Universidade de Aveiro
2011
Departamento de Electrónica, Telecomunicações e Informática

João Pedro Madaleno Pereira

Navegação baseada no terreno com dados de Sonar de Varrimento Lateral

Universidade de Aveiro
2011
Departamento de Electrónica, Telecomunicações e Informática

João Pedro Madaleno Pereira

Navegação baseada no terreno com dados de Sonar de Varrimento Lateral

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica de António Guilherme Rocha Campos, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e Francisco José Curado Mendes Teixeira, Professor Auxiliar Convidado do Departamento de Engenharia Mecânica da Universidade de Aveiro.

o júri

presidente

Profª. Dr.ª Ana Maria Perfeito Tomé
professora associada do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais

Prof. Dr. Miguel Angel Guevara Lopez
professor auxiliar do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto

Prof. Dr. António Guilherme Rocha Campos
professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Prof. Dr. Francisco José Curado Mendes Teixeira,
professor auxiliar convidado do Departamento de Engenharia Mecânica da Universidade de Aveiro

agradecimentos

Agradeço aos meus orientadores, Prof. Guilherme Campos e Prof. Francisco Curado, por toda a ajuda prestada ao longo de todo o trabalho e pela constante disponibilidade para esclarecer toda e qualquer dúvida.

Agradeço também a preciosa ajuda prestada pelo João Quintas, sem ela os testes com dados SSS não estariam presentes neste trabalho.

Agradeço aos meus amigos de Aveiro pela amizade e pelo forte sentido entreajuda criado nas várias sessões de estudo, ao longo destes 5 anos. Agradeço ainda aos meus amigos do Teixoso por serem um exemplo para mim de união e companheirismo.

Dedico este trabalho e um agradecimento muito especial aos meus pais, que sempre me incentivaram a não desistir nos momentos mais difíceis. Nada teria conseguido sem eles.

A todos, um grande obrigado.

palavras-chave

sonar de varrimento lateral, navegação robótica, robótica subaquática, entropia de Shannon, informação mútua, correlação cruzada, processamento de imagem.

resumo

O presente trabalho propõe uma técnica de *matching* de imagens de sonar de varrimento lateral, baseada nos conceitos de entropia de Shannon e informação mútua, com aplicação à navegação de veículos robóticos subaquáticos. Para tal, recorre-se não só à informação fornecida pela escala de cinzentos, como também à informação fornecida por um conjunto de *features* texturais (*Haralick Features*), extraídas dos dados de sonar.

Uma parte significativa do trabalho incide sobre o estudo e a implementação dos algoritmos de estimação de entropia, para cálculo da informação mútua. É feita também uma contextualização do problema proposto, onde, para além da apresentação dos conceitos teóricos envolvidos, são definidos os objectivos gerais, assim como uma revisão às tecnologias de navegação subaquática existentes.

Os métodos propostos são validados experimentalmente com dados obtidos de distribuições probabilísticas conhecidas e que permitem a validação dos mesmos analiticamente. São realizados testes adicionais com imagens fotográficas e com dados de SSS para validação daqueles métodos. Para a aplicação de navegação em vista, o cálculo da entropia das imagens baseia-se nos dados originais de SSS, representados em níveis de cinzento, complementados com a informação extraída desses dados segundo os métodos propostos originalmente por Haralick.

A selecção das features texturais a usar é feita tendo em conta a natureza dos dados, as limitações associadas à aquisição dos mesmos em ambiente submarino e os objectivos da sua aplicação em termos de navegação. Para tal apresenta-se uma proposta de classificação e escolha das *Haralick Features* mais adequadas a imagens de sonar de varrimento lateral.

Os testes finais aplicam a metodologia proposta a dados reais de sonar de varrimento lateral para demonstrar o potencial da sua utilização no âmbito da navegação de veículos robóticos subaquáticos.

keywords

side-scan sonar, robotic navigation, underwater navigation, underwater robotics, Shannon entropy, mutual information, cross-correlation, image processing.

abstract

The the work presented here proposes a technique for matching of Sidescan Sonar images, based on Shannon's Entropy and Mutual Information concepts, applied to the naviagtion of underwater robotic vehicles. For such, it is used not only the information given by the grey-scale levels, but also by a set of textural features (Haralick features), which are extracted from the sonar data.
A significant parto f this work was spent studying and implementing algorithms for entropy estimation and mutual information calculation. The theoretical concepts involved are presented, the objectives are defined as well as a revision of underwater navigation technologies in existence. All this is done in order to contextualize the proposed problem.
The proposed methods are validated experimentally through data acquired from known probability distributions which allow for na analytical validation.Aditional tests with both photographic images and sidescan sonar data are performed in order to validate such methods. For the intended navigational application, the images' entropy estimation is based on the original sidescan sonar data. This data is represented by grey-scale levels complemented by the information extracted in accordance to the method proposed by Haralick.
The selection of the textural features is done by taking into account the data's nature, limitations associated with data acquisition in submarine environments and the objectives of its application in terms of navigation. It is therefore proposed a more adequate classification and selection of the Haralick features.
The final tests are done by applying the proposed methodology to real sidescan sonar data in order to demonstrate its potential for use in assisting robotic underwater vehicles' navigation.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

- AUV: Veículo Subaquático Autónomo (do inglês *Autonomous Underwater Vehicle*)
- CC: Correlação Cruzada (do inglês *Cross-Correlation*)
- CUDA: Compute Unified Device Architecture
- DSP: *Digital Signal Processor*
- DVL: Registo de velocidade de Doppler (do inglês *Doppler Velocity Log*)
- GPS: Global Positioning System
- GPU: Graphics Processing Unit
- IM: Informação Mútua
- INS: Sistemas de Navegação Inercial (do inglês *Inertial Navigation Systems*)
- kdpee: *k-d Partitioning Entropy Estimator*
- kNN: *k-Nearest Neighbors*
- LBL: *Long Baseline*
- NCC: Correlação Cruzada Normalizada (do inglês *Normalized Cross-Correlation*)
- pdf: *Probability Density Function*
- RGB: Modelo de cor Vermelho-Verde-Azul (do inglês *Red-Green-Blue*)
- ROV: Veículos Controlados Remotamente (do inglês *Remotely Operated Vehicles*)
- SGLD: *Spatial Grey-Level Dependence Matrix*
- SLAM: *Simultaneous localization and mapping*
- SSS: Sonar de Varrimento Lateral (do inglês *Side-scan Sonar*)
- USBL: *Ultra Short Baseline*

1. Introdução

A navegação é um domínio de incontestável importância na evolução da Humanidade. A necessidade de localização e obtenção da melhor rota para chegar a um dado destino conduziu, ao longo da história, ao desenvolvimento de técnicas de navegação cada vez mais eficientes.

Desde a navegação baseada na posição de corpos celestes (Estrela Polar, Sol,…) até aos mais modernos sistemas de GPS (*Global Positioning System*), várias foram as técnicas utilizadas, e mais ainda as idealizadas, que fazem da navegação uma área em constante renovação.

Neste sentido, acompanhando a evolução natural da tecnologia assim como o crescente interesse pela exploração dos oceanos, surgiu no século XX uma nova área de aplicação dos conceitos de navegação: os veículos subaquáticos autónomos, AUV (*Autonomous Underwater Vehicle)*.

1.1. Motivação e objectivos

A principal motivação para este trabalho é o interesse em efectuar de forma eficaz e precisa a navegação de um AUV sem necessidade de interromper missões para recalibrar a instrumentação de navegação (através, por exemplo, de emersão do veículo para adquirir sinal GPS).

Para tal, pretende-se tirar partido de equipamentos de Sonar de Varrimento Lateral (SSS *Side-Scan Sonar*). Os sistemas do tipo SSS são muito utilizados no mapeamento de fundos marinhos e fluviais, razão pela qual têm vindo progressivamente a ser instalados em AUV e outros veículos subaquáticos.

A imagem adquirida com este tipo de sistemas fornece informação sobre determinadas propriedades acústicas do fundo (reflectividade) e permite inferir sobre outras características tais como a natureza e textura dos materiais que o constituem, assim como a localização aproximada de objectos presentes na sua superfície. Os registos assim obtidos assemelham-se a uma fotografia monocromática e fornecem uma imagem bastante realista dos fundos marinhos.

Através da análise destas imagens é possível detectar a presença de estruturas naturais, tais como afloramentos rochosos, de outras formações geológicas diferenciadas ou artefactos de origem humana tais como embarcações afundadas e condutas submarinas.

Dada a sua versatilidade, o sonar de varrimento lateral é actualmente usado em diversas aplicações científicas no domínio da oceanografia, na arqueologia subaquática, em prospecção geofísica para fins industriais, na pesquisa de navios ou aeronaves afundadas, na inspecção de infra-estruturas submersas e em aplicações militares tais como detecção de minas submarinas.

1.1.1. Técnicas de navegação subaquática

Esta secção tem por base [1].

A utilização de AUV, usados inicialmente onde a utilização de veículos controlados remotamente (ROV *Remotely Operated Vehicles*), era inadequada ou impossível, tem vindo a aumentar devido às suas características de estabilidade, autonomia em navegação de longo curso e versatilidade em termos de equipamentos instalados, aliadas à sua crescente vulgarização e diminuição de custos.

O desafio apresentado pela navegação de AUV deve-se, principalmente ao facto de que estes são utilizados em ambientes sem estruturas bem definidas e onde a navegação baseada em satélites (por exemplo GPS) não é possível.

Missões relacionadas com a inactivação de minas submarinas ou até operações executadas sob uma camada de gelo são exemplos de como a utilização de AUV constitui uma alternativa muito menos perigosa quando comparada com outros métodos, nomeadamente veículos de superfície ou submersíveis tripulados.

No entanto, para uma missão ter sucesso é essencial que o AUV siga o trajecto previamente especificado e que chegue com precisão à localização estabelecida para a sua recolha.

A qualidade da informação recolhida está dependente da precisão com que a navegação é efectuada. Por exemplo a mudança de rota ou velocidade pode provocar desalinhamentos na recolha de imagens.

Os métodos de navegação subaquática actualmente existentes podem ser divididos em 3 categorias: Navegação Inercial, Navegação Acústica e Navegação Geofísica (ou Baseada no Terreno).

- Navegação Inercial

A navegação inercial faz uso de sensores giroscópicos para fazer a detecção da aceleração do AUV, além de um registo de velocidade de Doppler (DVL *Doppler Velocity Log*), que mede a velocidade relativa do veículo.

Embora os sistemas de navegação inercial (INS *Inertial Navigation Systems*), poderem ser colocados em AUV de baixo custo, a utilização de INS de alto desempenho está limitada a AUV mais dispendiosos.

Dado que os acelerómetros de um INS estão sujeitos a fenómenos de deriva, em missões mais longas o AUV pode ser equipado com um sonar DVL. Contudo, isso aumenta o custo total da missão, sendo este custo tanto maior quanto maior for a precisão dos equipamentos usados.

Um problema adicional deste tipo de sistemas reside no facto de o DVL ter um alcance limitado, pelo que, para se obterem medições mais precisas em termos de navegação, deve ser usado perto do fundo marinho. É certo que também é possível o DVL obter medições em relação à água, não havendo assim necessidade de atingir o fundo marinho. Contudo, este cenário é propenso a causar erros de deriva, uma vez que a existência de correntes oceânicas que provoquem o arrastamento do veículo não é detectada pelo sonar DVL.

A navegação baseada em INS e DVL é vulgarmente designada por navegação *dead-reckoning.* Uma grande desvantagem deste tipo de navegação é que a estimativa

da posição do veículo degrada-se ao longo da missão. Desta forma, uma recalibração dos sistemas torna-se necessária, através do uso de um receptor GPS. Esta necessidade de acesso periódico à superfície limita o *dead-reckoning* a missões em águas pouco profundas, dado não ser possível receber sinal GPS debaixo de água, uma vez que esta provoca uma forte atenuação dos sinais electromagnéticos de alta frequência.

- Navegação Acústica

Esta forma de navegação baseia-se na utilização de balizas de sinalização acústicas (*beacons*) que permitem ao AUV determinar a sua localização ao longo da missão.

Existem diversos sistemas que fazem uso deste conceito; os mais utilizados são o LBL (*Long Baseline*) e o USBL (*Ultra Short Baseline*).

O sistema LBL requer a instalação de pelo menos duas balizas de sinalização acústicas, normalmente localizadas junto ao fundo marinho. Estas balizas de sinalização enviam um sinal acústico ao AUV que lhe permite obter uma estimativa da sua localização em relação a cada baliza. Para tal, o AUV necessita de saber, para além da posição de cada baliza, a velocidade e o tempo de propagação do som no meio local.

O sistema USBL funciona de uma forma semelhante. A diferença reside no facto de ser apenas utilizada uma baliza de sinalização acústica, normalmente fixa a um veículo à superfície.

Os sistemas USBL mais modernos usam balizas de sinalização equipadas com sistemas INS/GPS por forma a reduzir as necessidades de recalibração do veículo de superfície.

Apesar de teoricamente não haver limite para a extensão da rede de balizas de sinalização acústicas, o alcance limitado de cada baliza assim como o custo de instalação e manutenção de uma rede de grandes dimensões, torna estes sistemas impraticáveis num grande número de missões.

- Navegação geofísica (ou Baseada no Terreno)

Esta técnica faz uso de características físicas (*features*) localizadas no fundo marinho para obter uma estimativa da posição do AUV. Estas *features* podem ser preexistentes, como por exemplo formações rochosas ou embarcações, ou podem ser propositadamente colocadas no local onde se vai efectuar a missão.

A navegação usando este tipo de *features* é feita fornecendo um mapa já existente do local ao AUV, ou através da construção deste no decorrer da missão, através de técnicas SLAM (*Simultaneous localization and mapping*)[2].

Esta forma de navegação é vantajosa para missões onde não é possível ao AUV emergir para adquirir sinal GPS.

Uma desvantagem deste tipo de sistemas reside no facto de estarem dependentes da presença de *features* adequadas, assim como da capacidade do AUV ter sensores capazes de extrair e fazer um uso eficiente dessas mesmas *features*, o que inviabiliza a navegação tendo por base a utilização de sistemas de mapeamento automático.

A navegação geofísica tem como objectivo ter um funcionamento semelhante ao GPS, isto é, permitir uma localização do veículo em termos absolutos, em vez de uma localização relativa como a que é fornecida pelos métodos do tipo *dead-reckoning*.

1.1.2. O Sonar de Varrimento Lateral

Esta secção tem por base [3].

O Sonar de Varrimento Lateral (SSS *Side-scan Sonar*), é utilizado sempre que se pretende efectuar o mapeamento de grandes áreas do fundo marinho.

A sua principal característica é a alta resolução das imagens obtidas. De facto, variando as frequências com que o sinal acústico é emitido (desde 6.5 kHz até 1 MHz), é possível aos sonares de varrimento lateral obterem resoluções desde 60m até 1cm.

No entanto a utilização de frequências elevadas acarreta custos. Tornam-se necessários sistemas de aquisição de dados mais rápidos, assim como maiores velocidades de processamento. Além disto, altas frequências não se propagam tão longe. Desta forma, é necessário estabelecer um compromisso entre resolução e alcance do sonar.

Estão disponíveis cada vez mais sistemas multifrequência, cujo propósito é dar resposta aos problemas citados anteriormente. Geralmente estes sistemas permitem a configuração da frequência desejada antes de serem lançados nas missões. Apesar de raros, já existem também sistemas que permitem mudanças de frequência no decorrer da sua utilização.

O sonar de varrimento lateral baseia-se na emissão de dois feixes acústicos, um de cada lado do aparelho. Estes feixes apresentam uma forma cónica, bastante larga quando observada de uma perspectiva perpendicular ao movimento do sonar. Estes aparelhos são normalmente rebocados por uma embarcação e encontram-se instalados num veículo submerso, normalmente com um formato semelhante a um torpedo, usualmente designado “Peixe”.

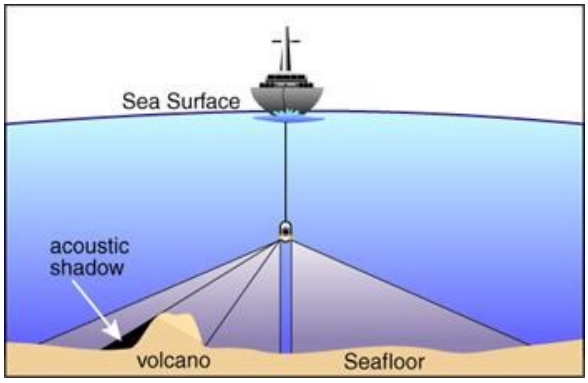


Figura 1 – Funcionamento de um sonar de varrimento lateral.[1]

[1] Imagem retirada de http://gralston1.home.mindspring.com/Sidescan.html

Figura 2 - Exemplo de um sonar de varrimento lateral instalado num veículo rebocado, "Peixe".[2]

A elevada qualidade das imagens, aliada ao facto de poderem ser obtidas faixas representando desde dezenas de metros até aproximadamente 60 km de largura do fundo marinho, faz com que o sonar de varrimento lateral seja a ferramenta de eleição para variados tipos de missões.

1.2. Estado da arte

1.2.1. Navegação geofísica baseada em dados de SSS: definição do problema

De entre os vários métodos de navegação subaquática, a navegação geofísica é aquela que apresenta maiores vantagens, principalmente quando se pretendem efectuar missões cada vez mais longas que requerem um elevado grau de autonomia por parte do AUV. Além disso, este tipo de missões é frequentemente caracterizado por não haver rota pré-definida. Tal deve-se ao facto de se pretender que os AUVs operem cada vez mais perto e de forma independente de estruturas que constituem um obstáculo ao seu movimento, por exemplo oleodutos ou outro tipo de estruturas subaquáticas presentes em portos. Contudo, estas estruturas representam também um conjunto de características (*features*) importantes para a navegação, pois têm elevada utilidade como pontos de referenciação.

Para além das estruturas artificiais, também as estruturas naturais presentes no fundo marinho têm extrema importância para a navegação dos AUV.

Neste contexto o sonar de varrimento lateral revela-se a ferramenta mais indicada. A capacidade de adquirir imagens de alta resolução, aliada ao facto de que os sistemas SSS têm um elevado alcance na sua aquisição potencia não só uma maior facilidade de identificação de *features* relevantes, como também a possibilidade de serem detectadas em quantidades muito maiores.

Apesar da grande vantagem do SSS ser a elevada qualidade da imagem obtida, em diversos ambientes subaquáticos é claramente impossível fazer uma correcta identificação das *features*.

Devido à constante modificação que o fundo marinho sofre, por exemplo devido a correntes, quanto maior for o intervalo entre a aquisição do mapa e o lançamento do AUV, maior será a possibilidade de o que é observado pelo veículo diferir do que é descrito no mapa.

[2] Imagem retirada de http://www.jwfishers.com/sss.htm

A erosão não é o único factor causador de perturbações. A própria forma como as imagens são adquiridas acarreta problemas. Note-se que o mapa da zona onde se efectuará a missão é tipicamente obtido por um SSS rebocado por uma embarcação. Ora, o AUV, que se pretende usar na navegação geofísica, irá obter imagens muito mais próximo do fundo. Desta forma, torna-se óbvio que os dois conjuntos de imagens serão diferentes, devido aos diferentes ângulos de incidência dos sinais de sonar.

De facto, é impossível garantir as mesmas condições de aquisição dos dados não só entre diferentes segmentos de uma mesma campanha oceanográfica mas mesmo no decurso de um desses segmentos. Estas alterações têm impactos significativos nas características das imagens adquiridas, não só em termos da intensidade dos sinais registados, como também em termos da sua orientação espacial relativa aos sistemas de aquisição.

A aquisição de dados de SSS pode ser perturbada por outros factores, tais como: a presença de cardumes de peixes; variações de densidade e temperatura ao longo da coluna de água, que provocam refracções dos sinais acústicos e atenuações da sua intensidade; interferências de outras fontes de ruído acústico tais como "*chirps*" emitidos por mamíferos marinhos, motores e sondas de embarcações.

Perante este cenário, torna-se necessário o recurso a técnicas de processamento de imagens que não sejam dependentes de contornos de *features* facilmente identificáveis. Caso se verificasse este cenário ideal, meras operações de correlação entre imagens bastariam para efectuar uma correcta identificação e localização dessas mesmas *features.* Dado que a realidade é bem distinta, é essencial introduzir técnicas de processamento que permitam "enriquecer" as imagens de sonar por forma a possibilitar a extracção de mais e melhor informação para a navegação.

1.2.2. Algoritmos de navegação subaquática baseados em métodos de correlação cruzada

A Correlação Cruzada (CC) de dados é um método bem conhecido em estatística aplicada e em processamento de imagem, sendo aplicado frequentemente em operações de "*template matching*".

Como tal, em princípio, este método seria adequado para o objectivo do presente trabalho. Verifica-se, no entanto, como tem sido referido na literatura, que a aplicação de CC ou da sua variante normalizada, NCC (do inglês *Normalized Cross Correlation*) não dá resultados satisfatórios quando aplicada a dados afectados por ruído, ou por transformações não lineares como as que são induzidas tipicamente nas imagens de sonar de varrimento lateral[4].

Este método poderá ser aplicável com maior sucesso à navegação baseada no terreno com recurso a dados de multifeixe, que fornecem directamente uma imagem acústica tridimensional do fundo marinho, permitindo uma correlação mais imediata com um mapa da topografia do terreno[5].

1.2.3. Utilização de Teoria da Informação para registo e alinhamento de imagens

Os métodos de classificação e alinhamento de imagem baseados na Teoria da Informação, nomeadamente o método de maximização da Informação Mútua (IM), têm sido referidos como mais robustos do que os métodos de correlação convencionais.

Tal é atribuído principalmente ao facto de a IM ser praticamente insensível a transformações não-lineares dos dados, que inviabilizam os procedimentos clássicos de correlação da intensidade das imagens [6].

1.2.4. Características de Haralick (introdução)

Uma das técnicas utilizadas na classificação de imagens - por exemplo, para classificar uma área geográfica com base em imagens de fotografia aérea ou imagens de satélite ou para classificação de tecidos orgânicos a partir de uma microfotografia utilizada em medicina [7] - consiste na utilização de características texturais que permitem identificar objectos ou regiões de interesse na imagem analisada. Neste tipo de aplicações, onde é difícil o estabelecimento de padrões pré-definidos facilmente reconhecíveis, a informação textural permite uma classificação mais robusta e eficaz do que a técnica de reconhecimento do tipo *template matching*. Para tal, revela-se potencialmente eficaz a metodologia proposta por Haralick et al. [7, 8] que estabeleceram um procedimento para a extracção de um conjunto de características texturais de blocos de imagens representadas em formato digital. Estas características, adiante designadas por características de Haralick (ou *Haralick features*) são calculadas no domínio espacial, tendo em conta a natureza estatística dos padrões de textura, a partir de imagens representadas por tons de cinzento. Naquele trabalho [7] é proposta a extracção de um conjunto de catorze características texturais a partir de blocos de uma imagem original.

Nesta abordagem, cada uma das características texturais é calculada para um conjunto de quatro direcções, pelo que as propriedades derivadas pelo método poderão ser sensíveis à orientação do bloco de imagem processado. Como se verá mais adiante, embora em diversas aplicações tal não seja necessário, esta propriedade do método pode ser interessante do ponto de vista da aplicação actual. Assim, na realidade, o conjunto de *Haralick features* originalmente proposto corresponde a um total de 14x4=56 características texturais, embora algumas delas possam não ser dependentes da orientação.

As características de Haralick mais usadas no tipo de problema que aqui se aborda são a Energia, o Contraste e, em alguns casos, a Homogeneidade. No presente trabalho, para além dos níveis de cinzento da imagem obtida, optou-se por se utilizar inicialmente as duas primeiras, calculadas em quatro direcções, totalizando assim um total de nove características atribuídas a cada ponto da imagem. Mais adiante será proposto um método de classificação e escolha das *Haralick features* potencialmente mais adequadas a imagens de sonar, cuja eficiência será testada neste trabalho com dados reais de SSS.

1.2.5. Utilização das características de Haralick em algoritmos de cálculo de Informação Mútua.

Tem sido referida na literatura (ver por exemplo [7]) a dificuldade de aplicar métodos de correlação convencionais às imagens de Sonar de Varrimento Lateral (SSS). Os dados obtidos com SSS representam a intensidade da reflexão de um sinal acústico numa dada área do fundo marinho e são geralmente representados em imagens que utilizam uma escala de níveis de cinzentos facilmente interpretáveis por um operador humano especializado.

Estas imagens são tipicamente muito texturadas mas geralmente isentas, ou com uma reduzida presença, de pontos notáveis (*landmarks*) que facilitem a sua georreferenciação ou o seu alinhamento com outras imagens obtidas previamente. Tais características tornam difícil a sua classificação e registo não supervisionados, bem como a sua utilização em sistemas automáticos de localização baseados em imagens (bi ou tridimensionais) do terreno (Navegação Geofísica).

Dadas as limitações referidas, torna-se importante extrair dos dados de sonar o máximo de informação possível, por forma a aumentar o potencial de utilização dos algoritmos de *matching.*

Este requisito é altamente relevante para os métodos baseados em teoria de informação, designadamente os que recorrem ao princípio de maximização da Informação Mútua entre duas imagens.

Para o efeito alguns autores têm utilizado com sucesso a extracção das características de Haralick (*Haralick Features*) a partir dos sinais "monocromáticos" de SSS. É o caso do trabalho [9] que, para além dos níveis de intensidade registados pelo equipamento de sonar, calcula a Energia e o Contraste segundo 4 orientações distintas.

Mais à frente no trabalho, será esclarecido o método de obtenção destas e de outras *features*, assim como a importância de serem extraídas segundo diferentes orientações. Também se pretende demonstrar que a utilização da informação adicional baseada nas características de *Haralick* aumenta significativamente a eficácia dos algoritmos de *matching*, o que justificaria a sua utilização.

2. Teoria da Informação e reconhecimento de padrões

O reconhecimento de padrões recorrendo a abordagens baseadas na Teoria da Informação e em particular no conceito de Informação Mútua (IM) calculada a partir da entropia de Shannon, tem vindo a aumentar significativamente em diversas áreas de aplicação, incluindo a imagem médica, a detecção remota e a classificação de imagens subaquáticas obtidas através de fotografia ou de sonar [9].

De acordo com a bibliografia disponível, os métodos baseados em IM são eficazes em domínios onde os algoritmos baseados na magnitude dos gradientes, na detecção de arestas e de outras características geométricas não funcionam adequadamente. No caso das imagens dos fundos marinhos, estas vantagens são muito relevantes, uma vez que as condições de aquisição dos dados variam significativamente ao longo do tempo, como já referido anteriormente.

Foram então elaborados conjuntos de testes, apresentados mais à frente neste capítulo, com o objectivo de avaliar a qualidade e o desempenho de diversos estimadores de entropia, passíveis de ser aplicados no cálculo da IM.

Primeiramente é feita uma exposição dos conceitos teóricos envolvidos, seguida da descrição dos testes efectuados e análise dos resultados neles obtidos.

2.1. Entropia

A bibliografia sobre teoria da informação [10] apresenta o conceito de entropia como uma medida da incerteza de uma variável aleatória.

Seja $p(x)$ a função densidade de probabilidade da variável aleatória discreta $x \in X$. A *entropia* $H(X)$ é definida por:

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{1}$$

O valor da entropia é função da variável aleatória X, nunca dependendo dos valores que esta variável pode tomar, mas sim das suas probabilidades.

De notar que quando a base do logaritmo é binária (base 2), o valor de entropia é expresso em *bits*. Caso a base do logaritmo seja neperiana (*e)*, o valor da entropia é expresso em *nats*.

Este conceito de entropia é também designado de entropia própria da variável aleatória considerada.

Apresenta-se de seguida o conceito de *entropia conjunta*.

Este conceito é em todo semelhante ao conceito base de entropia para uma única variável. A diferença reside no facto de agora serem tratados pares de variáveis aleatórias.

Considerando a distribuição de probabilidade conjunta $p(x, y)$, das variáveis aleatórias discretas $x \in X$ e $y \in Y$, a entropia conjunta é dada por:

$$H(X,Y) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(x,y) \tag{2}$$

Além disso define-se também *entropia condicional* como sendo:

$$H(Y|X) = \sum_{x\in X} p(x)H(Y|X=x) \tag{3}$$

Prova-se que esta definição pode ser reescrita da seguinte forma:

$$H(Y|X) = -\sum_{x\in X}\sum_{y\in Y} p(x,y)\log p(y|x) \tag{4}$$

A definição de entropia condicional é apresentada devido à sua relação com a entropia conjunta. Esta relação é expressa pelo seguinte teorema:

$$H(X,Y) = H(X) + H(Y|X) \tag{5}$$

Apesar de ser clarificado mais à frente, é importante referir desde já que o conceito de entropia conjunta é de extrema importância na definição de informação mútua.

De facto, uma vez que o pretendido é, essencialmente, a comparação de duas imagens, intuitivamente se percebe a importância de qualquer conceito estatístico que estabeleça uma relação entre elas.

2.2. Informação Mútua

Informação mútua (IM) é uma medida da quantidade de informação que uma variável aleatória contém sobre uma outra variável aleatória. Este conceito também pode ser interpretado como a redução da incerteza de uma dada variável aleatória devido ao conhecimento de uma outra.

Seja a distribuição de probabilidade conjunta das variáveis $x \in X$ e $y \in Y$ definida por $p(x,y)$, com funções de probabilidade marginais, respectivamente, $p(x)$ e $p(y)$. A Informação Mútua $I(X;Y)$ é dada por:

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \tag{6}$$

A partir da relação anterior, pode deduzir-se uma expressão para a informação mútua em função da entropia:

$$I(X;Y) = H(X) - H(X|Y) \tag{7}$$

Nesta expressão está claro o mencionado anteriormente: a informação mútua $I(X;Y)$ é a redução da incerteza de X por conhecimento de Y ($H(X|Y)$).

De igual forma é possível escrever:

$$I(X;Y) = H(Y) - H(Y|X) \tag{8}$$

É agora possível estabelecer a relação entre entropia conjunta e informação mútua.

Relacionando (5) com a expressão anterior, tem-se:

$$H(Y|X) = -H(X) + H(X,Y) \tag{9}$$

Relacionando (8) e (9), obtém-se:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{10}$$

É importante referir também que a informação mútua de uma dada variável aleatória com ela própria é dada pela entropia própria dessa mesma variável.

Prova-se que:

$$I(X;X) = H(X) - H(X|X) = H(X) \tag{11}$$

Por este motivo a entropia de uma dada variável aleatória é também designada por *informação própria* (*self-information*).

É igualmente fácil demonstrar que a entropia própria de uma variável aleatória é igual à entropia conjunta dessa variável com ela própria.

Sabendo que $I(X;X) = H(X)$, seja

$$\begin{aligned} I(X;X) &= H(X) + H(X) - H(X,X) \\ &= 2H(X) - H(X,X), \text{ tem-se que} \end{aligned}$$

$$H(X) = 2H(X) - H(X,X)$$

$$\Leftrightarrow H(X,X) = H(X) \tag{12}$$

Como se verá adiante, este último conceito revela-se fundamental do ponto de vista prático. A expressão da informação mútua pode ser reescrita como:

$$I(X;Y) = H(X,X) + H(Y,Y) - H(X,Y) \tag{13}$$

Deste modo é possível calcular os valores de IM pretendidos utilizando um único estimador de entropia, uma vez que os valores de entropia própria são calculados de igual forma que os valores de entropia conjunta.

2.3. Métodos de cálculo de entropia

Dado que o método de *matching* que se pretende utilizar se baseia na cálculo da informação mútua, que por sua vez se baseia no cálculo da entropia, torna-se imperativo dispor de um método de cálculo de entropia preciso e computacionalmente eficaz.

Exploram-se 3 métodos, a saber:

- histogramas;
- função *entropy* do Matlab;
- algoritmo de estimação de entropia *k-d partitioning entropy estimator* (*kdpee*)

2.3.1. Histogramas

Histograma é uma representação gráfica que permite a visualização da distribuição de probabilidade de um determinado conjunto de dados.

No caso de variáveis discretas corresponde a efectuar a contagem do número de ocorrências de cada combinação de valores que as variáveis aleatórias podem tomar dentro do conjunto de dados considerado.

Neste trabalho em particular, os histogramas são utilizados como forma de se obter uma aproximação à função de probabilidade dos dados utilizados, essencial para os cálculos de entropia pretendidos.

Dada a sua natureza claramente estatística, os histogramas constituem a forma mais precisa de se obter uma distribuição de probabilidades, desde que o espaço de amostras considerado seja suficientemente grande. Tal nunca se revela um problema neste trabalho, pois o volume de dados a tratar é bastante elevado.

No entanto o cálculo de histogramas é potencialmente o mais exigente a nível computacional. Obter histogramas de imagens monocromáticas não é muito complexo. No entanto quando se pretende aplicar os conceitos de cálculo de entropia a imagens policromáticas (por exemplo imagens *RGB*) essa complexidade aumenta muito, tornando-se inviável numa aplicação prática.

Deste modo, a utilização de histogramas, sendo o método que fornece resultados mais exactos, é considerada neste trabalho apenas como termo de comparação para verificar a eficácia de outros estimadores de entropia.

2.3.2. Entropy do Matlab®

Dada a sua facilidade de utilização, e popularidade como ferramenta de simulação, o Matlab® foi a principal plataforma para os programas e testes desenvolvidos neste trabalho.

Esta aplicação disponibiliza a função *entropy* capaz de calcular a entropia de uma dada imagem em escala de cinzentos (de 0 a 255).

A expectativa era que esta função constituísse uma referência tanto em termos de precisão como de eficiência computacional.

2.3.3. Estimador *k-d partitioning*

Utilizou-se também o estimador *kdpee* (*k-d partitioning entropy estimator*) [11]. Trata-se de um método de estimar o valor de entropia que pode ser aplicado de forma igualmente eficiente a dados multidimensionais.

A novidade introduzida por este método é a forma como é calculada a distribuição de probabilidades dos dados. Os autores usam um procedimento em tudo semelhante ao utilizado numa decomposição em árvore k-dimensional (*k-dimensional tree*). Nesta estrutura, a amostra inicial corresponde à raiz, as sucessivas divisões dos dados aos ramos, e as folhas aos intervalos finais em que os dados são decompostos.

A ideia do algoritmo é obter uma aproximação de uma função densidade de probabilidade genérica. Para tal, o espaço de amostragem é sucessivamente decomposto em intervalos de modo a que os dados contidos em cada intervalo sejam caracterizados por uma distribuição uniforme. De notar que esta divisão é feita ao longo de todas as dimensões da amostra de dados (caso seja multidimensional) e de forma independente.

A função densidade de probabilidade da amostra de dados considerada será a concatenação das probabilidades em cada um dos intervalos obtidos. Esta probabilidade é dada pela razão entre o número de pontos do intervalo considerado e o número total de pontos de amostra correspondente.

A partição de cada intervalo é feita de forma recursiva com base no valor da sua mediana até ser atingida um determinado critério de paragem. Esse critério consiste na verificação da uniformidade do intervalo, segundo condições especificadas pelos autores.

O algoritmo termina quando todos os intervalos obedecem ao critério de uniformidade imposto.

Uma vantagem de utilizar o algoritmo proposto reside no facto de que o código é fornecido pelos autores ao abrigo da *GNU General Public License*.

2.3.4. Testes de validação dos algoritmos

De seguida serão apresentados os testes realizados para identificar qual o melhor estimador de entropia a utilizar.

O primeiro passo consistiu em obter uma correcta estimação da distribuição de probabilidades de um conjunto de dados aleatórios, através de histogramas.

Começou-se por um conjunto de dados aleatórios segundo uma distribuição normal com média (μ) zero e desvio padrão (σ) unitário. A Figura 3 mostra o resultado obtido

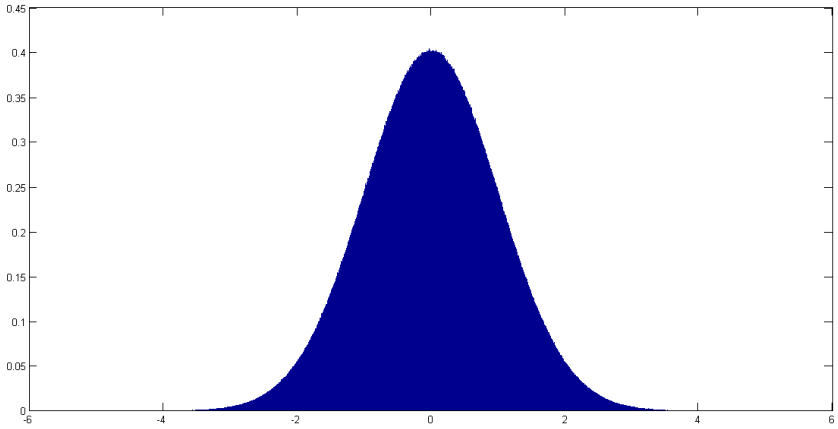


Figura 3 - Distribuição de probabilidades de dados gerados aleatoriamente segundo uma distribuição normal, obtida através do cálculo de histogramas.

Usou-se, como termo de comparação, a função *pdf* (*Probability Density Functions*) do Matlab. Esta função permite obter, segundo parâmetros bem especificados, a densidade de probabilidade de distribuições definidas analiticamente (por exemplo, Beta, Binomial, Exponencial, Normal). Escolheu-se neste caso a distribuição Normal.

O resultado é apresentado na Figura 4; o intervalo de dados foi o mesmo utilizado no cálculo com histogramas.

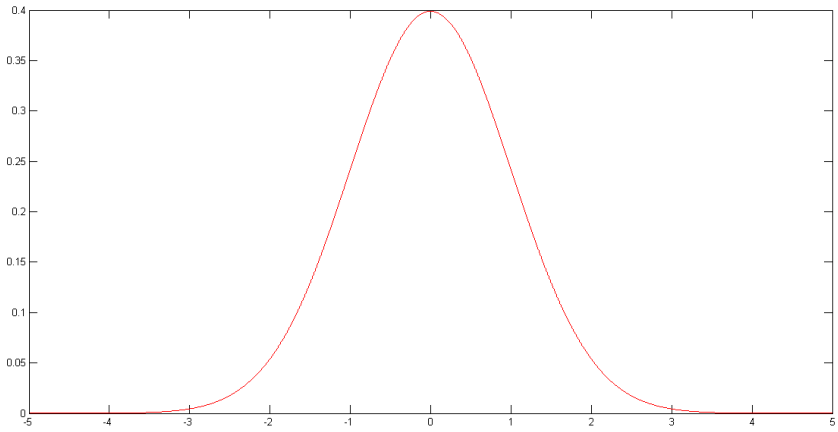


Figura 4 - Distribuição Normal, obtida usando a função *pdf* do Matlab

Tal como anteriormente, a distribuição foi especificada por forma a ter média (μ) nula e desvio padrão (σ) unitário.

Seguidamente, as duas imagens foram sobrepostas por forma a possibilitar a comparação directa dos resultados, (vide Figura 5).

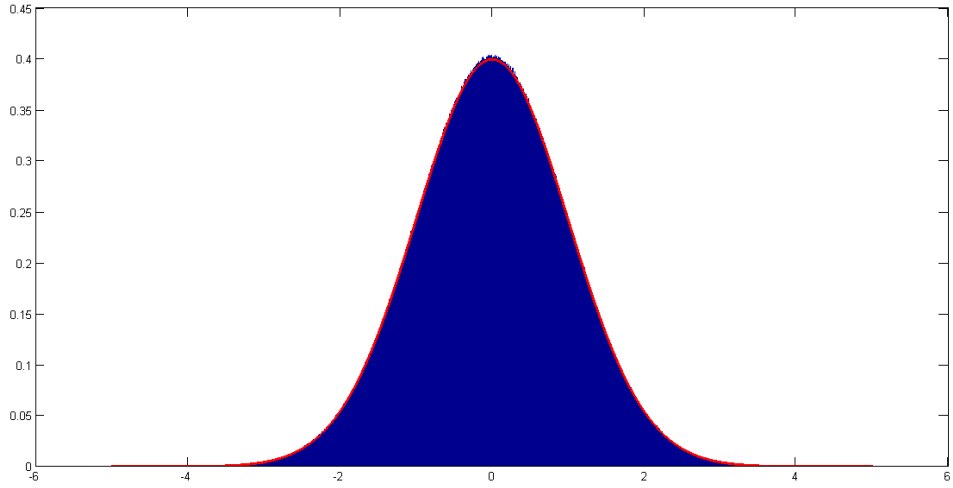


Figura 5 - Comparação entre as distribuições de probabilidade obtidas. A azul encontra-se a distribuição obtida através do cálculo de histogramas. A vermelho encontra-se a distribuição obtida utilizando a função *pdf*.

Este teste confirmou que a distribuição Normal obtida a partir do cálculo de histogramas constitui uma excelente aproximação à representação teórica obtida com a função *pdf*.

É assim possível aplicar a fórmula clássica da Entropia e obter a estimativa pretendida, utilizando as funções de probabilidade obtidas com recurso a histogramas.

No entanto esta comparação apenas serve de referência para atestar a validade do cálculo de histogramas na estimação de entropia.

A forma mais prática de atestar a validade dos estimadores utilizados consiste em recorrer à expressão teórica da entropia para uma distribuição de probabilidade definida analiticamente, como é o caso da distribuição Normal. Esta expressão é obtida por aplicação da fórmula da *entropia diferencial*, para variáveis contínuas, à expressão teórica da distribuição em causa.

Considerando a variável aleatória contínua $x \in X$, cuja função densidade de probabilidade é $f(x)$, a entropia diferencial é definida por

$$h(X) = -\int f(x) \log f(x)\, dx. \tag{14}$$

Para a distribuição normal, definida por

$$\phi(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}}, \tag{15}$$

prova-se que

$$h(\phi) = -\int \phi \ln \phi \tag{16}$$

$$= \frac{1}{2} \ln 2\pi e\sigma^2 \text{ (expresso em } \textit{nats}) \tag{17}$$

$$= \frac{1}{2} \log_2 2\pi e\sigma^2 \text{ (expresso em } \textit{bits}). \tag{18}$$

Obtêm-se assim os valores teóricos para a entropia de uma distribuição Normal. Uma vez que a distribuição Normal utilizada tem desvio padrão (σ) igual a 1, a variância (σ^2) também será 1.

O valor teórico da entropia de uma distribuição Normal de média nula e desvio padrão igual a 1 é então:

$$h(\phi) = \frac{1}{2} \ln 2\pi e\sigma^2 = 1.4189 \ (nats) \tag{19}$$

$$h(\phi) = \frac{1}{2} \log_2 2\pi e\sigma^2 = 2.0471 \ (bits) \tag{20}$$

A Tabela 1 compara os valores de entropia $H(X)$ obtidos, para um conjunto de 10^8 amostras, através de histogramas, com a função *pdf* do Matlab e com o algoritmo *kdpee*.

Tabela 1 - Tabela de valores de entropia, obtidos pelos diversos métodos testados.

Estimador utilizado	H(X) em *nats*	H(X) em *bits*
Histogramas	1,4189	2,0470
Função *pdf*	1,4189	2,0470
Algoritmo *kdpee*	1,4188	2,0469

Os vários métodos utilizados para estimar o valor da entropia revelam-se precisos, como era esperado.

Contudo, o objectivo deste trabalho passa por estimar a entropia de imagens em escala de cinzentos, com valores da intensidade de cada pixel na gama de 0 a 255 (8 *bits*). Neste facto reside o motivo de não se ter utilizado a função *entropy* do Matlab. A função *entropy* utiliza uma outra função do matlab para obter o valor da distribuição de probabilidades da amostra, para de seguida aplicar a fórmula da entropia. A função utilizada é a *imhist*, que obtém o histograma de uma imagem de acordo com a gama de valores dessa imagem. Por defeito a função *entropy* trata sempre as imagens como sendo de 8 bits, pelo que vai calcular sempre um histograma numa gama de 256 valores.

Nos testes anteriores usaram-se dados gerados aleatoriamente segundo uma distribuição Normal, compreende quer valores positivos, quer negativos. Nestas condições, a função *entropy* devolve resultados incorrectos.

Torna-se necessário elaborar um conjunto de testes adequado à estimação de entropia de uma imagem. Neste sentido, e seguindo o método de ter uma fórmula teórica que permita uma correcta validação dos testes, escolheu-se gerar dados aleatórios segundo uma *distribuição uniforme* entre 0 e 255.

Tendo uma variável aleatória $x \in X$, a distribuição uniforme $f(x)$ é definida por

$$f(x) = \frac{1}{\beta - \alpha}, \alpha \leq x \leq \beta. \quad (21)$$

Prova-se que a sua entropia é dada por

$$H(x) = \ln(\beta - \alpha)\ (nats) \quad (22)$$

$$H(x) = \log_2(\beta - \alpha)\ (bits) \quad (23)$$

Para a realização dos testes gerou-se um conjunto de dados aleatórios constituído por 10^7 amostras, na gama de valores de 0 a 255.

O valor teórico da distribuição uniforme no intervalo considerado é

$$H(x) = \ln(255 - 0) = 5.5413\ (nats) \quad (24)$$

$$H(x) = \log_2(255 - 0) = 7.9944\ (bits) \quad (25)$$

Os resultados obtidos são os apresentados na Tabela 2. Os dados da distribuição uniforme gerados eme Matlab são do tipo *double*.

Tabela 2 - Resultados dos diferentes estimadores de entropia, para uma distribuição uniforme no intervalo [0;255] (dados tipo *double*).

Estimador (amostras tipo *double*)	H(X) em *nats*	H(X) em *bits*
Histogramas	5,5413	7,9943
Função *entropy*	0,0474	0,0681
Algoritmo *kdpee*	5,5410	7,9948

Como é possível observar, os resultados obtidos, quer com histogramas, quer com o algoritmo *kdpee*, são condizentes com as previsões teóricas, o que confirma a eficácia destes dois estimadores. A função *entropy* do Matlab apresentou resultados completamente divergentes. No entanto, tal era previsível uma vez que a função *entropy* faz não só uma conversão dos dados para *uint8* (inteiros de 8 bits), como também uma mudança de escala.

Estes testes foram repetidos para a mesma amostra de dados, mas agora convertida para inteiros de 8 bits.

Os resultados obtidos são os apresentados na Tabela 3.

Tabela 3 - Resultados dos diferentes estimadores de entropia, para uma distribuição uniforme no intervalo [0;255] (dados do tipo *uint8*).

Estimador (amostras tipo *uint8*)	H(X) em *nats*	H(X) em *bits*
Histogramas	5,5440	7,9983
Função *entropy*	5,5440	7,9983
Algoritmo *kdpee*	não aplicável	não aplicável

Analisando esta tabela, verifica-se novamente a precisão dos resultados obtidos pelo cálculo de histogramas. Nesta situação, verifica-se que a função *entropy* apresenta resultados igualmente precisos o que confirma que ela funciona correctamente quando os dados estão no formato *uint8*. No entanto, desta vez não foi possível verificar o correcto funcionamento do algoritmo *kdpee*, uma vez que este algoritmo está escrito de forma a não aceitar qualquer outro tipo de dados que não *double*, sendo exibida uma mensagem de erro quando tal condição não é respeitada.

Repetiu-se este conjunto de testes usando intervalos diferentes para a distribuição uniforme, a saber: [0;1] e [10;110].

Os valores teóricos para uma distribuição Uniforme entre 0 e 1 são:

$$H(x) = \ln(1-0) = 0 \ (nats) \tag{26}$$

$$H(x) = \log_2(1-0) = 0 \ (bits) \tag{27}$$

Os valores teóricos para uma distribuição Uniforme entre 10 e 110 são:

$$H(x) = \ln(110-10) = 4.6052 \ (nats) \tag{28}$$

$$H(x) = \log_2(110-10) = 6.6439 \ (bits) \tag{29}$$

Os resultados obtidos no 1º caso são apresentados nas duas tabelas seguintes (respectivamente, para dados do tipo *double* e do tipo *uint8*).

Tabela 4 - Resultados dos diferentes estimadores de entropia, para uma distribuição uniforme no intervalo [0;1] (dados do tipo *double*).

Estimador (amostras tipo *double*)	H(X) em *nats*	H(X) em *bits*
Histogramas	0	0
Função *entropy*	5,5438	7,9981
Algoritmo *kdpee*	-0,0006	-0,0009

Tabela 5 - Resultados dos diferentes estimadores de entropia, para uma distribuição uniforme no intervalo [0;1] (dados do tipo *uint8*).

Estimador (amostras tipo *uint8*)	H(X) em *nats*	H(X) em *bits*
Histogramas	0,6931	1,0000
Função *entropy*	0,6931	1,0000
Algoritmo *kdpee*	não aplicável	não aplicável

Verifica-se que, para dados do tipo *double*, os resultados obtidos usando histogramas e o algoritmo *kdpee* correspondem aos valores teóricos previstos. Os obtidos pela função *entropy* são idênticos aos obtidos quando o intervalo de dados considerado corresponde a valores de 8 bits (vide Tabela 3). Tal resultado advém do facto, previamente referido, de que esta função faz uma mudança de escala convertendo os valores para a escala de 0 a 255.

Na Tabela 5 os resultados não estão de acordo com o esperado. Tal deve-se ao facto de que a geração em Matlab de uma distribuição Uniforme de dados do tipo *uint8*, conter apenas dois valores distintos: 0 e 1. Esta situação reflecte o caso particular da estimação da entropia da quantização *n*-bit de uma variável aleatória contínua.

Considerando um intervalo de quantização de $\Delta = \frac{1}{2^n}$ e a variável aleatória quantizada X^Δ cuja função de densidade probabilidade é dada por $f(x_i)\Delta$ tal que $x_i \in X^\Delta$, é possível provar que a entropia é dada por:

$$H(X^\Delta) = -\sum \Delta f(x_i) \log f(x_i) - \log \Delta \tag{30}$$

Desta forma é possível comprovar os resultados obtidos, uma vez que (para *n*=1)

$$\ln \frac{1}{2^1} = -0.6931 \tag{31}$$

$$\log_2 \frac{1}{2^1} = -1 \tag{32}$$

Contudo, tal verifica-se para níveis de quantização baixos, pois é possível provar que adicionando à entropia da variável aleatória quantizada X^Δ o valor de $\log_2 \Delta$ (ou $\ln \Delta$, conforme o pretendido), esta tende para o valor de entropia de uma variável aleatória contínua X à medida que Δ tende para zero.

$$H(X^\Delta) + \log_2 \Delta \to H(X), tal\ que\ \Delta \to 0 \tag{33}$$

De seguida apresentam-se os valores obtidos no 2º caso (entropia de uma distribuição Uniforme de valores entre 10 e 110), para dados *double* e *uint8*, respectivamente.

Tabela 6 - Resultados dos diferentes estimadores de entropia, para uma distribuição uniforme no intervalo [110;10] (dados do tipo *double*).

Estimador (amostras tipo *double*)	H(X) em *nats*	H(X) em *bits*
Histogramas	4,6051	6,6438
Função *entropy*	0	0
Algoritmo *kdpee*	4,6045	6,6429

Tabela 7 - Resultados dos diferentes estimadores de entropia, para uma distribuição uniforme no intervalo [110;10] (dados do tipo *uint8*).

Estimador (amostras tipo *uint8*)	H(X) em *nats*	H(X) em *bits*
Histogramas	4,6121	6,6539
Função *entropy*	4,6121	6,6539
Algoritmo *kdpee*	não aplicável	não aplicável

Mais uma vez se confirma a precisão que o cálculo de histogramas permite obter na estimação de entropia, revelando-se o único método que não mostra dependência do tipo de dados usados. Já a função *entropy* voltou a revelar problemas quando utiliza formato de dados diferente de inteiros de 8 bits, revelando-se contudo bastante precisa quando este formato de dados é utilizado. O algoritmo *kdpee* apresenta-se também como uma opção bastante precisa, como é observável pelos resultados apresentados.

Estes testes revelam que a estimação de entropia recorrendo a histogramas é o que mais imunidade oferece à variação no formato dos dados.

O próximo passo deste trabalho é aplicar a estimação de entropia ao problema de *matching* usando o conceito de informação mútua. Descrever-se-ão testes para a identificar as dificuldades inerentes, pretendendo-se simular não só diferentes metodologias de execução do *matching* como também situações adversas passíveis de correr num futuro ambiente de teste em tempo real.

3. Métodos de matching baseados em Informação Mútua

Tendo sido estudados os conceitos teóricos de entropia e informação mútua (IM), e verificada a eficiência dos estimadores considerados, o próximo passo consiste em aplicar estes conceitos ao *matching* de imagens.

Quando os *templates* são partes não alteradas do mapa de referência é expectável que o *matching* utilizando somente imagens na escala de cinzentos seja preciso e os resultados de acordo com o esperado. No entanto numa situação real tal não se verifica sendo assim necessário explorar uma vertente multidimensional através do uso de outro tipo de informação para além dos níveis de cinzento presentes na imagem. Esta necessidade advém do facto de ser difícil, senão mesmo impossível, que o AUV obtenha uma imagem que seja exactamente igual a uma dada parte de um mapa previamente adquirido. Diferenças resultantes da constante mudança que o fundo marinho sofre, devido a correntes e mesmo intervenção humana, aliado ao facto de os conjuntos de imagens serem obtidos a profundidades diferentes e muitas vezes por dispositivos de *SSS* diferentes, são factores que tornam bastante difícil o uso de métodos directos de *matching*.

Os diferentes atributos passíveis de fornecer informação adicional sobre imagem são designados de *features* e cada nova *feature* usada para classificação irá corresponder a uma nova dimensionalidade. Devido a factores como a qualidade da imagem de sonar utilizada, até à forma como são calculadas, o número de *features* necessárias a utilizar deverá ser definido pelo utilizador. Desta forma torna-se crítico que o algoritmo usado para detecção do máximo de informação mútua seja flexível ao ponto de permitir ser aplicado para o caso genérico de D dimensões.

3.1. Matching usando histogramas

3.1.1. Histogramas bidimensionais

Começou-se com um teste-piloto usando como mapa a imagem a) da Figura 6 (imagem da “Lena” usualmente utilizada em trabalhos realizados na área de processamento de sinal). Trata-se de uma imagem em escala de cinzentos de 8 bits (valores de 0 a 255), com 512x512 pixeis, e tendo um *template* correspondente a uma zona dessa imagem, com 50x50 pixeis. O algoritmo desenvolvido efectua o varrimento obtendo a cada iteração uma medida de IM, baseada no cálculo de Entropia Conjunto de duas imagens segundo o método de histogramas.

Os resultados obtidos encontram-se apresentados nas imagens c) e d) da Figura 6.

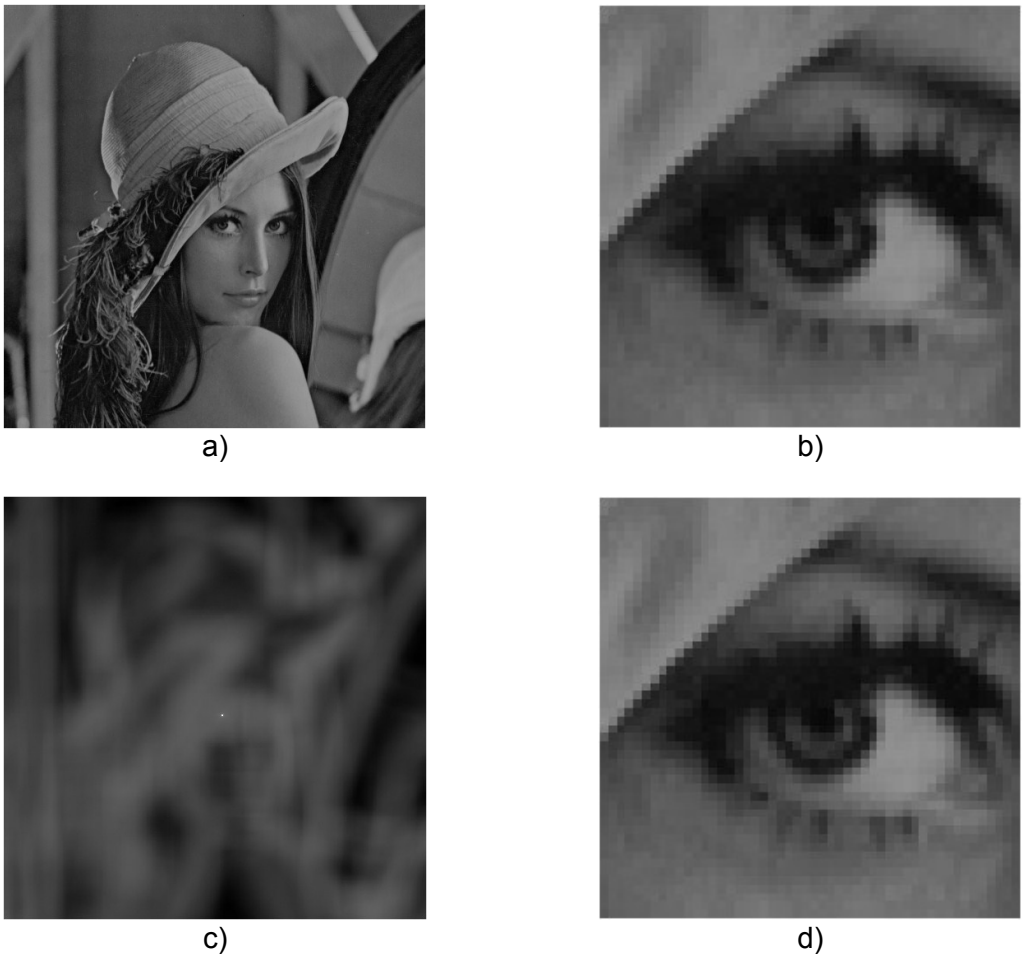


Figura 6 - Resultados de *matching* usando histogramas. Em a) temos o mapa (imagem "Lena") de 512x512 pixeis; em b) temos o *template* com 50x50 pixeis; em c) encontra-se uma visualização bidimensional da distribuição dos valores de IM ao longo do varrimento; em d) encontra-se a imagem localizada após o *matching*.

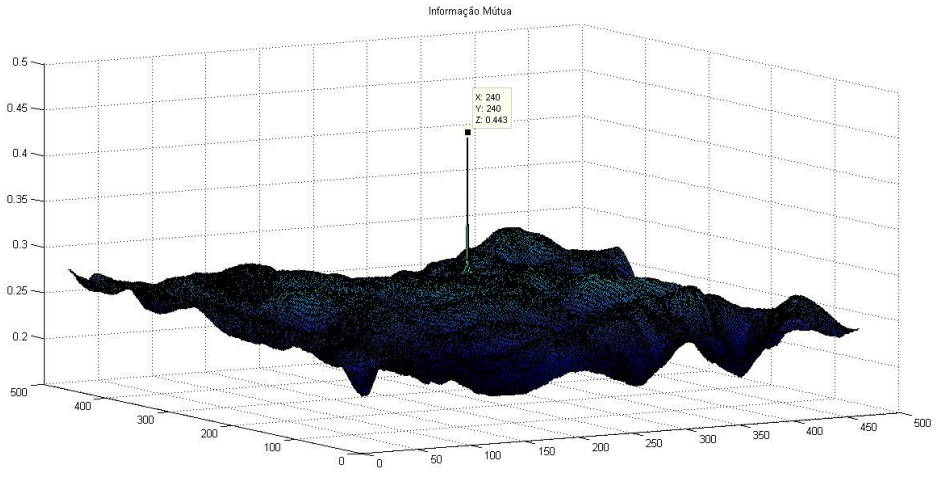


Figura 7 - Visualização tridimensional da imagem c) da Figura 6. O pico corresponde à posição do *template* na imagem original.

Os resultados correspondem ao esperado, pois o algoritmo de *matching* permitiu encontrar o *template* no mapa.

Numa primeira análise poder-se-ia concluir que o método funciona bem e que, portanto, os conceitos de entropia e informação mútua são passíveis de ser aplicados para situações de *matching*. No entanto o teste realizado fez uso apenas da informação presente na escala de cinzentos, que, como já referido anteriormente, poderá não ser suficiente para efectuar correctamente o *matching* em imagens reais de SSS.

Torna-se então necessário extrair mais informação das imagens em escala de cinzento, o que tem como principal consequência um aumento da sua dimensionalidade.

3.1.2. Histogramas multidimensionais

No caso da multidimensionalidade, o cálculo de informação mútua usando o método de histogramas revela-se computacionalmente ineficaz.

Considere-se de seguida a representação matricial do conjunto de pixeis de uma imagem, MxNxD, onde M corresponde ao número de linhas, N ao número de colunas e D à dimensionalidade da imagem. A título ilustrativo note-se que a uma imagem em formato *RGB* corresponde uma matriz MxNx3, sendo que D=3 representa os três níveis de cor da imagem (*Red*, *Green*, *Blue*).

Ao longo deste trabalho o termo “imagem monocromática” será usado para descrever imagens cuja representação matricial tem o formato MxNx1. Este formato corresponde às imagens em tons de cinzento de 8 bits. Sempre que forem mencionadas alterações na dimensionalidade da imagem está-se a referir a alterações do parâmetro D.

Tendo uma imagem A monocromática onde a única *feature* tida em consideração é a intensidade dada pela escala de cinzentos e tendo uma outra imagem B possuindo as mesmas características, o cálculo da entropia da própria imagem e da entropia conjunta entre A e B requer o cálculo de histogramas bidimensionais que são relativamente triviais de obter.

Se contudo forem extraídas mais *features* das imagens a dimensionalidade dos dados a tratar vai aumentar. Tendo agora uma imagem A que para além da intensidade é caracterizada por outras *features* como Contraste e Energia, por exemplo, e tendo uma imagem B nas mesmas condições, os cálculos necessários para obter o valor de entropia irão requerer o cálculo de histogramas com 6 dimensões.

Um histograma de 6 dimensões é entendido como sendo um histograma calculado para duas imagens, onde cada imagem tem a forma matricial MxNx3. Convém clarificar que o parâmetro D=3 representa o conjunto de 3 *features* considerado para as imagens e referidas no parágrafo anterior, sendo estas a intensidade (tons de cinzento), o Contraste e a Energia. De forma a reforçar a noção de histograma multidimensional refira-se histograma unidimensional é o caso onde é apenas pretendida uma medida da ocorrência de determinados valores numa imagem de formato MxNx1 (por exemplo, os valores de cada pixel da escala de cinzentos de 8 bits); por sua vez um histograma bidimensional é aquele onde é verificada a ocorrência conjunta dos valores dos pixeis de duas imagens de formato MxNx1; finalmente, histograma multidimensional é aquele onde é verificada a ocorrência conjunta dos valores da escala de cinzentos, de Contraste e de Energia, sendo que aqui a dimensionalidade igual a 6 é resultante da combinação das três *features* consideradas.

A informação dada por três *features* em cada imagem pode ainda não ser suficiente; o uso de um número maior de *features* irá fazer aumentar o peso computacional exigido do cálculo de histogramas multidimensionais.

Dada a pretensão de fazer *matching* em tempo real (ou próximo), tais exigências computacionais tornam o cálculo de histogramas inviável em aplicações reais.

Tal não impede que se utilize o cálculo de histogramas numa perspectiva de teste e comparação de resultados, pois estes permitem uma aplicação directa da expressão teórica da entropia e, de igual forma, da informação mútua, podendo ser utilizados para validar outros métodos que usam aproximações menos exactas.

3.2. Matching usando k-Nearest Neighbors

Perante as limitações encontradas com a utilização de histogramas para o cálculo de informação mútua para imagens multidimensionais, a operação de estimação de entropia passaria a ser efectuada pelo algoritmo *k-d partitioning entropy estimator* (*kdpee*) referido e testado anteriormente.

Como já verificado este algoritmo apresenta uma elevada precisão na estimação dos valores de entropia além de que, segundo os autores, o algoritmo estaria preparado e optimizado para o tratamento de imagens multidimensionais. No entanto detectou-se que este não funcionava quando se fornecia, como parâmetros de entrada, duas imagens iguais. Tal facto impossibilitava o cálculo da entropia própria de uma imagem, e desta forma, o cálculo da informação mútua entre duas imagens.

Foi então necessário encontrar uma alternativa ao algoritmo *kdpee* que permitisse obter a mesma precisão de resultados na estimação de entropia e que suportasse imagens multidimensionais.

A alternativa encontrada consiste na utilização do conceito de vizinhos. Baseia-se na determinação das distâncias euclidianas mínimas de um ponto definido no espaço multidimensional de *features*, que caracterizam cada pixel de uma imagem, aos pontos seus vizinhos de outra imagem.

Jan Kybic [11] propõe um estimador de entropia que tem por base a obtenção da distância mínima entre vizinhos, constituindo esse valor uma estimativa para a entropia.

De seguida apresenta-se o algoritmo proposto por Kybic onde *N* corresponde ao número de elementos na imagem considerada, *d* corresponde à dimensionalidade e $\gamma \approx 0.577$ à constante de Euler.

$$H(X) = \frac{d}{N}\sum_{i=0}^{N} \log Q_i + log\,\frac{(N-1)^{\frac{d}{2}}}{\left(1+\frac{d}{2}\right)} + \gamma \qquad (34)$$

De notar que $Q_i = min_{i\neq j} \parallel x_i - x_j \parallel$. Tal corresponde à distância euclidiana mínima entre as amostras x_i e os seus vizinhos.

Utilizando um algoritmo em Matlab® apropriado para o cálculo dos vizinhos foi então possível testar este algoritmo e verificar a sua aplicabilidade.

Além deste método proposto por Kybic existe outra possibilidade defendida por outros autores [12], que consiste numa média de todas as distâncias euclidianas calculadas. Esse algoritmo é designado por *MeanNN Entropy Estimator*.

Neste trabalho foi utilizada uma particularização desse estimador, representado pela seguinte expressão. Para efeitos práticos será designado ao longo do trabalho, sempre que justificável, de *Estimador MédiaNN*.

$$H(X) = \sum_{i \neq j} log \parallel x_i - x_j \parallel \tag{35}$$

Tal consiste em utilizar não apenas a mínima distância euclidiana entre os pontos de um par de imagens, mas sim ter em consideração todas as distâncias. Desta forma evita-se qualquer tipo de ordenação e/ou pesquisa do mínimo das distâncias obtidas, o que favoreceria a diminuição do peso computacional do algoritmo.

Torna-se então necessário averiguar qual o melhor algoritmo. Utilizando mais uma vez a ferramenta Matlab® é possível testar e verificar a robustez, eficiência e rapidez, em termos computacionais, de cada algoritmo.

3.2.1. O algoritmo k-Nearest Neighbors (kNN)

Observando-se que o segundo e terceiro termos da expressão referida por Kybic representam valores constantes optou-se inicialmente por não os introduzir no algoritmo testado, uma vez que aqueles não afectam as operações de maximização envolvidas no problema. Desta forma o algoritmo testado foi essencialmente o seguinte.

$$H(X) = \sum_{i=0}^{N} \log Q_i \tag{36}$$

Onde $Q_i = min_{i \neq j} \parallel x_i - x_j \parallel$ como já referido anteriormente.

Usou-se, mais uma vez, a imagem "Lena" (imagem a) da Figura 6) em duas variantes pretendidas, escala (tons) de cinzento e *RGB*. O uso de *RGB* neste contexto tem como objectivo o teste da multidimensionalidade que se pretende que o algoritmo seja capaz de suportar uma vez que é um tipo de imagem que já possuiu três dimensões dadas pelo conjunto de cada um dos seus componentes *RGB* (*Red*, *Green* e *Blue*).

Desta forma usaram-se as imagens da Figura 8 como base para testes. Ambas as imagens são do tipo *Bitmap*, sendo a imagem a) *RGB* e a imagem b) em escala de cinzentos com valores na gama de 0 a 255, ou seja, 8 bits. Optou-se por aplicar o algoritmo a imagens com diferentes tamanhos, obtidas a partir das imagens originais, não só para testar a sua robustez como também para verificar as possíveis diferenças no tempo de processamento. Assim a imagem original de formato 512x512 foi redimensionada obtendo-se então imagens com 50x50, 150x150 e 250x250 pixeis.

a)

b)

Figura 8 - Imagem a): Lena, formato RGB, tamanho 512x512 pixeis. Imagem b): Lena, formato escala de cinzentos, tamanho 512x512 pixeis.

As imagens usadas nos testes, obtidas a partir das imagens a) e b) da Figura 8, são as apresentadas de seguida.

Figura 9 - a) Imagem RGB, tamanho 50x50 pixeis; b) Imagem em escala de cinzentos, tamanho 50x50 pixeis; c) Imagem RGB, tamanho 150x150 pixeis; d) Imagem em escala de cinzentos, tamanho 150x150 pixeis; e) Imagem RGB, tamanho 250x250 pixeis; f) Imagem em escala de cinzentos, tamanho 250x250 pixeis.

3.3. Testes com kNN

Existem algoritmos *k-Nearest Neighbors* para Matlab® em grande quantidade, muitos disponibilizados em www.mathworks.com (por exemplo).

Utilizou-se um desses algoritmos [13] que, para além de fazer o cálculo das distâncias aos vizinhos de uma forma matricial bastante optimizada, faz ainda a ordenação desses valores. Cada linha da matriz de distâncias euclidianas obtida contém as distâncias ordenadas de um dado ponto aos seus vizinhos; a primeira posição (primeira coluna da matriz) contém as distâncias mínimas.

O único parâmetro que é necessário fornecer ao algoritmo, além das matrizes que representam cada uma das imagens a analisar, é o número de vizinhos que se pretende considerar.

A eficiência do algoritmo advém de a operação de cálculo da distância euclidiana entre pixeis ser efectuada de forma matricial. Mais concretamente, designando as duas imagens 'A' e 'B', a cada iteração do algoritmo, uma linha de B é replicada de modo a formar uma matriz à qual seguidamente se calcula a distância euclidiana a A. Este processo repete-se até que as distâncias euclidianas entre A e B sejam calculadas.

3.3.1. Testes iniciais

Todos os testes realizados, cujos resultados estão apresentados a seguir, foram executados usando o Matlab® R2010b 64 bits.

Estes testes destinaram-se a confirmar não só a robustez do código como também os pressupostos teóricos respeitantes ao conceitos de entropia e informação mútua.

De seguida apresenta-se o pseudo-código do algoritmo *kNN* utilizado.

$$
\begin{aligned}
&kNN(imageP, imageQ)\{\\
&\quad \boldsymbol{for}\ i = 1{:}N\\
&\qquad \boldsymbol{return}\ neighborDistances = \|\mathrm{q} - \mathrm{p}\|\\
&\quad \boldsymbol{end}\ \}
\end{aligned}
$$

Por forma a obter uma medida de Informação Mútua (IM) entre o mapa pré-existente e as imagens que vão sendo adquiridas pelo AUV, elaborou-se o algoritmo essencialmente descrito pelo seguinte pseudo-código.

$$imageA \leftarrow template(N \times D)$$
$$entropyA \leftarrow \sum^{N} \log\left(\min\left(kNN(imageA, imageA)\right)\right)$$
$$\boldsymbol{for}\ i = 1{:}\left(M_1 - (N_1 + 1)\right)$$
$$\quad \boldsymbol{for}\ j = 1{:}\left(M_2 - (N_2 + 1)\right)$$
$$\quad imageB \leftarrow partial\ matrix(N \times D)$$
$$\quad entropyB \leftarrow \sum^{N} \log\left(\min\left(kNN(imageB, imageB)\right)\right)$$
$$\quad entropyAB \leftarrow \sum^{N} \log\left(\min\left(kNN(imageA, imageB)\right)\right)$$
$$\quad mutual\ info(i,j) \leftarrow entropyA + entropyB - entropyAB$$
$$\quad \boldsymbol{end}$$
$$\boldsymbol{end}$$

O código apresentado destina-se a localizar, num mapa de dados pré-existente correspondente a uma imagem de M_1xM_2 pixeis (no caso geral multidimensional, sendo o número de dimensões D), uma porção (designada *template*), com N_1xN_2 pixeis. De notar que $N = N_1 * N_2$.

Nas aplicações de navegação de AUV em vista, tratar-se-á de dados de imagens obtidos por SSS. Por forma a melhor ilustrar a situação, a Figura 10 emprega a imagem 'Lena' já apresentada anteriormente.

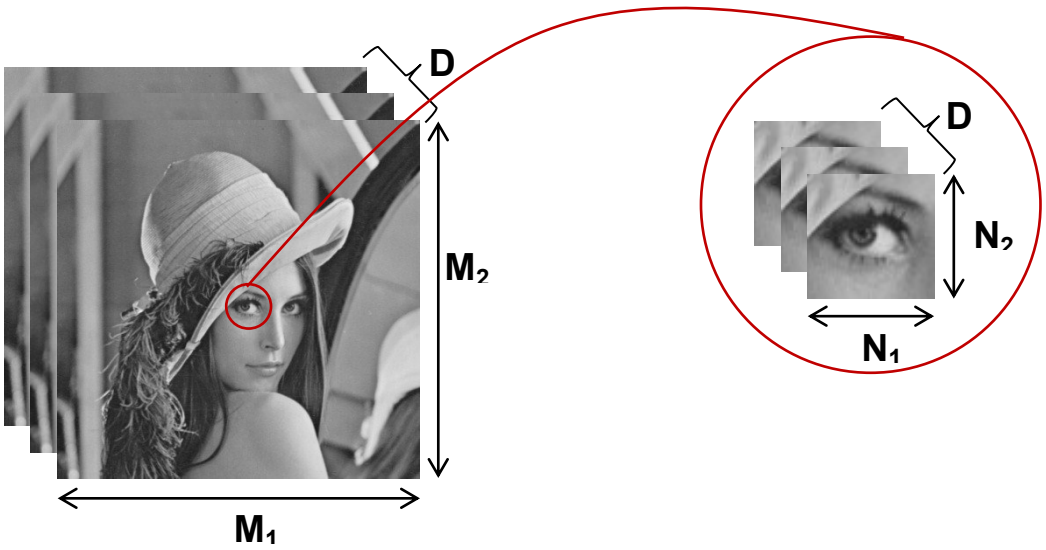


Figura 10 - Estrutura dimensional da imagem usada como mapa e do *template*

Por forma a localizar o *template* efectua-se a um varrimento do mapa, obtendo-se a cada iteração um pedaço desse mapa com as mesmas dimensões do *template*.

O varrimento do mapa é efectuado sempre ao longo das linhas da sua representação matricial. Como é descrito no pseudo-código, inicia-se no canto superior esquerdo e termina no inferior direito - vide Figura 11 e Figura 12.

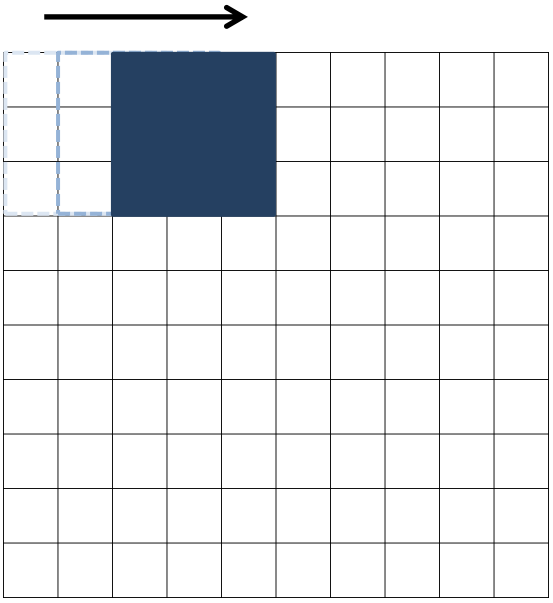


Figura 11 - Varrimento feito pixel a pixel ao longo das linhas, numa matriz de 10x10 pixeis

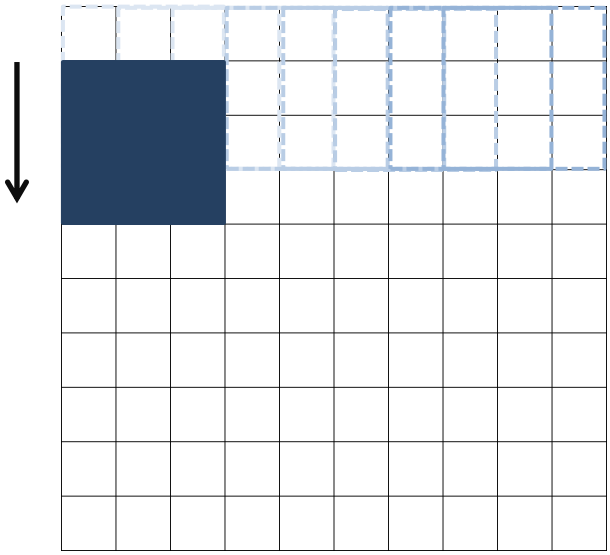


Figura 12 - Mudança de linha no varrimento. Ao chegar ao fim da linha "desce-se" um pixel e repete-se o processo.

A cada iteração, são obtidas estimativas da entropia própria da matriz parcial do mapa e da entropia conjunta entre esta e o *template*. Aliando a estes dois valores de entropia o valor da entropia própria do *template*, obtido previamente (antes de ser feito o varrimento do mapa), é então possível obter uma medida da informação mútua.

A execução do algoritmo resulta numa matriz que regista os valores de informação mútua entre o *template* e cada uma das matrizes parciais constituintes do mapa.

O último passo é encontrar nesta matriz o valor máximo de IM. As coordenadas deste valor na matriz irão corresponder às coordenadas no mapa, do pixel superior esquerdo do *template* que se pretende localizar.

Para exemplificar, observe-se a situação ilustrada na Figura 13.

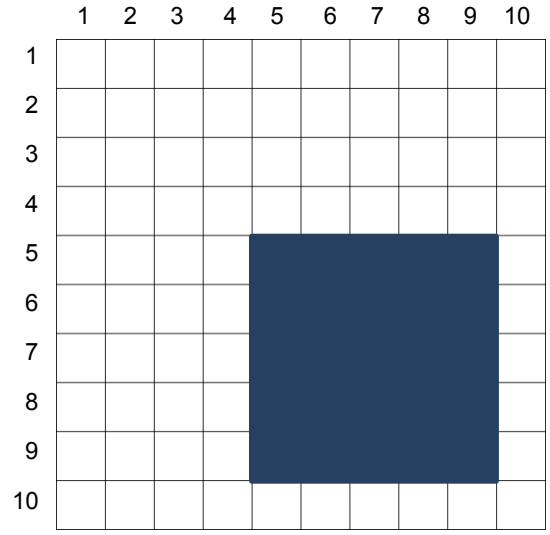


Figura 13 - Mapa com 10x10 pixeis e *template* com 5x5 pixeis.

Efectuando o varrimento da primeira linha obtêm-se 6 valores de IM (1x6).

Seguindo o mesmo raciocínio para as restantes linhas, obtém-se uma matriz, de valores de IM, de formato 6x6.

Após o varrimento, o valor máximo de IM, correspondente à localização do *template*, encontrar-se-á na posição (5,5) da matriz de IM. Estas coordenadas correspondem, como referido anteriormente, à posição no mapa do pixel do canto superior esquerdo do *template*.

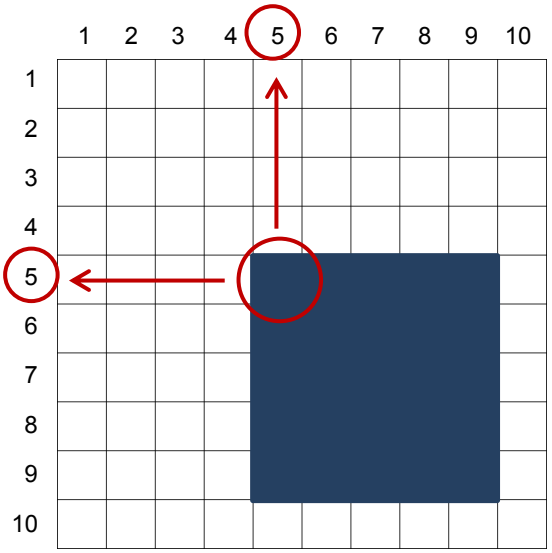


Figura 14 – Coordenadas do ponto máximo de IM, correspondentes à localização do *template* no mapa.

Os resultados dos vários testes apresentados neste capítulo devem ser interpretados tendo presente esta circunstância.

Os primeiros testes são respeitantes a imagens monocromáticas (tons de cinzento 8 bits) e *RGB* e têm como objectivo comparar o desempenho dos dois estimadores de entropia (estimador segundo método de Kybic e estimador MédiaNN).

A configuração dos testes realizados é a seguinte:

➔ Algoritmo segundo o método de Kybic aplicado a:

- Mapa *RGB* 50x50 pixeis, *template RGB* 10x10 pixeis;
- Mapa *RGB* 150x150 pixeis, *template RGB* 15x15 pixeis;
- Mapa *RGB* 250x250 pixeis, *template RGB* 25x25 pixeis;
- Mapa escala de cinzentos 50x50 pixeis, *template* escala de cinzentos 10x10 pixeis;
- Mapa escala de cinzentos 150x150 pixeis, *template* escala de cinzentos 15x15 pixeis;
- Mapa escala de cinzentos 250x250 pixeis, *template* escala de cinzentos 25x25 pixeis;

➔ Estimador MédiaNN aplicado a:

- Mapa *RGB* 50x50 pixeis, *template RGB* 10x10 pixeis;
- Mapa *RGB* 150x150 pixeis, *template RGB* 15x15 pixeis;
- Mapa *RGB* 250x250 pixeis, *template RGB* 25x25 pixeis;
- Mapa escala de cinzentos 50x50 pixeis, *template* escala de cinzentos 10x10 pixeis;
- Mapa escala de cinzentos 150x150 pixeis, *template* escala de cinzentos 15x15 pixeis;
- Mapa escala de cinzentos 250x250 pixeis, *template* escala de cinzentos 25x25 pixeis;

O *template* utilizado foi extraído de cada um dos mapas correspondentes. O tamanho de cada *template* foi escolhido sem nenhum critério específico, dado que o objectivo nesta altura do trabalho é apenas de verificar a funcionalidade dos métodos testados.

Os resultados obtidos para as diferentes configurações são apresentados

Tabela 8 - Resultados obtidos usando o método de Kybic

Imagem	coordenadas previstas (x,y)	coordenadas obtidas (x,y)	tempo (s)	tempo (min)
Mapa_RGB_50	25,25	25,25	29,6632	0,4944
Mapa_RGB_150	70,70	70,70	918,6270	15,3105
Mapa_RGB_250	120,120	120,120	11738,0000	195,6333
Mapa_EC_50	25,25	25,25	29,3378	0,4890
Mapa_EC_150	70,70	[70:71],[70:70]	880,5506	14,6758
Mapa_EC_250	120,120,	[113:120],[115:120]	10184,0000	169,7333

Tabela 9 - Resultados obtidos com estimador MédiaNN.

Imagem	coordenadas previstas (x,y)	coordenadas obtidas (x,y)	tempo (s)	tempo (min)
Mapa_RGB_50	25,25	23,27	28,3113	0,4719
Mapa_RGB_150	70,70	73,59	881,1936	14,6866
Mapa_RGB_250	120,120	124,149	13626,0000	227,1000
Mapa_EC_50	25,25	24,26	32,0020	0,5334
Mapa_EC_150	70,70	68,71	916,6455	15,2774
Mapa_EC_250	120,120	113,119	11345,0000	189,0833

Analisando a Tabela 8 verifica-se que o método proposto por Kybic revela-se igualmente exacto e preciso para o caso de imagens multidimensionais, pois as coordenadas obtidas são exactamente iguais às previstas. No caso das imagens em escala de cinzento os resultados foram igualmente bons para o mapa de 50x50 pixeis. Para os restantes dois mapas não foi encontrado um par de coordenadas único, mas sim um conjunto de pontos em redor das coordenadas previstas. Isto revela que estes resultados, não sendo precisos, apresentam alguma exactidão.

Os resultados obtidos sugerem que a utilização de imagens multidimensionais permite obter de resultados mais exactos.

Foi também verificado que no caso de imagens com tamanhos maiores (250x250 pixeis) os requisitos computacionais aumentam bastante. O tempo despendido na execução do programa na, plataforma computacional usada, passou de menos de ½ minuto para mais de 3 horas. Entre os dois tipos de imagens utilizadas não houve diferenças muito relevantes a nível de desempenho.

A Figura 15 ilustra os resultados apresentados na Tabela 8.

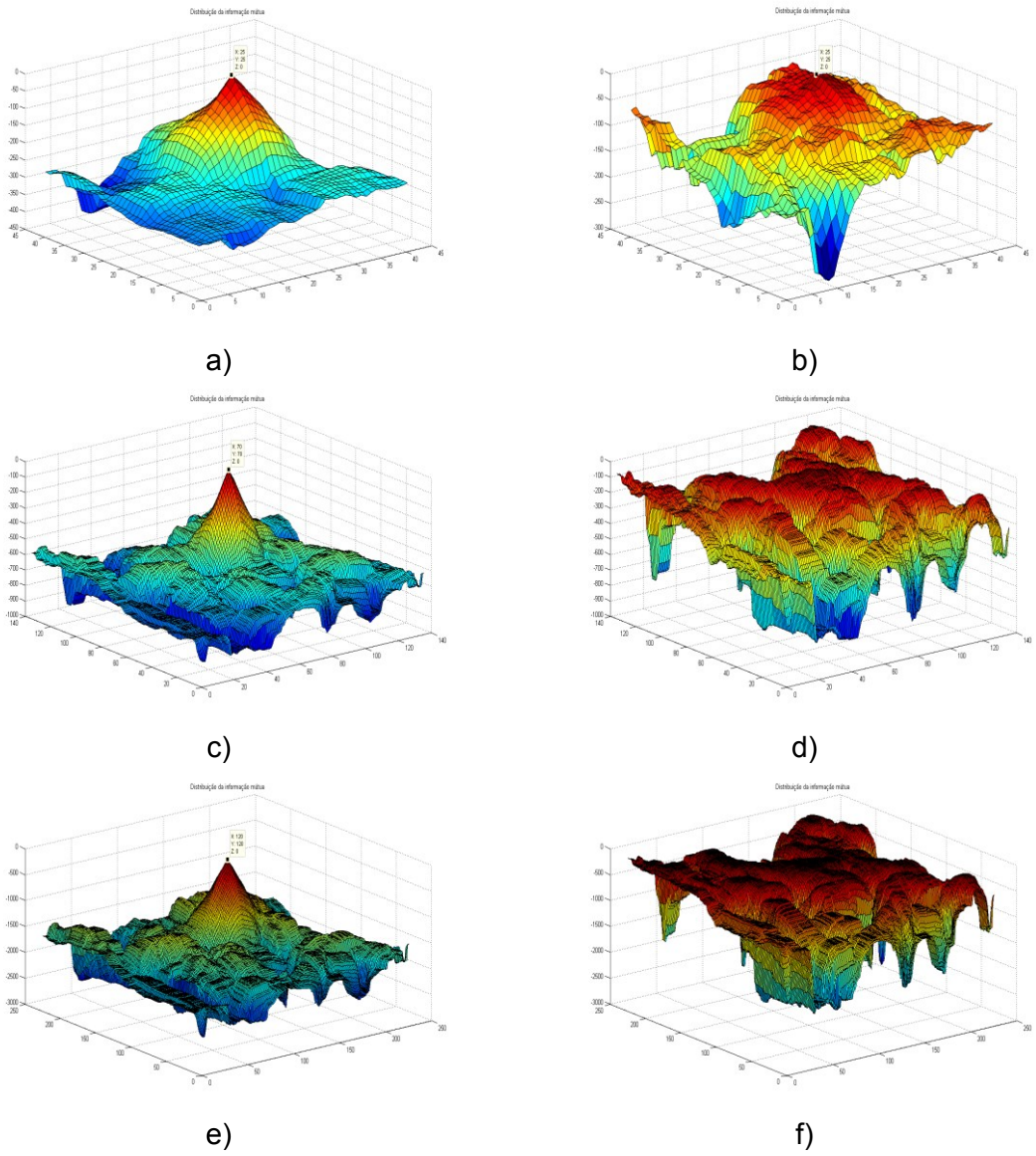


Figura 15 – Visualização tridimensional da IM obtida após o *matching*, o valor máximo corresponde à coordenada encontrada. Correspondência com a Tabela 8 é a seguinte: a) Mapa_RGB_50; b) Mapa_EC_50; c) Mapa_RGB_150; d) Mapa_EC_150; e) Mapa_RGB_250; f) Mapa_EC_250.

Analisando os resultados da Tabela 9, observa-se que o método obtém resultados próximos do esperado e muito semelhantes quer usando imagens *RGB*, quer usando imagens em escala de cinzentos. Apresenta no entanto uma exactidão bastante menor que os resultados da Tabela 8.

Em termo de execução do programa, se no caso de imagens com tamanho 50x50 e 150x150 pixeis o tempo despendido é semelhante, aplicando quer o método de Kybic quer o estimador MédiaNN, tal não se verifica no caso das imagens 250x250 pixeis onde este último método demora, aproximadamente, entre 12% e 15% mais que o método de Kybic.

A Figura 16 ilustra os resultados apresentados na Tabela 9.

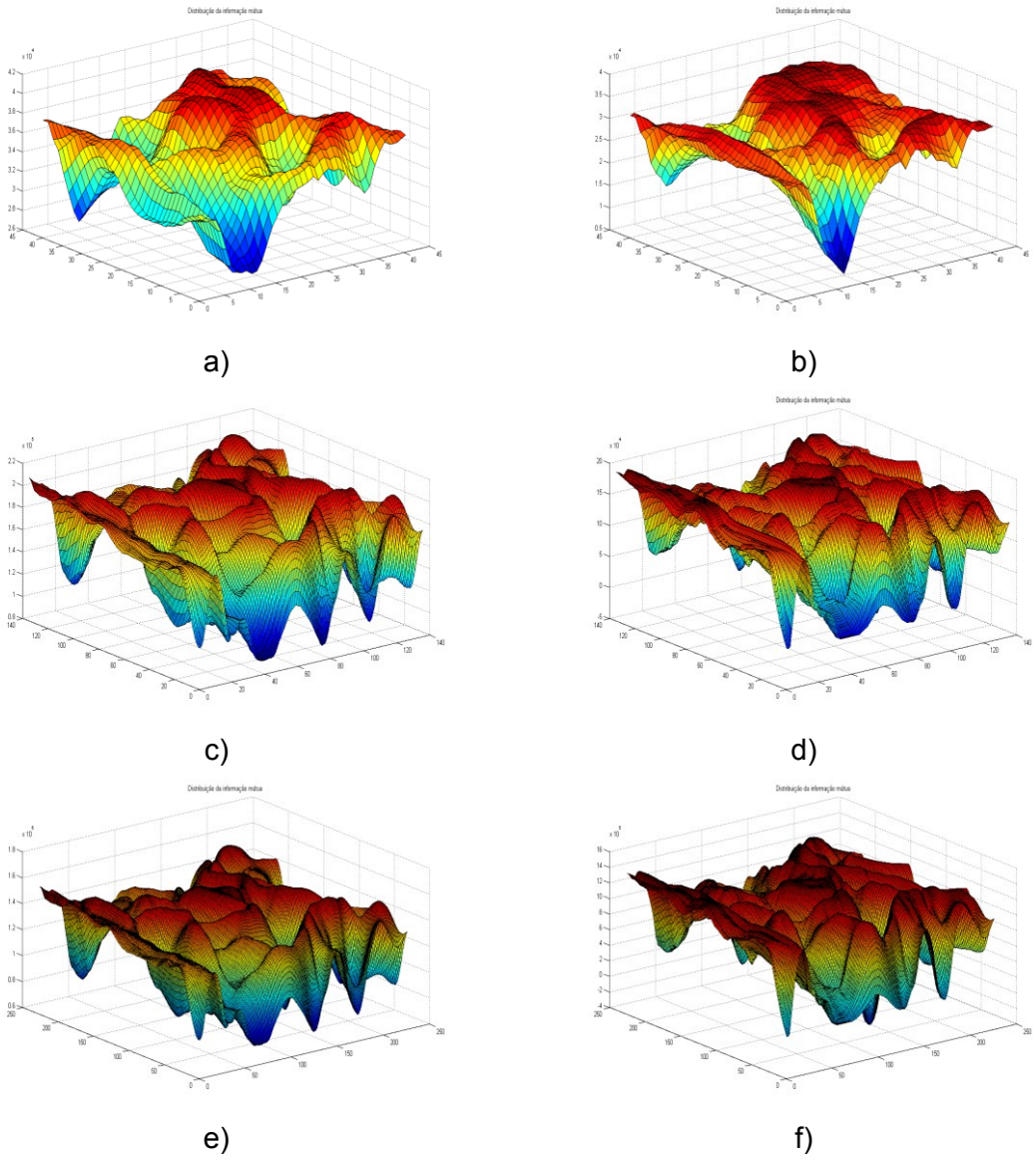


Figura 16 - Visualização tridimensional da IM obtida após o *matching*, o valor máximo corresponde à coordenada encontrada. Correspondência com a Tabela 8 é a seguinte: a) Mapa_RGB_50; b) Mapa_EC_50; c) Mapa_RGB_150; d) Mapa_EC_150; e) Mapa_RGB_250; f) Mapa_EC_250.

3.3.2. Testes com e sem ordenação

O conjunto de testes seguinte teve como objectivo averiguar qual o impacto no método de Kybic da operação de ordenação efectuada no cálculo dos vectores de distâncias aos *k-Nearest Neighbors*, que está integrada no algoritmo kNN. De referir que os mapas utilizados e os respectivos *templates* são os mesmos que nos testes realizados anteriormente.

O método de Kybic requer a determinação das distâncias mínimas de cada ponto; para tal, podem adoptar-se dois procedimentos funcionalmente equivalentes.

➔ Ordenar todos os valores das distâncias e tomar o primeiro;

➔ Pesquisar o mínimo de cada vector de distâncias, sem ordenação prévia.

Ambos foram testados usando imagens *RGB* e em escala de cinzentos. Os resultados encontram-se na Tabela 10 (1º caso), e Tabela 11 (2º caso).

Foi também testado o estimador MédiaNN, que considera todas as distâncias pelo que dispensa operações de ordenação e/ou de pesquisa de mínimo. Os resultados deste teste encontram-se na Tabela 12.

Tabela 10 - Resultados obtidos usando o método proposto por Kybic sem ordenação dos valores das distâncias entre vizinhos e fazendo a pesquisa do valor da distância mínima.

Imagem	coordenadas previstas (x,y)	coordenadas obtidas (x,y)	tempo (s)	tempo (min.)
Mapa_RGB_50	25,25	25,25	24,7972	0,4133
Mapa_RGB_150	70,70	70,70	626,8847	10,4481
Mapa_RGB_250	120,120	120,120	6840,9000	114,0150
Mapa_EC_50	25,25	25,25	21,7319	0,3622
Mapa_EC_150	70,70	[70:71],[70:70]	623,0599	10,3843
Mapa_EC_250	120,120,	[113:120],[115:120]	6721,6000	112,0267

Tabela 11 - Resultados obtidos usando o método proposto por Kybic efectuando ordenação dos valores das distâncias.

Imagem	coordenadas previstas (x,y)	coordenadas obtidas (x,y)	tempo (s)	tempo (min.)
Mapa_RGB_50	25,25	25,25	27,3307	0,4555
Mapa_RGB_150	70,70	70,70	840,6205	14,0103
Mapa_RGB_250	120,120	120,120	10636,0000	177,2667
Mapa_EC_50	25,25	25,25	27,5396	0,4590
Mapa_EC_150	70,70	[70:71],[70:70]	846,6321	14,1105
Mapa_EC_250	120,120	[113:120],[115:120]	10811,0000	180,1833

Tabela 12 - Resultados obtidos usando o estimador MédiaNN (que dispensa a ordenação dos valores das distâncias).

Imagem	coordenadas previstas (x,y)	coordenadas obtidas (x,y)	tempo (s)	tempo (minutos)
Mapa_RGB_50	25,25	23,27	22,3598	0,3727
Mapa_RGB_150	70,70	73,59	703,2135	11,7202
Mapa_RGB_250	120,120	124,149	7351,1000	122,5183
Mapa_EC_50	25,25	36,28	24,3901	0,4065
Mapa_EC_150	70,70	73,59	694,2556	11,5709
Mapa_EC_250	120,120	124,149	7150,7000	119,1783

Analisando os resultados, verifica-se que as coordenadas obtidas e registadas na Tabela 10 e Tabela 11 são idênticas às da Tabela 8. Tal era expectável uma vez que o algoritmo testado é o mesmo, pelo que as conclusões são igualmente as mesmas.

Em termos de tempo de execução, comparando os valores da Tabela 8 com os da Tabela 10 é possível observar uma redução de 15% a 25% para imagens de 50x50 pixeis, 33% para imagens 150x150 pixeis, e 33% a 40% para imagens com 250x250 pixeis. Para esta redução contribuiu o facto de os resultados da Tabela 8 serem obtidos utilizando um algoritmo que fazia não só a ordenação de todos os valores das distâncias, como também a pesquisa do valor mínimo.

Comparando os valores temporais da Tabela 11 com os da Tabela 8 verificam-se algumas diferenças, principalmente no caso das imagens com 250x250 pixeis. Eram expectáveis valores muito semelhantes, uma vez que a única diferença é não ser feita a pesquisa do valor mínimo das distâncias. Contudo foi ainda assim possível verificar que a operação de ordenação impõe um peso computacional indesejável, pelo que o melhor método consiste em efectuar a pesquisa do valor mínimo.

Analisando agora os resultados da Tabela 12, e comparando-os com a Tabela 9, verifica-se que as coordenadas obtidas utilizando imagens *RGB* são idênticas nas duas tabelas, não acontecendo o mesmo para imagens em escala de cinzentos. Contudo em ambos os casos os resultados não são exactos, colocando novamente em evidência a eficácia do método proposto por Kybic.

O desempenho computacional é, como esperado, semelhante ao registado na Tabela 10 uma vez que a operação de ordenação também não é utilizada.

3.3.3. Testes com *template* de tamanho fixo

Nos testes descritos até agora, as dimensões de cada *template* foram atribuídas tendo em conta as dimensões do mapa de onde eram retirados.

Como referido nos primeiros testes realizados, para as imagens de dimensão 50x50 pixeis foi usado um *template* de 10x10; no caso das imagens com 150x150 pixeis o *template* usado tinha 15x15; por último, para imagens com 250x250 pixeis usou-se um *template* com tamanho 25x25.

As imagens utilizadas como mapa e os respectivos *templates* estão ilustrados na Figura 17 para imagens *RGB*, e na Figura 18 para imagens em escala de cinzentos.

Figura 17 – a) Mapa *RGB* 50x50 pixeis; b) *template* 10x10 pixeis; c) Mapa *RGB* 150x150 pixeis; d) *template* 15x15 pixeis; e) Mapa *RGB* 250x250 pixeis; f) *template* 25x25 pixeis.

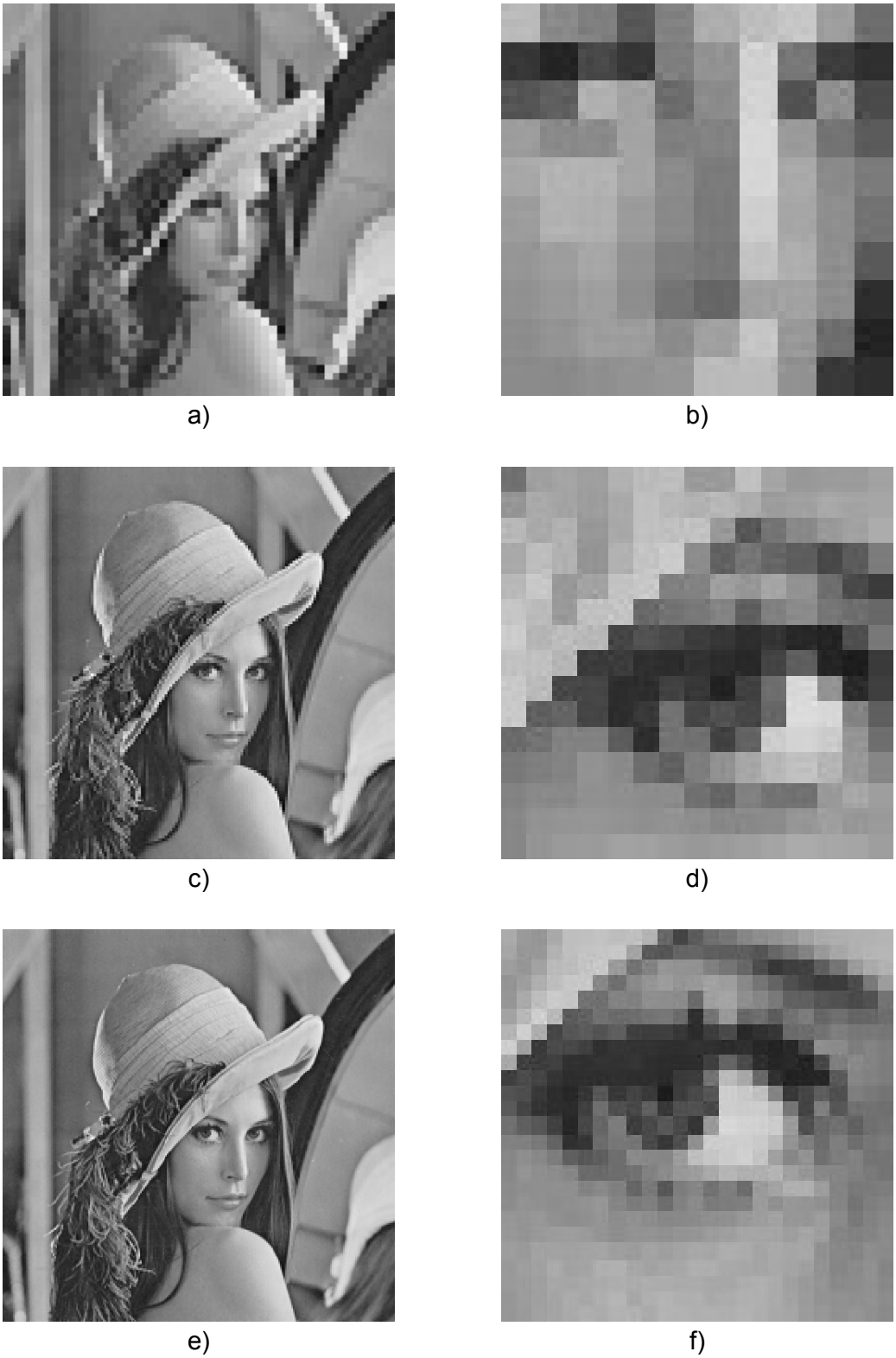


Figura 18 - a) Mapa escala de cinzentos 50x50 pixeis; b) *template* 10x10 pixeis; c) Mapa escala de cinzentos 150x150 pixeis; d) *template* 15x15 pixeis; e) Mapa escala de cinzentos 250x250 pixeis; f) *template* 25x25 pixeis.

Na aplicação ao caso real não irão ser usados tamanhos diferentes para o *template* obtido no momento. Na situação real será definida um tamanho fixo para a janela de aquisição de dados correspondente ao tamanho da imagem obtida pelo SSS ao fim de um determinado tempo de aquisição. Este tempo de aquisição e, por consequência, as dimensões da imagem obtida são definidos tendo em consideração factores como o tipo de missão, o local onde esta se realiza e as características do mapa (já conhecido) do local.

Assim, efectuou-se um novo conjunto de testes semelhantes aos anteriores mas adoptando um *template* de dimensões fixas (25x25 pixeis) e usando apenas os mapas de 150x150 pixeis e 250x250 pixeis. Os *templates* utilizados encontram-se ilustrados na Figura 19. Os resultados dos testes realizados estão apresentados na Tabela 13 e na Tabela 14.

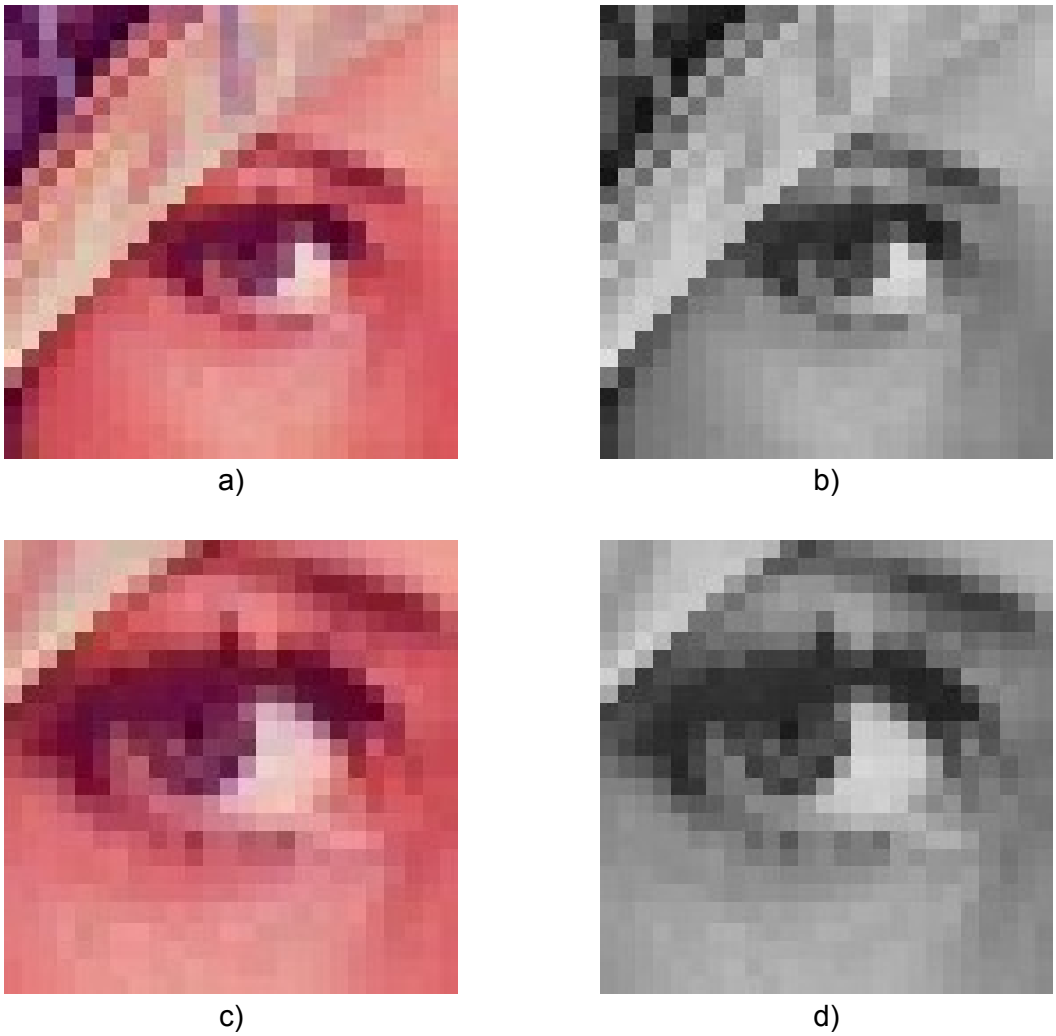


Figura 19 – a) *template* 25x25 pixeis obtido do mapa *RGB* 150x150 pixeis; b) *template* 25x25 pixeis obtido do mapa em escala de cinzentos 150x150 pixeis; c) *template* 25x25 pixeis obtido do mapa *RGB* 250x250 pixeis; d) *template* 25x25 pixeis obtido do mapa em escala de cinzentos 250x250 pixeis.

Tabela 13 - Resultados obtidos aplicando o método de Kybic (com e sem ordenação) usando *template* com tamanho fixo.

Imagem	coordenadas previstas (x,y)	coordenadas obtidas (x,y)	tempo (min) sem ordenação	tempo (min) com ordenação
Mapa_RGB_150	65,65	65,65	34,5300	51,6650
Mapa_RGB_250	120,120	120,120	86,9283	170,5167
Mapa_EC_150	65,65	[63:65],[62:65]	27,9567	53,5117
Mapa_EC_250	120,120	[113:120],[115:120]	88,3883	169,8333

Tabela 14 – Resultados obtidos com o estimador MédiaNN (que dispensa a operação de ordenação).

Imagem	coordenadas previstas (x,y)	coordenadas obtidas (x,y)	tempo (s)	tempo (minutos)
Mapa_RGB_150	65,65	68,58	1999,5000	33,3250
Mapa_RGB_250	120,120	124,149	6265,3000	104,4217
Mapa_EC_150	65,65	68,58	2011,6000	33,5267
Mapa_EC_250	120,120	124,149	6277,6000	104,6267

De novo o método proposto por Kybic para estimar entropia revela-se o mais eficaz. No caso multidimensional (imagens *RGB*) permite obter correctamente as coordenadas do *template* considerado, algo que usando o estimador MédiaNN não aconteceu. Mesmo no caso unidimensional (imagens em escala de cinzentos) o método proposto por Kybic revelou-se melhor, pois apesar de não ter identificado com precisão as coordenadas esperadas encontrou um conjunto de pontos situados em torno destas (e que as inclui). Estes resultados estão ilustrados na Figura 20 e na Figura 21.

Em termos de desempenho verifica-se que usando a ordenação o tempo de execução duplica. A excepção é o teste com o mapa *RGB* de 150x150 pixeis onde o aumento do tempo de execução é sensivelmente 45%.

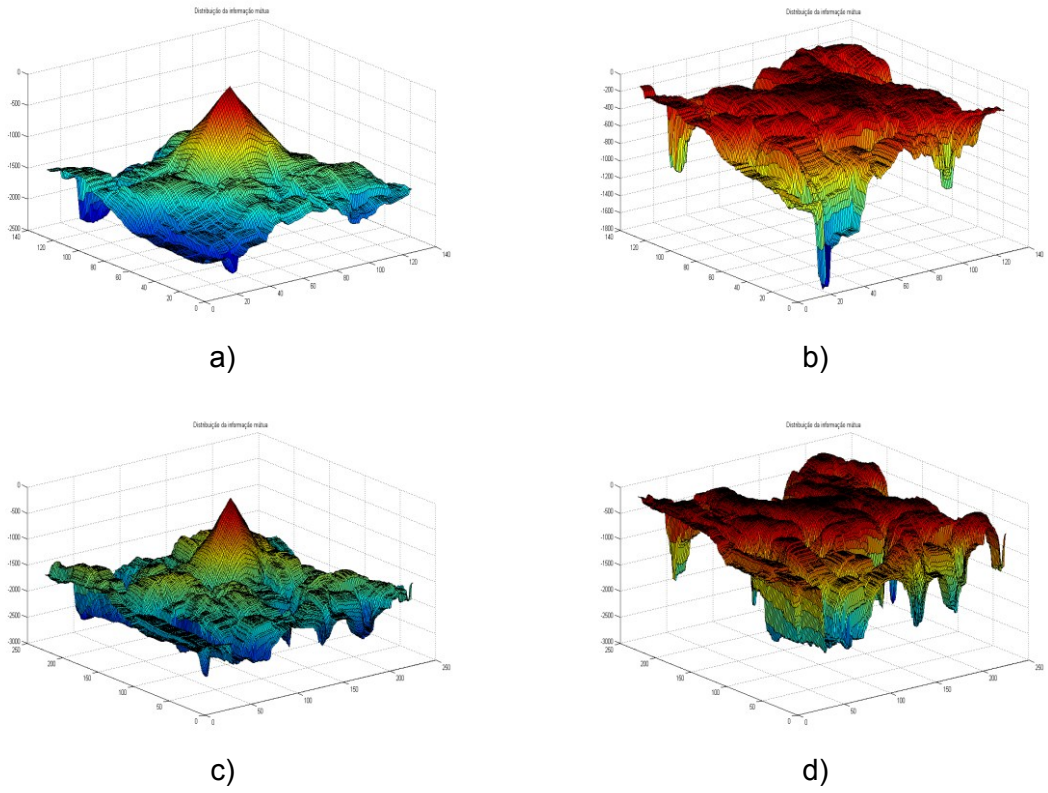
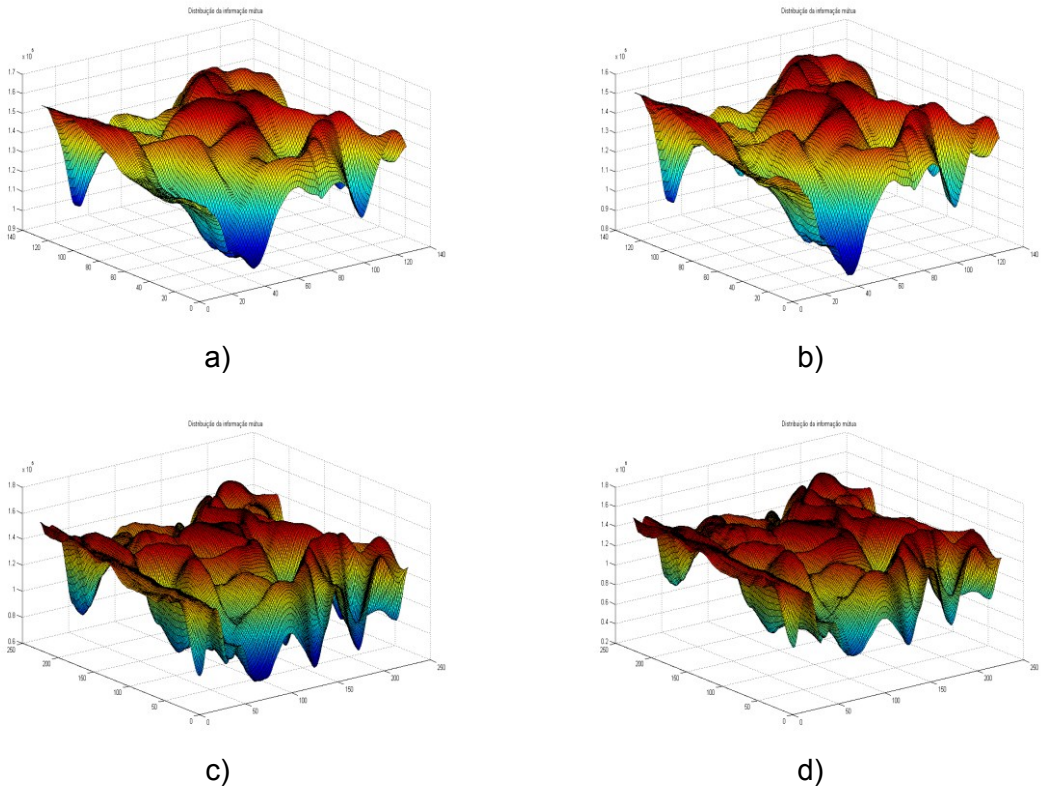


Figura 20 – Ilustração dos resultados da Tabela 13 a) mapa *RGB* 150x150 pixeis; b) mapa em escala de cinzentos 150x150 pixeis; c) mapa *RGB* 250x250 pixeis; d) mapa escala de cinzentos 250x250 pixeis.



a) b)

c) d)

Figura 21 – Ilustração dos resultados da Tabela 14 a) mapa *RGB* 150x150 pixeis; b) mapa em escala de cinzentos 150x150 pixeis; c) mapa *RGB* 250x250 pixeis; d) mapa em escala de cinzentos 250x250 pixeis.

Em resumo, os testes realizados indicam que o método com o estimador MédiaNN é menos eficiente que o de Kybic, quer funcionalmente quer computacionalmente.

Por outro lado, a operação de ordenação no algoritmo kNN pode ser suprimida, uma vez que não oferece qualquer vantagem computacional para o método de Kybic (pois este usa apenas os valores mínimos de distância, que podem ser obtidos por pesquisa directa).

Desta forma, passou-se a utilizar apenas o método proposto por Kybic, descrito pela equação (36), sem ordenação no algoritmo kNN.

3.3.4. Aplicação da Parallel Computing Toolbox™ do Matlab®

Devido à complexidade computacional dos algoritmos atrás apresentados, e aos longos tempos de execução que dela resultam, torna-se necessário encontrar ferramentas que possibilitem acelerar a execução.

Optou-se por utilizar a ferramenta Parallel Computing Toolbox™, presente no Matlab® desde a versão 3.3 (R2008a) com origem na Distributed Computing Toolbox™ existente desde a versão 3.0 (R2006b). Neste trabalho foi utilizada a versão 5.0 (R2010b).

A Parallel Computing Toolbox™ permite dividir a carga computacional de uma sessão Matlab® por várias sessões, chamadas de 'trabalhadores' (*workers*) operando em

paralelo. É possível usar até oito 'trabalhadores' na máquina local. Usando um processador *multicore* é possível utilizar um 'trabalhador' por cada *core*. É importante dispor de suficiente memória RAM, pois cada 'trabalhador' requer sensivelmente o mesmo que uma simples sessão de Matlab.

Apesar de esta 'toolbox' possuir instruções específicas para fazer processamento paralelo, elas não foram utilizadas. Para além de a paralelização de algoritmos não ser o foco principal deste trabalho, verificou-se também que o simples facto de serem activados oito 'trabalhadores' locais era suficiente para se ganhar uma redução significativa no tempo de execução.

Todos os testes a partir deste ponto foram efectuados usando a Parallel Computing Tool Box™ do Matlab.

3.3.5. Testes com Parallel Computing Toolbox™

Usando a Parallel Computing Toolbox™ repetiram-se os testes com o método de Kybic sem ordenação descritos na secção 3.3.2 (vide Tabela 10). Os resultados diferem apenas no tempo de execução, como evidencia a Tabela 15.

Tabela 15 – Aceleração do método proposto por Kybic com a Parallel Computing Toolbox™ (PCT).

Imagem	coordenadas previstas (x,y)	coordenadas obtidas (x,y)	tempo s/ PCT (min)	tempo c/ PCT (min)
Mapa_RGB_50	25,25	25,25	0,4133	0,2212
Mapa_RGB_150	70,70	70,70	10,4481	8,1225
Mapa_RGB_250	120,120	120,120	114,0150	90,4083
Mapa_EC_50	25,25	25,25	0,3622	0,2661
Mapa_EC_150	70,70	[70:71],[70:70]	10,3843	8,0330
Mapa_EC_250	120,120,	[113:120],[115:120]	112,0267	81,3483

Analisando a Tabela 15, observam-se reduções no tempo de execução na ordem dos 25% a 40%.

4. *Haralick Features*

4.1. Testes iniciais com *Haralick Features*

Como o método de localização aqui utilizado se baseia no princípio de encontrar o máximo de informação mútua entre duas imagens, torna-se óbvio que quanto mais informação de cada imagem puder ser tida em conta, mais eficaz será a sua aplicação. As *Haralick Features* surgem então como uma resposta a esta necessidade de "mais informação".

Uma vantagem das *Haralick Features* é o facto de serem sensíveis à orientação da imagem uma vez que são passíveis de ser calculadas segundo 4 orientações possíveis, 0^o, 45^o, 90^o e/ou 135^o.

Como sugerido em [9] decidiu-se usar as *features* "Energia" e "Contraste" e as 4 orientações possíveis. Explorou-se também, como alternativa, a utilização das médias dos valores de Energia e Contraste.

Temos então duas situações distintas a testar. Por um lado, o caso em que se utiliza uma imagem em escala de cinzentos e as *features* Energia e Contraste, cada *uma* obtida segundo cada uma das quatro orientações já referidas, obtendo-se, desta forma, dados de imagem com dimensionalidade 9. Por outro lado, o uso dos dados de imagem apenas com dimensionalidade 3, resultantes de não serem utilizadas 4 matrizes para cada *feature* (uma para cada orientação), mas sim uma única resultante da média aritmética dos valores dessas 4 matrizes.

Note-se que o cálculo das *Haralick Features* se aplica quer à imagem (*template*) que se pretende localizar no mapa, quer às porções desse mapa com que se pretende compará-la.

O cálculo compreende dois passos: obtenção da matriz de co-ocorrência, denominada SGLD (*Spacial Grey-Level Dependence Matrix*); aplicação a essa matriz das equações correspondentes às *features* pretendidas (no caso, Energia e Contraste).

As equações correspondentes à Energia e Contraste são as que se seguem, onde $p(i,j)$ representa a matriz SGLD normalizada:

$$Energia = \sum_i \sum_j p(i,j)^2 \tag{37}$$

$$Contraste = \sum_i \sum_j (i-j)^2 * p(i,j) \tag{38}$$

O uso da matriz SGLD baseia-se na premissa de que a variação espacial dos tons de cinzento contém a informação de textura da imagem. Essa informação de textura é especificada pela frequência com que cada variação de nível de cinzento ocorre entre dois pixeis vizinhos nessa mesma imagem. A relação de vizinhança é estabelecida pela orientação angular e pelo parâmetro *d* (*delta*) que indica a distância ao pixel vizinho, como ilustra a Figura 22.

É o facto de a matriz SGLD ser também função da relação angular entre pixeis que permite que as *Haralick Features* sejam sensíveis a mudanças de orientação da imagem.

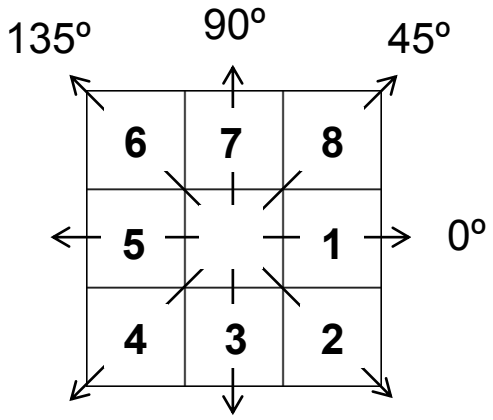


Figura 22 - Relação angular e de vizinhança entre pixeis da matriz SGLD, *d*=1. Para 0º as células 1 e 5 são vizinhas. Para 45º as células 4 e 8 são vizinhas. Para 90º as células 3 e 7 são vizinhas. Para 135º as células 2 e 6 são vizinhas.

É necessário ter em especial consideração, no código computacional [14] para obtenção das várias *features* os pixeis localizados perto dos limites da imagem. Numa primeira abordagem ao problema foi adicionada uma moldura com um tom de cinzento arbitrário (entre 0 e 255) à imagem que se pretende processar. Optou-se por escolher o tom intermédio, 128, com a intenção de reduzir o impacto negativo que poderia trazer para o cálculo a adição de pixeis não pertencentes à imagem. O tamanho da moldura foi definido de acordo com o valor especificado para *d*.

Nestas condições, foi realizado um conjunto de testes análogos aos realizados anteriormente. Note-se que não foi necessária qualquer adaptação do algoritmo kNN para estimação de entropia pois ele está preparado para cálculos num espaço multidimensional. (Os testes realizados com imagens *RGB* destinaram-se justamente a testar o seu comportamento nessa situação).

Testaram-se então duas imagens em tons de cinzento, considerando diferentes tamanhos (novamente 50x50, 150x150 e 250x250 pixeis),e diferentes distâncias entre pixeis vizinhos (aquando do cálculo da SGLD): *d*=1 e *d*=5. Os resultados obtidos são apresentados nas tabelas Tabela 16 e Tabela 17. De notar que nas tabelas a designação '9D' representa dados de imagem com dimensionalidade 9 e '3D' dados de imagem com dimensionalidade 3 (de acordo com o referido anteriormente).

Tabela 16 – Teste de *matching* baseado em *Haralick features* usando d=1.

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
Mapa_9D_50	20,20	21,20	11,3672	1,2623	51,3103
Mapa_9D_150	70,70	[71:71],[69:70]	102,8848	0,9951	554,6650
Mapa_9D_250	120,120	121,120	302,0397	2,5354	7205,5000
Mapa_3D_50	20,20	[21:21],[20:21]	12,0774	1,4006	41,7262
Mapa_3D_150	70,70	69,70	101,0272	1,0306	465,3833
Mapa_3D_250	120,120	121,120	264,9462	2,5853	5709,7000

Tabela 17 - Teste de *matching* baseado em *Haralick features* usando d=5

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
Mapa_9D_50	20,20	24,22	12,4366	1,4596	50,7715
Mapa_9D_150	70,70	[72:73],[71:71]	103,0859	0,9482	518,0045
Mapa_9D_250	120,120	122,121	280,1535	2,8341	7382,8000
Mapa_3D_50	20,20	34,01	12,3104	1,4279	40,2133
Mapa_3D_150	70,70	7,136	104,0690	1,0353	435,9782
Mapa_3D_250	120,120	1,226	309,9336	2,8602	5623,7000

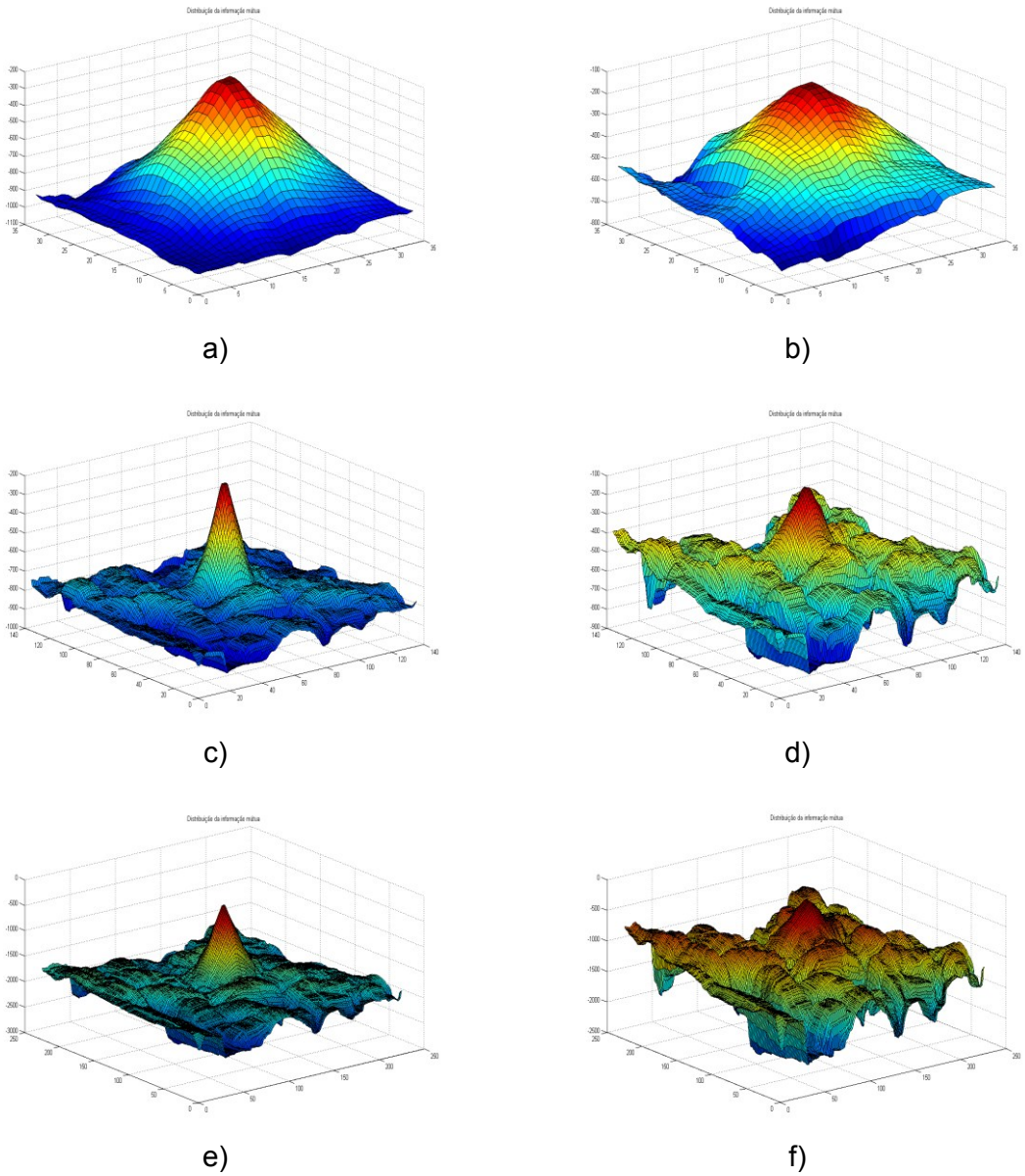


Figura 23 – Ilustração dos resultados descritos na Tabela 16.

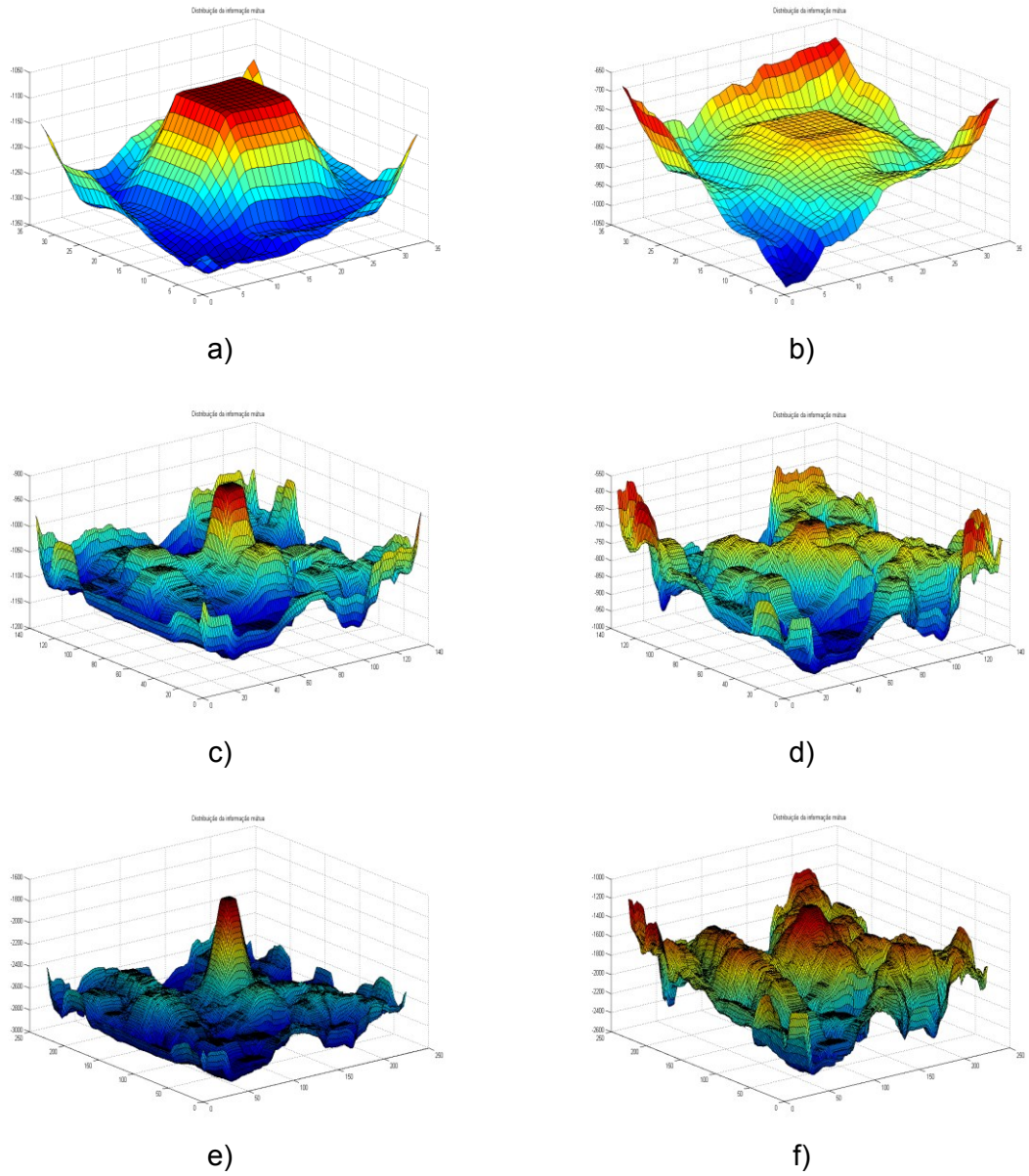


Figura 24 – Ilustração dos resultados descritos na Tabela 17.

Com *d*=1, os resultados foram bastante próximos do que era pretendido. De notar que em alguns casos não foi encontrado o ponto na coordenada prevista mas sim um pequeno conjunto de pontos em torno daquele pretendido.

Um ponto a realçar é o facto de o pré-processamento (cálculo das *Haralick features*) do mapa atrasar ainda mais a execução do programa, que já de si era cada vez mais lenta à medida que o tamanho das imagens aumentava. É observável também que usando a média das *features* o peso computacional diminui sem, aparentemente, comprometer os resultados.

Com *d*=5, os resultados degradam-se, de forma especialmente dramática, quando foram usados os dados baseados na média das *features* (3D).

Em relação ao tempo de execução do programa, não se verificou nenhuma alteração relevante, como aliás era de prever.

4.2. Testes com *Haralick Features* sem usar moldura de pixeis

Perante os resultados obtidos nos testes anteriores ponderou-se qual seria o impacto da adição da moldura na imagem original. De facto, a adição desta moldura constitui uma introdução de ruído que se esperaria que não tivesse um impacto muito significativo mas que, no futuro, aquando da realização de testes com imagens de SSS, poderá ter uma influência de tal modo negativa que a sua utilização não seja de todo aceitável.

Optou-se portanto, uma abordagem alternativa: descartar os pixeis localizados na periferia da imagem têm de ser descartados. Uma consequência do descarte destes pixeis é a diminuição do tamanho da imagem usada como mapa. Considerando um mapa de 150x150 pixeis, se o *delta* utilizado for *d*=1, após o cálculo das *Haralick Features* é descartado um pixel da periferia, ficando o mapa com um formato 148x148. Tal operação, aquando da aquisição do *template*, podia resultar numa perda de informação necessária para o *matching*, por força do reduzido tamanho deste. Assim, ao ser seleccionado o *template*, são também seleccionados pixeis periféricos extra, constituindo uma situação semelhante a uma moldura que vai salvaguardar a informação nele presente. Considerando, por exemplo, que se pretende localizar um *template* com 25x25 pixeis na situação de *d*=1, o cálculo das *Haralick features* vai ser efectuado sobre uma imagem de 27x27 pixeis, pois são seleccionados pixeis extra para esse efeito. Após a obtenção das *features*, o *template* utilizado tem de novo 25x25 pixeis devido ao descarte dos pixeis da periferia, tal como acontece no processamento do mapa. A Figura 25 ilustra esta situação.

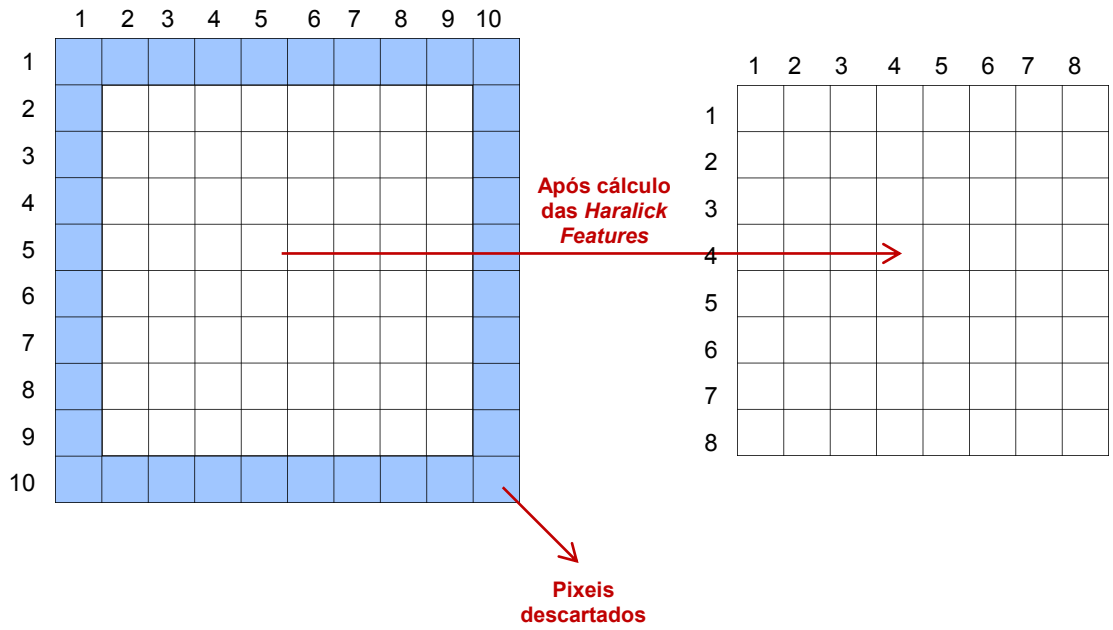


Figura 25 - Exemplo (para *d*=1) de descarte de pixeis periféricos, num mapa de 10x10, após o cálculo das *Haralick Features*. O resultado é um mapa de 8x8 pixeis.

Esta situação faz com que as coordenadas inicialmente previstas sejam afectadas pela operação de descarte dos pixeis periféricos do mapa. Assim, se as coordenadas previstas forem (20,20), tendo sido usado *d*=1, as coordenadas obtidas (se o *matching* for exacto) serão (19,19), uma vez que o descarte de um pixel do mapa faz com que as coordenadas sejam todas reduzidas de um valor.

Um exemplo claro desta situação é o descarte do pixel na coordenada (1,1). Isto faz com que o pixel que estava previamente localizado em (2,2) passe a localizar-se em (1,1), ou seja, a origem do referencial de coordenadas é modificado de acordo com o delta utilizado. Este raciocínio reflecte o que acontece aos restantes pixeis após a actualização de todas as coordenadas. A Figura 26 serve de suporte visual a esta explicação.

Perante esta situação, os resultados apresentados, a partir deste ponto do trabalho, serão todos referidos às coordenadas originais.

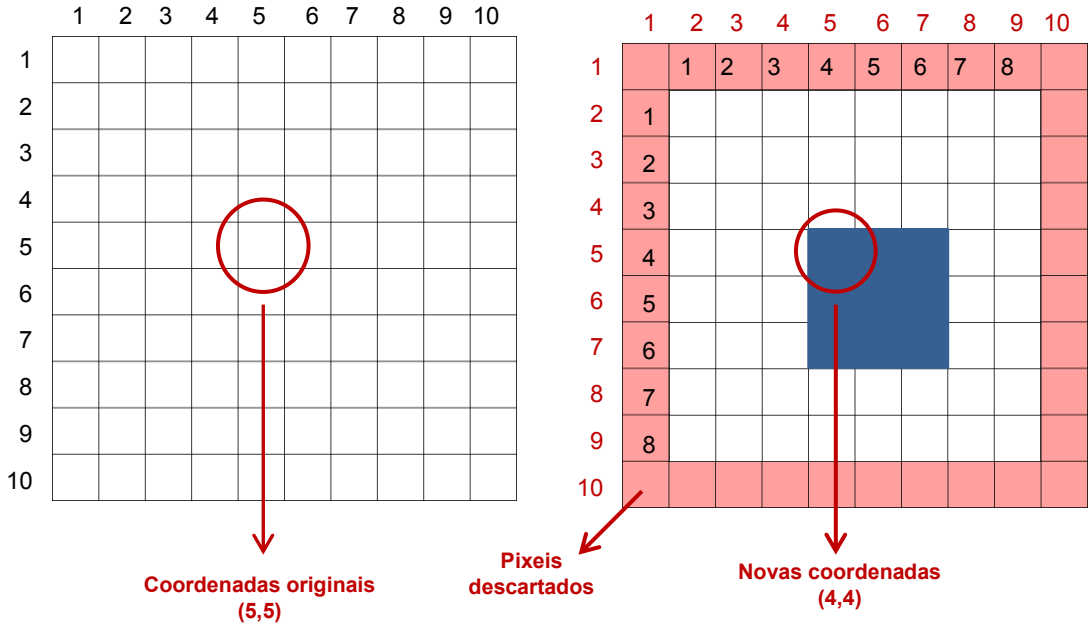


Figura 26 - Exemplo do processo de mudança das coordenadas do *template* devido ao descarte de pixeis do mapa. Situação para o caso de *d*=1.

Além de *d*=1 e *d*=5, testou-se também o caso de um *delta* maior, tendo-se optado por *d*=10. Os resultados obtidos foram os observados nas tabelas Tabela 18, Tabela 19 e Tabela 20. Neste conjunto de testes foram usados *templates* de tamanho 15x15 pixeis para mapas com 50x50 e 150x150 pixeis, e *templates* de tamanho 25x25 pixeis para mapas com 250x250 pixeis.

Tabela 18 – *Matching* baseado em *Haralick features* sem moldura e com *d*=1.

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
Mapa_9D_50	20,20	20,20	8,6137	1,2030	40,7551
Mapa_9D_150	70,70	70,70	90,2688	0,9557	521,5986
Mapa_9D_250	120,120	120,121	238,7417	2,5297	6863,4000
Mapa_3D_50	20,20	21,19	8,9638	1,2173	37,2718
Mapa_3D_150	70,70	70,70	87,5508	0,9252	450,1696
Mapa_3D_250	120,120	117,120	265,1130	2,6030	5412,6000

Tabela 19 - *Matching* baseado em *Haralick features* sem moldura e com *d*=5.

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
Mapa_9D_50	20,20	20,20	6,8853	1,2779	24,8588
Mapa_9D_150	70,70	70,70	84,0821	0,9788	463,7122
Mapa_9D_250	120,120	120,120	240,2907	3,0219	6440,3000
Mapa_3D_50	20,20	20,19	8,0448	1,4851	19,4347
Mapa_3D_150	70,70	70,70	83,0646	1,0360	383,9003
Mapa_3D_250	120,120	118,119	267,9544	2,7756	4828,6000

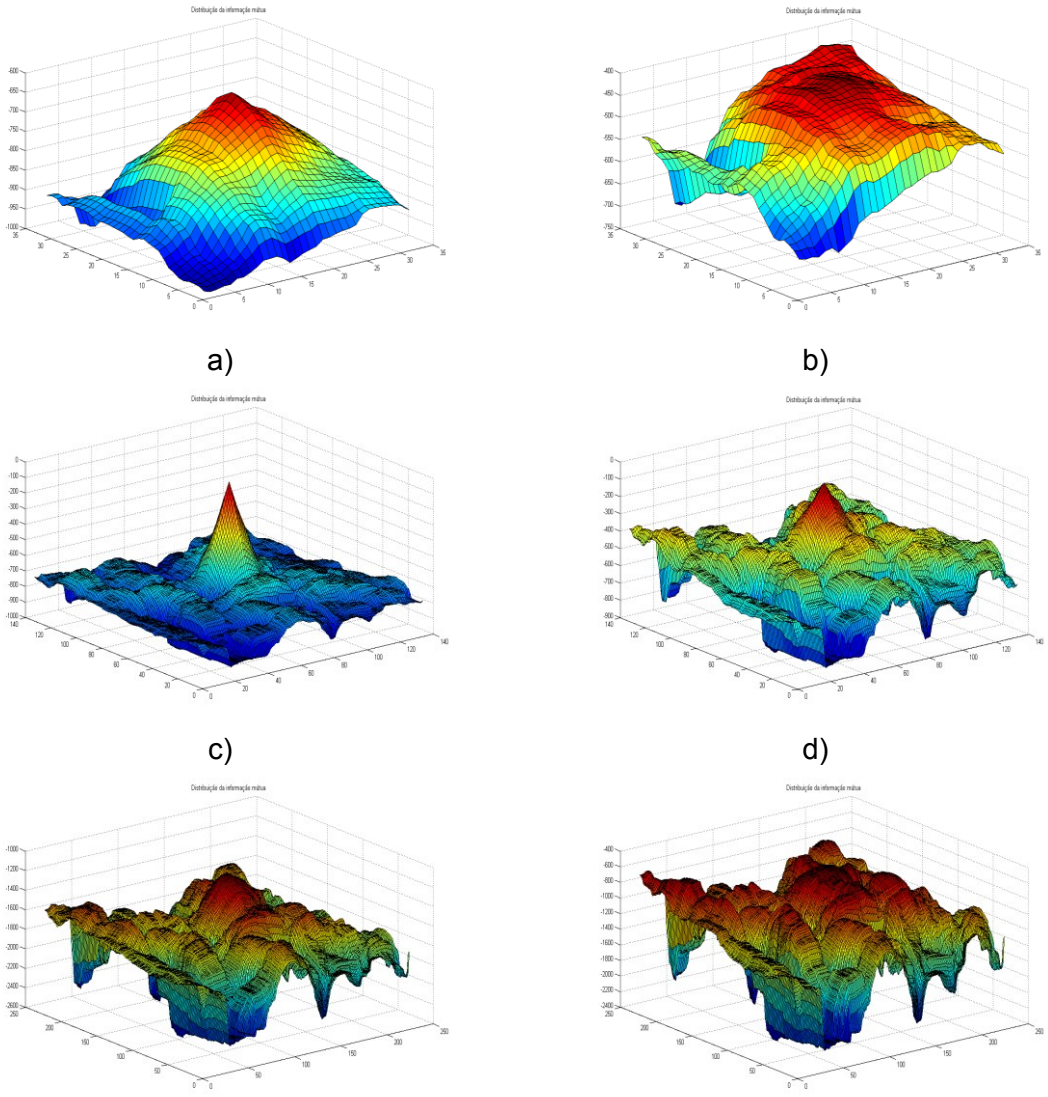
Tabela 20 - *Matching* baseado em *Haralick features* sem moldura e com *d*=10.

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
Mapa_9D_50	20,20	20,20	4,4916	1,4694	8,0709
Mapa_9D_150	70,70	70,70	74,8819	1,0386	394,0406
Mapa_9D_250	120,120	121,120	257,3860	3,0136	5929,6000
Mapa_3D_50	20,20	21,20	4,5986	1,4865	6,6265
Mapa_3D_150	70,70	70,70	76,5741	1,0120	324,8623
Mapa_3D_250	120,120	119,120	287,4333	3,2006	4157,6000

Analisando as 3 tabelas de resultados, verifica-se que em todos os casos estes se aproximaram bastante ou coincidiram com o que era previsto. Tal é uma indicação de que a moldura previamente acrescentada às imagens para o cálculo das *Haralick features* é apenas ruído desnecessariamente acrescentado e nada benéfico para o resultado final.

A ligeira diminuição do tempo de processamento em relação aos testes realizados com moldura deve-se ao facto de neste caso ter diminuído ligeiramente o tamanho efectivo dos dados. Além disso verifica-se que com *d=10* os tempos de processamento foram muito menores que com *d=1* e *d=5*, sem prejuízo nos resultados de *matching*.

É possível então concluir que a inserção de uma moldura é desnecessária e que um *delta* maior para o cálculo das *Haralick features* pode ser benéfico não só em termos de exactidão como também em termos de tempo de cálculo.



a) b) c) d) e) f)

Figura 27 – Ilustração dos resultados da Tabela 18. a) mapa 9D 50x50 pixeis; b) mapa 3D 50x50 pixeis; c) mapa 9D 150x150 pixeis; d) mapa 3D 150x150 pixeis; e) mapa 9D 250x250 pixeis; f) mapa 3D 250x250 pixeis.

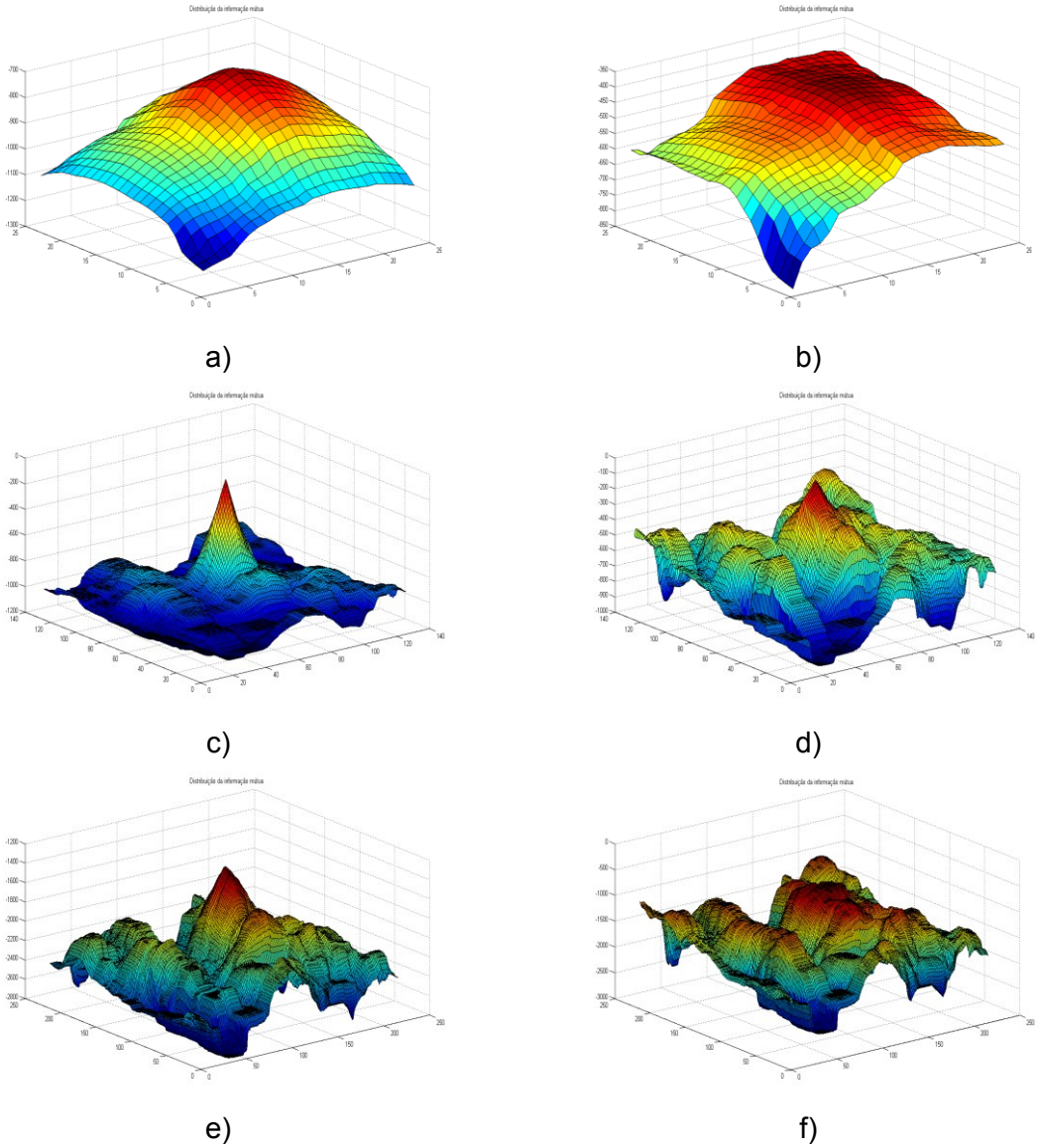


Figura 28 – Ilustração dos resultados da Tabela 19. a) mapa 9D 50x50 pixeis; b) mapa 3D 50x50 pixeis; c) mapa 9D 150x150 pixeis; d) mapa 3D 150x150 pixeis; e) mapa 9D 250x250 pixeis; f) mapa 3D 250x250 pixeis.

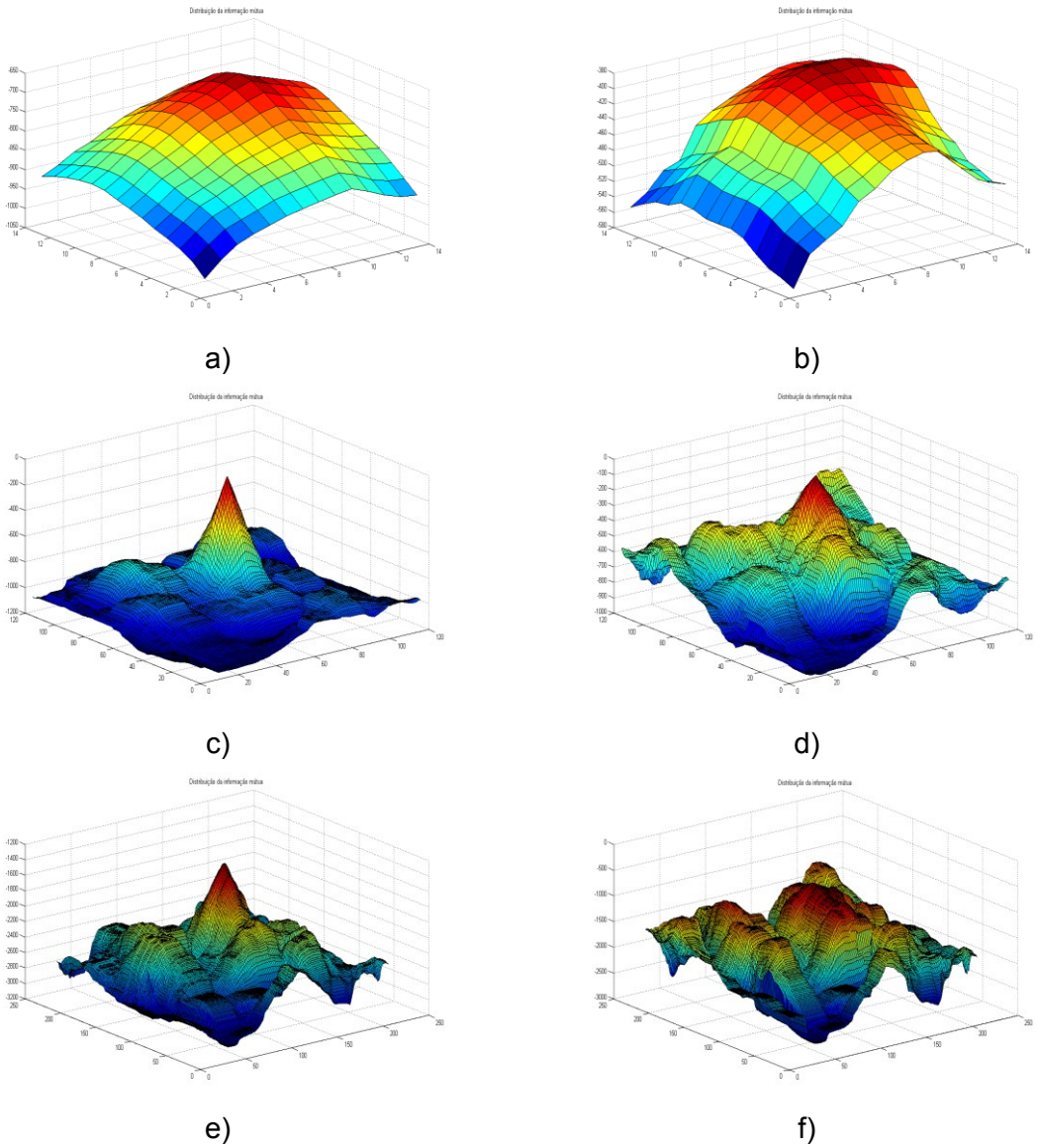


Figura 29 – Ilustração dos resultados da Tabela 19. a) mapa 9D 50x50 pixeis; b) mapa 3D 50x50 pixeis; c) mapa 9D 150x150 pixeis; d) mapa 3D 150x150 pixeis; e) mapa 9D 250x250 pixeis; f) mapa 3D 250x250 pixeis.

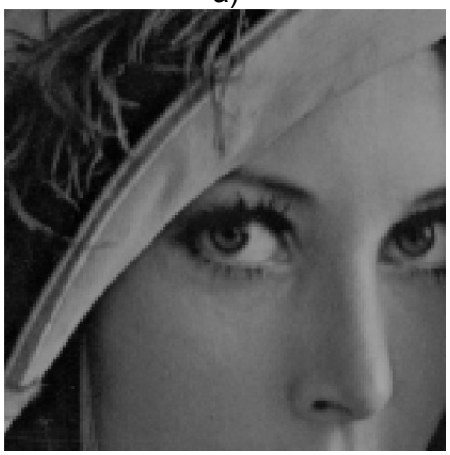
4.3. Testes com *template* de tamanho fixo e resolução constante

Perante os resultados anteriores, que apontavam para a não necessidade do uso de moldura, optou-se por não voltar a considerar essa opção. Desta forma, e doravante, todos os pixeis da periferia da imagem utilizada como mapa são descartados aquando do cálculo das *Haralick features*. De notar que o descarte dos pixeis periféricos do mapa praticamente não prejudica a informação nele presente, uma vez que, numa futura aplicação deste método a um AUV, existe sempre um conhecimento relativo da posição do veículo, o que permite evitar a situação de navegação nos limites do mapa conhecido.

Verificou-se que nos testes anteriores não tinha sido tomada em consideração a resolução das imagens utilizadas. De facto os 3 conjuntos de imagens (50x50, 150x150 e 250x250 pixeis) foram originalmente obtidos a partir de uma imagem maior (512x512 pixeis) à qual foi aplicado um redimensionamento por forma a ter o tamanho desejado. Contudo, este processo implica que a informação presente na imagem, a utilizar como mapa, seja alterada. Ao reduzir-se a resolução de uma imagem está a perder-se informação. Certos pormenores que na imagem maior eram representados por um elevado número de pixeis, o que permitia identifica-los claramente, são agora representados por muito menos pixeis, dificultando uma correcta identificação. Esta situação não reflecte o que ocorre numa aplicação real. Num veículo subaquático autónomo, que venha a usufruir de um sistema de navegação que possua este tipo de tratamento de imagem, o mapa será, tanto quanto possível, restrito local onde o veículo se encontra no momento. Não fará qualquer sentido processar um mapa onde estejam representados vários quilómetros do fundo marinho quando só é necessária a zona circundante ao veículo, dado que existe sempre alguma informação *a priori* sobre a localização aproximada do veículo, além de que o ponto de partida da missão é sempre conhecido.

Tendo isto em consideração, foram utilizadas, nos testes descritos doravante, imagens com os mesmos tamanhos (50x50, 150x150 e 250x250 pixeis) mas obtidas por corte da imagem original maior (512x512 pixeis) mantendo-se, desta forma, a resolução da imagem original. Estas imagens encontram-se representadas na Figura 30.

a)



b)

c)

Figura 30 – Imagem para testes de *template* fixo e resolução constante. a) mapa 50x50 pixeis; b) mapa 150x150 pixeis; c) mapa 250x250 pixeis.

Quanto ao *template*, decidiu-se não o fazer depender do tamanho do mapa; testaram-se, em todos os casos, tamanhos de 15x15 e 25x25 pixeis.

Em resumo, nos testes que se seguem não foram inseridas molduras (antes descartam-se bandas periféricas de acordo com *d*), o tamanho do *template* é

independente da dimensão do mapa e todas as imagens mantêm a resolução da imagem original de onde foram extraídas.

Comparou-se o desempenho com dados 9D e 3D (obtidos da forma já descrita), dois tamanhos de *template* e 3 valores de *d*: *d*=1, *d*=5 e *d*=10 num total de 12 combinações de testes. Os resultados encontram-se nas tabelas Tabela 21, Tabela 22, Tabela 23 e Tabela 24.

Tabela 21 – Resultados obtidos para imagens com 9 features, template 15x15 pixeis.

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
d=1					
Mapa_9D_50	20,20	20,20	9,2828	0,9666	32,7866
Mapa_9D_150	70,70	70,70	85,7768	0,9178	521,5265
Mapa_9D_250	90,90	90,90	243,6825	0,9890	1541,9000
d=5					
Mapa_9D_50	20,20	20,20	7,0493	0,9876	19,1139
Mapa_9D_150	70,70	70,70	80,6482	0,9437	467,7113
Mapa_9D_250	90,90	90,90	244,0401	0,9822	1427,3000
d=10					
Mapa_9D_50	20,20	20,20	4,3438	1,0766	7,2919
Mapa_9D_150	70,70	70,70	74,9677	1,0664	387,0338
Mapa_9D_250	90,90	90,90	237,4947	1,0295	1291,2000

Tabela 22 – Resultados obtidos para imagens com 9 features, template 25x25 pixeis.

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
d=1					
Mapa_9D_50	11,11	11,11	9,2677	2,5569	78,3228
Mapa_9D_150	70,70	70,70	91,1790	2,6205	2120,9000
Mapa_9D_250	90,90	90,90	272,8442	2,6009	6494,5000
d=5					
Mapa_9D_50	11,11	11,11	6,7614	2,6590	35,6173
Mapa_9D_150	70,70	70,70	81,1711	2,7047	1900,3000
Mapa_9D_250	90,90	90,90	261,6155	2,6918	5984,7000
d=10					
Mapa_9D_50	11,11	11,11	4,1462	2,9630	5,0584
Mapa_9D_150	70,70	70,70	79,4118	2,9114	1610,1000
Mapa_9D_250	90,90	90,90	257,2510	3,3633	5278,6000

Tabela 23 – Resultados obtidos para imagens com 3 features, template 15x15 pixeis.

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
d=1					
Mapa_3D_50	20,20	20,20	9,9064	0,9716	27,0949
Mapa_3D_150	70,70	70,70	99,6436	0,9499	424,5554
Mapa_3D_250	90,90	90,90	263,3827	1,0064	1348,5000
d=5					
Mapa_3D_50	20,20	20,20	6,8597	1,0542	16,1813
Mapa_3D_150	70,70	70,70	91,2118	1,0008	384,5648
Mapa_3D_250	90,90	90,90	268,0400	1,0101	1258,8000
d=10					
Mapa_3D_50	20,20	20,20	4,5326	1,1511	5,9617
Mapa_3D_150	70,70	70,70	79,7698	1,1074	323,1230
Mapa_3D_250	90,90	90,90	255,5267	1,1296	1148,2000

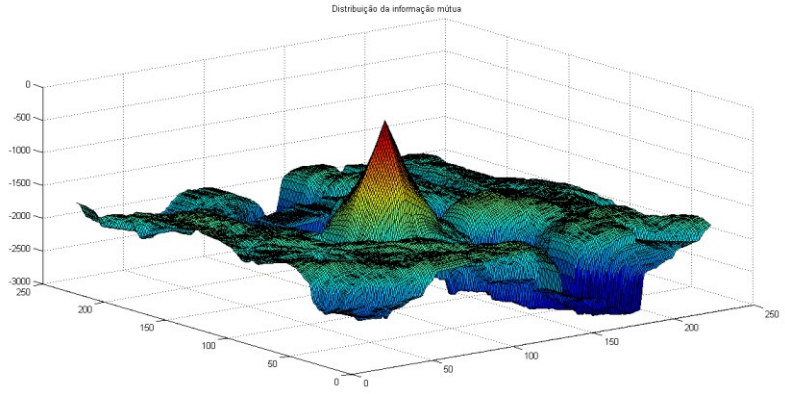
Tabela 24 – Resultados obtidos para imagens com 3 features, template 25x25 pixeis.

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
d=1					
Mapa_3D_50	11,11	11,11	9,3608	2,3942	60,1328
Mapa_3D_150	70,70	70,70	93,2268	2,5450	1737,8000
Mapa_3D_250	90,90	90,90	278,2861	2,5892	5228,8000
d=5					
Mapa_3D_50	11,11	11,11	6,9261	2,8451	27,2322
Mapa_3D_150	70,70	70,70	90,4580	2,7751	1534,4000
Mapa_3D_250	90,90	90,90	263,7340	3,9824	4834,3000
d=10					
Mapa_3D_50	11,11	11,11	4,2812	2,9795	4,0283
Mapa_3D_150	70,70	70,70	82,1061	3,1455	1199,8000
Mapa_3D_250	90,90	90,90	259,7439	2,9900	4383,6000

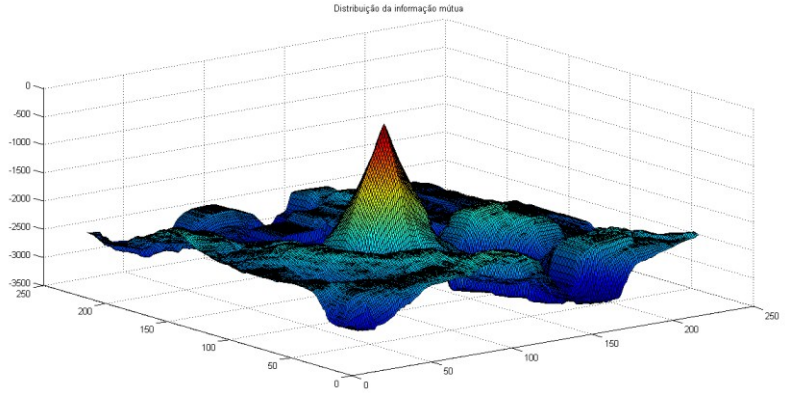
Nestes testes verifica-se que a localização foi exacta em todos os casos, o que vem confirmar as expectativas iniciais de que a não adição da moldura e a utilização de imagens retiradas de uma maior, ao invés de se fazer um redimensionamento, seriam factores que iriam contribuir para o aumento da qualidade dos resultados. Além disso é confirmada a eficácia do algoritmo de cálculo de informação mútua, reforçando assim a eficácia do método proposto por Kybic.

Em termos de processamento, observa-se que os tempos de pré-processamento do mapa sofrem poucas alterações no conjunto total de testes; no caso do *template* os tempos de execução variam consoante o tamanho deste, sem contudo haver mudanças muito significativas. Observa-se que as maiores diferenças a nível temporal correspondem ao tempo necessário para efectuar o cálculo da informação mútua onde se

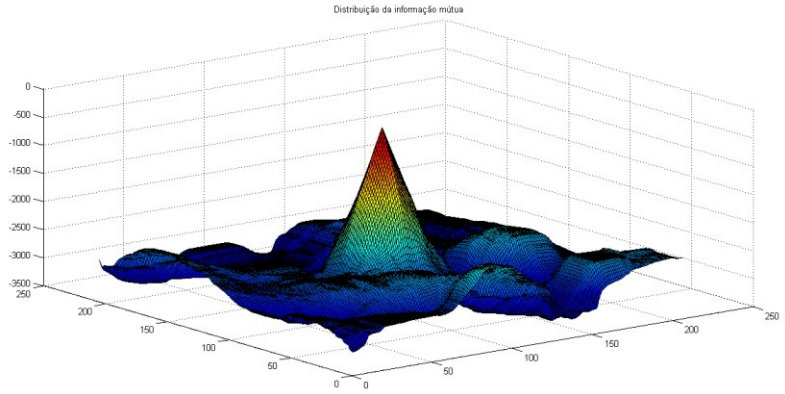
verifica que o tempo de execução diminui drasticamente com a diminuição do tamanho do *template*. Verifica-se também que o tempo de processamento diminui com o aumento do *d* (*delta*). Além disso uma valor do parâmetro *d* superior aponta para um aumento da fiabilidade dos resultados do *matching*, como sugere a Figura 31.



a)

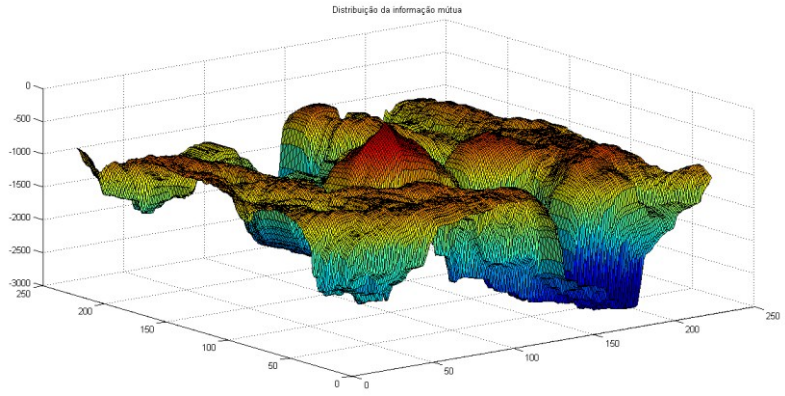


b)

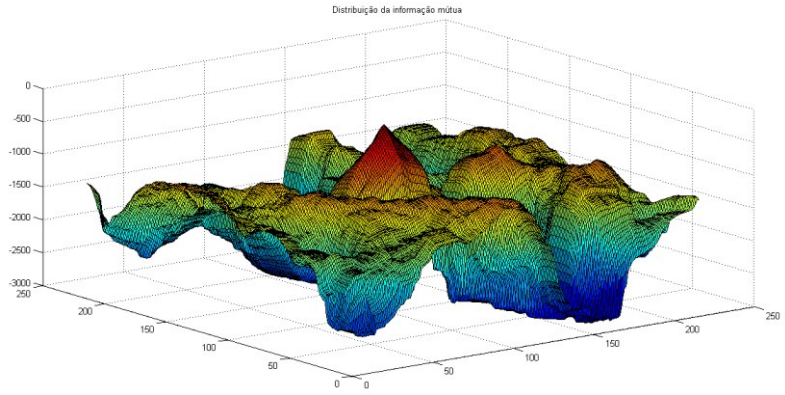


c)

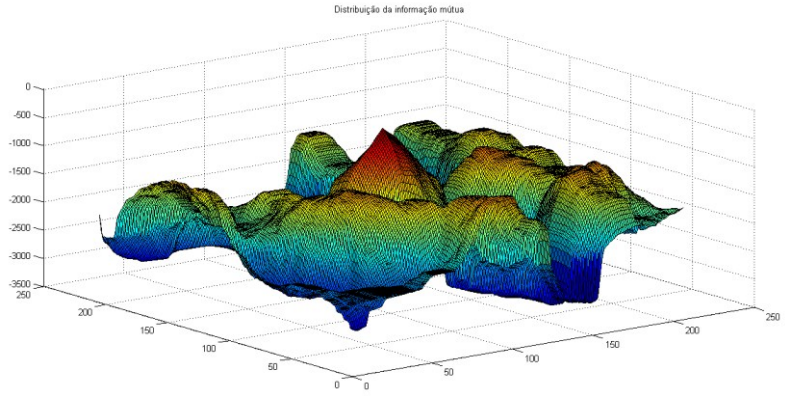
Figura 31 – Distribuição dos valores de informação mútua, considerando 9 dimensões e mapa 250x250 pixeis. a) *d*=1; b) *d*=5; c) *d*=10.



a)



b)



c)

Figura 32 – Distribuição dos valores de informação mútua, considerando 3 dimensões e mapa 250x250 pixeis. a) *d*=1; b) *d*=5; c) *d*=10.

Como o propósito é encontrar o máximo da informação mútua mapa e *template* os picos representados nas figuras Figura 31 e Figura 32 indicam a localização do *template*

no mapa. Observa-se que este pico de informação mútua é tanto mais destacado quanto mais alto é *d*.

As imagens aqui utilizadas têm padrões claramente identificáveis não havendo, desta forma, possibilidade de encontrar resultados ambíguos. Contudo, no caso das imagens de sonar tal não se verifica e por isso será importante verificar se este efeito de *d* nos resultados se mantém.

O facto de a localização melhorar com o aumento de *d* poderá estar relacionado com fenómenos de correlação dos dados em microescala. Este é um assunto que extravasa o âmbito desta dissertação mas que poderia ser estudado aplicando técnicas de análise da continuidade espacial dos dados utilizadas no domínio da geoestatística; ver por exemplo [15], [16].

4.4. Testes com ruído

De forma a melhor averiguar não só a robustez do algoritmo, como também se é justificável considerar as *Haralick features* (Energia e Contraste) para cada orientação, ao invés de se utilizar apenas uma média dessas 4 orientações, procedeu-se à realização de testes com ruído.

Apesar de se ter adicionado deliberadamente ruído às imagens utilizadas, em imagens de sonar o maior problema é a variabilidade do fundo marinho. Pequenas alterações provocadas quer por erosão, quer por diferenças na aquisição dos dados, introduzem alterações nas imagens capturadas (*template*) em relação ao mapa considerado. Apesar de não se tratar de "ruído" no verdadeiro significado da palavra, estas alterações dificultam a aplicação das técnicas aqui estudadas à navegação.

Com cálculo das *Haralick Features* para as 4 orientações é esperado que o impacto deste "ruído" seja minimizado. De notar que, para uma determinada rota do AUV, o fundo marinho circundante pode ser visto como muito ruidoso, segundo uma *feature*, mas no entanto segundo outra (calculada numa diferente orientação) pode possuir informação muito relevante para a qual a influência do ruído não seja significativa.

Para limitar o número de testes necessário optou-se por utilizar apenas imagens com 150x150 pixeis, uma vez que apresentam um bom compromisso entre tempo de execução e qualidade dos resultados.

Utilizou-se a função '*imnoise*' do Matlab para adicionar ruido branco gaussiano com média e variância constantes.

O ruido foi adicionado à imagem em tons de cinzento que serve de base a todos os testes realizados, sendo de seguida extraído o *template* pretendido. Deste modo, o cálculo das *Haralick features* correspondentes ao *template* é afectado pela adição do ruído.

A *média* do ruído branco gaussiano foi testada para valores de 0.2 e 0, enquanto que para a *variância* se utilizaram os valores 0.01, 0.001 e 0.0005. De notar que o pior dos casos (o mais ruidoso) ocorre quando a *média* é diferente de zero e a *variância* elevada, o que aqui corresponde aos valores 0.2 e 0.01, respectivamente.

O efeito desta adição de ruído pode ser observado na Figura 33 onde se apresenta primeiro a imagem original sem ruído seguida das imagens com mais e menos ruído adicionado.

a)

b)

c)

Figura 33 – a) Imagem original sem ruído, 150x150 pixeis; b) Imagem com ruído branco gaussiano, média = 0 e variância = 0,0005; c) Imagem com ruído branco gaussiano, média = 0,2 e variância = 0,01.

Como é possível observar, apesar da adição de ruído, a imagem representada em b) não apresenta muita diferença em relação à original. Contudo no caso da imagem c) é previsível que a adição de ruído possa vir a ter uma interferência muito negativa nos resultados pretendidos.

É também observável que tendo uma *média* diferente de zero está-se a modificar a intensidade dos níveis de cinzento, enquanto que modificando a *variância* altera-se a homogeneidade da imagem; por exemplo, zonas onde a intensidade média da cor seria constante passam a ter uma maior variação e portanto um maior contraste entre pixeis vizinhos.

Os testes realizados dividiram-se em 4 grupos, todos compreendendo testes com dados de imagem a 9 dimensões e a 3 dimensões.

A configuração é a seguinte:

- ➔ Testes com *template* de tamanho 15x15 pixeis e *delta* igual a 1;
- ➔ Testes com *template* de tamanho 15x15 pixeis e *delta* igual a 10;
- ➔ Testes com *template* de tamanho 25x25 pixeis e *delta* igual a 1;
- ➔ Testes com *template* de tamanho 25x25 pixeis e *delta* igual a 10;

Tabela 25 – Resultados com *d*=1 e *template* 15x15 pixeis.

Imagem	variância	média	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
Mapa_9D_150	0,01	0	69,69	72,02	94,9920	0,9205	501,1249
Mapa_9D_150	0,001	0	69,69	71,69	94,9298	0,9536	542,1347
Mapa_9D_150	0,0005	0	69,69	69,69	91,8506	0,9508	506,5759
Mapa_9D_150	0,01	0,2	69,69	81,66	93,5452	1,0637	522,1296
Mapa_9D_150	0,001	0,2	69,69	70,76	95,1441	1,0196	533,4939
Mapa_9D_150	0,0005	0,2	69,69	69,75	99,7108	1,0365	515,3449
Mapa_3D_150	0,01	0	69,69	69,02	94,7178	0,9380	426,8866
Mapa_3D_150	0,001	0	69,69	71,65	93,1179	1,0043	429,9534
Mapa_3D_150	0,0005	0	69,69	69,68	88,6293	0,9308	425,9274
Mapa_3D_150	0,01	0,2	69,69	82,66	91,4723	1,0338	459,3887
Mapa_3D_150	0,001	0,2	69,69	59,23	93,0508	0,9402	468,2896
Mapa_3D_150	0,0005	0,2	69,69	9,96	102,1947	1,0862	457,4627

Tabela 26 – Resultados com *d*=10 e *template* 15x15 pixeis.

Imagem	variância	média	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
Mapa_9D_150	0,01	0	60,60	66,57	82,3865	1,1437	414,4472
Mapa_9D_150	0,001	0	60,60	60,60	84,8309	1,1775	415,3312
Mapa_9D_150	0,0005	0	60,60	60,60	82,5417	1,1545	422,2073
Mapa_9D_150	0,01	0,2	60,60	[56:56],[50:51]	85,7821	1,1190	410,6671
Mapa_9D_150	0,001	0,2	60,60	[66:66],[44:45]	83,2755	1,0556	414,8179
Mapa_9D_150	0,0005	0,2	60,60	[66:66],[44:45]	84,6644	1,1762	420,3688
Mapa_3D_150	0,01	0	60,60	64,56	83,4224	1,1296	317,9621
Mapa_3D_150	0,001	0	60,60	61,60	83,3109	1,0499	320,7833
Mapa_3D_150	0,0005	0	60,60	60,60	87,4662	1,0996	314,7468
Mapa_3D_150	0,01	0,2	60,60	25,38	82,2049	1,1445	357,5628
Mapa_3D_150	0,001	0,2	60,60	50,14	78,4223	1,0785	363,5365
Mapa_3D_150	0,0005	0,2	60,60	50,14	81,5555	1,0669	357,5696

Tabela 27 – Resultados com *d*=1 e *template* 25x25 pixeis.

Imagem	variância	média	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
Mapa_9D_150	0,01	0	69,69	71,55	96,0130	2,8087	2234,4000
Mapa_9D_150	0,001	0	69,69	73,65	101,6348	2,7648	2196,0000
Mapa_9D_150	0,0005	0	69,69	71,68	94,5963	2,7159	2195,6000
Mapa_9D_150	0,01	0,2	69,69	65,53	95,9180	2,7587	2230,1000
Mapa_9D_150	0,001	0,2	69,69	77,64	95,5455	2,5244	2216,1000
Mapa_9D_150	0,0005	0,2	69,69	77,63	103,0913	2,5778	2168,1000
Mapa_3D_150	0,01	0	69,69	2,115	89,0361	2,7654	1638,0000
Mapa_3D_150	0,001	0	69,69	73,65	84,3205	2,6914	1591,4000
Mapa_3D_150	0,0005	0	69,69	71,65	96,4555	2,8103	1601,2000
Mapa_3D_150	0,01	0,2	69,69	2,114	93,9289	2,7807	1575,8000
Mapa_3D_150	0,001	0,2	69,69	122,64	90,1523	2,6556	1649,6000
Mapa_3D_150	0,0005	0,2	69,69	122,70	88,5140	2,7753	1614,4000

Tabela 28 – Resultados com *d*=10 e *template* 25x25 pixeis.

Imagem	variância	média	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
Mapa_9D_150	0,01	0	60,60	64,53	79,2899	3,0104	1626,3000
Mapa_9D_150	0,001	0	60,60	60,59	79,5769	3,0329	1594,2000
Mapa_9D_150	0,0005	0	60,60	60,59	79,3830	3,0033	1593,8000
Mapa_9D_150	0,01	0,2	60,60	66,36	79,0615	2,8900	1592,1000
Mapa_9D_150	0,001	0,2	60,60	64,35	78,9160	2,8798	1568,8000
Mapa_9D_150	0,0005	0,2	60,60	64,35	79,5114	2,9912	1594,3000
Mapa_3D_150	0,01	0	60,60	63,53	77,0101	3,0088	1262,3000
Mapa_3D_150	0,001	0	60,60	61,58	77,2654	3,1045	1224,9000
Mapa_3D_150	0,0005	0	60,60	61,59	76,4835	2,9429	1255,0000
Mapa_3D_150	0,01	0,2	60,60	106,54	79,0231	2,9463	1241,5000
Mapa_3D_150	0,001	0,2	60,60	23,69	76,7524	2,8454	1241,4000
Mapa_3D_150	0,0005	0,2	60,60	23,69	79,5998	2,9721	1231,5000

Comparando todos os resultados obtidos é possível concluir que os piores resultados verificam-se quando o parâmetro *média* é diferente de zero e, regra geral, a dimensionalidade das imagens utilizadas é 3 (3D).

Em contrapartida, os melhores resultados são obtidos para as imagens 9D e parâmetro *média* igual a zero, o que é claramente indicativo de que imagens cujas *Haralick features* são individualmente consideradas apresentam uma maior robustez do que quando é feita uma média dessas mesmas *features*.

O único resultado contraditório é apresentado na Tabela 28, para o caso de imagens 9D e *média* diferente de zero, onde as coordenadas obtidas se afastaram muito das previstas. Note-se no entanto que os conjuntos de valores obtidos foram bastante semelhantes entre si, o que revela uma grande precisão do método, apesar da fraca exactidão. Tendo sido este resultado diferente do que até então se tinha verificado nos testes realizados, e tendo em consideração não só a precisão alcançada como também o facto de que o ruído adicionado ser constituído por valores aleatórios, existe a possibilidade destes resultados terem sido um infeliz acaso. Apenas realizando um elevado número de repetições destes testes se poderia verificar para que valores tendem realmente os resultados.

No entanto, apesar de pertinente, tal análise só faria de facto sentido aquando da realização de testes usando dados reais de sonar e em situações onde o tipo de ruído introduzido simulasse correctamente uma situação real de navegação de AUV. Os testes realizados e apresentados nas tabelas anteriores têm como principal objectivo ter uma melhor percepção de como diferentes configurações dos dados utilizados se comportam perante situações ruidosas, de forma a permitir indicar um caminho a seguir na elaboração de testes futuros.

Relativamente aos tempos de execução obtidos, em todos os caso se verificou que estes se encontram dentro daquilo que era esperado e que já tinha sido verificado em testes anteriores.

4.5. Diminuição da resolução da matriz SGLD

Perante os resultados obtidos até este ponto pode-se tirar a importante conclusão de que uma imagem para a qual foram calculadas as *Haralick Features* Energia e Contraste para cada uma das 4 orientações, sendo portanto constituída por 9 *features* (contando com os tons de cinzento), produzem resultados com uma precisão bastante elevada sendo, inclusive, mais imunes a ruído do que quando se utilizam as imagens constituídas pelos tons de cinzento e pela média das 4 orientações das *features* Energia e Contraste.

Além disto, verificou-se que seleccionando valores de *delta* maiores que 1, aquando do cálculo das *Haralick Features*, os resultados obtidos são tão exactos quanto os verificados usando apenas *d*=1, com a vantagem de serem também processados mais rapidamente que neste último caso.

Outro ponto importante é o tamanho da janela de aquisição do sonar que corresponde ao tamanho do *template* utilizado nos testes realizados. Verificou-se que tamanhos maiores são preferíveis principalmente por serem uma representação mais próxima da realidade pois num sistema real de SSS o *template* obtido terá dimensões consideráveis.

Para que os testes aproximem melhor um caso real só falta passar a utilizar imagens reais, obtidas por um sonar de varrimento lateral.

Contudo, devido ao tamanho das imagens de sonar e consequente peso computacional é importante procurar formas de diminuir os tempos de processamento. Uma forma de conseguir isso é diminuir a resolução da matriz SGLD aquando do cálculo

das *Haralick Features*. Tal pode ser feito especificando um número de bits menor que o número de bits de resolução da imagem original.

As imagens utilizadas têm todas uma escala de cinzentos de 0 a 255 (8 bits). Ao especificar-se para a matriz SGLD apenas metade da gama (7 bits), na realidade está-se a realizar o cálculo de uma matriz de co-ocorrência cuja área é 4 vezes menor que a calculada em todos os testes realizados anteriormente.

Desta forma, pelo simples facto de se fazer um cálculo matricial cujo peso é muito menor, é de esperar um ganho bastante grande em termos de tempo de execução.

Testou-se esta hipótese de uma forma simples. Utilizando uma imagem de tamanho 150x150 pixeis com 9 *features*, um *template* com 25x25 pixeis, realizaram-se 3 testes para os casos de *delta* igual a 1, 5 e 10. Os resultados obtidos são os apresentados na seguinte tabela.

Tabela 29 – Resultados com redução da resolução da matriz SGLD (7 bits).

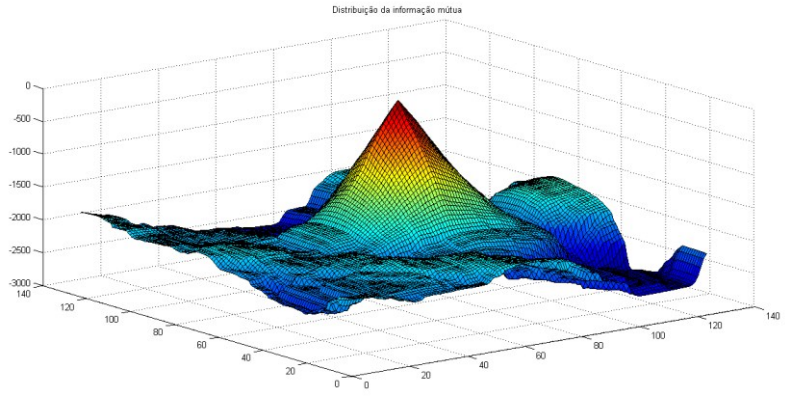
Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN (s)
d=1					
Mapa_9D_150	70,70	70,70	39,1536	1,2022	2132,7000
d=5					
Mapa_9D_150	70,70	70,70	36,0786	1,1709	1845,7000
d=10					
Mapa_9D_150	70,70	70,70	32,7806	1,2409	1540,7000

Da análise destes resultados é possível observar que as diferenças em relação às medidas temporais obtidas em testes realizados anteriormente não se verificam no processamento do algoritmo kNN mas sim, como era esperado, no cálculo das *Haralick features*.

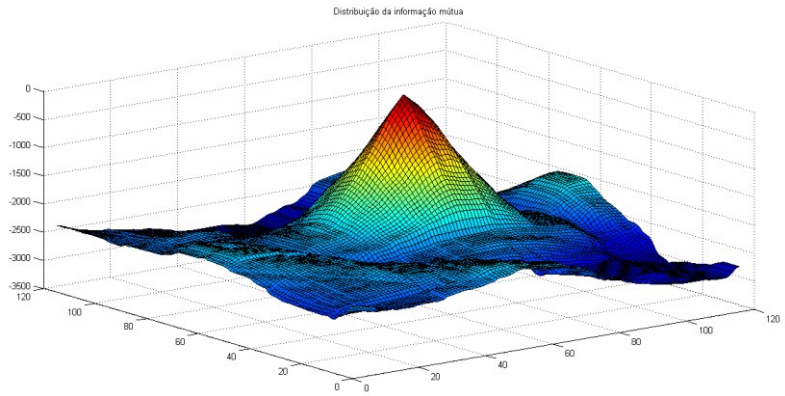
De facto comparando os resultados da Tabela 29 com os de outras tabelas (por exemplo Tabela 22) é possível ver que para o caso em que as imagens consideradas tinham um tamanho de 150x150 pixeis a redução do tempo de processamento das *Haralick features* foi bastante drástico, chegando mesmo a cair para menos de metade do que verificado anteriormente.

Além disto, em todos os 3 casos testados os resultados obtidos foram exactamente os que eram pretendidos, ou seja, a diminuição da resolução da matriz SGLD para metade não afectou a qualidade da informação fornecida pelas *Haralick features*. Assim obtém-se um método igualmente exacto ao verificado anteriormente mas com a vantagem de o pré-processamento do mapa e do *template* serem bastante mais rápidos.

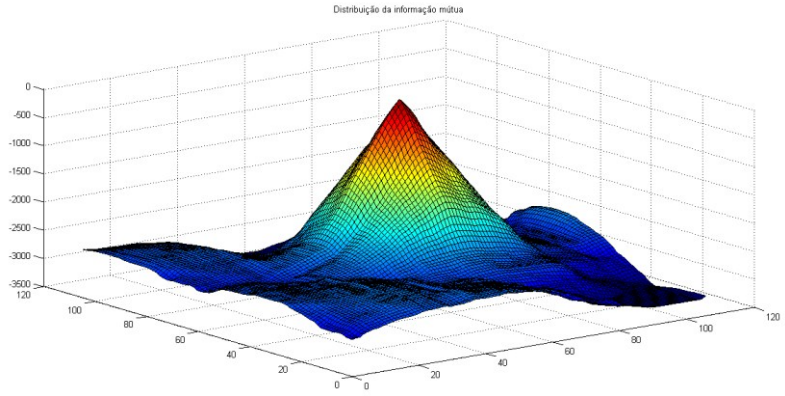
A Figura 34 ilustra os resultados descritos na Tabela 29.



a)



b)



c)

Figura 34 – Distribuição dos valores de informação mútua, considerando 9 dimensões e mapa 150x150 pixeis, para uma matriz SGLD representada a 7 bits. a) *d*=1; b) *d*=5; c) *d*=10.

4.6. Estimador de entropia kNN-Batch

Uma outra forma de reduzir o tempo de processamento foi proposta por Kybic [11], que não só propôs um estimador de entropia mas também uma forma de tornar esse estimador mais rápido, ao qual chamou de estimador *batch* (daí a designação kNN-batch usada neste trabalho).

O método é relativamente simples e consiste em dividir o espaço de amostras consideradas, que corresponderá neste caso ao *template* e à matriz parcial obtida a cada iteração do algoritmo, em grupos e estimar a entropia de cada grupo de amostras. A entropia final será uma média dos valores de entropia estimados para cada grupo.

Utilizando a versão *batch* do algoritmo kNN foi executado um conjunto de testes usando duas imagens com 9 *features* com tamanhos 150x150 e 250x250 pixeis, *template* com 25x25 pixeis e *delta* com valores iguais a 1, 5 e 10. Os resultados obtidos são os presentes na Tabela 30, de notar que o factor de divisão (*M*) considerado é de *M*=4.

Tabela 30 - Resultados obtidos para algoritmo kNN-batch, matriz SGLD com 7bits, template 25x25 pixeis, M=4

Imagem	Coord. previstas (x,y)	Coord. obtidas (x,y)	tempo cálculo HF mapa (s)	tempo cálculo HF *template* (s)	tempo kNN batch (s)
d=1					
Mapa_9D_150	70,70	70,70	38,0047	1,1130	1076,5000
Mapa_9D_250	90,90	90,90	108,2975	1,1217	3337,0000
d=5					
Mapa_9D_150	70,70	70,70	36,4759	1,1813	944,1784
Mapa_9D_250	90,90	90,90	104,6139	1,1022	3075,2000
d=10					
Mapa_9D_150	70,70	70,70	34,0185	1,2796	791,0323
Mapa_9D_250	90,90	90,90	104,4338	1,2484	2818,7000

De uma primeira análise aos resultados obtidos pode-se observar que as coordenadas obtidas foram exactamente aquelas que eram pretendidas.

Analisando os valores obtidos para o tempo de execução do algoritmo kNN-batch, e comparando esses valores com os da Tabela 22, é clara a redução no tempo despendido com a execução do algoritmo. De facto é observável que essa redução no tempo de execução está na ordem dos 50%. De notar que o factor de divisão utilizado é de apenas 4 pois o espaço de amostras não é muito grande (25x25 pixeis); para um número de amostras maior será necessário um factor de divisão maior para que este método seja justificável.

Este método consegue uma grande melhoria do algoritmo kNN, claramente evidenciada pelos resultados obtidos desde os testes iniciais até este ponto. Observa-se uma grande redução do tempo de processamento.

Conclui-se então que, em termos do algoritmo, é possível uma exactidão bastante elevada. Além disso, usando o método *batch* é possível obter reduções muito significativas no peso computacional do algoritmo. Uma outra conclusão importante está relacionada com o número de *features* utilizadas; verifica-se que um espaço dimensional

maior é plenamente justificável, pois a informação adicional garante uma maior imunidade ao ruído. O tamanho do *delta* considerado para o cálculo das *Haralick features* também tem influência nos resultados finais. No entanto, apesar de pelos resultados analisados *d=10* ser a melhor configuração, só realizando testes com dados de sonar reais se poderá ter uma conclusão definitiva sobre se de facto *deltas* maiores produzem melhores resultados. De igual forma, o tamanho ideal da janela para aquisição do *template* só poderá ser estabelecido aquando da utilização de dados reais de sonar. Nos testes realizados, optou-se por usar um *template* maior, contudo a situação testada não apresenta um paralelo directo com o esperado no caso real.

A imagem seguinte ilustra os resultados presentes na Tabela 30.

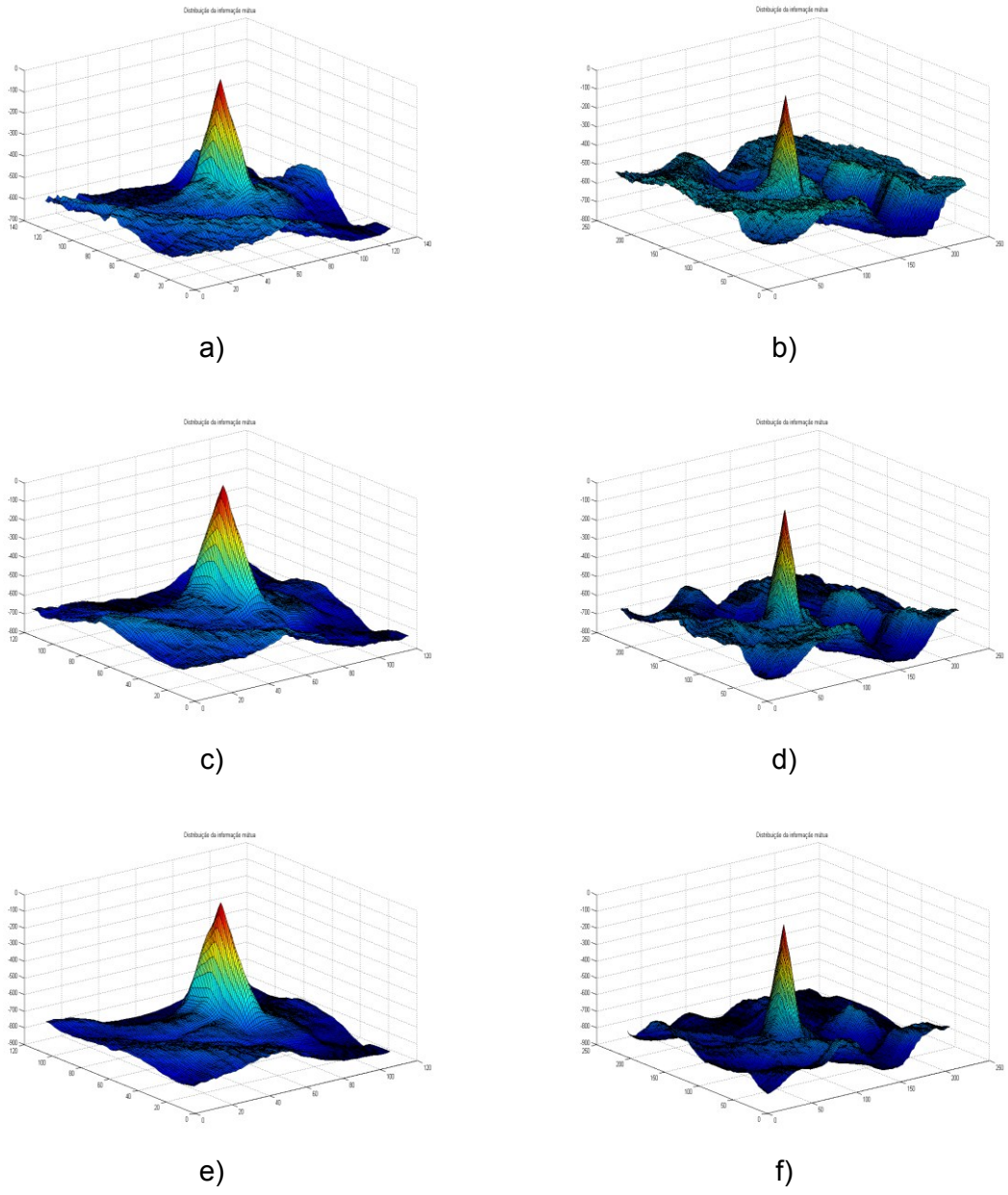


Figura 35 - Distribuição dos valores de informação mútua obtidos pelo algoritmo kNN-batch. a) mapa 150x150 pixeis *d*=1; b) mapa 250x250 pixeis *d*=1; c) mapa 150x150 pixeis *d=5*; d) mapa 250x250 pixeis *d*=5; e) mapa 150x150 pixeis *d*=10; f) mapa 250x250 pixeis *d*=10.

5. Complexidade Computacional do algoritmo de Informação Mútua

Apresenta-se de seguida um estudo sobre a complexidade computacional imposta pelo algoritmo de cálculo da Informação Mútua.

Tenha-se novamente em consideração o pseudo-código já previamente apresentado.

$kNN(imageP, imageQ)\{$
$\quad \boldsymbol{for}\ i = 1{:}N$
$\quad\quad \boldsymbol{return}\ neighborDistances = \|\mathrm{q} - \mathrm{p}\|$
$\quad \boldsymbol{end}\ \}$

$imageA \leftarrow template(N \times D)$
$entropyA \leftarrow \sum^{N} \log\left(\min\left(kNN(imageA, imageA)\right)\right)$
$\boldsymbol{for}\ i = 1{:}\left(M_1 - (N_1 + 1)\right)$
$\quad \boldsymbol{for}\ j = 1{:}\left(M_2 - (N_2 + 1)\right)$
$\quad imageB \leftarrow partial\ matrix(N \times D)$
$\quad entropyB \leftarrow \sum^{N} \log\left(\min\left(kNN(imageB, imageB)\right)\right)$
$\quad entropyAB \leftarrow \sum^{N} \log\left(\min\left(kNN(imageA, imageB)\right)\right)$
$\quad mutual\ info(i,j) \leftarrow entropyA + entropyB - entropyAB$
$\quad \boldsymbol{end}$
$\boldsymbol{end}$

Para o cálculo da complexidade computacional do algoritmo de obtenção de Informação Mútua entre duas imagens é necessário considerar que se pretende localizar num *mapa,* cujo formato é (M_1xM_2xd), uma imagem (*patch*) de formato (N_1xN_2xd). Os pixeis do *patch*, e restantes imagens parciais obtidas à medida que se faz o varrimento do mapa, são reorganizados de modo a formarem uma matriz com formato (Nxd) onde:

- *N* – número de pixeis em cada *patch* ($N_1 * N_2 = N$);
- *d* – número de dimensões das imagens, ou seja, número de *features*;

No início do algoritmo é calculada a entropia do patch. Para tal é necessário obter as distâncias euclideanas entre vizinhos usando o algoritmo kNN (k-Nearest Neighbor).

No algoritmo kNN

$kNN(imageP, imageQ)\{$
$\quad \boldsymbol{for}\ i = 1{:}N$
$\quad\quad \boldsymbol{return}\ neighborDistances = \|\mathrm{q} - \mathrm{p}\|$
$\quad \boldsymbol{end}\ \}$

temos, por cada iteração, d multiplicações e d somas o que, após todas as iterações, o peso computacional totaliza:

- $N * d = Nd$ Multiplicações;
- $N * d = Nd$ Somas;

Tendo os valores das distâncias, obtém-se a entropia de acordo com a fórmula $entropyA \leftarrow \sum^{N} \log(\min(kNN(imageA, imageA)))$. Nesta caso temos um total de:

- $N - 1$ Somas;

Nesta fórmula encontra-se também presente a operação de ordenação, necessária para encontrar o conjunto das mínimas distâncias euclideanas entre vizinhos. Usando, por exemplo, o algoritmo de ordenação *merge sort*, é adicionado um peso computacional na ordem de $Nlog(N)$. O peso computacional total pode então ser aproximado a:

- $N^2 log(N)$;

Totalizando o peso computacional imposto pelo algoritmo kNN e pelo cálculo da entropia temos:

- $Nd + N^2\log(N)$ Multiplicações;
- $Nd + N^2\log(N)$ Somas;

Comparando os dois elementos da expressão é possível constatar que o peso computacional imposto por Nd pode ser desprezado pois é muito menor que $N^2\log(N)$. Desta forma, o processo de obtenção de uma estimativa para o valor de entropia impõe um peso computacional de:

- $N^2\log(N)$. Multiplicações;
- $N^2\log(N)$. Somas;

De seguida é efectuado o varrimento do mapa onde se pretende encontrar o *patch*. Nesta fase é necessário calcular a entropia de cada imagem parcial assim como a entropia conjunta entre essas imagens e o *patch*. Para tal procede-se de igual forma, obtendo-se uma estimação dos valores de entropia usando os resultados da aplicação do algoritmo kNN. Nesta fase o peso computacional será duas vezes o verificado no caso acima referido. Desta forma temos:

- $2 * N^2\log(N)$ Multiplicações;
- $2 * N^2\log(N)$ Somas;

Para concluir existe o cálculo da informação mútua, que consiste em efectuar a operação $mutual\ info(i,j) \leftarrow entropyA + entropyB - entropyAB$ que adiciona apenas mais duas somas ao peso computacional de cada iteração, no entanto esta operação não

tem qualquer influência no desempenho geral do programa podendo portanto ser ignorada.

Com a excepção do cálculo da entropia do *patch*, todos os restantes valores são obtidos no decorrer dos ciclos que permitem fazer o varrimento do mapa. Estes ciclos repetem-se $[M_1 - (N_1 + 1)] * [M_2 - (N_2 + 1)]$ vezes, sendo que M_1 corresponde ao número de linhas, da matriz correspondente ao mapa, e M_2 ao número de colunas. Esta expressão pode ser aproximada a $M_1 M_2 - (M_1 + M_2)$.

No final, e tendo em consideração todas as iterações do algoritmo, o peso computacional pode ser expresso da seguinte forma:

$$Multiplicações: N^2 \log(N) + [M_1 M_2 - (M_1 + M_2)] * 2N^2 \log(N)$$
$$Somas: N^2 \log(N) + [M_1 M_2 - (M_1 + M_2)] * 2N^2 \log(N)$$

Desenvolvendo obtém-se:

$$Multiplicações: N^2 \log(N) + 2N^2 \log(N) * M_1 M_2 - 2N^2 \log(N) * (M_1 + M_2)$$

$$Somas: N^2 \log(N) + 2N^2 \log(N) * M_1 M_2 - 2N^2 \log(N) * (M_1 + M_2)$$

Que resulta em:

$$Multiplicações: N^2 \log(N) * (1 + 2M_1 M_2 - (M_1 + M_2))$$

$$Somas: N^2 \log(N) * (1 + 2M_1 M_2 - (M_1 + M_2))$$

Simplificando estas expressões podemos concluir que o peso computacional total pode ser expresso aproximadamente pelas fórmulas:

$$Multiplicações: 2N^2 \log(N) M_1 M_2$$

$$Somas: 2N^2 \log(N) M_1 M_2$$

Tendo em consideração o facto de que as imagens podem ser quadradas (número de linhas igual ao número de colunas, $M_1 = M_2 = M$), a expressão final, que expressa o peso computacional imposto pelo algoritmo de obtenção dos valores de Informação Mútua, será:

$$\boldsymbol{\mathcal{O}}(N^2 \log(N) M^2)$$

6. Selecção de *Haralick Features* para aplicação em dados de *Side-Scan Sonar*

Os testes até agora realizados tiveram como objectivo principal demonstrar que a utilização de *Haralick features* representa um acréscimo de informação muito importante, permitindo obter resultados muito mais exactos que usando apenas a informação fornecida pela escala de cinzentos.

Em muita da documentação consultada é referida a utilização das *features* Energia e Contraste, tendo sido esse o primeiro motivo para a sua utilização neste trabalho. Contudo, e tendo em conta o facto de que no seu artigo Haralick apresenta muitas mais *features* para classificação de imagem, não foi encontrada uma justificação satisfatória para o uso destas duas *features* em particular.

Neste sentido, e perante a possibilidade de as *features* utilizadas não serem aquelas que maior informação poderão fornecer acerca das imagens de SSS, optou-se por fazer uma análise e selecção das *features* adequadas, tendo em conta critérios adequados ao presente problema da navegação subaquática.

6.1. *Haralick Features* de imagens de SSS

A determinação das melhores *features* a utilizar em imagens de SSS, impõe quantificar a informação que cada uma contém. Desta forma, recorreu-se à definição de Informação Própria (*Self-Information*), utilizando uma metodologia semelhante a já usada previamente. Foram então calculados os valores de informação própria de cada *Haralick feature*, tendo estas sido obtidas de uma imagem em escala de cinzentos (8 bits) do fundo marinho e para d=1, d=5 e d=10.

Na seguinte figura encontra-se representada uma imagem do fundo marinho, obtida de um SSS, utilizada para obter e classificar as diferentes *Haralick features*. Nas tabelas 46 e 47 estão apresentadas as *Haralick features* calculadas, segundo a orientação 0º, para d=1, d=5 e d=10.

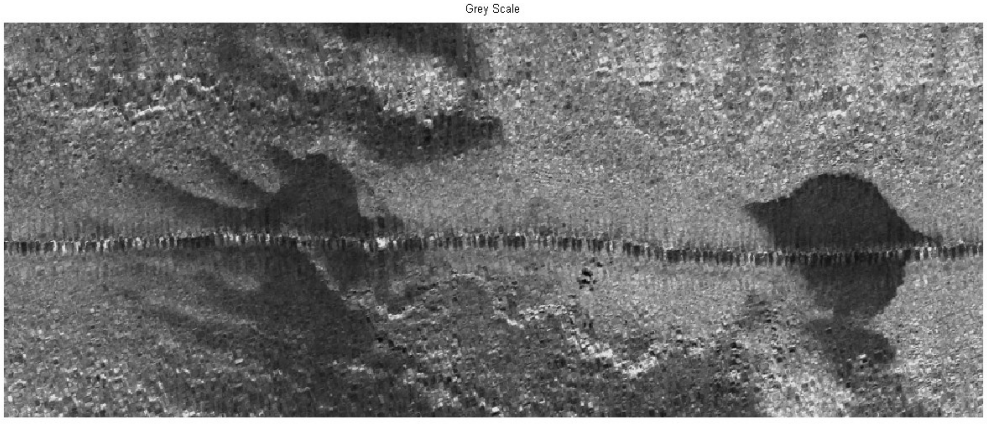


Figura 36 - Imagem em escala de cinzentos (8 bits) do fundo marinho, obtida através de Sonar de Varrimento Lateral. *Feature* 01 (F01). A figura apresenta uma área de aproximadamente 500m x 250m.

Tabela 31 - *Haralick Feature Energy* calculada para *d*=1, *d*=5 e *d*=10.

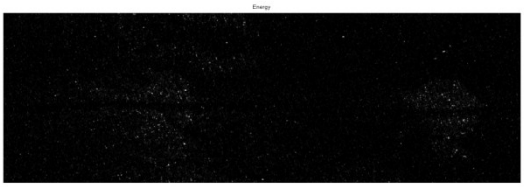
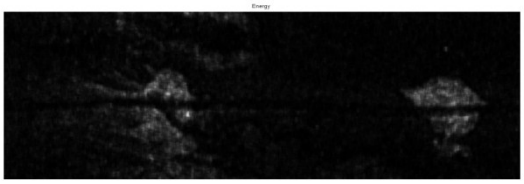
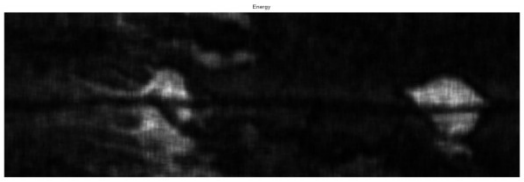
Haralick Feature – Energy, feature 02 (F02)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 32 - *Haralick Feature Correlation* calculada para *d*=1, *d*=5 e *d*=10.

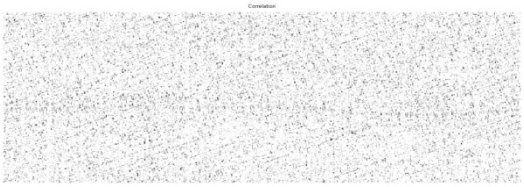
Haralick Feature – Correlation, feature 03 (F03)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 33 - *Haralick Feature Contrast* calculada para *d*=1, *d*=5 e *d*=10.

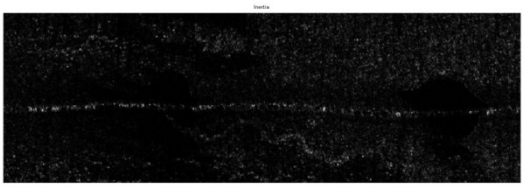
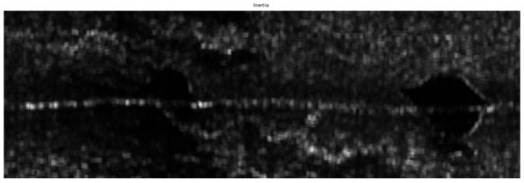
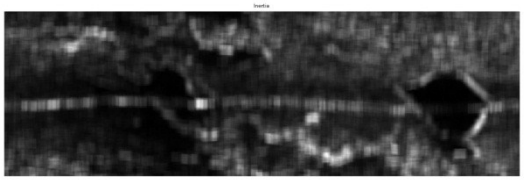
Haralick Feature – Contrast, feature 04 (F04)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 34 - *Haralick Feature Entropy* calculada para *d*=1, *d*=5 e *d*=10.

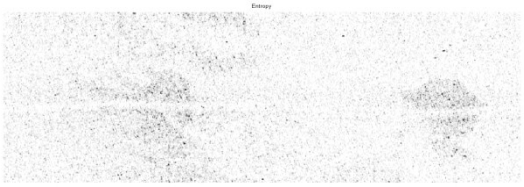
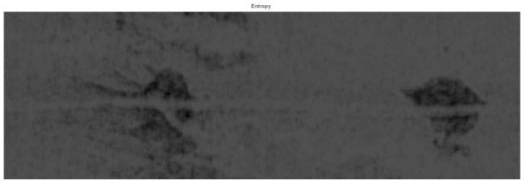
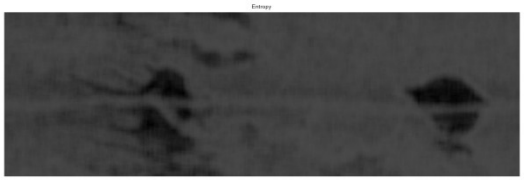
Haralick Feature – Entropy, feature 05 (F05)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 35 ***Haralick Feature Inverse Difference Moment*** **calculada para *d*=1, *d*=5 e *d*=10.**

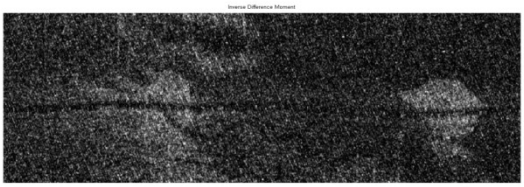
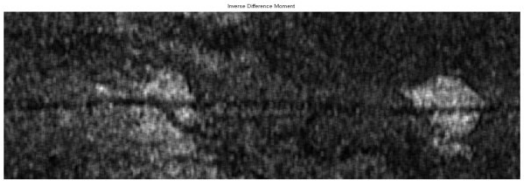
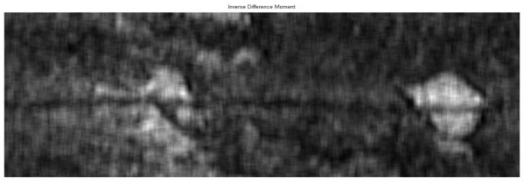
Haralick Feature – Inverse Difference Moment, feature 06 (F06)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 36 - ***Haralick Feature Sum Average*** **calculada para *d*=1, *d*=5 e *d*=10.**

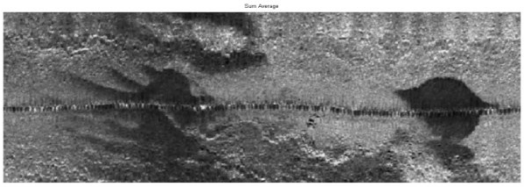
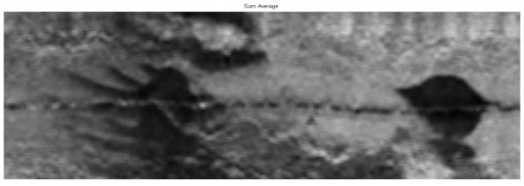
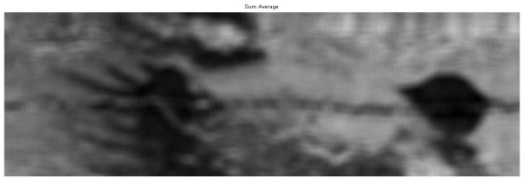
Haralick Feature – Sum Average, feature 07 (F07)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 37 - *Haralick Feature Sum Variance* calculada para *d*=1, *d*=5 e *d*=10.

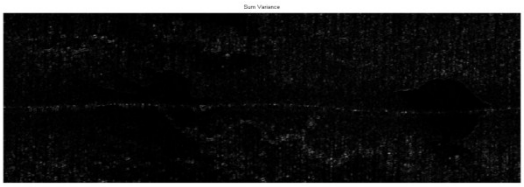
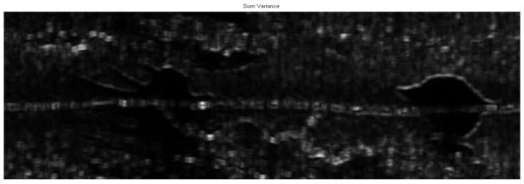
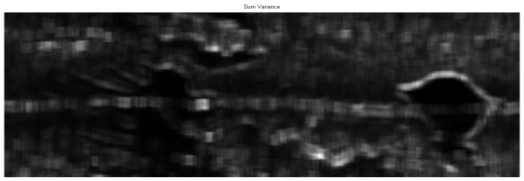
Haralick Feature – Sum Variance, feature 08 (F08)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 38 - *Haralick Feature Sum Entropy* calculada para *d*=1, *d*=5 e *d*=10.

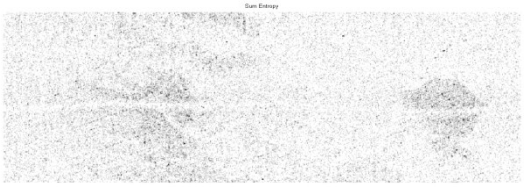
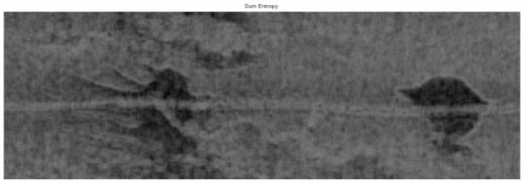
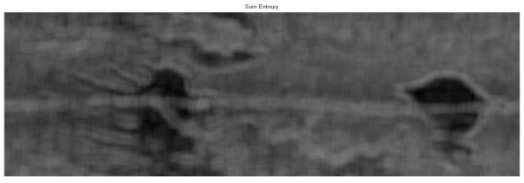
Haralick Feature – Sum Entropy, feature 09 (F09)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 39 - *Haralick Feature Difference Average* calculada para *d*=1, *d*=5 e *d*=10.

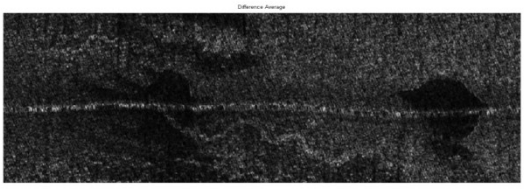
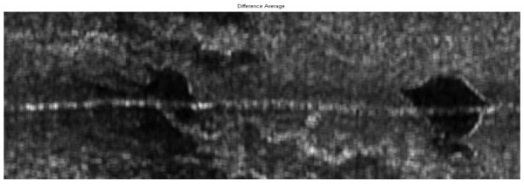
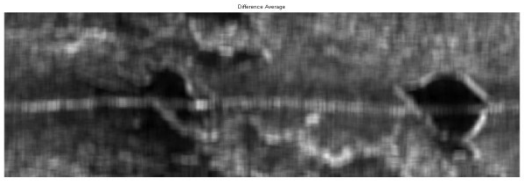
Haralick Feature – Difference Average, feature 10 (F10)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 40 - *Haralick Feature Difference Variance* calculada para *d*=1, *d*=5 e *d*=10.

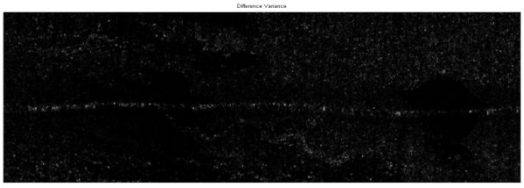
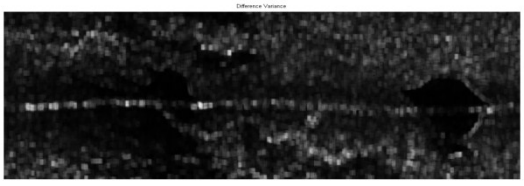
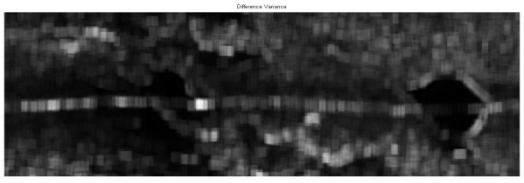
Haralick Feature – Difference Variance, feature 11 (F11)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 41 - *Haralick Feature Difference Entropy* calculada para *d*=1, *d*=5 e *d*=10.

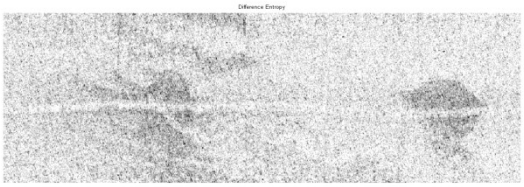
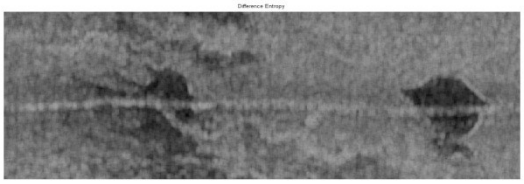
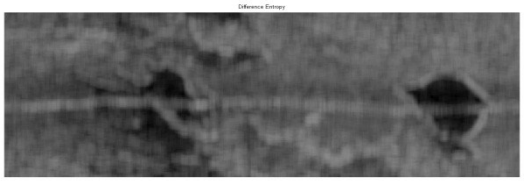
Haralick Feature – Difference Entropy, feature 12 (F12)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 42 - *Haralick Feature Information Measure of Correlation 1* calculada para *d*=1, *d*=5 e *d*=10.

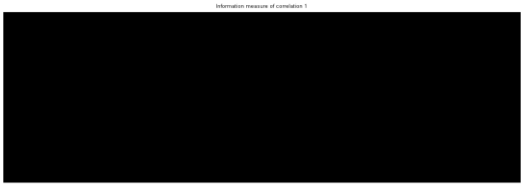
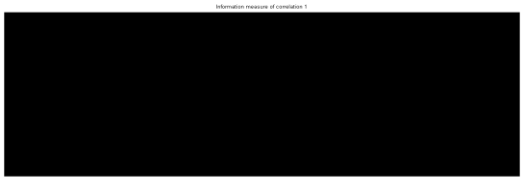
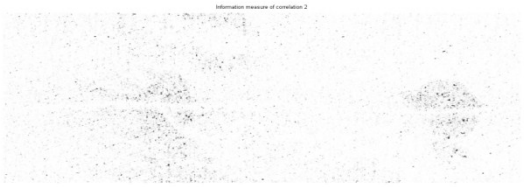
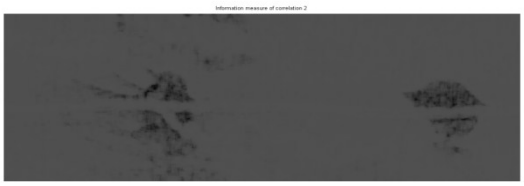
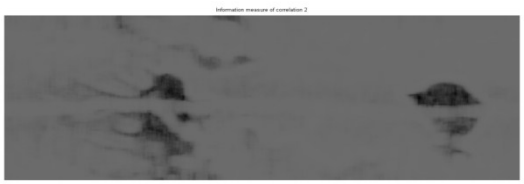
Haralick Feature – Information Measure of Correlation 1, feature 13 (F13)	
***d*=1**	
***d*=5**	
***d*=10**	

Tabela 43 - *Haralick Feature Information Measure of Correlation 2* calculada para *d*=1, *d*=5 e *d*=10.

Haralick Feature – Information Measure of Correlation 2, feature 14 (F14)	
***d*=1**	
***d*=5**	
***d*=10**	

6.2. Critérios e Metodologia propostos para a selecção das *features*

Após o cálculo da informação mútua, foram obtidos os resultados apresentados nas seguintes tabelas, correspondendo a *d*=1, *d*=5 e *d*=10, respectivamente.

Tabela 44 - Resultados da informação própria para *d*=1

Nº da *Feature*	Informação mútua (*self-information*)
1	25,5995
2	-2,2956
3	3,1211
4	13,3189
5	-2,3626
6	23,5526
7	25,7929
8	9,1283
9	-5,5154
10	21,5090
11	10,9364
12	-1,2085
13	-12,6136
14	4,7292

Tabela 45 - Resultados da informação própria para *d*=5

Nº da Feature	Informação mútua (self-information)
1	25,4270
2	11,2053
3	17,3482
4	18,9538
5	10,1416
6	21,8021
7	25,3960
8	17,1270
9	18,6056
10	22,8599
11	20,0346
12	20,6483
13	-12,0047
14	-0,2438

Tabela 46 - Resultados da informação própria para *d*=10

Nº da Feature	Informação mútua (self-information)
1	24,3527
2	14,2052
3	18,9972
4	22,4170
5	11,9372
6	21,6767
7	24,7529
8	19,6125
9	18,3227
10	23,6733
11	21,6148
12	19,8466
13	-11,2639
14	4,4737

Analisando os resultados obtidos e fazendo uma comparação entre estes e as várias *Haralick features*, verificou-se que existia uma correspondência. De facto é possível observar que valores mais elevados de informação própria correspondem às *features* que contêm mais características identificativas dos diversos elementos presentes na imagem original em escala de cinzentos.

Contudo verificaram-se situações não desejáveis ao processo de selecção pois algumas *features* não acrescentavam informação adicional àquela fornecida pela imagem original. Um exemplo claro desta situação é a *feature* número 7 (*sum average*), onde é patente a similaridade com a imagem em escala de cinzentos; não haveria qualquer vantagem em utilizá-la para *matching*, apesar do elevado valor de informação própria. A utilização desta *feature* em particular constituiria uma redundância que apenas provocaria um aumento indesejável do peso computacional.

Desta forma estabeleceram-se, para além dos valores de informação própria, outros critérios de classificação com o objectivo de ser obtida uma métrica passível de ser utilizada como referência aquando da escolha das *Haralick features*.

Os critérios adicionalmente definidos são a Complementaridade, a Invariância com a Orientação e o Conteúdo de Informação. Com o critério Complementaridade pretende-se estabelecer se uma dada *feature* introduz e/ou complementa a informação já presente na imagem original em escala de cinzentos. Com o critério Invariância com a Orientação pretende-se que os elementos presentes nas diversas *features* sejam constantes independentemente da orientação segundo a qual estas são calculadas. Com o critério Conteúdo de Informação pretende-se, à semelhança do que se passa com a informação própria, obter uma medida da quantidade de características identificativas dos diversos elementos da imagem original. De facto este último critério idealmente corresponderá a uma validação visual dos valores obtidos de informação própria.

É importante referir que estes critérios resultam de uma observação e comparação visual entre as diversas *features*, calculadas para os diferentes *deltas*, estando portanto sujeitas a erros por parte do observador.

Como forma de unificar estes critérios segundo uma métrica, definiram-se duas fórmulas que traduzem o peso que cada critério vai ter na classificação final das *features*. Uma delas estabelece uma classificação por inspecção visual das várias *features*; outra junta a esta informação a informação proveniente do cálculo da informação própria. Desta forma, relacionam-se os critérios visuais com os critérios estatísticos, formando uma métrica verossímil.

De seguida são apresentadas as fórmulas, tendo em conta a notação considerada: C (Complementaridade), O (Invariância com a Orientação), I (Conteúdo de Informação) e SI (Informação própria – *self information*).

$$Classificação\ Visual = I * (1 + C * O) * C$$
$$Classificação\ Visual\ e\ Estatistica = I * (1 + C * O) * C + SI * C$$

Nestas fórmulas pretende-se evidenciar a influência do critério Complementaridade, uma vez que com a informação adicional fornecida pelas *Haralick features* pretende-se completar a informação presente na escala de cinzentos. Se uma dada *feature* não trouxer um conteúdo informativo distinto do existente então essa *feature* não pode ser considerada de grande utilidade, daí o peso atribuído ao critério Complementaridade. A segunda fórmula constitui um reforço da primeira por inclusão do critério estatístico Informação Própria.

De seguida são apresentados os resultados obtidos para *d*=1, *d*=5 e *d*=10. Nas tabelas figura também a classificação atribuída às diversas *features* segundo os 3 critérios visuais considerados e organizadas segundo o critério *Classificação Visual e Estatística* (nas tabelas representado pela coluna Classificação Visual + SI).

Tabela 47 - Classificação das *Haralick Features*, para *d*=1

Escala: 1-5(1 menor, 5 maior)		d=1				Resultados	
Feature	**Nº**	**Conteúdo de Informação**	**Invariância com orientação**	**Complementaridade**	**Info. própria**	**Classif. Visual**	**Classif. Visual + SI**
Info. Measure of Correl. 1	13	1	5	1	-12,6136	6,0	-6,6
Correlation	3	2	2	2	3,1211	20,0	26,2
Sum Entropy	9	2	5	2	-5,5154	44,0	33,0
Entropy	5	2	5	2	-2,3626	44,0	39,3
Difference Variance	11	2	2	2	10,9364	20,0	41,9
Sum Variance	8	2	3	2	9,1283	28,0	46,3
Info. Measure of Correl. 2	14	2	5	2	4,7292	44,0	53,5
Sum Average	7	5	5	1	25,7929	30,0	55,8
Difference Entropy	12	3	5	2	-1,2085	66,0	63,6
Energy	2	2	5	3	-2,2956	96,0	89,1
Difference Average	10	4	3	2	21,5090	56,0	99,0
Inverse Differ. Moment	6	3	4	2	23,5526	54,0	101,1
Contrast	4	3	4	3	13,3189	117,0	157,0

Tabela 48 - Classificação das *Haralick Features*, para *d*=5

Escala: 1-5(1 menor, 5 maior)		d=5				Resultados	
Feature	**Nº**	**Conteúdo de Informação**	**Invariância com orientação**	**Complementaridade**	**Info. própria**	**Classif. Visual**	**Visual+ Self.Inf.**
Info. Measure of Correl. 1	13	1	5	1	-12,0047	6,0	-6,0
Correlation	3	1	4	1	17,3482	5,0	22,3
Info. Measure of Correl. 2	14	2	5	2	-0,2438	44,0	43,5
Sum Average	7	4	5	1	25,3960	24,0	49,4
Difference Entropy	12	3	2	2	20,6483	30,0	71,3
Inverse Differ. Moment	6	3	2	2	21,8021	30,0	73,6
Difference Average	10	4	2	2	22,8599	40,0	85,7
Entropy	5	3	5	2	10,1416	66,0	86,3
Energy	2	3	5	2	11,2053	66,0	88,4
Contrast	4	3	2	3	18,9538	63,0	119,9
Sum Entropy	9	3	3	3	18,6056	90,0	145,8
Difference Variance	11	3	3	3	20,0346	90,0	150,1
Sum Variance	8	4	3	4	17,1270	208,0	276,5

Tabela 49 - Classificação das *Haralick Features*, para *d*=10

Escala: 1-5(1 menor, 5 maior)		d=10				Resultados	
Feature	**Nº**	**Conteúdo de Informação**	**Invariância com orientação**	**Complementaridade**	**Info. própria**	**Classif. Visual**	**Visual+ Self.Inf.**
Info. Measure of Correl. 1	13	1	5	1	-11,2639	6,0	-5,3
Correlation	3	2	1	3	18,9972	24,0	81,0
Entropy	5	3	5	2	11,9372	66,0	89,9
Inverse Differ. Moment	6	3	1	3	21,6767	36,0	101,0
Difference Entropy	12	3	2	3	19,8466	63,0	122,5
Sum Average	7	4	5	2	24,7529	88,0	137,5
Info. Measure of Correl. 2	14	3	5	3	4,4737	144,0	157,4
Energy	2	3	4	3	14,2052	117,0	159,6
Sum Entropy	9	4	4	3	18,3227	156,0	211,0
Difference Average	10	4	2	4	23,6733	144,0	238,7
Difference Variance	11	3	3	4	21,6148	156,0	242,5
Sum Variance	8	4	2	5	19,6125	220,0	318,1
Contrast	4	4	2	5	22,4170	220,0	332,1

Analisando os resultados das tabelas e comparando-os com as imagens de cada uma das *Haralick features* constata-se que existe uma maior coerência entre os valores obtidos e aquilo que é realmente observado nos casos *d*=5 e *d*=10. De facto é facilmente observável que para *d*=1 é mais difícil distinguir os pormenores presentes nas várias *Haralick features* fazendo com que exista uma maior ambiguidade e incoerência dos resultados obtidos.

Uma conclusão passível de ser retirada desta classificação é que os melhores casos se verificam para *d*=5. De facto, é observável que as *features Sum Variance* e *Difference Variance* se apresentam como aquelas que melhor complementam a imagem em escala de cinzentos, realçando também as características presentes. Outras *features* como *Contrast* (Contraste, previamente utilizado) ou mesmo *Sum Entropy* apresentam também uma classificação elevada, corroborada pela observação das próprias *features*. No entanto, é fundamental ter em consideração o facto, já mencionado, de que esta classificação resulta de análises visuais sujeitas a erro, sendo portanto fundamental estabelecer critérios e proceder de acordo com estes.

Esta classificação, mais do que uma análise das diversas *Haralick Features*, representa a necessidade de se estabelecer uma metodologia fundamentada para a escolha das melhores *features* para o problema apresentado. Neste caso (imagens de

SSS), concluiu-se que as duas “melhores *features*” eram *Sum Variance* e *Difference Variance* para *d*=5. Contudo, é necessário ter presente que apenas realizando testes de *matching* com dados de sonar é possível confirmar os resultados obtidos.

É também importante referir que a metodologia utilizada pode não ser passível de ser generalizada para qualquer tipo de imagens em escala de cinzento; mesmo para imagens de SSS pode realizar-se uma análise diferente consoante os critérios pretendidos, assim como a aplicação a que se destina.

7. Testes com dados de Sonar de Varrimento Lateral

Neste capítulo são apresentados os testes com dados obtidos através de sonar de varrimento lateral.

Ainda não tinha sido feito qualquer teste com este tipo de dados devido ao peso computacional do algoritmo de cálculo de informação mútua, uma vez que com a máquina utilizada os tempos de execução estimam-se que fossem da ordem das dezenas de horas.

A solução encontrada, e que permitiu elaborar os testes apresentados neste capítulo, partiu da possibilidade de se usar as capacidades de computação paralela das placas gráficas Nvidia®. A arquitectura que possibilita o uso de GPU (*Graphics Processing Unit*) para computação paralela é chamada CUDA (*Compute Unified Device Architecture*) [17]. Além disso, as versões mais recentes do Matlab permitem (através da já mencionada Parallel Computing Tool Box™) [18] utilizar as capacidades de computação CUDA de forma relativamente transparente para o utilizador.

No entanto o que permitiu de facto ter uma plataforma computacional que permitisse executar testes exigentes com dados reais de SSS foi a ferramenta Jacket desenvolvida pela empresa AccelerEyes® [19, 20]. Esta ferramenta funciona como uma ligação entra a linguagem M (do Matlab) e a arquitectura CUDA da GPU através de uma extensa biblioteca de funções cujo funcionamento, do ponto de vista do utilizador, é exactamente igual ao de vulgares funções Matlab. Desta forma o utilizar pode desenvolver um pograma em Matlab da forma usual. Quando se pretende que um determinado trecho de código seja executado paralelamente basta sinalizar as variáveis que têm de ser armazenadas na memória da GPU.

7.1. *Matching* usando Informação Mútua e *Normalized Cross-Correlation* (NCC)

Foi então utilizado como mapa, para os testes realizados de seguida, a imagem da Figura 37. Como *template* foi escolhida uma região deste mapa, como se ilustra na Figura 38.

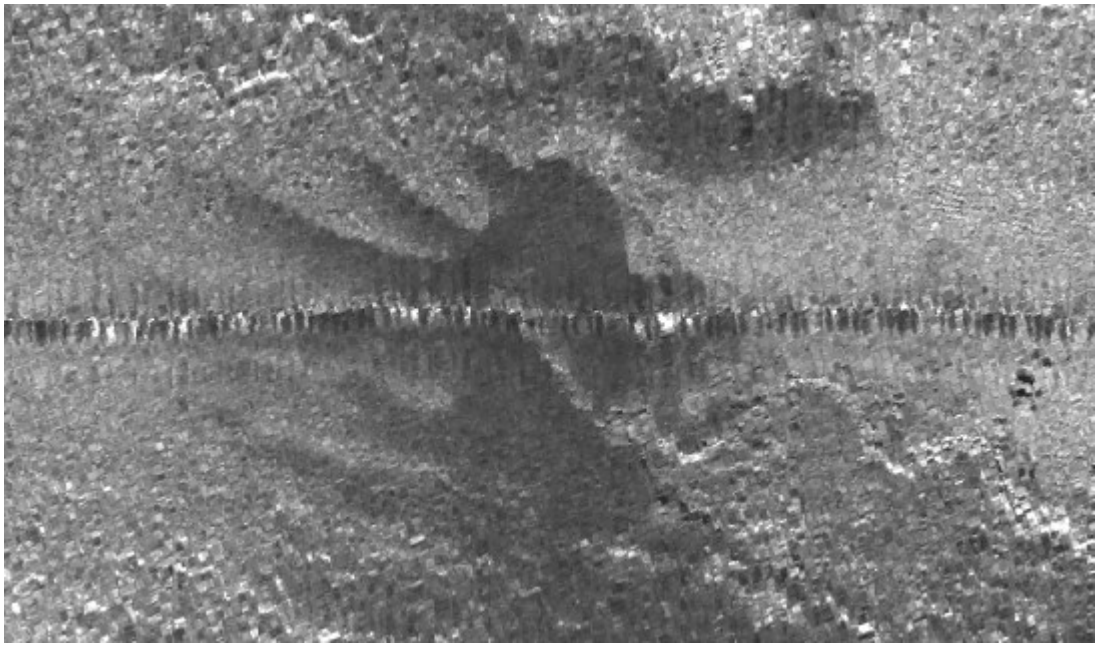


Figura 37 – Imagem obtida de SSS, tamanho 535x309 pixeis.

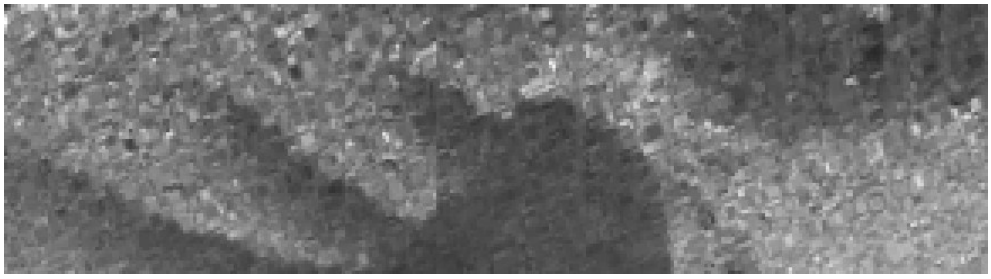


Figura 38 – *Template* simples, de tamanho 281x70 pixeis.

Dado que a Correlação Cruzada [21-24] é uma das técnicas mais utilizadas em problemas de *matching*, a comparação desta técnica com o método de cálculo da informação mútua é de extrema importância.

Assim foram elaborados um conjunto de testes com o objectivo não só de comparar os dois métodos, mas também simular situações passíveis de ocorrer na aquisição de dados de SSS e com efeitos prejudiciais para o *matching*, tendo sido, para este efeito, introduzidas modificações no *template*.

A principal alteração ao *template* foi a introdução de uma faixa central de ruído cuja finalidade é simular a faixa correspondente à zona situada na vertical do sonar e que fica alinhada com a trajectória do veículo (peixe). Na Figura 37 observa-se claramente a faixa de sentido horizontal. No *template* é adicionada uma faixa idêntica mas por forma a descrever o deslocamento na direcção Norte-Sul (perpendicular à faixa registada no mapa).

Foi também modificado o *template* por forma a simular um caso quando existe uma ligeira rotação deste em relação ao mapa. Escolheu-se para o efeito uma rotação de 5º, dado que numa aplicação real recorre-se a métodos de minimização deste tipo de efeito (alinhamentos).

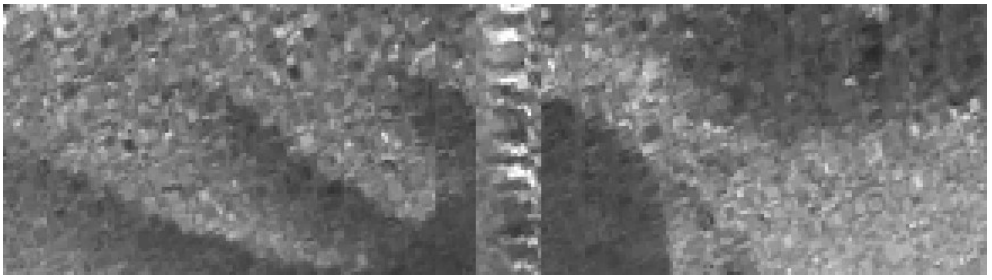
A utilização de um *template* com uma resolução inferior tem como objectivo representar a situação em que o este é adquirido a uma distância do fundo marinho diferente daquela a que é adquirido o mapa. Este problema também pode ocorrer quando se usam sonares com frequências diferentes.

Uma outra situação comum em sistemas de SSS é a diferença de velocidade do veículo em relação à velocidade de aquisição dos dados. Tal provoca uma imagem "alongada" ou "comprimida" consoante a velocidade de aquisição de dados é inferior ou superior à de deslocação.

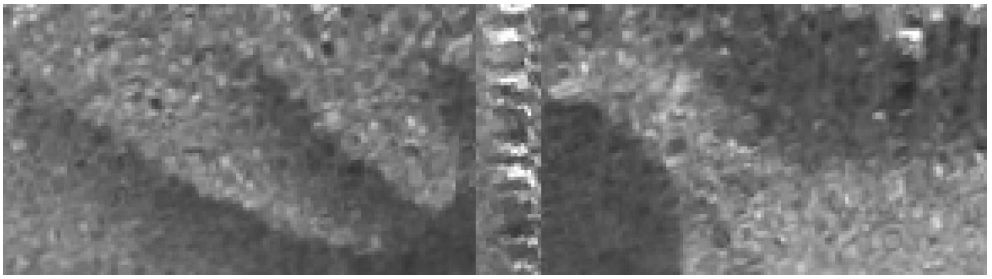
Foi também simulada a adição de ruído de alta intensidade ao *template*. E por fim testou-se uma situação altamente prejudicial que consiste na combinação das perturbações mais prováveis. O último teste utiliza então um *template* "alongado", rodado 5º e com uma resolução menor que a do mapa.

Importa salientar que apenas no primeiro teste (usando o *template* simples) é que a faixa ruidosa não foi acrescentada.

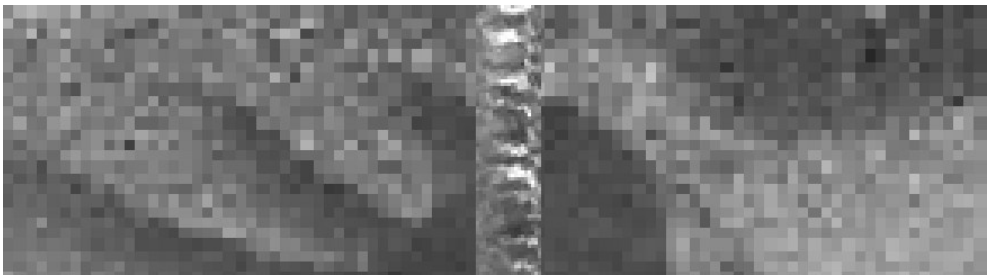
A Figura 39 ilustra as várias modificações mencionadas.



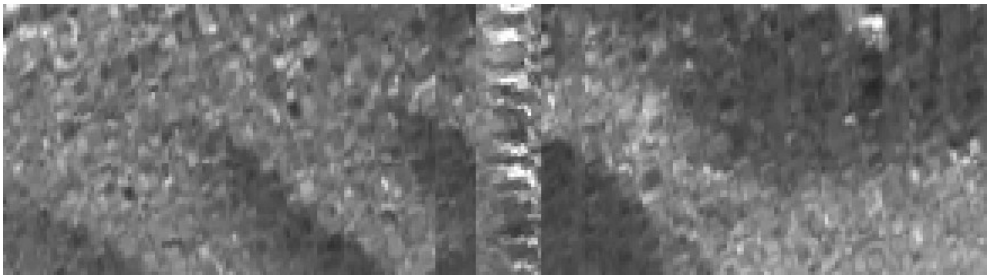
a)



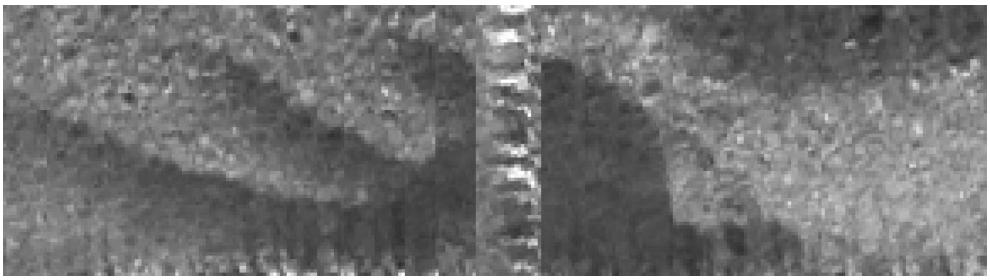
b)



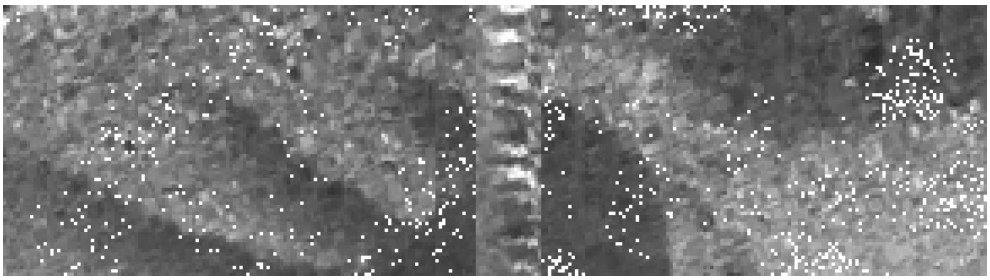
c)



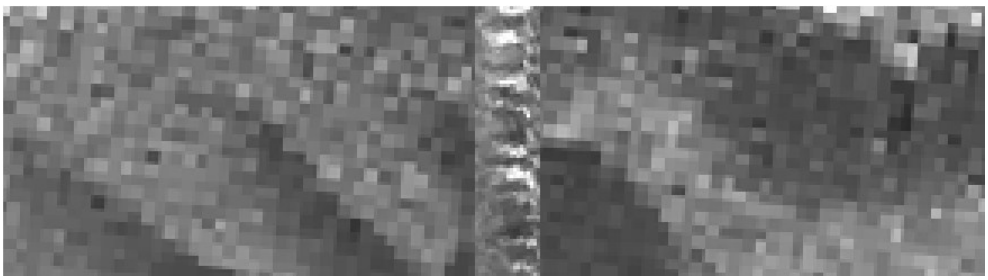
d)



e)



f)

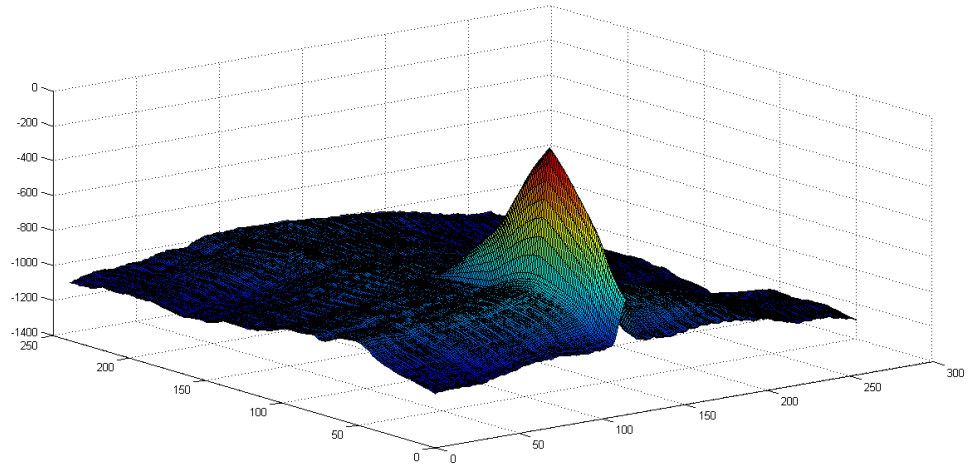


g)

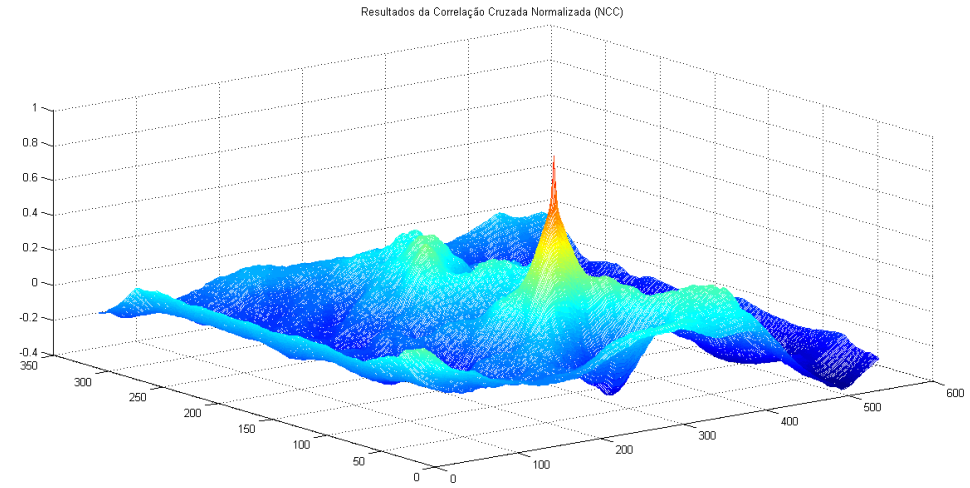
Figura 39 – *Templates* com modificações adicionadas: a) faixa central; b) rotação de 5º; c) resolução inferior; d) alongado; e) comprimido; f) com ruído de alta intensidade; g) alongado + rotação 5º + resolução inferior.

Os testes comparativos foram realizados utilizando os dois algoritmos: kNN-batch com o parâmetro M=40 (uma vez que o tamanho dos dados assim o permite) e NCC (*Normalized Cross-Correlation*). As *Haralick features* utilizadas no cálculo da IM foram as seleccionadas no capítulo anterior (*Sum Variance e Difference Variance*) para *d*=5.

Foi também calculada a distância euclidiana entre as coordenadas obtidas e as previstas por forma a obter uma métrica passível de comparar os resultados entre IM e NCC.

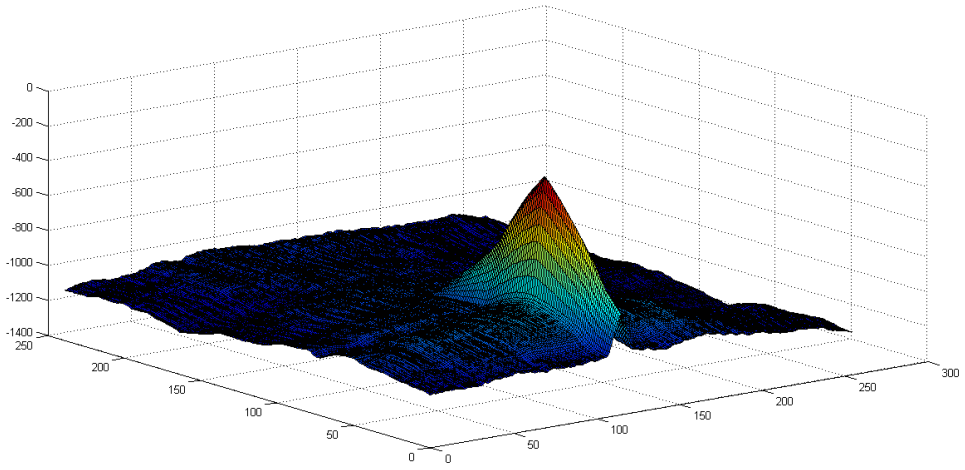


a)

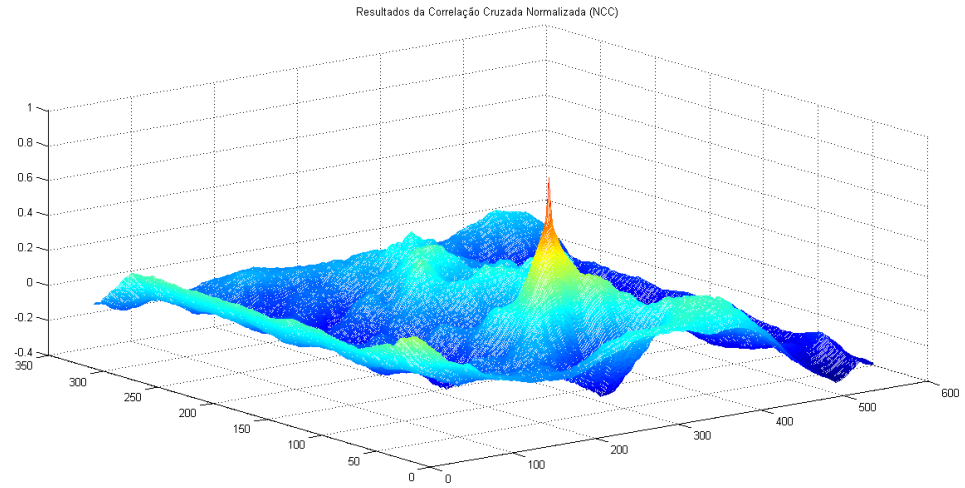


b)

Figura 40 – Testes com *template* simples: a) resultados IM; b) resultados NCC.

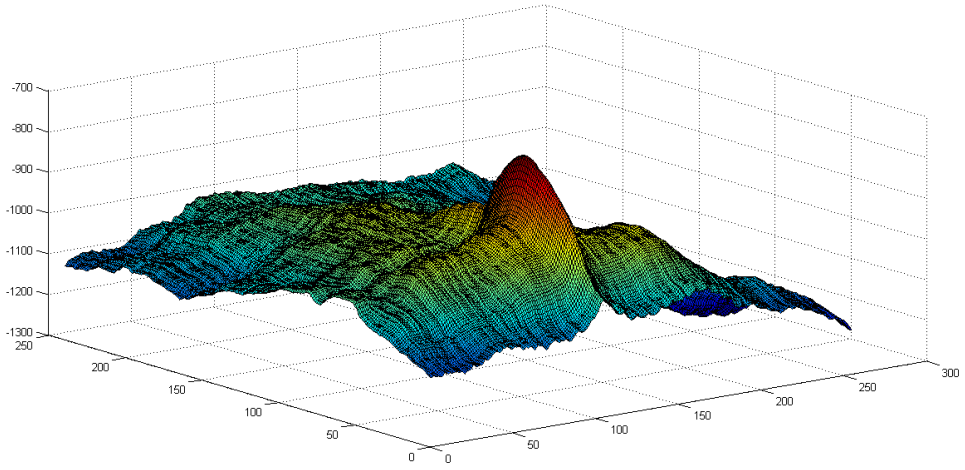


a)

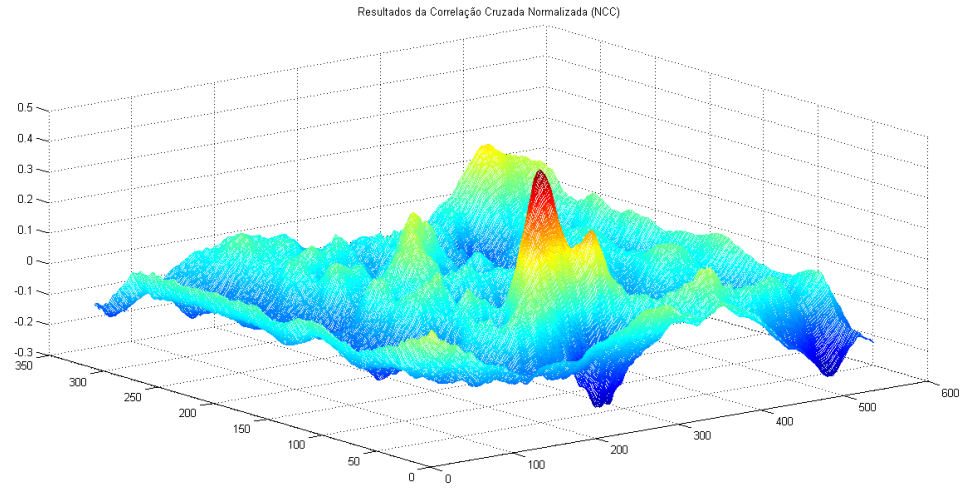


b)

Figura 41 – Testes com *template* com faixa central: a) resultados IM; b) resultados NCC.

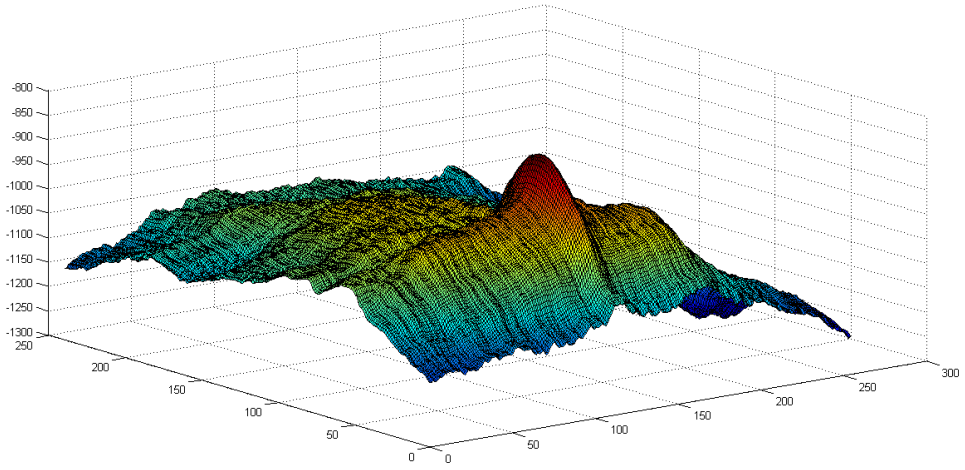


a)

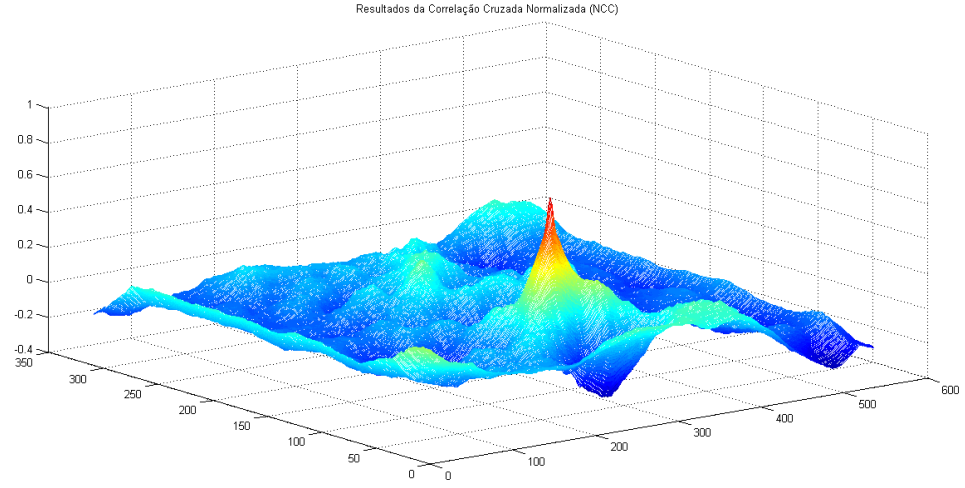


b)

Figura 42 – Testes com *template* rodado 5º: a) resultados IM; b) resultados NCC.

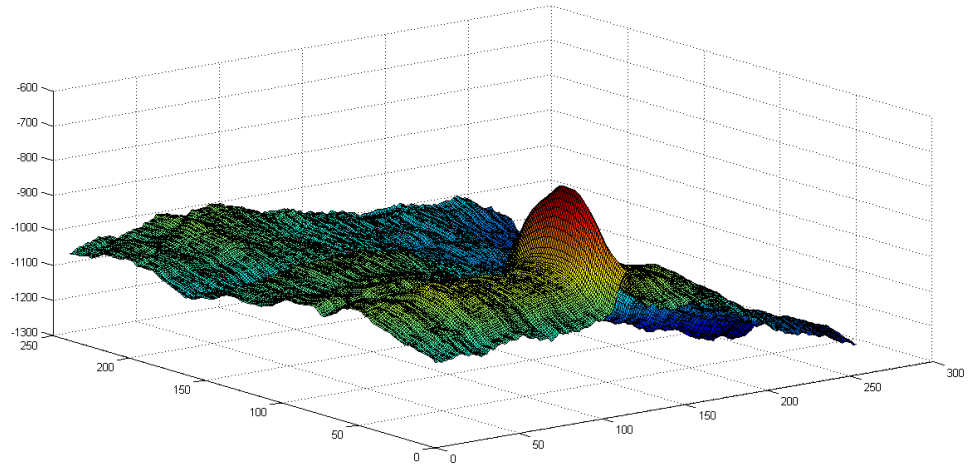


a)

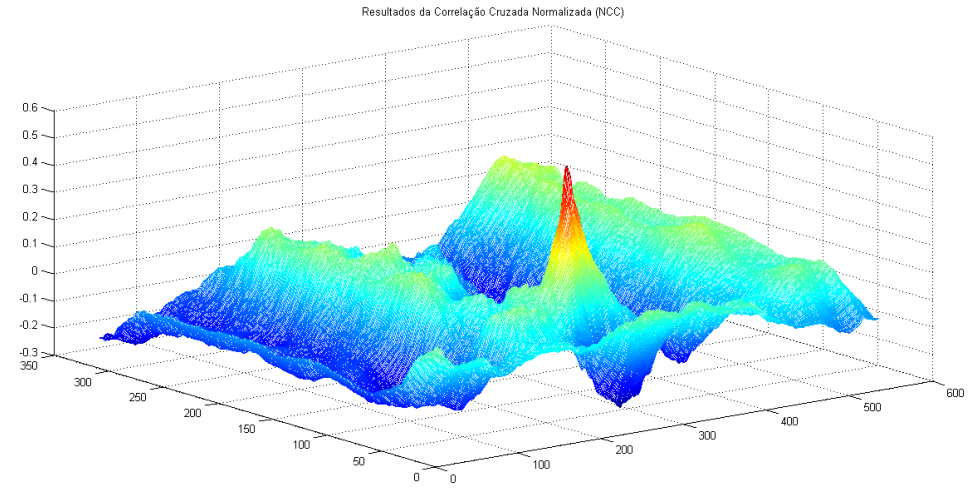


b)

Figura 43 – Testes com *template* com resolução inferior: a) resultados IM; b) resultados NCC.

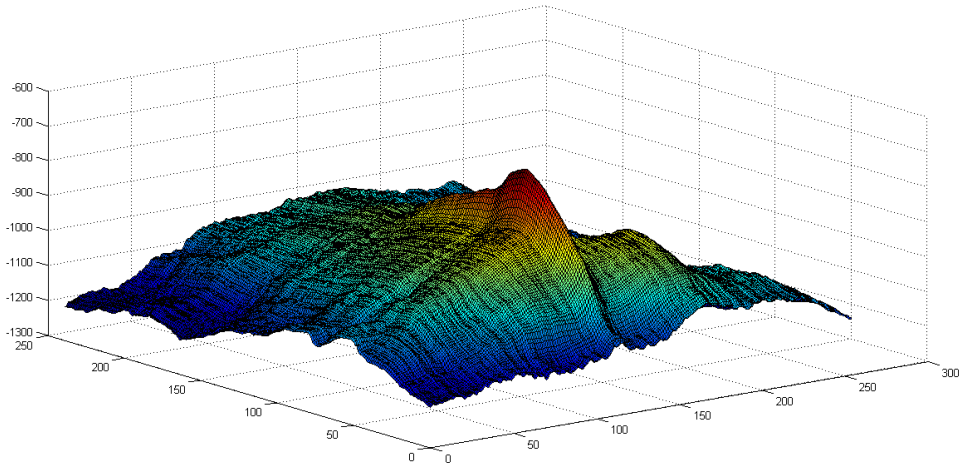


a)

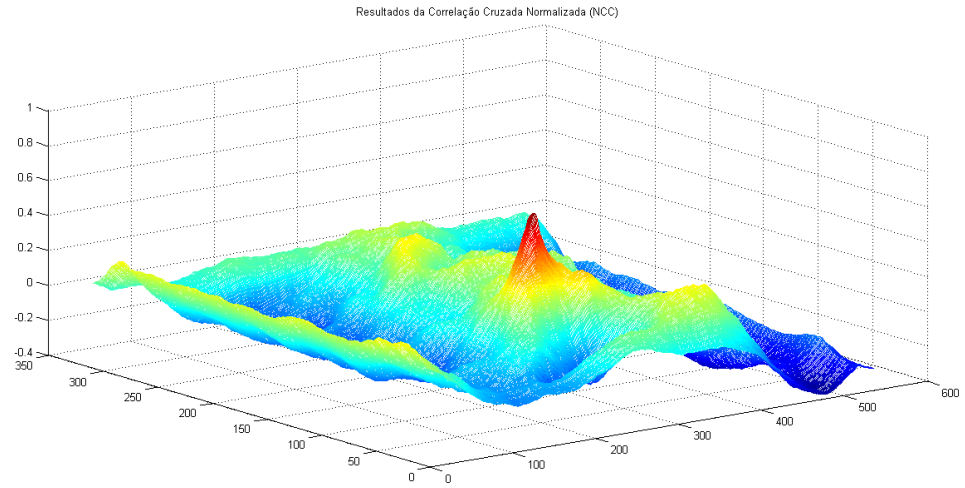


b)

Figura 44 – Testes com *template* alongado: a) resultados IM; b) resultados NCC.

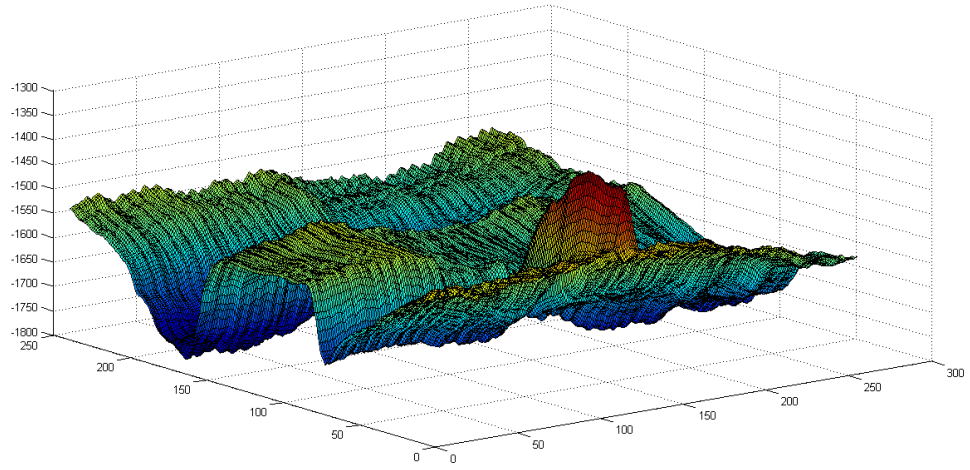


a)

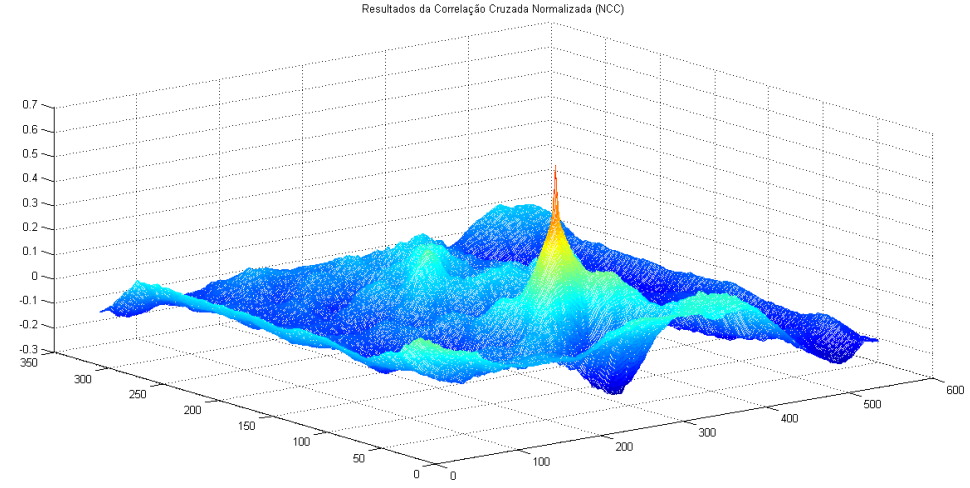


b)

Figura 45 – Testes com *template* comprimido: a) resultados IM; b) resultados NCC.

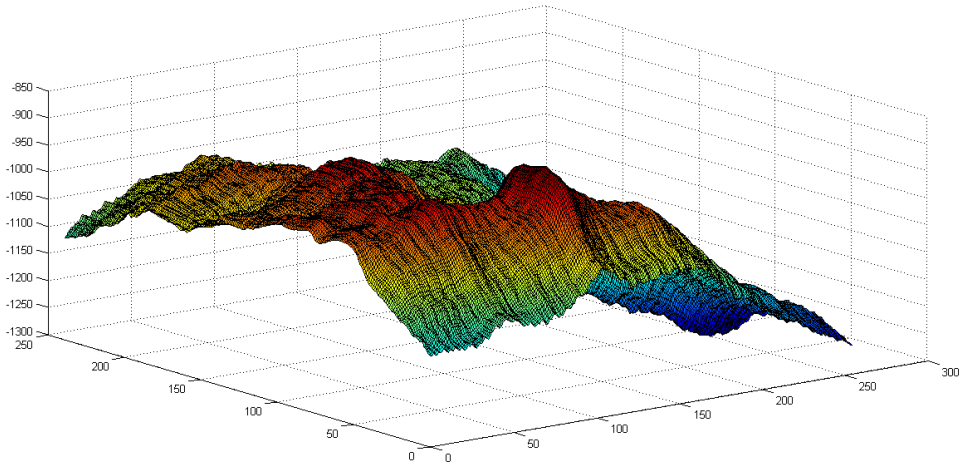


a)

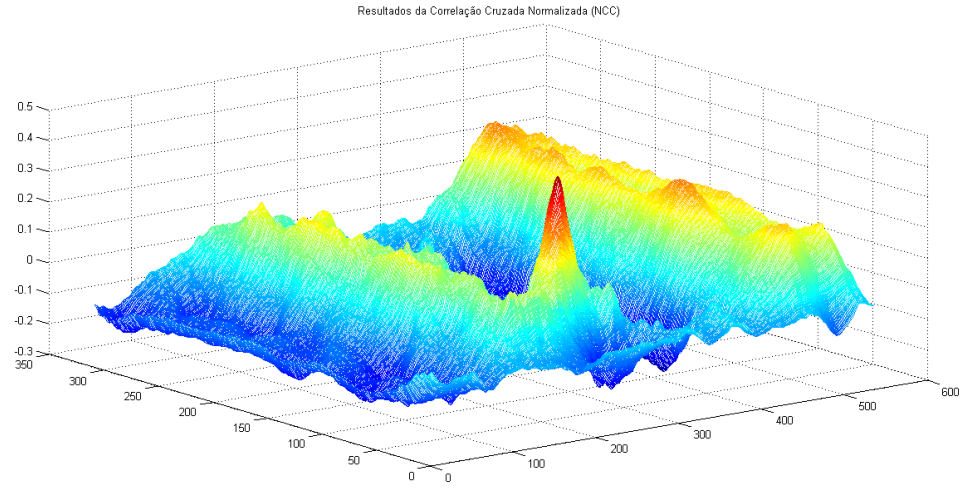


b)

Figura 46 – Testes com *template* com ruído de alta intensidade: a) resultados IM; b) resultados NCC.



a)



b)

Figura 47 – Testes com *template* alongado + rotação 5º + resolução inferior: a) resultados IM; b) resultados NCC.

Tabela 50 – Comparação dos resultados obtidos com IM e NCC.

Template	Coordenadas				Métrica (distância euclidiana)	
	previstas IM (x,y)	obtidas IM (x,y)	previstas NCC (x,y)	obtidas NCC (x,y)	IM	NCC
simples	115,50	115,50	115,50	115,50	0	0
com faixa central	115,50	115,50	115,50	115,50	0	0
rodado 5º	115,50	108,55	115,50	100,48	8,6023	15,1327
com resolução baixa	115,50	115,51	115,50	110,45	1	7,0711
alongado	115,50	115,41	115,50	110,35	9	15,8114
comprimido	115,50	115,62	115,50	110,59	12	10,2956
com ruído	115,50	115,24	115,50	110,45	26	7,0711
alongado + 5º + resolução baixa	115,50	108,41	115,50	100,31	11,4018	24,2074

No geral, o aspecto dos gráficos sugere que a NCC dá resultados mais precisos pois o pico de *matching* resultante é melhor definido do que o da IM. No entanto, tal não corresponde à realidade, além de que em alguns testes com NCC encontraram-se máximo locais, o que não ocorre nos resultados de *matching* com IM.

Na maioria dos testes observa-se que *matching* com IM é significativamente mais exacto do que com NCC; em geral o erro de localização (métrica) obtido com IM é 2 vezes menor.

Outro resultado proveniente dos testes com IM é o facto de estes crescerem monotonicamente para o pico na maioria dos testes. Tal sugere que se poderá encontrar um método baseado em gradientes ascendentes para determinação do máximo de IM, reduzindo assim o peso computacional exigido pelo algoritmo.

Verificou-se que a adição de uma faixa central altamente ruidosa não afecta o *matching*, quer com IM quer com NCC. Isto é algo de muito positivo pois a existência dessa faixa é inevitável nos dados SSS, significando que os tipos de dados e métodos de *matching* aqui testados são adequados à resolução do problema de navegação baseada no terreno.

No que respeita a rotações do *template*, o método de IM também demonstra ser mais robusto que o de NCC.

Um dos resultados mais importantes e reveladores da robustez do método de IM é o do teste com alteração da resolução. De facto, o erro obtido com IM é mínimo; o erro obtido com NCC é 7 vezes maior. Este resultado é muito importante porque representa um dos cenários mais prováveis de ocorrer na prática.

No caso com o *template* alongado mais uma vez se obteve um erro de *matching* menor com IM. Contudo, no caso do *template* comprimido, os resultados obtidos de IM e NCC não são significativamente diferentes.

O único caso onde a diferença de resultados entre IM e NCC é bastante significativa é no teste realizado com *template* com ruído de alta intensidade (Figura 39 f)). Uma das causas prováveis pode residir no facto de que as *Haralick features* deste *template* sejam muito alteradas pelo tipo de ruído. Resultados mais detalhados implicariam a realização de novos testes com diferentes *Haralick features* e com ruído de características diferentes.

Por fim o resultado do último testes (combinação de perturbações mais significativas e com maior probabilidade de ocorrência em situações reais), mostra mais uma vez a elevada robustez do método de IM pois, apesar das distorções a que os dados

foram sujeitos, obtém um erro de localização que é menos de metade do erro obtido com NCC.

Em resumo, pode dizer-se que os resultados obtidos com IM mostram que o método é aplicável à resolução do problema de localização com base em dados SSS e que este método se revela superior ao de NCC em termo de robustez na presença de transformações não lineares dos dados dos *templates* e do erro de localização no mapa. Em rigor os resultados com NCC parecem ser mais precisos, mas revelam-se menos exactos do que os obtidos com IM. Na verdade, o ideal seria ter as duas características (precisão e exactidão) no mesmo estimador, não sendo possível, neste caso a exactidão é preferível.

8. Conclusões e trabalho futuro

O trabalho desenvolvido tinha como objectivo estudar a aplicabilidade de um método de *matching* baseado no conceito de informação mútua com aplicabilidade ao problema da navegação de veículos autónomos subaquáticos.

Estudaram-se inicialmente estimadores de entropia e testaram-se diversas situações que viram aperfeiçoar o algoritmo de *matching*. Conclui-se que o método de estimação de entropia proposto por Kybic se revela como uma solução robusta e com um nível de exactidão bastante alto.

Foram também realizados testes de *matching* usando imagens multidimensionais onde a informação fornecida pela escala de cinzentos é complementada pela introdução do cálculo de *Haralick features*.

Foi feita uma análise das diversas *Haralick features* calculadas para dados de SSS. Desta análise implementou-se um método de classificação das mesmas com o propósito de se ter uma forma sistemática e concreta de seleccionar quais as *features* que melhor se adequavam aos tipos de dados.

Finalmente, foram realizados testes de *matching* com dados de SSS de dimensões consideráveis, usando os métodos baseado em IM e correlação cruzada normalizada (NCC). Concluiu-se que apesar da elevada precisão dos resultados obtidos com NCC, utilizando o método de IM os resultados obtidos são muito mais exactos apresentando também uma robustez muito maior na presença de transformações indesejadas.

Num trabalho futuro é imperativo optimizar o algoritmo de *matching* segundo o método de Kybic. O ideal é conseguir efectuar o *matching* em tempo real (ou bastante próximo). Para tal poderá explorar-se a utilização de técnicas baseadas em determinação de gradientes evitando-se, desta forma, o varrimento sequencial (pixel a pixel) efectuado neste trabalho.

A utilização de *hardware* dedicado também é uma forte solução. Equipamentos como o DSP podem contribuir de forma muito positiva para aumentar o desempenho dos métodos aplicados ao problema da navegação. Aliando *hardware* dedicado a um algoritmo altamente optimizado torna possível obter-se uma técnica de *matching* mais precisa e robusta.

Outro aspecto a explorar poderá ser o método de selecção das *Haralick features* mais adequadas aos tipos de dados que se possuam. O método aqui proposto aparentemente adequa-se, no entanto a selecção automática das melhores *features* constituiria uma forma de aumentar fortemente a aplicabilidade do algoritmo.

9. Referências

1. Stutters, L., et al., *Navigation Technologies for Autonomous Underwater Vehicles.* IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews, 2008. **38**(4): p. 581-589.
2. Ruiz, T., Y. Petillot, and D.M. Lane. *Improved AUV navigation using side-scan sonar*. in *Oceans 2003. Proceedings*. 2003. San Diego, CA, USA.
3. Blondel, P., *The Handbook of Sidescan Sonar*. Springer-Praxis Books on Geophysical Sciences, ed. Springer2009, Chichester, UK: Praxis Publishing Ltd.
4. Thisen, E., H.B.D. Sorensen, and B. Stage. *Sidescan Sonar Image Matching Using Cross Correlation*. in *Proceedings of SPIE*. 2003.
5. Nygren, I. and M. Jansson, *Terrain Navigation for Underwater Vehicles Using the Correlator Method.* IEEE Journal of Oceanic Engineering, 2004. **29**(3): p. 906-915.
6. Viola, P. and W.M. Wells III. *Alignment by Maximization of Mutual Information*. in *Fifth International Conference on Computer Vision*. 1995. Cambridge, MA , USA.
7. Haralick, R., K. Shanmugam, and I.h. Dinstein, *Textural Features for Image Classification.* IEEE Trans. on Systems, Man, and Cybernetics, 1973. **SMC-3**(6): p. 610-621.
8. Haralick, R. and L.G. Shapiro, *Computer and Robot Vision*. Vol. I. 1992: Addison Wesley.
9. Mignotte, P.-Y., M. Lianantonakis, and Y. Petillot. *Unsupervised registration of textured images: applications to side-scan sonar*. in *Oceans' 2005 - Europe* 2005.
10. Cover, T.M. and J.A. Thomas, *Elements of Information Theory*. 2nd ed2006: Wiley.
11. Kybic, J. *High-dimensional mutual information estimation for image registration*. in *ICIP '04. 2004 International Conference on Image Processing*. 2004.
12. Faivishevsky, L. and J. Goldberger. *ICA based on a Smooth Estimation of the Differential Entropy*. 2008.
13. *K Nearest Neighbors*. Available from: http://www.mathworks.com/matlabcentral/fileexchange/15562-k-nearest-neighbors.
14. *Haralick Texture Features Matlab Toolbox v0.1b*. Available from: http://read.pudn.com/downloads114/sourcecode/graph/479658/Haralick.m .htm.
15. Isaaks, E.H. and R.M. Srivastava, *Applied Geostatistics*1989: Oxford University Press.
16. Wen, R. and R. Sinding-Larsen, *Geostatistics for Image Analysis. Theory and applications related to GLORIA sidescan sonography from the Mississippi Fan, Gulf of Mexico*, in *Geostatistics Wollongong '96*, E.Y. Baafi and N.A. Schofield, Editors. 1996. p. 1233-1243.
17. *AccelerEyes libJacket*. Available from: http://developer.nvidia.com/accelereyes-libjacket.
18. *MATLAB GPU Computing with NVIDIA CUDA-Enabled GPUs*. Available from: http://www.mathworks.com/discovery/matlab-gpu.html.
19. *JACKET*. Available from: http://wiki.accelereyes.com/wiki/index.php/JACKET.

20. *AccelerEyes*. Available from: http://www.accelereyes.com/.
21. Di Stefano, L. and S. Mattoccia. *Fast template matching using bounded partial correlation*. in *Machine Vision and Applications*. 2003.
22. Di Stefano, L. and S. Mattoccia, *A sufficient condition based on the Cauchy-Schwarz Inequality for efficient template matching.* IEEE, 2003.
23. Di Stefano, L., S. Mattoccia, and M. Mola. *An Efficient Algorithm for Exhaustive Template Matching based on Normalized Cross Correlation*. in *Proceedings of the 12th International Conference on Image Analysis and Processing*. 2003. IEEE Computer Society.
24. Sarvaiya, J.N., S. Dr. Patnaik, and S. Bombaywala. *Image Registration by template matching using Normalized Cross-Correlation*. in *International Conference on Advances in Computating, Control, and Telecommunications Technologies*. 2009.