78 só é usado caso se esteja a fazer uma gravação e fica assume o valor 1. No último grupo de Bytes* (78 até aos 99) relativos ao *File* Header são *Bytes Reserved* que ostentam o valor 0.

Analisando agora o *Sonar Return Data Header* que ostenta os *Bytes* do 100 até ao 111. Começando pelo 100, 101 e 102 em que cada um destes tem uma letra em código ASCII sendo estas ‘I’ o ‘H’ e o ‘X’. No *Byte* 103 temos o *Head ID* que tem sempre o

valor de 0x10. No *Byte* 104 temos o Serial *Status* onde é confirmado se a ligação para transição de dados está feita corretamente. No *Byte* 105 temos o tipo de SportScan que sendo de uma só frequência assume o valor 0, alterando para o valor 1 caso tenha duas frequências. O *Byte 106* é relativo ao canal que se está a usar, assumindo os seguintes valores: 1 a estibordo, 2 a bombordo e o valor 3 se estiverem ambos a ser usados. O *Byte 107* é relativo ao alcance sonar informando se está a operar a se está a operar a 15, 30, 60, 90 ou 120 metros. Os *Bytes* 108 e 109 são R*eserved e tem o valor de 0. Os últimos dois* bytes do *Sonar Return Data Header são Data Bytes que contem o número de Bytes* que proveem do *SportScan que são 500 em que o cabeçalho não está incluído.* No *Sonar Return Echo Data* que vai dos 112 até aos 611 contem a informação que provem do eco sonar. Se tivermos o *Byte* 106 a 1 (estibordo) ou a 2 (bombordo) temos então 500 *Bytes* que são usados para transportar a informação do eco. Caso o *Byte* 106 estiver a 3 (ambos os bordos) temos então 250 *Bytes* por bordo. O *Byte* 612 é um Byte de fim de dados que vem na forma de 0xFC. É de salientar que o nível de intensidade dos dados varia entre 0 e 127.

3.3 Deteção através do Ficheiro Sonar

O fluxograma apresentado na figura 9 apresenta a estrutura do algoritmo d*eteção através de ficheiro sonar* que possibilita a deteção de náufragos como o próprio nome o diz através de um ficheiro. Este procura inspiração na estrutura clássica de um problema de reconhecimento de padrões conforme desenvolvido no capítulo 2, e é composto por três fases distintas:

- Simplificação da imagem;
- Extração de características relevantes para a classificação da imagem;
- Classificação da imagem (resposta positiva ou negativa na presença de um náufrago).

A fase da simplificação da imagem tem por objetivo a melhoria no desempenho final do processo de identificação de náufragos. Esta visa a redução de alguns efeitos indesejáveis, tais como o ruído. Para esse efeito cada imagem presente no ficheiro a analisar passa pela função *linha de água. Nesta é feita a* segmentação da imagem para se ficar apenas com a zona de possível aparecimento do náufrago. Posteriormente e obtendo a zona de interesse esta passa por um filtro onde é removido parcialmente o ruído.

A fase da extração de características e de classificação de imagem encontram-se no mesmo módulo (*Função Detect*), onde primeiramente se extrai as características da imagem a ser analisada e depois se procede à comparação das características da mesma com as características das imagens presentes na base de dados.

Como é possível visualizar no Fluxograma da Figura 9 o ficheiro sonar começa a ser analisado. Ao mesmo tempo vai enviando imagens que serão visualizadas no *Display*. A imagem que esta a ser visualizada vai se deslocar para o módulo de simplificação onde será encontrada a linha de água e reduzido o ruido. Após este processo a imagem em análise passa para o módulo de extração de características e é comparada com a base de dados. Caso se tenha uma resposta positiva na deteção esta é assinalada no *Display*. *A*pós estes módulos é feita a verificação se ainda existe mais dados no ficheiro sonar se houver o ciclo é recomeçado com uma nova imagem por analisar. Este Algoritmo só irá parar quando o ficheiro sonar chegar ao fim e não tiver mais dados para serem analisados.

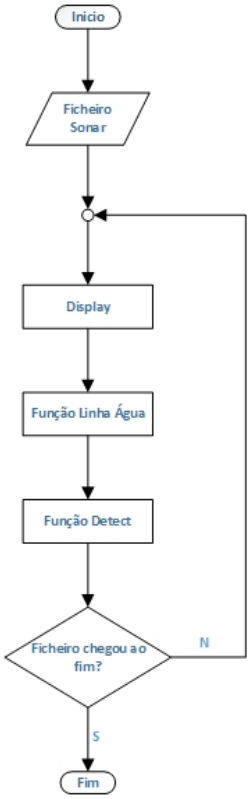


Figura 9: Fluxograma da função Ficheiro Sonar

3.4 Deteção através de vídeo Sonar

A deteção através de um vídeo sonar baseia-se nos mesmos princípios da secção anterior sendo estes a simplificação da imagem, a extração das características e a classificação da imagem. A utilização de um vídeo sonar surge como um protótipo para a deteção através do ficheiro sonar. O que difere em ambos, são os dados de entrada ou seja em vez de serem provenientes do próprio ficheiro sonar são eram gravados no formato AVI (*Audio Video Interleave*) para depois serem analisados.

O fluxograma apresentado na figura 10, apresenta uma estrutura do algoritmo *Deteção através de Vídeo Sonar* que possibilita a deteção de náufragos através de um vídeo sonar. Este é composto pelas três fases de um problema de reconhecimento de padrões, começando pelo módulo de simplificação em que o vídeo sonar após ter sido decomposto pelas suas *frames* estas irão passar pela função *linha de água onde serão* segmentadas e filtradas ao nível do ruido por forma a ficar apenas a área de possível aparecimento de um náufrago. *Cada frame* constituinte do vídeo sonar após ter passado este módulo é gravada numa pasta para depois ser analisada. Os módulos de extração de características e de classificação encontram-se na função *detect* onde as imagens constituintes do vídeo sonar após terem sido tratadas vão ser comparadas com a base de dados. Caso se obtenha uma resposta positiva de um possível náufrago este é assinalado no *Display*, após estes módulos é feita uma verificação se ainda existe mais imagens na pasta por serem analisadas. Caso exista, uma nova imagem entra no módulo de extração de características e classificação. O Algoritmo de deteção através de vídeo sonar para quando não houver mais imagens na pasta por analisar.

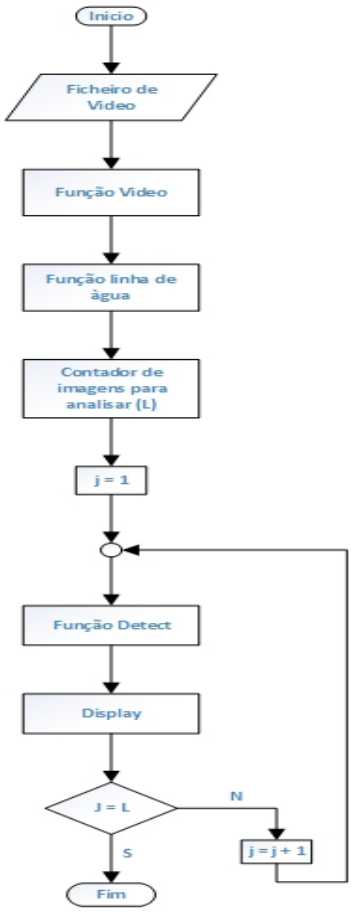


Figura 10: Fluxograma da função Deteção através de Vídeo Sonar

3.5 Função Linha de Água

O fluxograma apresentado na figura 12, apresenta a estrutura da função *linha de água* que tem como objetivo segmentar a imagem por forma a ficar com a zona de possível aparecimento de um náufrago e reduzir o ruido da imagem sonar à superfície. O filtro usado na imagem é o do *Sobel*. Esta função tem duas variáveis de decisão que são o limiar de intensidade (LI) e o limiar agua (LA), em que o LI é a variável que tem a intensidade mínima necessária para que o *pixel* seja considerado pertencente da superfície, enquanto a variável LA tem o número mínimo de pixels numa linha para que esta seja considerada como a linha de água. A função inicia-se com a receção de uma imagem em que esta é vista como uma matriz de intensidades (m x n), ou seja cada linha

e cada coluna são constituídas por níveis de intensidade que podem variar dos 0 aos 255. Após a entrada da imagem retira-se quantas linhas tem a matriz (m), por forma a iniciar um ciclo que corra linha à linha a matriz. O processo de deteção da linha de água consiste em correr a imagem de cima para baixo, e verificando se a linha que está a ser analisada tem *pixels* que superem a variável LI, caso tenha estes são contabilizados para depois se verificar com a variável LA se existe o número suficiente para que a linha seja considerada como linha de água. Caso não seja encontrada nenhuma linha em que se tenha verificado a condição da variável LA, a linha de água é considerada como a altura da matriz (m). Na figura 11 pode-se visualizar exemplos da função *Linha de Água* onde na figura 11 (a) temos uma imagem sonar que após passar pela função em questão fica segmentada com a zona de possível aparecimento de náufragos, como é possível de visualizar na figura 11 (b).

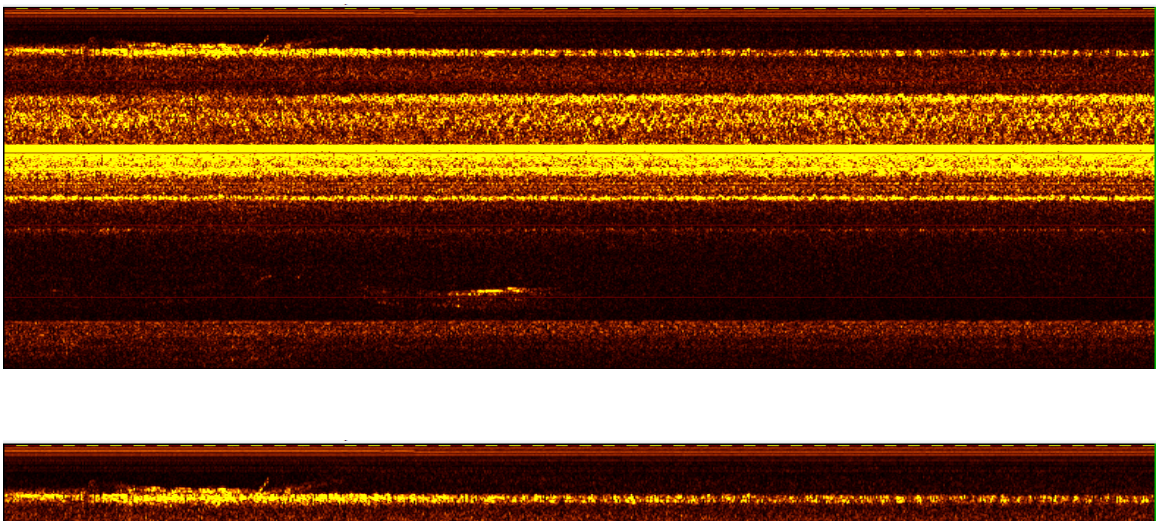


Figura 11: Função Linha de Água. (a) Imagem Sonar Original, (b) Imagem Segmentada pela Função.

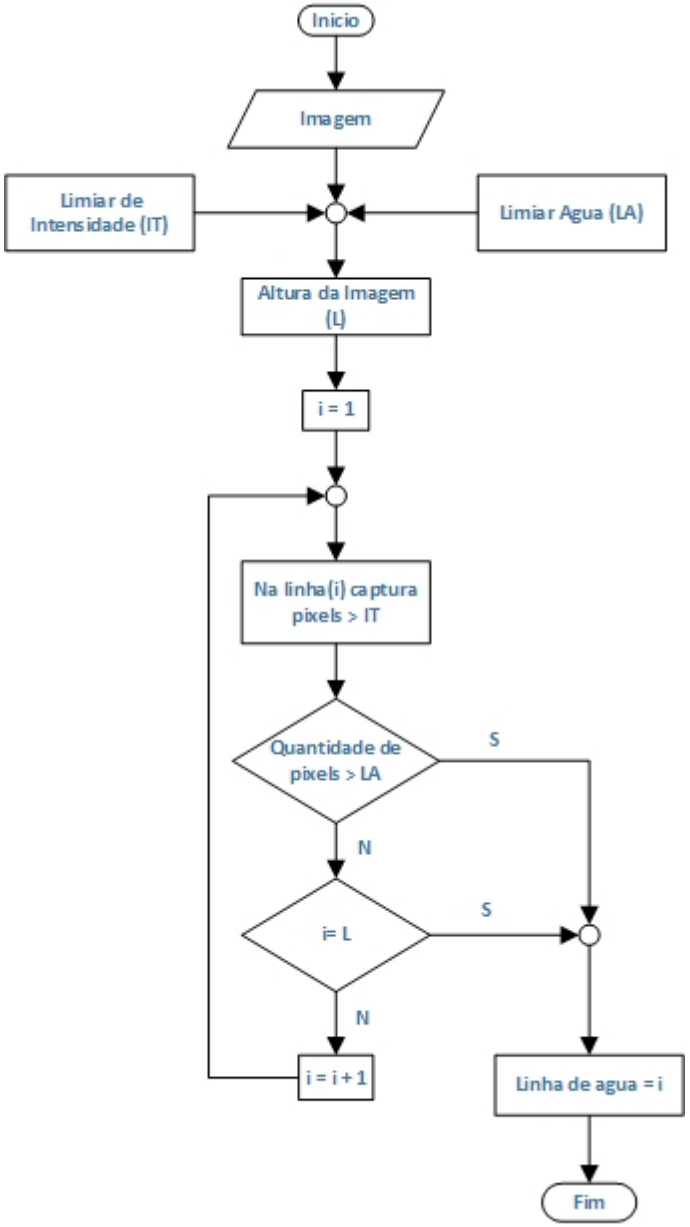


Figura 12: Fluxograma da Função Linha de Água

3.6 Função Vídeo

A função *vídeo* tem como objetivo principal receber um ficheiro vídeo e depois separa-lo pelas suas *frames* e grava-las para depois serem analisadas. Como se pode visualizar no fluxograma da figura 13, esta função ao receber o ficheiro vídeo vai em primeiro lugar verificar quantas *frames* são usadas no vídeo, Após ter o total de *frames* (F), esta inicia um ciclo em que a *frame* em análise passa pelo módulo de simplificação ou seja a função *linha de água. Findo este processo a frame* já segmentada e com o ruído

reduzido é gravada numa pasta para mais tarde ser analisada. Este processo é repetido até que não haja mais *frames* por passarem pelo módulo de simplificação.

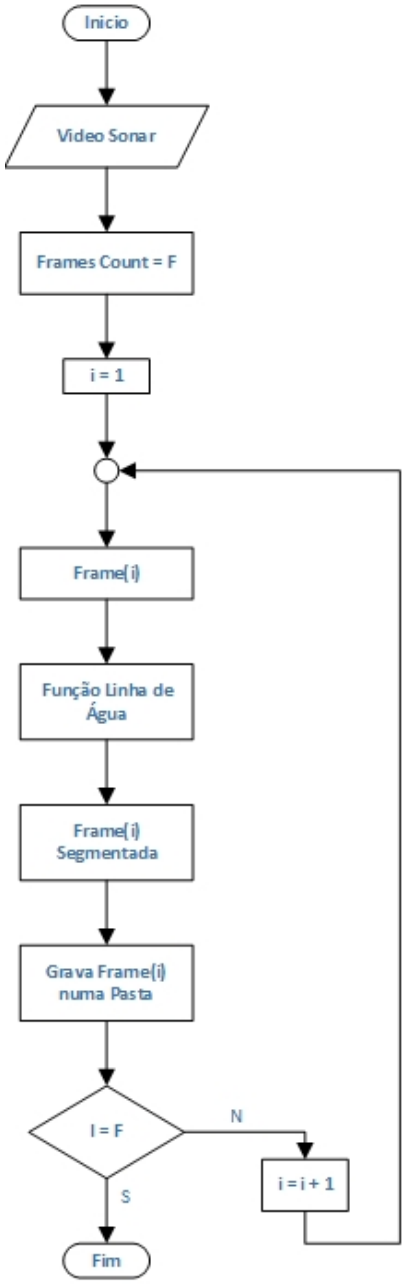


Figura 13: Fluxograma da Função Vídeo

3.7 Função Detect

A função *Detect* representa os módulos de extração de características e o de classificação. É nesta função que se verifica se existe alguma semelhança na imagem em análise com alguma presente na base de dados. No fluxograma da figura 14 pode-se ver que esta função é iniciada com três dados de entrada, sendo eles a imagem em análise, a frequência e o ganho. Os dados de entrada, quer a frequência, quer o ganho são dados que irão permitir agilizar o processo na busca por semelhança na base de dados. Após os dados de entrada serem carregados é feita a contabilização da dentro da pasta *base de dados* para a frequência e ganho respetivo. Inicia-se um ciclo em que se extrai as características da imagem em análise e das imagens presentes na base de dados. Realiza-se a comparação das mesmas e caso se verifique que existe um determinado número de pares de semelhança, a imagem é considerada como tendo um náufrago presente nela e este é assinalado nesta. Este ciclo termina tendo em conta duas condições, caso seja encontrado um possível náufrago ou não haja mais imagens na base de dados, esta é classificada como não havendo náufrago. Este processo de deteção, até obter ou não a resposta positiva do náufrago demora cerca de 2,3 segundos.

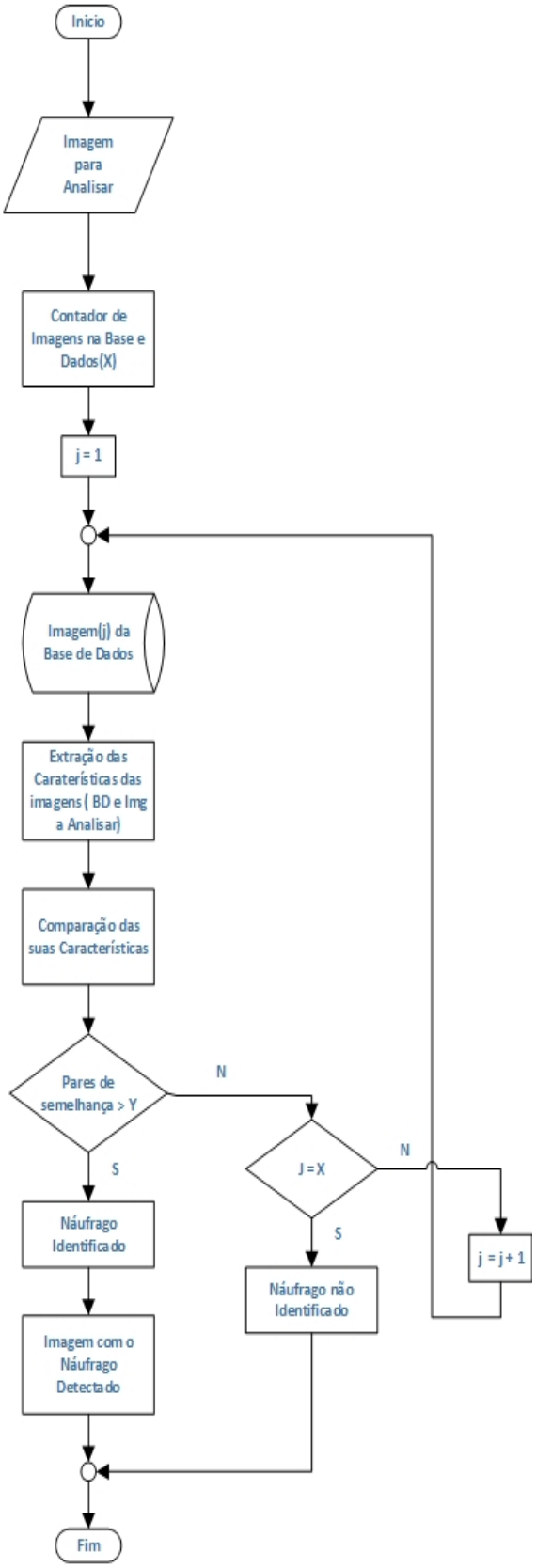


Figura 14: Fluxograma da Função Detect

Na figura 15 pode-se visualizar exemplos da função *Detect* onde foi detetado possíveis náufragos.

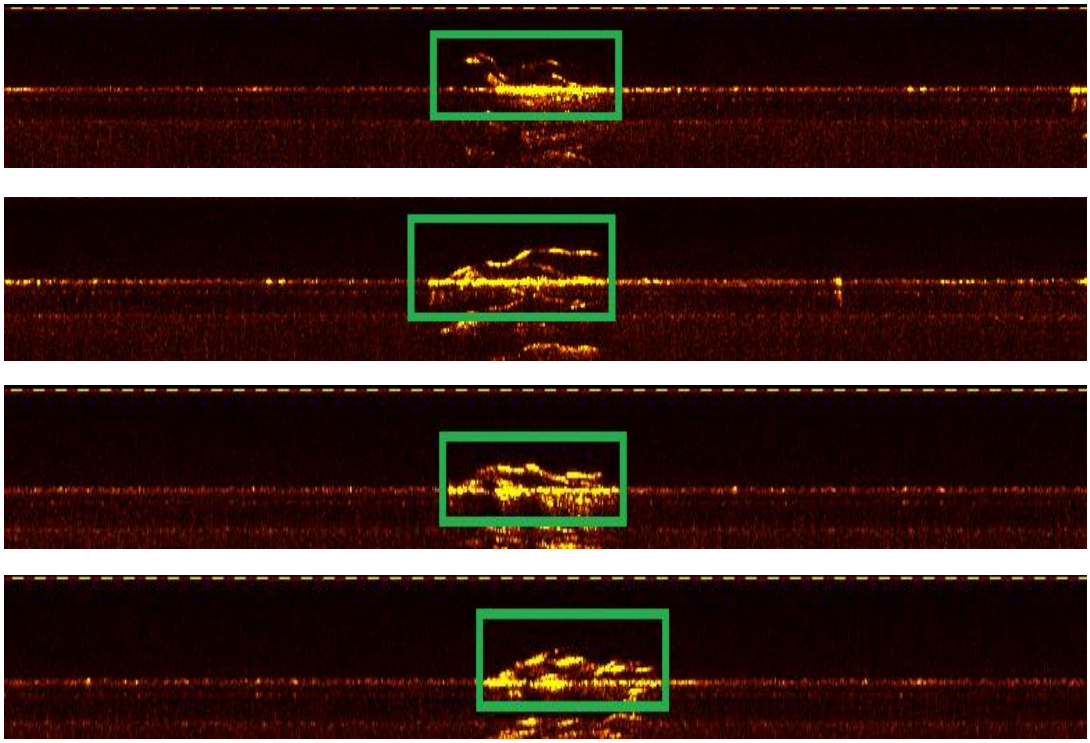


Figura 15: Função Detect

Capitulo 4 – Resultados e sua Discussão

A tese apresenta um método para localizar automaticamente náufragos a partir de uma imagem sonar, traduzindo-se numa tarefa complexa resultado dos diversos aspetos discutidos ao longo da dissertação. Um dos aspetos relevantes é a baixa qualidade de imagem. Para o desenvolvimento e obtenção dos resultados, foram realizados diversos testes com o intuito de avaliar o melhor método para tal.

Neste capítulo serão apresentados e discutidos os testes realizados para ajuste do método de detenção. Ao longo dos testes, foram recolhidas 300 imagens em que se encontrava um corpo humano e 50 imagens com objetos. As mesmas fazem parte de uma amostragem de imagens sonar, que têm como resolução 786x506 *pixels*, no formato *Portable Network Graphics* (png). Da Figura 18 à Figura 34 é apresentado o conjunto de imagens recolhidas nos testes. A apresentação destas imagens junto ao texto, embora em uma resolução menor, servem para que o leitor perceba algumas peculiaridades e diferenças visuais entre elas.

As imagens sonar foram conseguidas através de um sonar SideScan da marca IMAGENEX SportScan, recolhidas em ambiente controlado (piscina de água salgada).

4.1 Ambiente de Testes

Em primeiro lugar é de salientar que os resultados obtidos nesta secção foram conseguidos em parceria com o camarada ASPOF EN-AEL Ramos da Palma que também irá usar estes resultados na sua dissertação de mestrado.

Para a formulação de uma base de dados foram selecionadas quatro possíveis posições de se encontrar um náufrago e utilizou-se três objetos de materiais diferentes para verificar o seu eco sonar.

Para a recolha de imagens sonar usaram-se duas frequências que o sonar possibilita, que são 330kHz (*Low Frequency*) e 800KHz (*High Frequency*), e fez-se variar o ganho entre 20 dB e 30 dB. O sonar dentro de água encontrava-se a 2m de profundidade e virado para a superfície.

As posições selecionadas passiveis da existência de um náufrago à deriva consistem em barriga para baixo, de costas e de lado. Foram utilizados um guarda-chuva, uma tabua e três latas unidas, como objetos adicionais. Em cada uma destas posições

fizeram-se vinte passagens por cima do sonar com o ganho em 20 dB e com a frequência em High e em *Low* Em seguida repetiu-se o processo para o ganho em 30 dB. Em *Low Frequency* com ganho de 30 dB em cada uma das posições os resultados afastaram-se do que era expectável devido à grande quantidade de ruido presente nas imagens sonar, por isso apenas foram realizadas cinco passagens por cima do sonar.

O processo realizado para as posições selecionadas repetiu-se nos objetos com a diferença no número de passagens por cima do sonar, que em vez de vinte foram cinco em cada frequência e ganho. O intuito dos objetos era ter o conhecimento do contraste do eco sonar de um corpo humano para objetos feitos de diferentes materiais tais com plástico, madeira e latão.

4.2 Base de Dados

Nesta secção é apresentada as imagens que constituem a base de dados, onde a imagem que vai ser analisada é comparada com as imagens presentes na base de dados para a mesma frequência e ganho.

As posições presentes na base de dados ostentam a posição de barriga para baixo, de costas e de lado. A base de dados também contém imagens relativas a objetos tais como o guarda-chuva, uma tabua e três latas unidas, por forma a testar o protótipo. A base de dados está dividida pelas posições, e dentro da pasta de cada posição, temos uma subdivisão pelas frequências (*High* e *Low*), e dependendo da frequência selecionada temos os dois ganhos estudados o ganho de 20dB e o de 30dB. Dentro da respetiva posição, frequência e ganho temos 20 imagens sonar onde temos um corpo à deriva.

4.2.1 Náufrago de Barriga para Baixo

Cada uma das Figuras 16, 17 e 18 são constituídas por 20 imagens de um corpo à deriva na posição em que o náufrago se encontra de barriga para baixo, à exceção da figura 19 onde só temos 5 imagens nesta posição devido ao facto que para *Low Frequency* com ganho de 30 dB o ruido é tal que se torna impossível de diferenciar a linha água do náufrago.

Nesta posição ao nível visual consegue-se diferenciar duas zonas a do tronco e a das pernas. Nota-se que os resultados em *High Frequency* são melhores do que os de *Low*

Frequency ao nível de ruido, e quando variamos o ganho este amplifica o eco de náufrago mas também o do ruido envolvente.

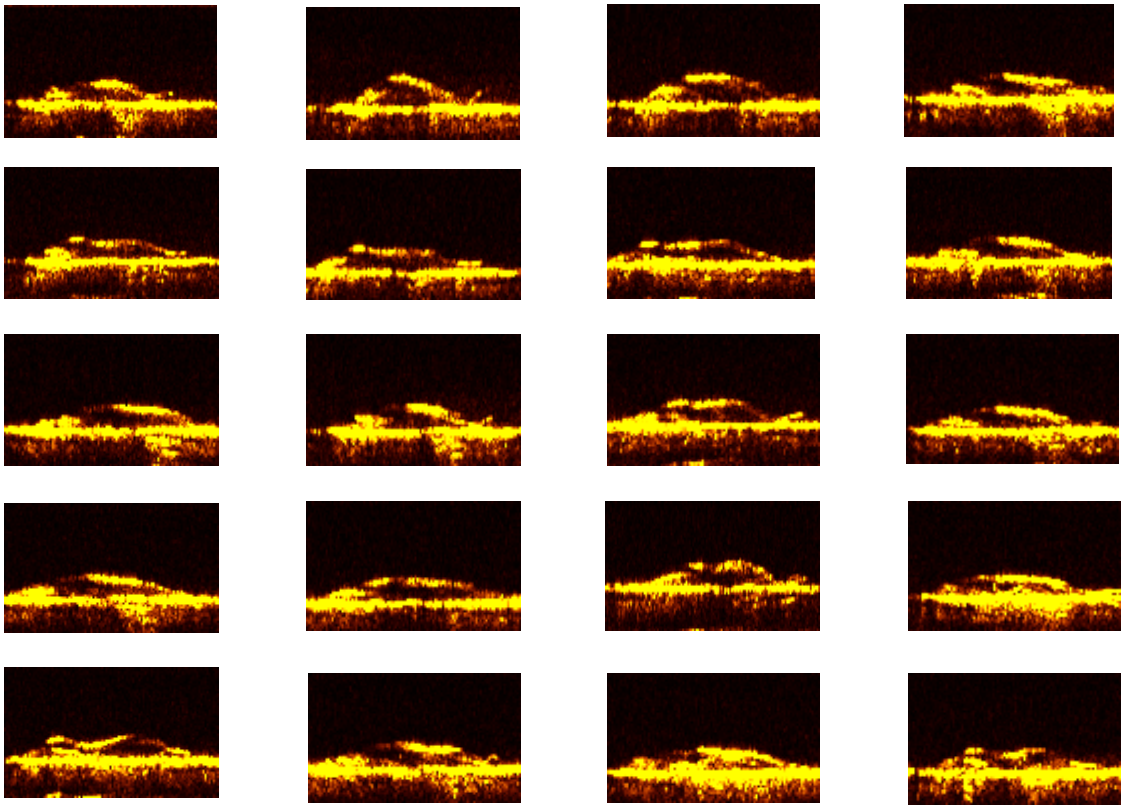


*Figura 16: Náufrago de Barriga para Baixo (*High Frequency */ 20dB)*

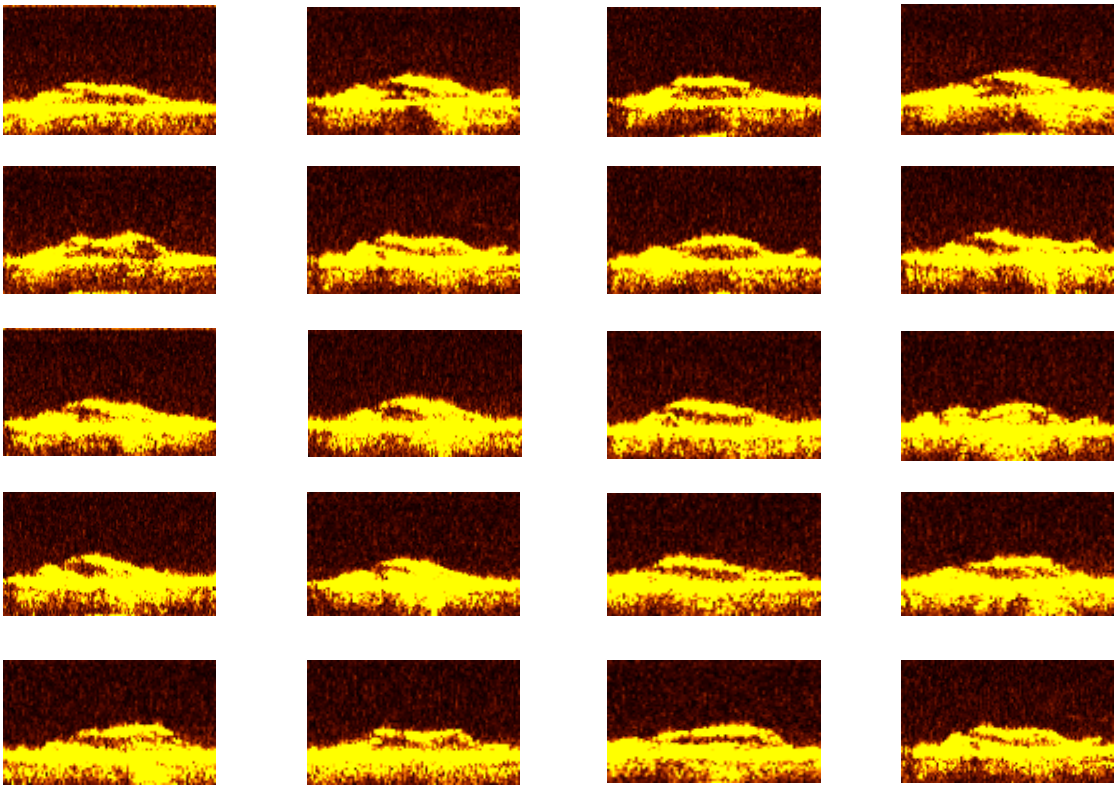


*Figura 18: Náufrago de Barriga para Baixo (*High Frequency */ 30dB)*

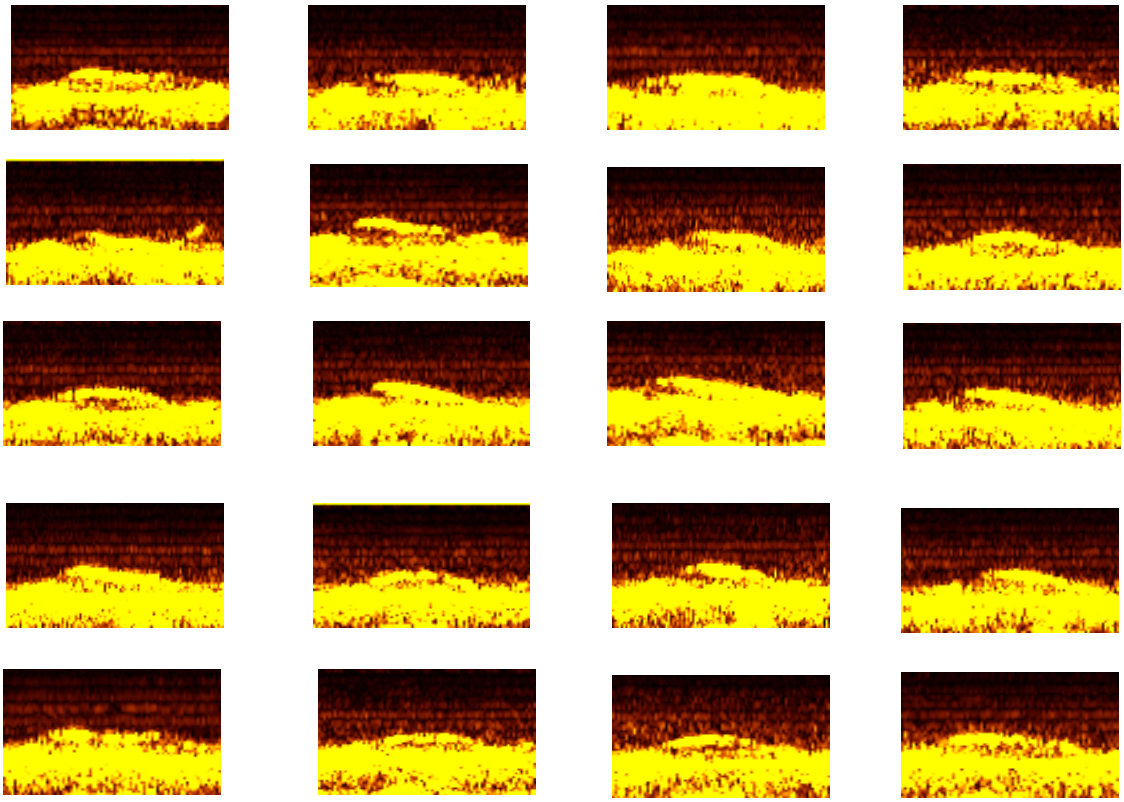


*Figura 17: Náufrago de Barriga para Baixo (*Low Frequency */ 20dB)*

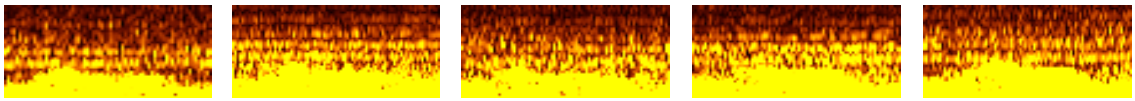


*Figura 19: Náufrago de Barriga para Baixo (*Low Frequency */ 30dB)*

4.2.2 Náufrago de Costas

Cada uma das Figuras 20, 21 e 22 são constituídas por 20 imagens de um corpo à deriva na posição em que o náufrago se encontra de costas para o fundo, à exceção da figura 23 onde só temos 5 imagens nesta posição devido ao facto que para *Low Frequency* com ganho de 30 dB o ruido é tal que se torna impossível de diferenciar a linha água do náufrago.

Nesta posição ao nível visual consegue-se diferenciar duas zonas a do tronco e a das pernas. Nota-se que os resultados em *High Frequency* são melhores do que os de *Low Frequency* ao nível de ruido, e quando variamos o ganho este amplifica o eco de náufrago mas também o do ruido envolvente.

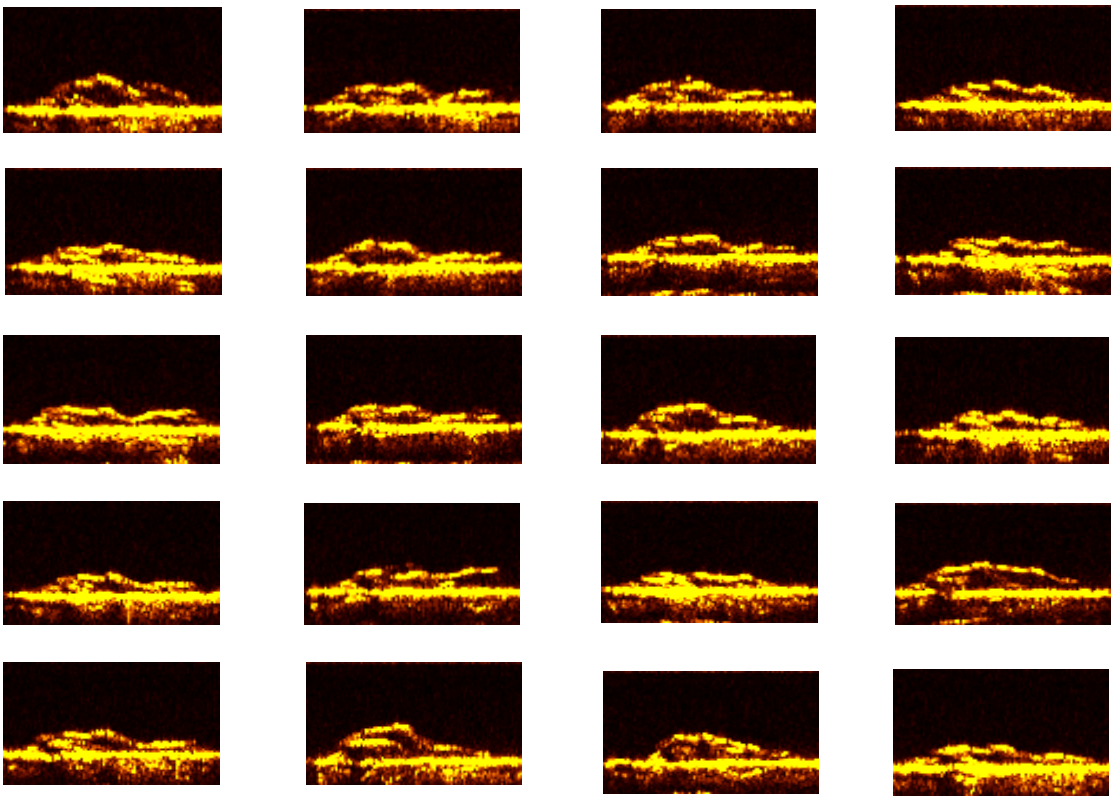


*Figura 20: Náufrago de Costas (*High Frequency */ 20db)*

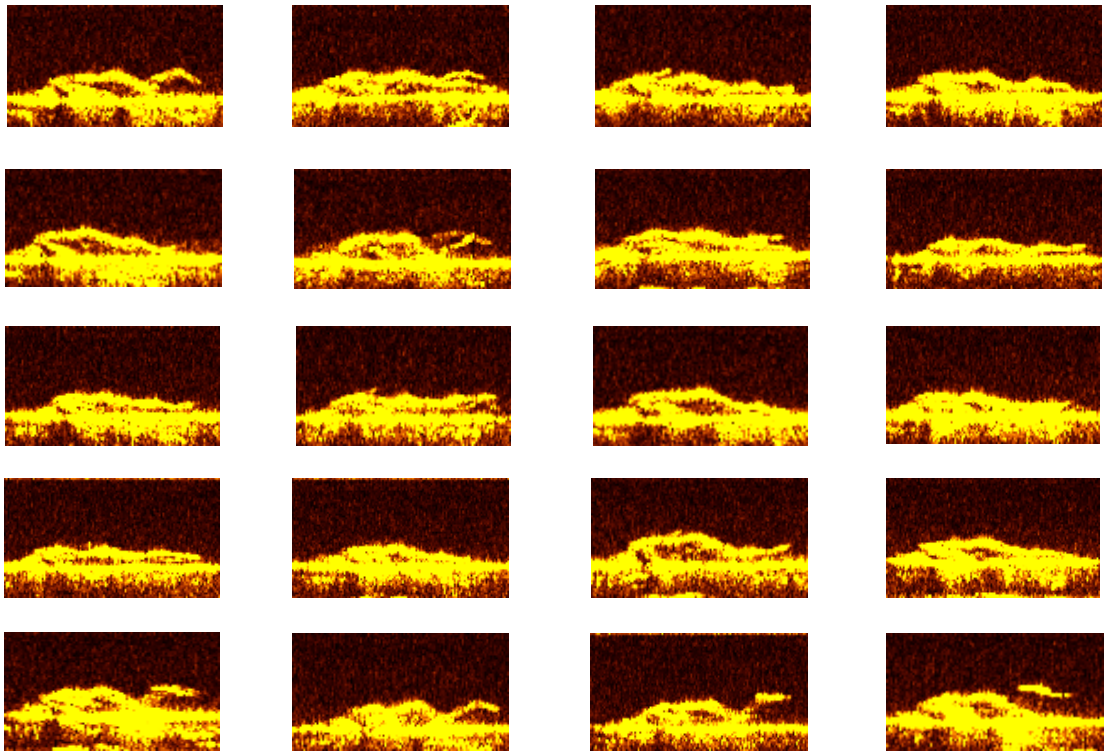


*Figura 22: Náufrago de Costas (*High Frequency */ 30dB)*

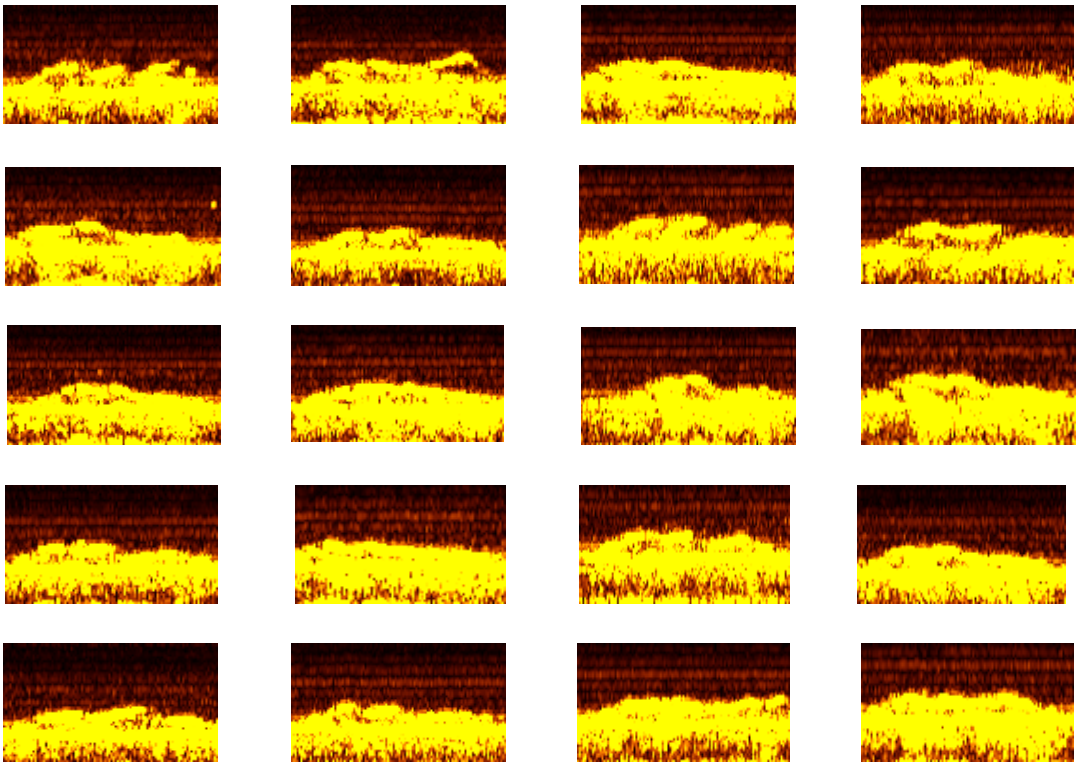


*Figura 21: Náufrago de Costas (*Low Frequency */ 20dB)*

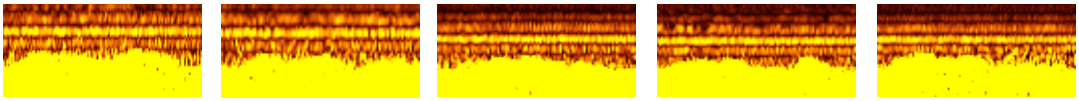


*Figura 23: Náufrago de Costas (*Low Frequency */ 30dB)*

4.2.3 Náufrago de Lado

Cada uma das Figuras 24, 25 e 26 são constituídas por 20 imagens de um corpo à deriva na posição em que o náufrago na lateral, à exceção da figura 27 onde só temos 5 imagens nesta posição devido ao facto que para *Low Frequency* com ganho de 30 dB o ruido é tal que se torna impossível de diferenciar a linha água do náufrago.

Nesta posição ao nível visual consegue-se perceber que o eco do corpo é mais estreito devido ao facto da exposição do corpo ser menor do que nas restantes posições. Nota-se que os resultados em *High Frequency* são melhores do que os de *Low Frequency* ao nível de ruido, e quando variamos o ganho este amplifica o eco de náufrago mas também o do ruido envolvente.

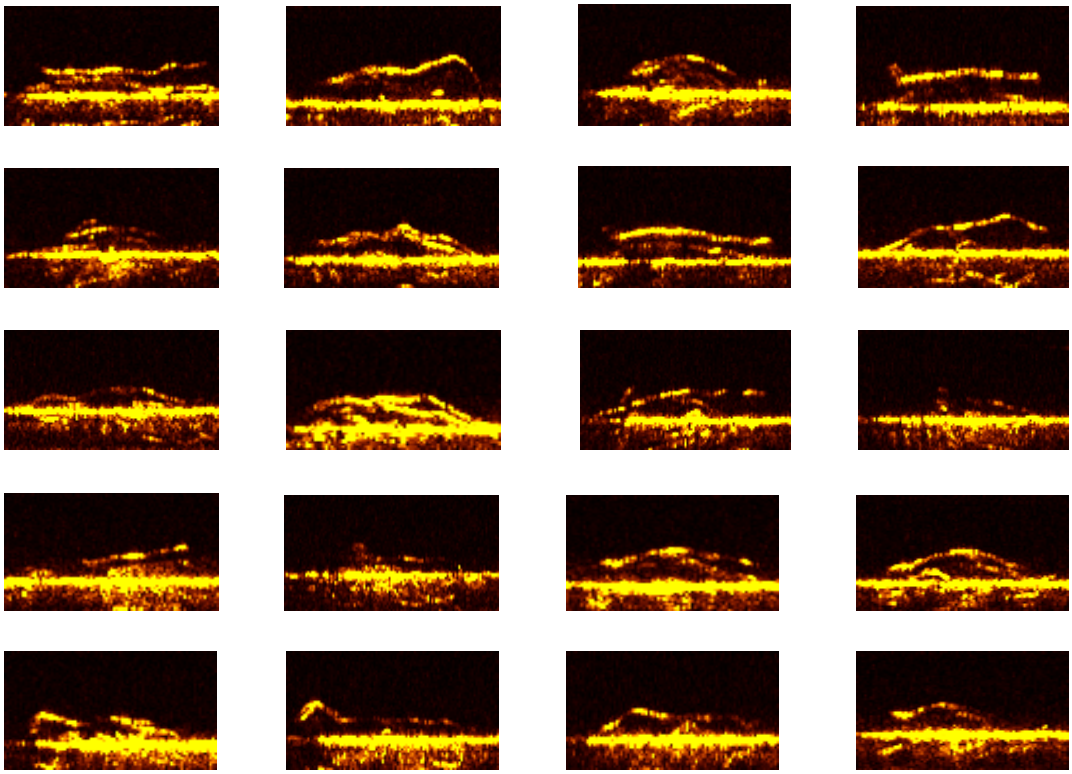


*Figura 24: Náufrago de Lado (*High Frequency */ 20dB)*

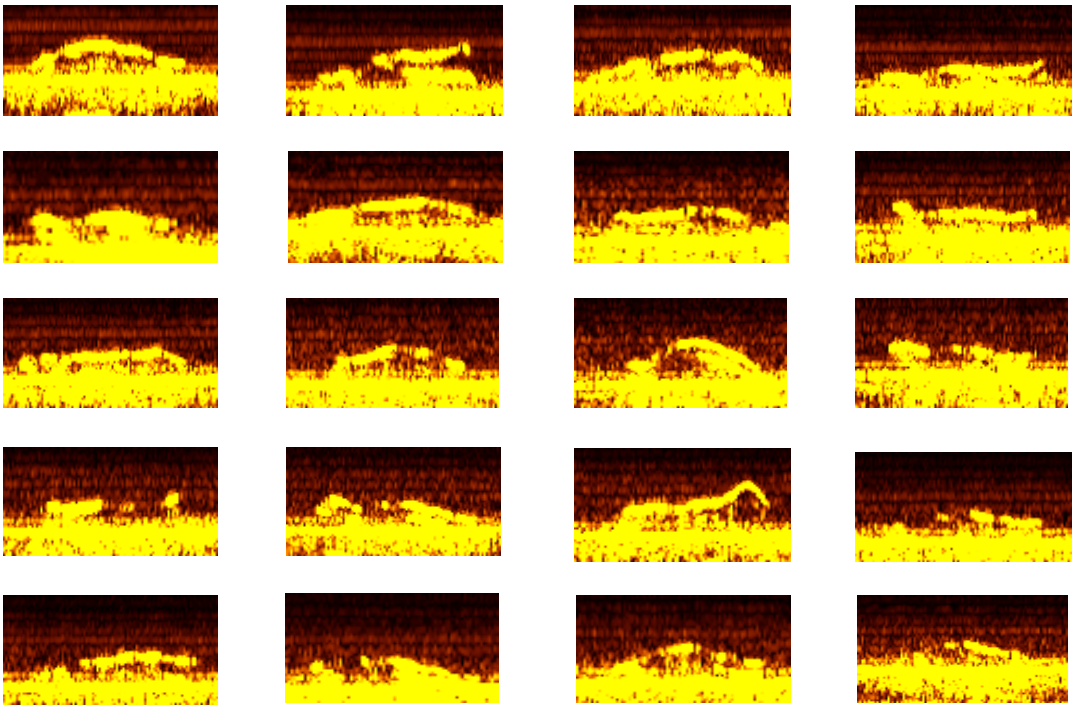


Figura 26: Náufrago de Lado (High Frequency */ 30dB)*

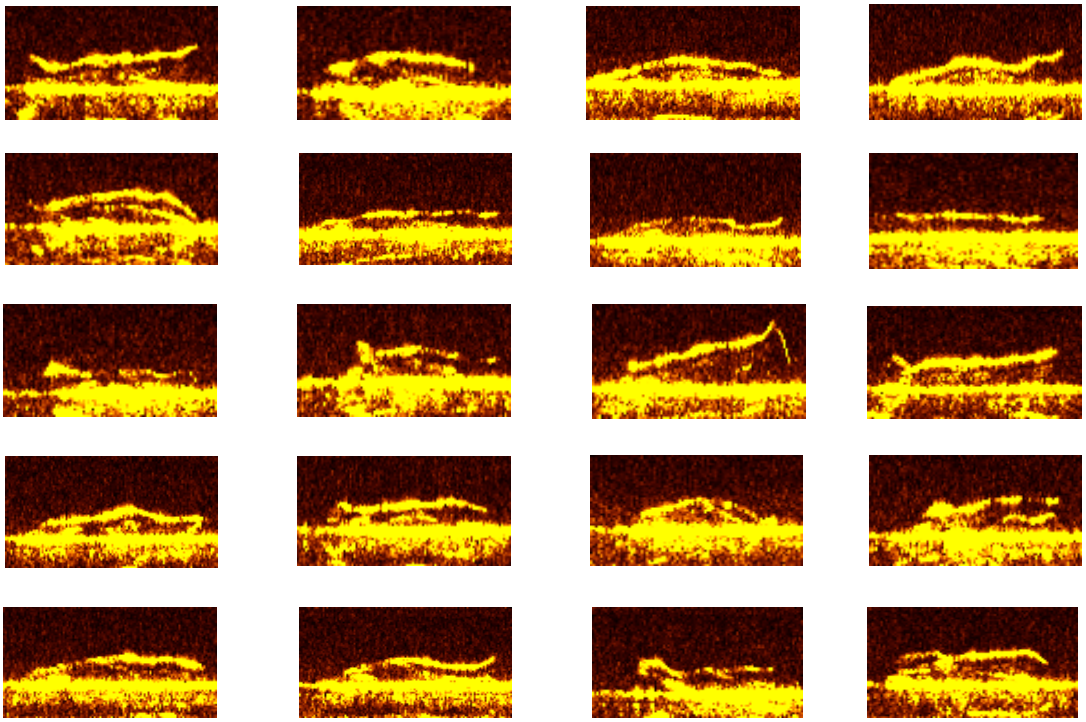


Figura 25: Náufrago de Lado (Low Frequency / 20dB)

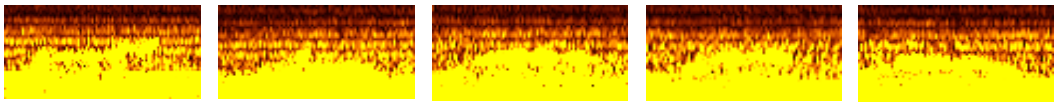


*Figura 27: Náufrago de Lado (*Low Frequency */ 30dB)*

4.2.4 Objetos

O processo realizado para os objetos é semelhante aos das posições possíveis de o náufrago ser encontrado à deriva, varia é na quantidade da amostragem que foram recolhidas apenas 15 imagens por cada objeto. Descartou-se a *Low Frequency* com ganho 30dB devido ao facto de se ter notado na recolha das imagens das diferentes posições, que com *Low Frequency* ganho 20 o ruido era demasiado.

O intuito de conhecer os ecos sonar de diferentes objetos é de ter o conhecimento do contraste do eco sonar destes em relação ao do corpo humano, para objetos feitos de diferentes materiais tais com plástico, madeira e latão. É notável que o eco das latas de latão produz um eco mais forte devido ao tipo de material, mas sua forma não se assemelha ao do eco de um corpo.

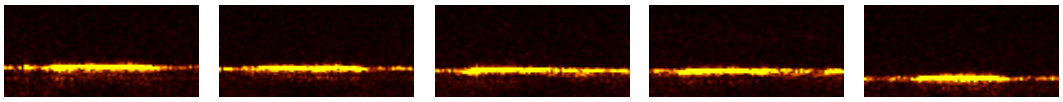


Figura 28: Tábua (High Frequency / 20dB)

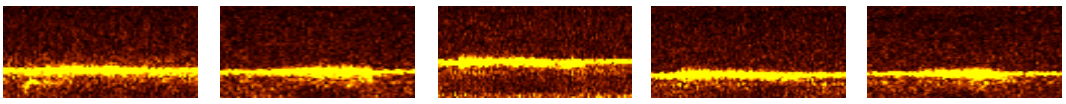


Figura 29: Tábua (High Frequency / 20dB)

Figura 30: Tábua (Low Frequency / 20dB)

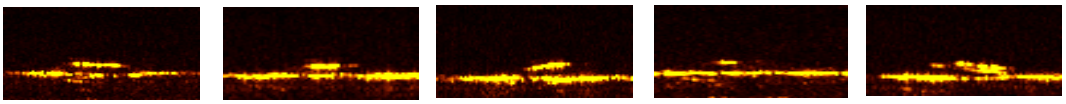


Figura 31: Guarda-Chuva (High Frequency / 20dB)

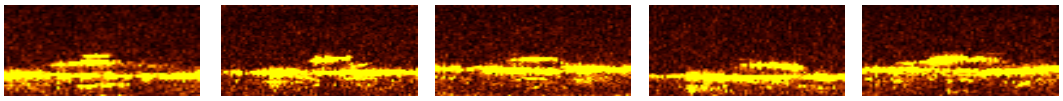


Figura 32: Guarda-Chuva (High Frequency / 30dB)

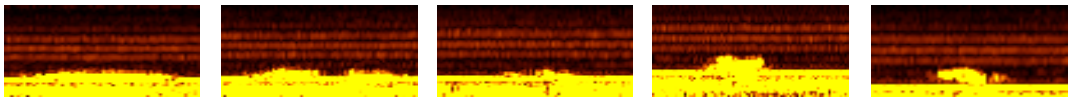


Figura 33: Guarda-Chuva (Low Frequency / 20dB)

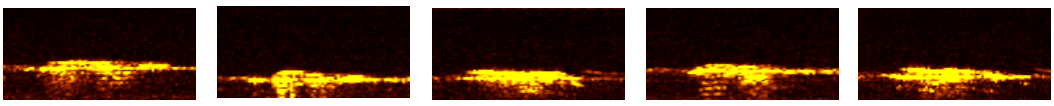


Figura 34: Baldes de lata (High Frequency / 20dB)

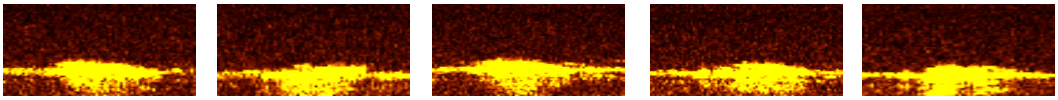


Figura 35: Baldes de lata (High Frequency / 30dB)

Figura 36: Baldes de lata (Low Frequency / 20dB)

4.3 Métricas de Desempenho

Para avaliar o desempenho do algoritmo *SURF* foram utilizados os métodos precisão e *recall*, por serem os métodos mais referenciados e utilizados para avaliar o desempenho de algoritmos de sistemas de reconhecimento de informação (Palma, 2004)

O método precisão (Murphy, Torralba, Eaton, & Freeman, 2005) procura a relação entre o número de imagens corretamente classificadas (*True Positives*, **TP**) e o número total de imagens detetadas, seja corretamente (TP) ou incorretamente (*False Positives* **FP**).

$$Precisão = \frac{TP}{TP + FP}$$

Equação 2: Precisão

Onde,

$$FP = FISI + FICI$$

Equação 3: Falso Positivo

O método *recall* (Murphy, Torralba, Eaton, & Freeman, 2005) procura a relação entre o número de imagens que foram corretamente classificadas e o número de imagens que fazem parte dos conjuntos de teste (**nP**).

$$Recall = \frac{TP}{nP}$$

Equação 4: Recall

A variável **FP** (*False Positives*), será a soma de todas as imagens que pertencem ao subconjunto de imagens sem interesse (**FISI** – Falsas imagens sem interesse), mas foram classificadas como pertencentes ao subconjunto de imagens com interesse com as imagens classificadas como pertencendo ao subconjunto de imagens com objetos de interesse (**FICI** – Falsas imagens com interesse), mas que de fato, pertencem ao subconjunto de imagens sem interesse.

A variável **TP** (True Positives), corresponde a todas as imagens que pertencem ao subconjunto de imagens sem interesse e foram corretamente classificadas.

A variável **nP** será relativa ao número de exemplos existentes no conjunto de teste em estudo e que correspondem a 100 imagens sonar.

Para determinação de *False positives* (FP) e *True positives* (TP) foi definido um conjunto de imagens 100 imagens de teste (figura 37), dividido em dois subconjuntos:

- Imagens (786x506) sem objetos de interesse – 20 imagens;
- Imagens (786x506) com objetos de interesse – 80 imagens;

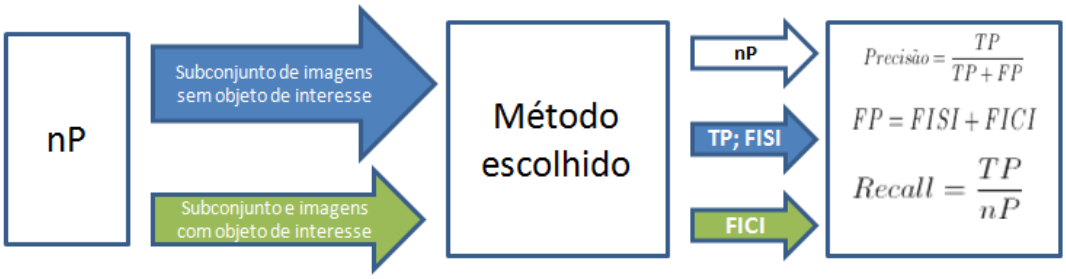


Figura 37: Processo de determinação das métricas de avaliação de desempenho

4.4 Condições e Metodologia de Avaliação de Desempenho

Com o objetivo de avaliar o desempenho do algoritmo *Deteção através de ficheiro sonar* foi desenvolvido uma metodologia onde o método de segmentação escolhido e o de deteção foram testados.

A ideia subjacente à metodologia implementada, é utilizar o conjunto de imagens de teste (imagens que contêm objetos de interesse e imagens que não contêm objetos de interesse) e através dos métodos de avaliação determinar quais os parâmetros que apresentam as melhores *performances* nas métricas escolhidas, precisão e *recall*.

Como foi descrito no capítulo anterior, o processo de *Deteção através de ficheiro sonar* passa pela definição de dois *threshold's*: um associado à segmentação e outro associado à classificação.

O *threshold* relativo à segmentação da imagem está subdividido em dois, ou seja é necessário estabelecer um limiar de intensidade por forma a considerar a partir de que valores se considera que estes fazem parte da linha água e quantos são necessários para fazer a segmentação da mesma.

O *threshold* relativo à classificação necessita de um limiar que a partir de um certo número de pares de semelhança classifique como sendo um náufrago. Dada a sua dimensão, forma e características, esta variável é a principal foco de ajuste pois é esta que vai definir a resposta positiva ou negativa de haver um corpo à deriva.

4.4.1 Avaliação do desempenho para a localização de náufragos

A tabela 1 demonstra se a segmentação das imagens que vão ser analisadas, é realizada de forma correta. Analisando as imagens na zona da linha de água, é notável

que os valores dos pixéis que pertencem à linha de água variam entre os 130 e os 230 em relação ao seu nível de intensidade.

Foi então testado com a variação do Limiar de Intensidade, a quantidade de pixéis que se encontram por linha em relação ao LI. Os resultados obtidos mostram que para valores de limiar entre 130 e 150 a segmentação é feita de forma correta, pois temos em média mais 59% de *pixels* maiores que o LI na zona da linha de água, e onde pode-se verificar que a segmentação é feita de forma correta.

Limiar de Intensidade	***Pixels* na linha maiores ou iguais que o LI (%)**	**Segmentação da imagem**
130	78	Correto
140	71	Correto
150	59	Correto
160	49	Incorreto
170	41	Incorreto
180	32	Incorreto
190	24	Incorreto
200	15	Incorreto
220	13	Incorreto
230	11	Incorreto

Tabela 1: Resultados obtidos em função à segmentação da imagem

A tabela 2 demonstra o comportamento da classificação do conjunto de teste de imagens, com a variação do *threshold* de pares de semelhança:

Pares de Semelhança	**Precisão**	**Recall**
4	50/(50+17) = 0.75	50/80 = 0.63
6	56/(56+14) = 0.8	56/80 = 0.7
8	61/(61+10) = 0.85	61/80 = 0.76
10	69/(69+7) = 0.91	69/80 = 0.86
12	71/(71+6) = 0.92	71/80 = 0.89

14	73/(73+2) = 0.97	73/80 = 0.91
16	76/(76+0) = 1	76/80 =0.95
18	77/(77+0) = 1	77/80 = 0.96
20	79/(79+0) = 1	79/80 = 0.99

Tabela 2: Resultados obtidos em função do threshold

A avaliação do desempenho do algoritmo SURF em relação ao número de pares de semelhança para a classificação do náufrago é descrita, na tabela 2. A percentagem ótima de *recall* é obtida quando se utiliza o valor de *threshold* vinte pares de semelhança para a classificação do náufrago, e temos uma percentagem de 99%. Quanto à precisão conclui-se que o valor ótimo de threshold é obtido quando o algoritmo SURF utiliza dezasseis pares de semelhança, tendo associada uma percentagem de 100%.

Em relação ao protótipo desenvolvido, pode-se concluir que os valores de *threshold* recomendados são 140 para o *threshold* LI e em relação aos pares de semelhança o valor 20 de *threshold*. É de salientar que estes valores de threshold foram obtidos nas condições ideais, pois o ruido em ambiente controlado é muito menor em relação ao ambiente real (mar aberto), onde o ruido se faz sentir muito à superfície devido à ondulação e ao vento sentido no local.

Capitulo 5 – Sumário e Trabalho Futuro

5.1 Sumário

Para sumarizar, o caminho seguido, que teve como objetivo, conseguir detetar náufragos através de imagens sonar com um elevado grau de certeza. Para tal farei uma breve descrição do caminho seguido através de uma breve síntese de cada capítulo.

No capítulo 1, foi procurada uma introdução que colocasse o leitor dentro do problema em questão. Para tal, contextualizou-se o tema e foi definido o objetivo perseguido.

No capítulo 2, foi desenvolvido o estado da arte do processamento de imagem digital, procurando sempre fazer a ligação a imagens sonar. Ao longo do capítulo 2, foi definido o caminho a seguir, desde a fase de pré-processamento da imagem, passando pelos diversos métodos de segmentação da imagem e reconhecimento de padrões em imagens, por forma a conseguir criar um algoritmo robusto e eficaz.

No capítulo 3, procurou-se desenvolver o processo que irá permitir localizar náufragos através de imagens sonar. Dividiu-se o processo em três fases: simplificação, extração de características e classificação.

A fase da simplificação da imagem tem como objetivo a melhoria no desempenho final do processo de identificação de náufragos. Esta visa a redução de alguns efeitos indesejáveis, tais como o ruído. Para esse efeito cada imagem é pela função *linha de água. Nesta é feita a* segmentação da imagem para se ficar apenas com a zona de possível aparecimento do náufrago e onde é removido parcialmente o ruído.

A fase da extração de características e de classificação de imagem encontram-se no mesmo módulo (*Funçao Detect*), onde primeiramente se extrai as características da imagem a ser analisada e depois se procede à comparação das características da mesma com as características das imagens presentes na base de dados.

É importante salientar que as imagens sonar são contaminadas com ruido, que atrapalha o processamento e a interpretação das mesmas. Durante o desenvolvimento da dissertação de mestrado houve preocupação em melhorar a qualidade das imagens, para tal utilizou-se o filtro *Sobel* antes de passarem para a fase de extração de características, sendo que a perda da informação não é significante.

O ambiente implementado para a criação dos métodos e para a realização dos testes foi desenvolvido no Matlab, ferramenta computacional muito empregue no trabalho com imagens.

Conforme apresentado no capítulo 3, o algoritmo SURF foi apresentado por Herbert Bay na Conferência Internacional sobre Visão Computacional realizada na Áustria, em maio de 2006, mostrando que pode ser usado em tarefas como o reconhecimento de objetos, sendo este muito mais eficaz e mais rápido do que o SIFT, tendo a característica de ser invariante a rotação.

Por fim, no capítulo 4, temos a base de dados que foi criada, e os resultados obtidos. Dos resultados obtidos é demostrado o porque da escolha dos limiares de decisão que o algoritmo usa e são retiradas conclusões da eficácia do algoritmo empregue para a deteção de náufragos.

Através do capítulo 4 pode-se concluir que o *threshold* LI que indica o limiar de intensidade, a partir do qual o *pixel* é constituinte da linha da água pode variar entre os 130 e os 150 por forma a realizar a segmentação da imagem de forma correta. Em relação ao segundo *threshold* dos pares de semelhança que são necessários para que se consiga obter classificação foi escolhido o valor de *threshold* 20, pois foi este que apresentou uma percentagem ótima de *recall* de 99% porcento para a classificação positiva de náufragos à deriva. É de salientar que estes valores de threshold foram obtidos nas condições ideais, pois o ruido em ambiente controlado é muito menor do que em ambiente real (mar aberto), onde o ruido se faz sentir muito à superfície devido à ondulação e ao vento sentido no local.

Neste capítulo também foi possível notar que para cumprir com o objetivo final desta dissertação, que era fazer deteção em tempo real de náufragos, será necessário aumentar a velocidade de processamento dos dados de uma forma mais expedita. Na secção seguinte deixo uma solução possível que fará com que o tempo de análise diminua drasticamente.

5.2 Trabalho Futuro

É importante definir algumas linhas de trabalho que poderão ser abordadas por outros investigadores, para que a pesquisa tenha sequencia. Neste contexto foram

identificadas algumas possibilidades de continuação para este trabalho que devem contar com futuros investigadores. Um dos possíveis incrementos seria colocar a posição vertical do náufrago à superfície na base de dados, recolhendo imagens sonar nas duas frequências (*high and Low Frequency*) e com os dois ganhos (20db e 30 dB) por forma a enriquecer–la. Com isto é expectável que se consiga aumentar taxa de eficiência na classificação. A partir do método deteção através de ficheiro sonar, adaptar por forma a desenvolver um método em que se consiga a deteção em tempo real, ou seja a cada imagem proveniente do feixe acústico esta seja instantaneamente analisada e obtida uma resposta. Como foi referido na secção anterior esta deteção a velocidade de processamento não é a desejável (2,3 segundos). Uma das soluções possíveis é abordar o problema com uma arquitetura CUDA (*Compute Unified Dispositivo Architecture*), em que esta é uma plataforma utilizada para programação em paralelo, sendo o primeiro modelo de programação que permitiu alto nível de programação para GPUs (unidades de processamento gráfico). A função dos GPUs é o processamento gráfico, assim sendo o desenvolvimento feito pelos programadores CUDA, consiste em acelerar os processamentos dos algoritmos, através da quantidade de paralelismos dos GPUs. Outro incremento de extrema relevância seria a utilização de outro tipo de sonar, por exemplo o de multifeixe por forma a comparar a qualidade de imagens e os *thresholds* usados para a segmentação da imagem e classificação. Por último a programação noutra linguagem para que se consiga a implementação do *software* num UUV, em que o veículo encontrar-se-á submerso sempre a analisar a superfície e só se deslocará à superfície caso tenha encontrado um possível náufrago.

Bibliografia

ALLAN, L. D. (1986). *Manual of Fetal Echocardiography*. Norwell, MA.USA: MTP Press.

AWCOCK, G. J., & THOMAS, R. (1996). *Applied Image Processing.* New York: McGraw-Hill.

BAY, H. (2008). *Speeded-Up Robust Features (SURF).* Computer Vision and Image Understanding.

BEDNAR, J., & WATT, T. (1984). *Alpha-trimmed and their relationship to the median filters.* New York: IEEE Transactions on Acoustic.

BISHOP, C. M. (1995). *Neural Networks for Pattern Recognition.* New York: Oxford University Press.

BUSSE, L., CRIMMINS, T. R., & FIENUP, J. R. (1995). *Ultrasonics Symposium* (v.2 ed.). IEEE Computer Society.

CAETANO, T. S. (2000). *Estudo sobre Técnicas Estatísticas Paramétricos para Reconhecimento de Padrões.* Porto Alegre: PPGC UFRGS.

CRIMMINS, T. R. (1985). *Geometric filter for specke reduction.*

FACON, J. (1993). *Processamento e Análise de Imagens.* Cordoba: Escuela Brasileno - Argentina de Informática.

FISHER, R., PERKINS, S., WALKER, A., & WOLFART, E. (2000). *Crimmins Speckle Removal*. (Edinburgh: Univerty of Edinburgh, Division of Informatics Artificial Intelligence) Obtido em Acesso em: Jan. 2015, de <http://www.dai.ed.ac.uk/HIPR2/crimmins.htm#1>

GONZALEZ, R. C., & WOODS, R. E. (1993). *Digital Image Processing.* Massachusetts: Addison-Wesley.

GONZALEZ, R. C., WOODS, R., & EDDINS, S. (2004). *Digital Image Processing: Using Matlab* (1.ed. ed.). Upper Saddle River: Pearson Prentice Hall.

GRAHAN, D., & BARRETT, A. (1997). *Knowledge-Based Image Processing.* London: Springer Verlag.

JAHNE, B. (1997). *Digital Image Processing.* Berlin: Springer-Verlag.

JAIN, A. K. (1989). *Fundamentals of Digital Image Processing.* Englewood Cliffs: Prentice Hall.

LINDEBERG, T. (1998). *Feature Detection with Automatic Scale Selection.* International Journal on Computer Vision.

LOWE, D. G. (1999). *Object Recognition from Local Scale-Invariant Features.* International Conference on Computer Vision.

MALLADI, R., & SETHIAN, J. (1996). *Graphical Models and Image Processing* ([s.1.], v.58, n.2 ed.). IEEE Transaction on Medical Imaging.

MURINO, V. (1998). *Structured Neural Networks for Pattern Recognition* (v.28 ed.). New York: IEEE Transactions on Systems, Man and Cybernetics.

Murphy, K., Torralba, A., Eaton, D., & Freeman, W. (2005). *Object Detection and Localization Using Local and Global Features.* University of British Columbia.

Palma, D. (2004). *Extracção automática de texto em sequências de video.* Lisboa: Instituto Superior Técnico.

PEDRINI, H., & SCHWARTZ, W. R. (2007). *Análise de Imagens Digitais - Prinicipios, Algoritmos e Aplicações* (1.ed. ed.). São Paulo: Thompson Learning.

SANTANA, J. (1999). *Pré-processamento de Imagens Ecocardiográficas.* Porto Alegre: PPGC da UFRGS.

SONKA, M., HLAVAC, V., & BOYLE, R. (1998). *Image Processing, Analysis and Machine Vision.* Pacific Grove: PWS Publishing.

Apêndices

Apêndice 1 - Código que vai fazer a deteção do náufrago

```
function testar(data)

D1 = 'C:\Users\Hugo Fonseca\Desktop\base de dados\HF20\';%pasta que
contem img´s da base de dados
imgcount = contadorIMG(D1);%variavel com nº de img´s na BD
i = lineofwater(data);%funçao que descobre a linha de àgua
ToAnalyze = data(1:i,:);%imagem ja segmentada com a linha de àgua

for i = 1 : imgcount
source = imread(strcat(D1,int2str(i),'.png'));%imagem da base de dados
detect(source,ToAnalyze,i); % funçao que detecta naúfrago
end
end
```

Apêndice 2 - Código que vai contar quantas imagens estão presentes na Base de Dados

```
function imgcount = contadorIMG(Dcaminho)
D = dir(Dcaminho);%caminho da pasta
 imgcount = 0;%inicia contador de ficheiros

for i=1 : size(D,1)
    if
not(strcmp(D(i).name,'.')|strcmp(D(i).name,'..')|strcmp(D(i).name,'Thu
mbs.db'))%verifica se tem ficheiro
        imgcount = imgcount + 1; % Numero das imagens que estao na
pasta da base de dados
    end
end
end
```

Apêndice 3 - Código da Função Linha de água

```
function agua = lineofwater(data)
LIMIAR=20;%intesidade minima do pixel
 [height,length] = size(data);%[506x700]
for i=15:length
    line = data(i,:);%Analisa linha à linha a imagem
    indice = line >LIMIAR;%substitui os valores maiores que o LIMIAR
por "1" e os outros abaixo por "0"
    y = line(indice);%compara a matriz inicial com o indice e forma
uma matriz so com esses valores
    z = size(y);%da as dimensoes da matriz
        if z(1,2) > 30%se tiver mais que X valores
            agua = i+20; %linha de àgua
            return;
         end
    end
agua =  500;%linha de àgua
end
```

Apêndice 4 - Código da Função Video

```
function video(VideoIN)
%limpa a pasta onde vao ser guardadas as imgs
delete('C:\Users\Hugo Fonseca\Desktop\imagens a testar\*.png');
%carrega o video
vidobj=VideoReader(VideoIN);
%nº de frames que estao no video
frames=vidobj.Numberofframes;
k=1; % variavel que permite saltar de 4 em 4 frames
o=1; % gera o nome das imagens em nº (1.png,2.png,3png,...,n.png)

for f=1:frames

    if k<=frames

        thisframe=read(vidobj,k);% le a frame k do video
        % funçao que nos da a posiçao da linha de agua
        i = lineofwater(thisframe);
        % descarta tudo o que tiver abaixo da linha de agua
        i2=thisframe(1:i,1:786,:);
        % caminho das imagens que vao ser analisadas
        pathname = 'C:\Users\Hugo Fonseca\Desktop\imagens a testar\';
        % grava as imagens ja recortadas
        imwrite( i2,[pathname, num2str(o),'.png']);

        k=k+4;% variavel que permite saltar de 4 em 4 frames
        o=o+1;% gera o nome das imagens em nº
(1.png,2.png,3png,...,n.png)

    else
        break
    end
end
end
```

Apêndice 5 - Código da Função Detect

```
function detect(source,ToAnalyze,i)

I= rgb2gray (source); % converte para cizento a imagem
K= ToAnalyze;

sourcePoints = detectSURFFeatures(I); %analisa as caracteristicas da
imagem
ToAnalyzePoints = detectSURFFeatures(K);
 %extração das caracteristicas
[sourceFeatures, sourcePoints] = extractFeatures(I, sourcePoints);
[ToAnalyzeFeatures, ToAnalyzePoints] = extractFeatures(K,
ToAnalyzePoints);
%compara as caracteristicas das duas imagens (da-nos as coordenadas)
boxPairs = matchFeatures(sourceFeatures,
ToAnalyzeFeatures,'MaxRatio',1);

numPairs = length(boxPairs); % Numero de pares entre as duas imagens

if numPairs >= 5

    matchedsourcePoints = sourcePoints(boxPairs(:, 1), :);
    matchedToAnalyzePoints = ToAnalyzePoints(boxPairs(:, 2), :);
% figure;
% showMatchedFeatures(source, ToAnalyze,
matchedsourcePoints,matchedToAnalyzePoints, 'montage');
% title('Pontos Correspondentes (Incluindo os Outliers)');


 tform = estimateGeometricTransform(matchedsourcePoints,
matchedToAnalyzePoints, 'projective');

    % buscar as dimensoes da imagem da base de dados
boxPolygon = [1, 1;...                           % canto sup esquerdo
(1,1)
          size(source, 2), 1;...                 % canto sup direito
          size(source, 2), size(source, 1);... % canto inf direito
          1, size(source, 1);...                 % canto inf esquerdo
            1, 1];         % canto sup esquerdo para fechar o poligno

    % adapta a img da base de dados com a posiçao da imagem a testar
    newBoxPolygon = transformPointsForward(tform, boxPolygon);

                figure;
                imshow(ToAnalyze);%colaca a imagem que esta a ser
analisada
                hold on;

                line(newBoxPolygon(:, 1), newBoxPolygon(:, 2),
'Color', 'G','LineWidth', 3); %desenha a box
                title('Caixa ao Redor do Naúfrago');

                mensagem = strcat('Na imagem' , int2str(i), ' da BD
temos --->  ', int2str(numPairs), 'pontos de semelhança --->
POSITIVO');
```

```
                disp(mensagem);
        else

        mensagem = strcat('Na imagem' , int2str(i), ' da BD temos --->
', int2str(numPairs), 'pontos de semelhança ---> NEGATIVO');
        disp(mensagem);

end
```

Apêndice 6 - Código que permite visualizar dados do sonar em Matlab (FEUP)

```
function varargout = offlineViewer(varargin)
% OFFLINEVIEWER M-file for offlineViewer.fig
% OFFLINEVIEWER, by itself, creates a new OFFLINEVIEWER or raises the
% existing
% singleton*.
%
% H = OFFLINEVIEWER returns the handle to a new OFFLINEVIEWER or the
handle to
%      the existing singleton*.
%
%      OFFLINEVIEWER('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in OFFLINEVIEWER.M with the given input
arguments.
%
%      OFFLINEVIEWER('Property','Value',...) creates a new
OFFLINEVIEWER or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before offlineViewer_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to offlineViewer_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help offlineViewer

% Last Modified by GUIDE v2.5 27-May-2014 13:26:53

% Begin initialization code - DO NOT EDIT

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @offlineViewer_OpeningFcn, ...
                   'gui_OutputFcn',  @offlineViewer_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before offlineViewer is made visible.
function offlineViewer_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to offlineViewer (see VARARGIN)

% Choose default command line output for offlineViewer
handles.output = hObject;

handles.chunkSize = 640;
handles.numReads = 500;

handles.fName = '';
handles.fPath = '';
handles.fIndex = 0;
handles.running = 0;
handles.pause = 0;

set(handles.pushbuttonStartStop, 'Enable', 'off');
set(handles.textFileName, 'String', '(no file selected)');
set(handles.pushbuttonPause, 'Enable', 'off');
set(handles.sliderSpeed, 'Value', handles.numReads);
set(handles.sliderSpeed, 'SliderStep', [1/9, 1/9]);
set(handles.editSpeed, 'String', handles.numReads);

handles.t = timer();
set(handles.t, 'Name', 'OffViewerT');
set(handles.t, 'ExecutionMode', 'fixedDelay');
set(handles.t, 'Period', 0.2);
handles.t.TimerFcn = {@offViewerT_Callback, handles};

handles.fId = 0;

handles.A = zeros(handles.chunkSize, 1, 'uint8');

handles.ssWidth = 500;
handles.ssLength = 300;

handles.ssImage = zeros(handles.ssLength, handles.ssWidth, 'uint8');
handles.stbd = 113 + (0:2:handles.ssLength-1);
handles.port = 113 + (1:2:handles.ssLength-1);

set(handles.axes1, 'Position', [0.03 0.03 0.95 0.78]);
set(handles.axes1, 'DrawMode', 'fast');
set(handles.axes1, 'DataAspectRatio',
[handles.ssWidth, handles.ssLength, 1]);
```

```
set(handles.axes1, 'Parent', hObject);
handles.iHandle = image(handles.ssImage, 'Parent', handles.axes1);
%handles.iHandle = image('CDATA', handles.ssImage, 'Parent',
handles.axes1,
%'BusyAction', 'cancel', 'EraseMode', 'normal', 'HitTest', 'off');

colormap(handles.axes1, bone(128));%desenha as 5 linhas vermelhas
for b = 1:5
    linePoint = round(handles.ssLength * b/6);
    line([0, handles.ssWidth],[linePoint, linePoint], 'Color', 'r',
    'LineWidth', 1, 'LineStyle', ':');
end
axis off;
%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%

% Update handles structure
guidata(hObject, handles);


% UIWAIT makes offlineViewer wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = offlineViewer_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


function offViewerT_Callback(obj, event, handles)
data = guidata(handles.figure1);

if (data.fId == 0)
    return;
end

data.A, count] = fread(data.fId, [data.chunkSize, data.numReads],
'*uint8');

if feof(data.fId)
    frewind(data.fId);
    if count == 0
        return
    end
end


if data.A(613, :) ~= 252
```

```
   disp('Error in reading file');
   disp(data.A(613, :));
end

%imshow(data.A(data.port,:));
testar(data.A(110:600,:));
imshow(data.A(110:600,:));

B = [flipud(data.A(data.stbd, :)); data.A(data.port, :)];

realNumReads = data.numReads;
if size(data.A, 2) ~= realNumReads
    realNumReads = size(data.A, 2);
end
data.ssImage = [data.ssImage(:, realNumReads+1:end), B];

set(data.textDateData, 'String', char(data.A(9:20, end)'));
set(data.textTimeData, 'String', char(data.A(21:29, end)'));
set(data.textPosData, 'String', char(data.A(48:71, end)'));
set(data.textSpeedData, 'String', double(data.A(74, end)) * 0.0514 );
set(data.textHeadingData, 'String', hex2dec(strcat(dec2hex(data.A(75,
end),
2), dec2hex(data.A(76, end), 2)))/10 );

set(data.figure1, 'CurrentAxes', data.axes1);
set(data.iHandle, 'CData', data.ssImage);
drawnow expose;



guidata(handles.figure1, data);

% --- Executes on button press in pushbuttonStartStop.
function pushbuttonStartStop_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonStartStop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.fIndex == 1
    % if file selected
    if handles.running == 0
        % Start playback
        try
            handles.fId = fopen(strcat(handles.fPath, handles.fName));
            start(handles.t);
            handles.running = 1;
            set(handles.pushbuttonStartStop, 'String', 'Stop');
            set(handles.pushbuttonSelectFile, 'Enable', 'off');
            set(handles.pushbuttonPause, 'Enable', 'on');
            set(handles.pushbuttonPause, 'String', 'Pause');
        catch mException
            errordlg(mException.message, 'Error opening file.');
        end
    else
        % Stop playback
        try
            fclose(handles.fId);
            handles.fId = 0;
```

```
            stop(handles.t);
            handles.running = 0;
            set(handles.pushbuttonStartStop, 'String', 'Start');
            set(handles.pushbuttonSelectFile, 'Enable', 'on');
            set(handles.pushbuttonPause, 'Enable', 'off');
        catch mException
            errordlg(mException.message, 'Error closing file.');
        end
    end
end

guidata(handles.figure1, handles);


% --- Executes on button press in pushbuttonSelectFile.
function pushbuttonSelectFile_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonSelectFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[fName, fPath, fIndex] = uigetfile('*.81s', 'Choose Sidescan File');
if fIndex ~= 0
   handles.fName = fName;
   handles.fPath = fPath;
   handles.fIndex = 1;
   set(handles.textFileName, 'String', fName);
   set(handles.pushbuttonStartStop, 'Enable', 'on');
else
   handles.fName = '';
   handles.fPath = '';
   handles.fIndex = 0;
   set(handles.textFileName, 'String', '(no file selected)');
   set(handles.pushbuttonStartStop, 'Enable', 'off');
end

guidata(handles.figure1, handles);


% --- Executes on button press in pushbuttonPause.
function pushbuttonPause_Callback(hObject, eventdata, handles)
% hObject    handle to pushbuttonPause (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.pause == 0
    % pause state
    stop(handles.t);
    handles.pause = 1;
    set(handles.pushbuttonPause, 'String', 'Resume');
else
    % un-pause state
    start(handles.t);
    handles.pause = 0;
    set(handles.pushbuttonPause, 'String', 'Pause');
end
guidata(handles.figure1, handles);
```

```
% --- Executes on slider movement.
function sliderSpeed_Callback(hObject, eventdata, handles)
% hObject    handle to sliderSpeed (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max')
%to determine range of slider

sliderV = get(hObject,'Value');
set(handles.editSpeed, 'String', sliderV);
handles.numReads = sliderV;

guidata(handles.figure1, handles);


% --- Executes during object creation, after setting all properties.
function sliderSpeed_CreateFcn(hObject, eventdata, handles)
% hObject    handle to sliderSpeed (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function editSpeed_Callback(hObject, eventdata, handles)
% hObject    handle to editSpeed (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editSpeed as text
%        str2double(get(hObject,'String')) returns contents of
% editSpeed as a double


% --- Executes during object creation, after setting all properties.
function editSpeed_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editSpeed (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
    get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object deletion, before destroying properties.
function figure1_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
data = guidata(hObject);
try
    stop(data.t);
    delete(data.t);
    if data.fId ~= 0
        fclose(data.fId);
    end
catch mException
    errordlg(mException.message, 'Error deleting file.');
end
handles.t = 0;
handles.fId = 0;
guidata(handles.figure1, handles);


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```