João Pedro Teixeira Dias

APLICAÇÃO EM TEMPO-REAL PARA DETEÇÃO AUTOMÁTICA DE FRATURAS ÓSSEAS EM IMAGENS DE ULTRASSOM

UNIVERSIDADE DO ALGARVE
Faculdade de Ciências e Tecnologia
2017

João Pedro Teixeira Dias

APLICAÇÃO EM TEMPO-REAL PARA DETEÇÃO AUTOMÁTICA DE FRATURAS ÓSSEAS EM IMAGENS DE ULTRASSOM

Mestrado Integrado em Engenharia Eletrónica e Telecomunicações

Trabalho efetuado sob a orientação de: Maria da Graça Cristo dos Santos Lopes Ruano,

e,

Hélder Aniceto Amadeu de Sousa Daniel

UNIVERSIDADE DO ALGARVE
Faculdade de Ciências e Tecnologia
2017

Aplicação em tempo-real para deteção automática de fraturas ósseas em imagens de ultrassom

Declaração de autoria do trabalho

Declaro ser o(a) autor(a) deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

Copyright©

Dedicatória e agradecimentos

Agradeço à Professora Doutora Maria da Graça Ruano e ao Professor Doutor Hélder Daniel por toda a orientação, disponibilidade e apoio ao longo desta dissertação. Em especial à Professora que durante todo o tempo que frequentei a Universidade do Algarve demonstrou o seu apoio e amizade.

Agradeço a todos os professores que fizeram parte deste meu percurso académico e que contribuíram para o meu conhecimento e aprendizagem.

Agradeço aos meus amigos de infância, os amigos de Moncarapacho, pelo companheirismo, apoio e lealdade demonstrada desde os dias em que ainda estávamos no jardim de infância.

Agradeço a todos os meus colegas e amigos que fiz durante este percurso na Universidade do Algarve pelo companheirismo e união demonstrada nesta etapa da minha vida.

Agradeço à minha namorada, Ana Rute Santos, pelo apoio, compreensão e companheirismo em todos os momentos.

Quero dedicar esta dissertação aos meus avós, pais e irmã, que desde o primeiro momento me apoiaram e nunca deixaram de acreditar em mim, proporcionando todas as condições que necessitei para concluir este curso.

A todos, um enorme

OBRIGADO!

Resumo

A população mundial está atualmente exposta a um nível de radiação considerável. Não se refere apenas à radiação solar, que só por si é bastante preocupante, mas também à radiação proveniente de outras fontes, nomeadamente as relacionadas com métodos imagiológicos que utilizam radiação ionizante. A utilização de raios-X, é uma prática bastante comum nos métodos de diagnóstico e a sua frequência de utilização é alarmante quando os pacientes são ainda jovens ou crianças por o seu organismo ainda se encontrar em crescimento e ser potencialmente afetado pelas radiações.

A utilização de métodos imagiológicos baseados em ultrassom, devem ser levados em consideração nestes casos, por serem métodos com radiação não-ionizante. Apesar de estes métodos necessitarem de maior experiência por parte do técnico, a instrumentação é de menor dimensão, são mais económicos e mais portáveis. No sentido de promover a utilização de métodos imagiológicos alternativos, este projeto de dissertação deu seguimento a projetos anteriormente desenvolvidos pelo Nuno Fernandes e pelo Luís Nascimento, de forma a facilitar a utilização da imagiologia ultrassónica de diagnóstico mediante a criação de uma aplicação em tempo-real para deteção de fraturas ósseas em imagens de ultrassom.

As imagens de ultrassom são caracterizadas pela presença de ruído aleatório causado por reflexões microscópicas que ocorrem nas diversas estruturas internas do paciente. O ruído *Speckle*, é caracterizado por apresentar um efeito de pequenos salpicos, em vários tons de cinza ao longo de toda a imagem. Para reduzir o problema gerado pela presença deste tipo de ruído, são utilizados dois filtros intitulados como *Mean Filter* e SSR (*Simple Speckle Removal*). No que diz respeito à identificação da linha de osso na imagem ultrassonográfica, os métodos *leftright*, *middle* ou *3split+middle* apresentaram bons resultados, segundo a literatura, e posteriormente, através de uma soma de intensidade de pixéis na região da linha de osso identificada, é possível encontrar a fratura óssea.

Tendo presente esta informação, o primeiro objetivo foi pesquisar se estaria disponível algum aplicativo móvel que permitisse a identificação de fraturas ósseas em imagens de ultrassom. Após pesquisa, verificou-se que apenas estão disponíveis aplicativos de caracter educacional. Aplicativos que exemplificam o modo de se obter as imagens, ao mostrar a posição e a zona onde a sonda deverá ser ocupada e sugestões de possíveis patologias encontradas nessas regiões. Para além destes, apenas existe um aplicativo que permite a

identificação de fraturas ósseas ao utilizar este tipo de imagens e foi o aplicativo desenvolvido pelo Luís Nascimento.

Foi definido como segundo objetivo a ligação do algoritmo desenvolvido em MATLAB, com o aplicativo móvel que se iria desenvolver. Este objetivo foi conseguido com a utilização de um pacote que após instalação, torna possível a comunicação entre a linguagem *Python*, JAVA ou C++, e o *engine* do MATLAB. Este pacote, o MATLAB *engine* para *Pyhton*, permite a utilização de funções previamente desenvolvidas em MATLAB, numa aplicação desenvolvida em *Python*.

O terceiro e último objetivo definido, foi criar a aplicação que conseguisse gerar a resposta no mesmo formato da aplicação desenvolvida pelo Luís Nascimento e aproximadamente no mesmo espaço temporal. Tratando-se de um aplicativo em tempo-real, o tempo de resposta era fulcral para o sucesso deste aplicativo. Era necessário ter alguns fatores em consideração, tais como, o tempo de resposta da aplicação em MATLAB e o tempo da comunicação entre servidor e cliente. O objetivo inicialmente delineado foi conseguido apenas na segunda versão do aplicativo, em que se colocou a inicialização do MATLAB *engine* no momento em que o servidor era inicializado, evitando esta perda de tempo no momento em que utilizador submeteria a imagem para ser analisada pelo aplicativo. Tendo como comparação um teste preliminar feito à aplicação do Luís, verificou-se que os resultados, no que diz respeito ao tempo de resposta da nova aplicação, são bastante satisfatórios. Pelo que se considera que a aplicação em tempo-real desenvolvido apresentou uma boa *performance*.

Palavras chave: Aplicativos web, framework Django, sistemas de tempo-real, imagens de ultrassom, identificação de fraturas ósseas.

Abstract

The world's population is currently exposed to a considerable radiation level. It does not refer only to solar radiation, which alone is of concern, but also to radiation from other sources, particularly those related to imaging methods using ionizing radiation. The use of X-rays is a common practice in diagnostic methods and its frequency of use is alarming when patients are still young or children because their body is still growing and potentially affected by radiation.

The use of imaging methods based on ultrasound should be considered in these cases because they are non-ionizing radiation methods. Although these methods require more experience on the part of the technician, the instrumentation is smaller, are more economical and most portable. In order to promote the use of alternative imaging methods, this dissertation project followed projects previously developed by Nuno Fernandes and Luís Nascimento in order to facilitate the use of ultrasonic diagnostic imaging by creating a real-time application for detection of bone fractures in ultrasound images.

Ultrasound images are characterized by the presence of a random noise caused by microscopic reflections that occur in the various internal structures of the patient. Speckle noise, is characterized by having a small splash effect, in various shades of gray, throughout the image. To reduce the problem generated by the presence of this type of noise, two filters are used, *Mean Filter* and *SSR* (Simple Speckle Removal). Regarding the identification of the bone line in the ultrasound image, the *leftright*, *middle* or *3split* + *middle* methods presented good results, according to the literature, and later, through a sum of pixel intensity in the region of the identified bone line, it is possible to find bone fracture.

Taking this information into account, the first objective was to search if any mobile application would be available that would allow the identification of bone fractures in ultrasound images. After researching, it has been found that there are only available educational applications. Applications that exemplify the way of obtaining the images, showing the position and the area where the probe should be occupied and suggestions of possible pathologies found in these regions. Besides those, there is only one application that allows the identification of bone fractures when using this type of images and this was the application developed by Luís Nascimento.

It was defined as second objective the connection of the algorithm developed in MATLAB, with the algorithm that would be developed. This objective was achieved with the

use of a package that, after installation, makes possible the communication between the programming language Python, JAVA or C++, and the MATLAB *engine*. This package, the MATLAB engine for Pyhton, allows the use of functions previously developed in MATLAB, in the algorithm of an application developed in Python.

The third and last defined objective, was to create the application that could generate the answer in the same format of the application developed by Luís Nascimento and approximately in the same time response. Being a real-time application, response time was central to the success of this application. It was necessary to have some factors in mind, such as the response time of the application in MATLAB and the time of communication between server and client. The objective initially outlined was achieved only in the second version of the application, in which MATLAB engine was set to initialize at the time the server was initialized, freeing each image submission from this delay. Comparing a preliminary test made to the application of Luís, it was verified that the results, regarding the response time of the new application, are quite satisfactory. Therefore, it is considered that the real-time application developed showed a good performance.

Keywords: Web applications, Django framework, real time systems, ultrasound images, bone fracture identification.

Índice

Índice de figuras

Índice de tabelas

Capítulo 1 – Introdução

1.1 Motivação

O atual diagnóstico clinico é fortemente suportado por modalidades de imagiologia médica ionizantes como a tomografia computorizada (TC) e a radiologia convencional (raios-X). O aumento da prescrição de exames radiológicos tem sido um dos maiores contribuintes para o dramático aumento de exposição a radiação por parte da população [1].

Diversos estudos têm sido feitos pondo em causa as doses de radiação, as suas consequências e as medidas a tomar [2], [3]. Em contrapartida, a imagiologia não-ionizante baseada em ultrassons (US), apesar de ser tão antiga como os raios-X, ainda não constitui prática corrente no diagnóstico de fraturas ósseas, campo em que a radiologia convencional e a TC são sistematicamente utilizadas. A consulta de bibliografia reporta experiências relativamente recentes de utilização dos US como meio de diagnóstico em lugares remotos [4], [5], em alguns casos de emergência médica [6] [7], e, em emergência pediátrica [8], [9]. Fatores como a dificuldade de interpretação das imagens de US [10], constituem entraves à aplicação generalizada dos US na identificação de fraturas ósseas.

Neste contexto, a motivação do presente projeto de dissertação é dar continuidade ao estudo de sistemas de apoio ao diagnóstico clinico de imagens de fraturas ósseas de ultrassom, contribuindo com o desenvolvimento de uma aplicação em tempo-real para a deteção e análise das mesmas.

1.2 Objetivos

Com base nas metodologias de deteção de fraturas ósseas desenvolvidas no passado nas teses de dissertação de Nuno Fernandes [11] e de Luís Nascimento [12] pretende-se nesta tese adaptar as funcionalidades dos algoritmos propostos para se testar a aplicabilidade e desempenho dos algoritmos numa versão do sistema que satisfaça requisitos de tempo-real através de uma aplicação web, para que o mesmo possa ser utilizado em ambiente hospitalar ou outro local, desde que com ligação a rede de comunicação, por um utilizador não especialista em ultrassons.

1.3 Organização do relatório

No capítulo 1, é explicada a motivação e os objetivos propostos para esta dissertação e apresentar uma descrição do conteúdo da tese.

No capítulo 2, descreve-se um pouco da teoria adjacente a este projeto. Inicia-se com uma descrição genérica de conceitos relacionados com o ultrassom e o ruído presente nas imagens ultrassónicas, seguindo-se uma revisão do algoritmo, desenvolvido anteriormente, para deteção de fraturas ósseas nessas mesmas imagens.

No capítulo 3, faz-se uma breve apresentação de conceitos gerias de aplicativos móveis, passando pela apresentação de alguns desses aplicativos existentes no mercado. Será também apresentado o conceito de *web framework*, juntamente com 3 das *frameworks* mais utilizadas em linguagem Java, C++ e Python.

O capítulo 4 tem como objetivo apresentar a metodologia utilizada neste projeto, onde se encontrará uma breve apresentação da web Framework utilizada e o mecanismo utilizado para efetuar a ligação entre o algoritmo anteriormente desenvolvido em MATLAB e o algoritmo web aqui desenvolvido.

O capítulo 5 tem como objetivo a verificação de resultados obtidos anteriormente na análise das imagens ultrassonográficas, comparando com resultados atuais e verificação dos mesmos na ligação entre cliente e servidor.

No capítulo 6, são apresentadas as conclusões finais e serão sugeridas linhas futuras para investigação.

Capítulo 2 – Revisão dos algoritmos de deteção de fraturas ósseas em imagens ultrassónicas

2.1 Introdução

Neste capítulo serão apresentados fundamentos teóricos relacionados com as imagens ultrassónicas, tais como, um pouco de história do ultrassom, a forma como são formadas as imagens de ultrassonografia e problemática de interpretação das mesmas como seja a afetação das imagens por ruído designado por *speckle noise*. Será também apresentada a metodologia seguida em teses anteriores ([11], [12]) para identificação de possíveis fraturas ósseas, ou seja, métodos de pesquisa da linha de osso seguidos então de métodos de identificação da fratura óssea nas imagens ultrassónicas, que permitirá uma visão global da forma de atuação do algoritmo do aplicativo desenvolvido. Mais detalhes devem ser pesquisados em [11], [12].

2.2 Imagens Ultrassónicas

Nesta secção é apresentada, de forma geral, um pouco de história e informação sobre ultrassom e a forma como as imagens são criadas, o tipo de ruído existente nas imagens ultrassónicas, os métodos de redução deste mesmo ruído e o modelo desenhado no trabalho anterior [12] para deteção da linha do osso e fraturas nas imagens ultrassónicas.

2.2.1 História

As técnicas do ultrassom foram inspiradas na forma de comunicação usada entre animais, tais como morcegos, golfinhos e baleias. Como se sabe, os morcegos, por não conseguirem ter muita visibilidade, têm de emitir sons e esperar pela receção do eco para conseguir identificar as distâncias a que se encontram de determinados objetos.

E foi na altura da Primeira Guerra Mundial que surgiu o sonar que acabou por ser aperfeiçoado durante a Segunda Guerra Mundial. Esta técnica era usada para identificar objetos submersos através da emissão de ondas sonoras na água e receção dos respetivos ecos.

Com isto, após a Guerra, vários investigadores médicos começaram a trabalhar no sentido de aplicar estes conceitos para o diagnóstico médico. E assim, durante a década de 1950, conseguiram desenvolver dois modos de aplicar o ultrassom à medicina. A primeira

unidade desenvolvida fornecia imagens, através de um monitor, que representavam uma série de pontos cuja altura era o reflexo da intensidade do eco recebido. Na segunda unidade foram introduzidos aparelhos de ultrassonografia bidimensional de escalas de cinza. Esta escala permitia que, após um conversor de imagens amplificar e processar esses ecos, fossem mostradas num monitor as imagens com as intensidades dos ecos com diferentes tons de cinza[13][14].

2.2.2 Princípios do ultrassom

O ultrassom é uma onda sonora a uma frequência que o ouvido humano não consegue detetar. A banda de frequências que o nosso ouvido consegue captar está entre os 20 Hz e os 20 kHz, enquanto que para o ultrassom utilizado na medicina a banda de frequências está entre os 1 a 17 MHz [14]. O ultrassom pode ser produzido por cristais, em que a vibração dos cristais produz ondas mecânicas com frequências na gama do ultrassom; estas ondas propagam-se da mesma forma que as ondas sonoras audíveis pelo que interagem com o meio circundante através de efeitos de reflexão, refração, difração e interferência de acordo com as impedâncias acústicas de cada meio de propagação.

2.2.3 Princípios da Ultrassonografia

A ultrassonografia ou ecografia é uma técnica de diagnóstico por imagem que tem como base o uso de ondas de ultrassom, e que normalmente são usadas para produzir imagens de tecidos moles e órgãos do corpo humano. A imagem é representativa dos ecos recebidos do interior do corpo humano, ecos esses, resultantes maioritariamente da reflexão da estrutura que se pretende visualizar, mas sujeitos à interferência de refrações dessa e das estruturas vizinhas.

Mais recentemente, e principalmente em ambientes remotos (longe de centros médicos), a ultrassonografia tem vindo a ser usada na deteção de fraturas ósseas em alternativa a métodos de diagnósticos por radiação ionizante.

Esta técnica torna-se mais vantajosa em relação a outras modalidades de diagnóstico por imagem pois não usa radiação ionizante, evitando assim a exposição dos pacientes a mais radiação, fator relevante principalmente se o paciente for jovem.

2.2.4 Redução de *Speckle noise*

As imagens de ultrassom são sujeitas a corrupção por um tipo de ruído designado por ruído *Speckle.* Este ruído é um fenómeno complexo, imprevisível e difícil de remover, que introduz aleatoriamente na imagem uma espécie de salpicos em tons de cinzento afetando partes da imagem [15] que podem ser de grande importância para o diagnóstico médico, por exemplo a deteção da zona do osso e a sua envolvente. Este ruído ocorre devido ao aparecimento de múltiplas reflexões microscópicas ocasionadas pelas estruturas internas. [16]

Como tal, foram testados, nos trabalhos realizados anteriormente, quatro métodos que são referenciados na literatura como os melhores para a redução de ruído: *Simple Speckle Removal (SSR), Mean Filter, Median Filter e Gamma Filter.* São considerados os melhores, porque se tratam de filtros espaciais e assim apresentam uma computação mais rápida e fácil. Os métodos *Median Filter e Gamma Filter*, após testes efetuados pelo Luís Nascimento [12], acabaram por ser descartados pois apresentavam um tempo de computação mais demorado que o *SSR* e o *Mean Filter*, que são brevemente apresentados de seguida.

2.2.4.1 *Simple Speckle Removal (SSR)*

Trata-se de um método em que todos os cálculos são efetuados no domínio do tempo, o que evita o uso de transformadas ou expressões complexas e que o torna num método de matemática simples e de rápida implementação computacional.

Este algoritmo foi selecionado de entre a bibliografia para integrar [11], tendo sido mantido na dissertação [12]. Este algoritmo assume por princípio que as imagens ultrassónicas de fraturas ósseas apresentam três zonas distintas (Figura 1): zona de ruído, zona do osso e zona de sombra.

Numa primeira fase do método, tem de se escolher uma zona da imagem onde exista ruído em maior quantidade. Em geral essa zona corresponde à zona do corpo que o ultrassom deverá atravessar desde a pele até chegar ao osso, composta por músculos e outros tecidos, traduzindo-se numa zona onde a informação da imagem pode ser considerada exclusivamente ruído. A zona do osso, como o próprio nome indica, será a zona da imagem onde esperamos encontrar/visualizar o osso. A zona de sombra corresponde à zona em que a maioria das ondas do ultrassom não conseguem chegar, em consequência da reflexão a que estas são sujeitas devido à presença do osso. Ainda assim, e como já foi verificado no projeto em [12], por vezes esta zona poderá conter algumas variações de tons de cinzento que podem ser

relacionados a artefactos ou à passagem das ondas de ultrassom quando o osso em estudo tem acentuadas interrupções.

Foi determinado em [11] que a zona de ruído está compreendida entre os primeiros 20% de altura e 80% de largura das imagens de ultrassom em estudo, considerando que se analisa a imagem do canto superior esquerdo que, no caso então em estudo, correspondia a imagens de ossos longos. Esta foi a zona considerada exclusivamente por ruído (Figura 2).

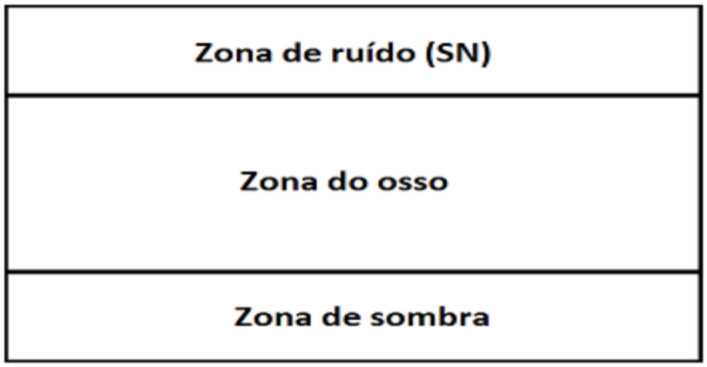

Figura 1 - Zonas típicas de uma imagem ultrassom de osso[12]

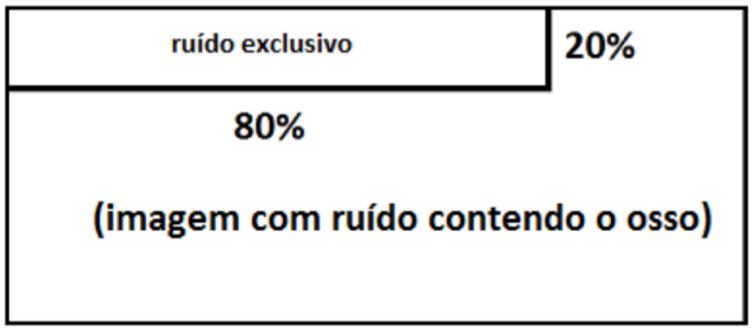

Figura 2 - Representação gráfica da percentagem e localização na imagem de ultrassom de ossos longos das zonas de mais ruído [12]

Estando identificada a zona constituída exclusivamente por ruído, segue-se para o passo seguinte que é o cálculo do valor médio de intensidade de pixel nessa zona que gera o valor médio de ruído na mesma, através da expressão seguinte:

$$VdR = \sum_{j=1}^{N_j} \sum_{i=1}^{N_i} \frac{I(i,j)}{N_j . N_i} \qquad (1)$$

Onde $\boldsymbol{VdR}$ é o valor médio de ruído, $\boldsymbol{i}$ e $\boldsymbol{j}$ correspondem às linhas e colunas, respetivamente, $\boldsymbol{N_i}$ e $\boldsymbol{N_j}$ representam o número total de linhas e colunas da zona de ruído, $\boldsymbol{I}$ representa a matriz das intensidades de pixéis da imagem em análise.

Determinado o valor médio de ruído, subtrair-se-á o mesmo à matriz das intensidades dos pixéis da imagem (***I***), subtração que é feita pixel-a-pixel, ou seja, a cada elemento da matriz irá subtrair-se o ***VdR***, que resultará numa nova matriz que representa a imagem filtrada (I_{filt}).

$$I_{filt} = I(i,j) - VdR \qquad (2)$$

2.2.4.2 *Mean Filter*

Apesar do método *SSR* ser simples e de fácil computação, o método *Mean Filter* também é muito conhecido na literatura de filtragem de imagens com ruído por ser um método ainda mais simples e generalista que o *SSR*. Enquanto o método *SSR*, extrai a todos os pixéis da imagem apenas a quantidade de ruído que foi identificada na zona acima referida de "ruído exclusivo", o *Mean Filter* calcula o valor médio de cada pixel e é feita uma subtração pixel-a-pixel desse valor médio a cada pixel da imagem original, obtendo-se assim a imagem filtrada. Ainda assim, há imagens em que o ruído não só é evidente na zona de ruído, mas também na zona de sombra. O método SSR não leva em consideração esse excesso de ruído e, por isso, através de experiências que foram realizadas em [12], ficou demonstrado que deve ser utilizado o *Mean Filter* em vez do *SSR*.

2.2.5 Pesquisa da linha do osso

Após a aplicação de métodos de redução de ruído, o passo seguinte constitui a deteção da linha do osso nas imagens ultrassónicas. Como tal, foram testados três métodos de procura da linha de osso, que acabaram por não ser utilizados por não apresentarem resultados tão bons como o método desenvolvido por [12]. Segundo este autor, é necessário aplicar um filtro chamado Energia Externa para detetar apenas contornos de objetos, que neste caso será a linha de osso, porque como se trata de um filtro que aumenta o contraste da imagem pode acabar por "retirar" a fratura da imagem. Em seguida, é feito o somatório dos pixéis, por cada linha, dessa imagem filtrada e é gerado um gráfico representativo dessa operação (Figura 4).

A determinação da linha de início da pesquisa da linha de osso é feita percorrendo a representação gráfica do somatório dos pixéis, desde a linha zero até à linha máxima, para se conseguir identificar qual a linha que corresponde a um valor de intensidade que se encontre dentro de uma percentagem denominada como ***isline***, isto é, quantos mais pixéis se tiver

numa determinada linha, maior será a intensidade dos pixéis, como tal, define-se uma percentagem (***isline***), relativa ao número de pixéis, para se iniciar a pesquisa da linha do osso através de uma linha que se considere com um número de pixéis suficientes para ser considerado como uma proximidade do osso. Ex.: *isline*=50%, a linha escolhida seria perto da linha 100, como se pode ver através da Figura 4.

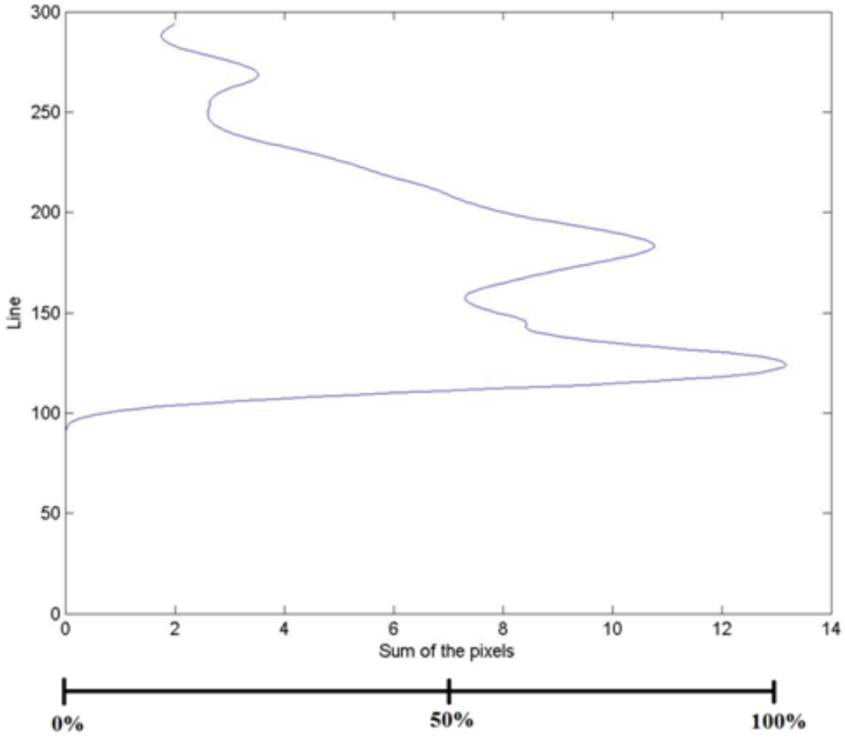

Figura 3 - Representação gráfica do somatório dos pixeis em cada linha da imagem[12]

Por fim, antes de começar a procura da linha de osso, é necessário definir a variável ***L*** que indica a área em que será feita a procura e o seu valor deve corresponder a uma linha situada acima da linha que foi anteriormente escolhida para iniciar a pesquisa. A variável ***L*** corresponde a uma área por necessidade de alargamento da zona de pesquisa, porque por vezes a pesquisa pode não acompanhar a curvatura do osso, ou por a percentagem ***isline*** escolhida corresponder a ruído e não concretamente ao osso.

Passamos então para os três tipos de pesquisa de linha de osso propostos por [12].

2.2.5.1 Pesquisa *leftright*

A procura é iniciada na primeira coluna situada no lado esquerdo da imagem e dentro da área L já definida, é escolhido e marcado o local do valor máximo de intensidade pixel.

Em seguida, é procurado na coluna seguinte o máximo valor de intensidade pixel entre os três pixéis adjacentes ao escolhido, ou seja, o pixel que está na mesma linha, o da linha acima e o da linha abaixo, prosseguindo desta forma até ao final da imagem. Durante o processo podem surgir situações em que os valores de intensidades dos pixéis em análise sejam iguais e, por isso, criou-se a necessidade de se definir um critério de "desempate": - Se as intensidades dos pixéis forem iguais, a pesquisa continua para o lado direito; - Se as intensidades dos pixéis forem iguais na linha abaixo e acima, é escolhido o pixel da linha abaixo por haver menor probabilidade da procura se perder no ruído; - Se o valor dos três pixéis forem simultaneamente zero, o valor é descartado e passa-se para a coluna seguinte seguindo o critério definido anteriormente.

Através da Figura 3 podemos observar um exemplo esquemático de como é efetuado o processo de pesquisa para este tipo.

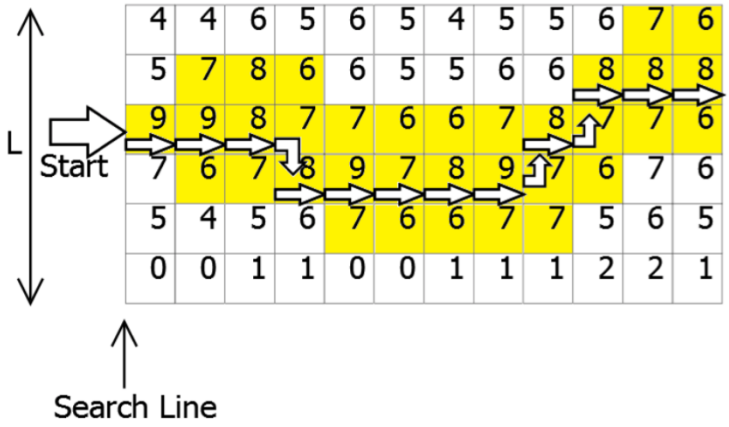

Figura 4 - Exemplo esquemático de um pesquisa simples para o tipo leftright [12]

2.2.5.2 Pesquisa *middle*

No tipo de pesquisa anterior o nome estava diretamente relacionado com a forma de como era feita a pesquisa da linha do osso, neste caso também está, o que indica que a coluna a ser considerada inicialmente é a coluna central da matriz da imagem. Em seguida, a metodologia de escolha de pixel, é similar ao processo anterior em que a diferença está na direção da pesquisa, que neste caso será efetuada para a esquerda e para a direita.

Caso a matriz contenha um número de colunas par, a coluna a escolher para se iniciar o processo, será a coluna imediatamente à esquerda do centro (Ex.: imagem com 10 colunas, a coluna a escolher seria a coluna 4), no entanto foi verificado experimentalmente em [12] que essa escolha não tem qualquer influência no resultado.

Através da Figura 5 podemos observar um exemplo esquemático de como é efetuado o processo de pesquisa nesta alternativa.

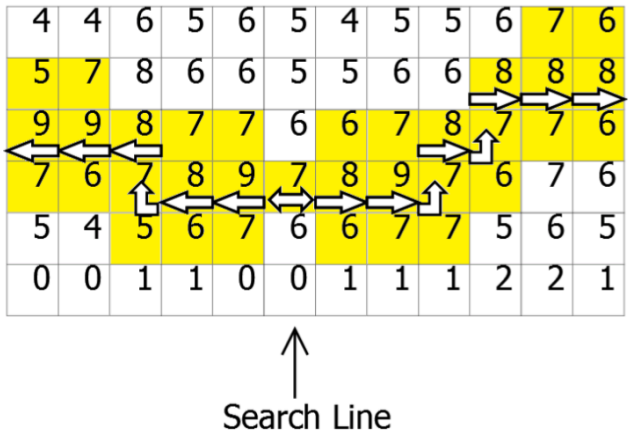

Figura 5 - Exemplo esquemático de uma pesquisa simples para o tipo middle [12]

2.2.5.3 Pesquisa *3split+middle*

Neste caso, para se iniciar a pesquisa, é necessário dividir a imagem em três partes (Figura 6), e se não se conseguir dividir em três partes iguais, por o número de colunas não ser divisível por 3, as duas primeiras partes terão o mesmo número de colunas e a terceira parte ficará com as restantes. Este processo foi verificado experimentalmente por [12], que concluiu que independentemente do número de colunas existentes na imagem e desde que haja uma distribuição uniforme pelas três partes da imagem, os resultados obtidos da pesquisa, serão sempre os mesmos.

Após a divisão da imagem, volta-se a fazer o somatório dos pixéis, linha a linha, para cada uma das partes, e no fim aplica-se então a pesquisa do tipo *middle*. Através da Figura 7 podemos observar um exemplo esquemático da aplicação deste critério. Para finalizar todo o processo basta apenas voltar a unir as três partes da imagem (Figura 8).

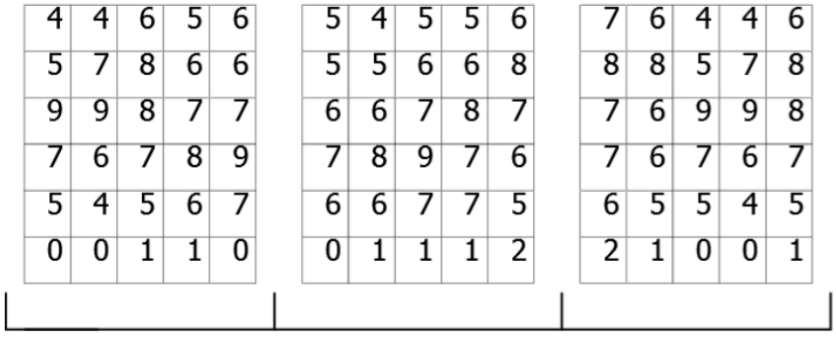

Figura 6 - Exemplo de uma imagem dividida em três partes [12]

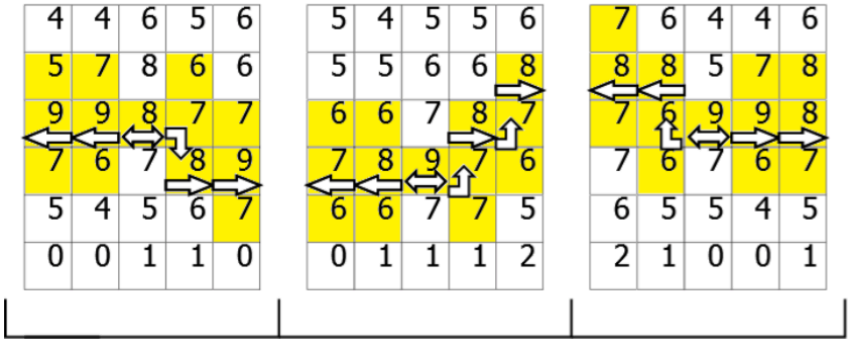

Figura 7 - Exemplo esquemático do método 3split+middle [12]

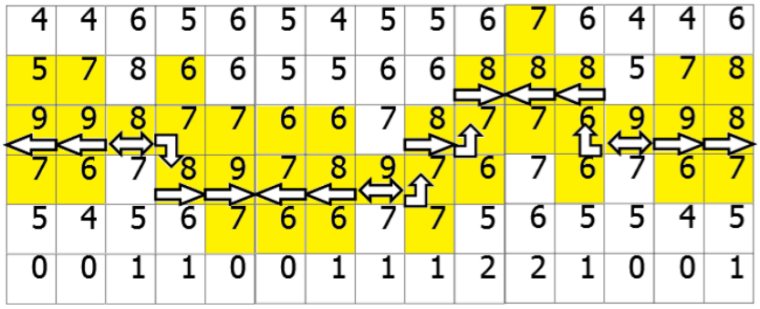

Figura 8 - Passo final do método 3split+middle (juntar as três partes da imagem)[12]

2.2.6 Identificação da fratura

Após a linha de osso ter sido calculada através de um dos métodos anteriormente apresentados, a mesma é colocada sobre a imagem filtrada por um dos filtros espaciais e a imagem filtrada pela energia externa deixa de ser considerada. Tendo em conta que existem alguns pixéis à volta da linha que se podem assumir como fazendo parte de irregularidades do

contorno do osso, foi decidido alargar um pouco a linha branca que tinha sido determinada acrescentando valores acima e outro abaixo da mesma. Os valores a adicionar variam consoante o tipo de osso que está a ser analisado, sendo que esses valores foram estipulados e estão presentes em [12].

Com a linha de osso totalmente encontrada, passa-se então para o somatório das intensidades dos pixéis dessa linha de osso para cada coluna e apresenta-se uma representação gráfica com esses valores. Dentro dessa representação gráfica, podemos encontrar alguns valores com picos mínimos que podem significar a existência de uma fratura na coluna respetiva a esse mínimo, ou então poderá ser apenas um valor baixo de intensidade dos pixéis. Como solução para esta questão, é efetuada a média de todos os pixéis de osso que existem e abaixo dessa média, é feita a escolha de uma percentagem que será denominada como ***isfract*** e só os picos mínimos que se encontram abaixo dessa percentagem é que serão considerados como indicadores de uma fratura óssea. Através da Figura 9, pode-se ver um exemplo retirado de [12] relativamente a uma fratura de *stress* na tíbia.

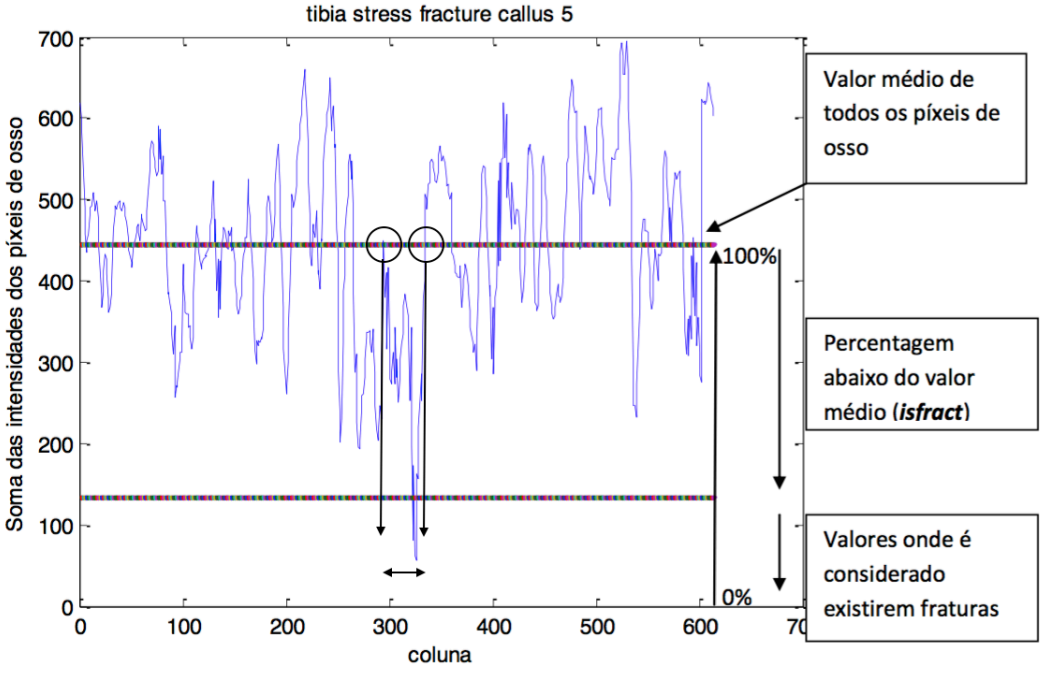

Figura 9 - Representação gráfica da soma dos pixeis com os valores médios e percentagem isfract [12]

Como principal condição para que exista fratura, é que esta esteja presente na linha de osso e por isso, procura-se o mínimo que esteja abaixo dessa percentagem (***isfract***) e verificam-se os valores à esquerda e direita do mesmo para ver se estes apresentam valores que estejam na linha da média de todos os pixéis (Figura 9). Se eventualmente não apresentarem esses valores, esse mínimo é descartado e é procurado um novo mínimo.

Quando verificada a existência de um mínimo que está entre o valor médio de todos os pixéis de osso, o local do mesmo é marcado e todos os valores, quer do mínimo como os valores à esquerda e direita são descartados. Como por vezes podem haver vários mínimos abaixo da linha da percentagem ***isfract***, é feita uma nova pesquisa para verificar essa existência. Caso ocorra, o mínimo mais baixo que se encontre abaixo da linha ***isfract***, é marcado com um círculo de acordo com a definição encontrada em [12]. O segundo mínimo é marcado com um quadrado e o terceiro com uma cruz. Estes símbolos são, pois, representativos da ocorrência de uma probabilidade alta, média ou fraca de existência de fratura no local assinalado pelo sistema por um círculo, quadrado ou cruz, respetivamente.

2.3 Conclusão

Este capítulo iniciou-se com uma breve apresentação sobre as imagens ultrassónicas, onde se falou um pouco sobre a história do ultrassom e os princípios do ultrassom e da ultrassonografia. Sendo o speckle noise um problema existente nas imagens ultrassónicas, apresentou dois filtros que permitem a remoção de uma grande parte desse ruído sem afetar em demasia a informação da imagem, o filtro SSR e o Mean Filter. Posteriormente foram apresentados os métodos desenvolvidos em [12] para a deteção da linha de osso nas imagens ultrassónicas, o método leftright, middle e 3split+middle, tal como o método de identificação de fratura utilizando um gráfico que é gerado após efetuar o somatório da intensidade dos pixéis por cada coluna da linha do osso encontrada através de um dos 3 métodos, sendo que a fratura é detetada através de uma ou mais interrupções dessa linha de osso.

Capítulo 3 – Conceitos gerais sobre aplicativos

3.1 Introdução

Com o passar dos anos o desenvolvimento dos dispositivos móveis, tais como *smarthphones* e *tablets,* foi notável, através de melhorias em vários aspetos tais como CPUs mais eficientes, ecrãs maiores e com melhor resolução, aumento da memória e a facilidade de acesso à internet e *download* de variados tipos de *software*. Este conjunto de fatores possibilitou o uso destes dipositivos móveis em diversas áreas, através do uso de aplicações, por parte de estudantes e profissionais no seu local de trabalho, mais especializadas em cada área. Em geral, as aplicações apresentam alto nível de fiabilidade pelo facto de serem desenvolvidas com apoio de investigadores da área para a qual a aplicação está direcionada.

Sendo a temática em estudo nesta dissertação direcionada para a área da medicina, verificou-se que vários estudos foram realizados ao longo dos últimos anos acerca do uso de *smartphones* em ambientes hospitalares como forma de apoio aos médicos recentemente formados e um destes estudos revela que um acesso rápido a um aplicativo móvel com acesso à internet para verificar livros de texto com informação fundamental, ajuda a aprendizagem e prática de médicos recém-formados nesta fase mais crítica do seu desenvolvimento [17]. Mas claro que não são apenas os médicos recentemente formados que utilizam os dispositivos como forma de apoio, desde o inicio de 1900 que a prática da telemedicina se começou a ser praticada por vários médicos, sendo que inicialmente foi através de comunicações rádio e atualmente através de videoconferências. Cada vez mais, esta prática é utilizada até para que haja troca de informação em tempo real entre médicos espalhados pelo mundo e que torna o diagnóstico à distância possível.

Após pesquisa, encontrou-se um conjunto de aplicativos móveis que foram desenvolvidos ao longo dos últimos anos que estão relacionados com a técnica imagiológica referida neste projeto, o ultrassom. Verifica-se que todos os aplicativos existentes são de características educacionais, que servem de apoio aos técnicos de ultrassom. De seguida, apresentar-se-á alguns exemplos desses aplicativos.

3.2 Aplicações de género *handbook* e educacionais

A maioria das aplicações existentes no mercado são do género handbook, ou seja, aplicações que através de texto e imagens e outras até vídeos, fornecem uma grande à classe médica ou estudantes de medicina. Algumas destas aplicações são gratuitas e outras não, como consequência algumas estão mais completas do que outras. "*Pocket Atlas of Emergency Ultrasound*" [18], "*SonoSupport: a clinical emergency medicine and critical care ultrasound reference tool*" [19], "*Emergency Ultrasound Handbook*" [20], são três exemplos deste género de aplicações.

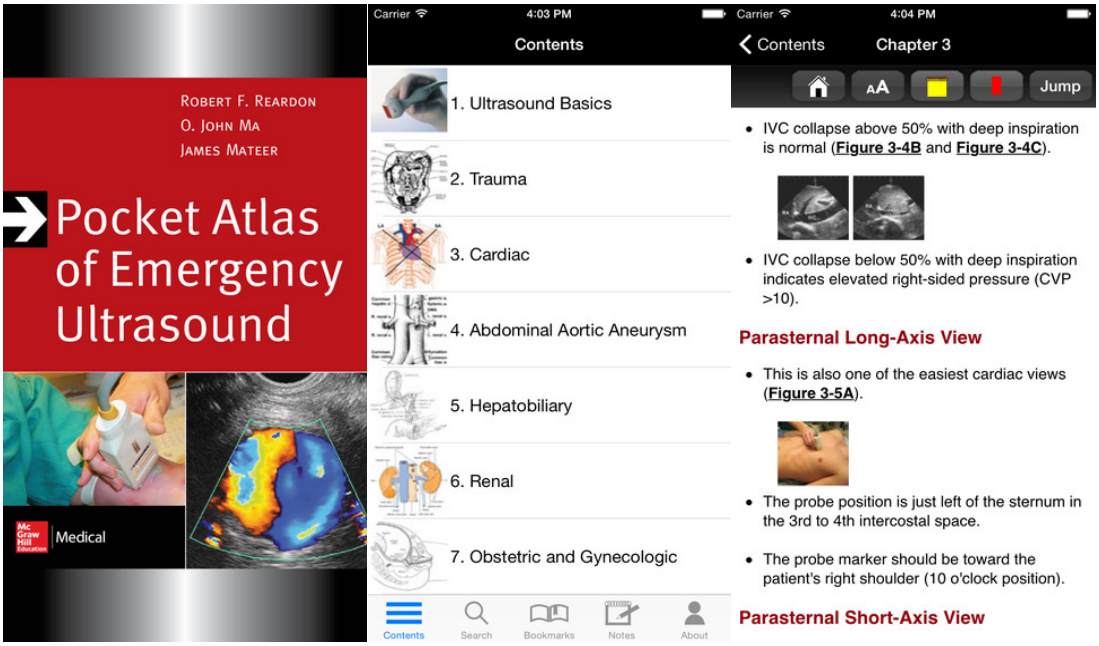

Figura 10 - Screenshots da Pocket Atlas of Emergency Ultrasound [8]

As duas primeiras são aplicações pagas e mais completas do que a terceira aplicação, ao nível de detalhe da descrição das patologias e sugestões para se efetuar os exames, se conseguir obter as melhores imagens através da escolha da melhor sonda, e/ou a melhor posição para a colocar durante a realização do mesmo.

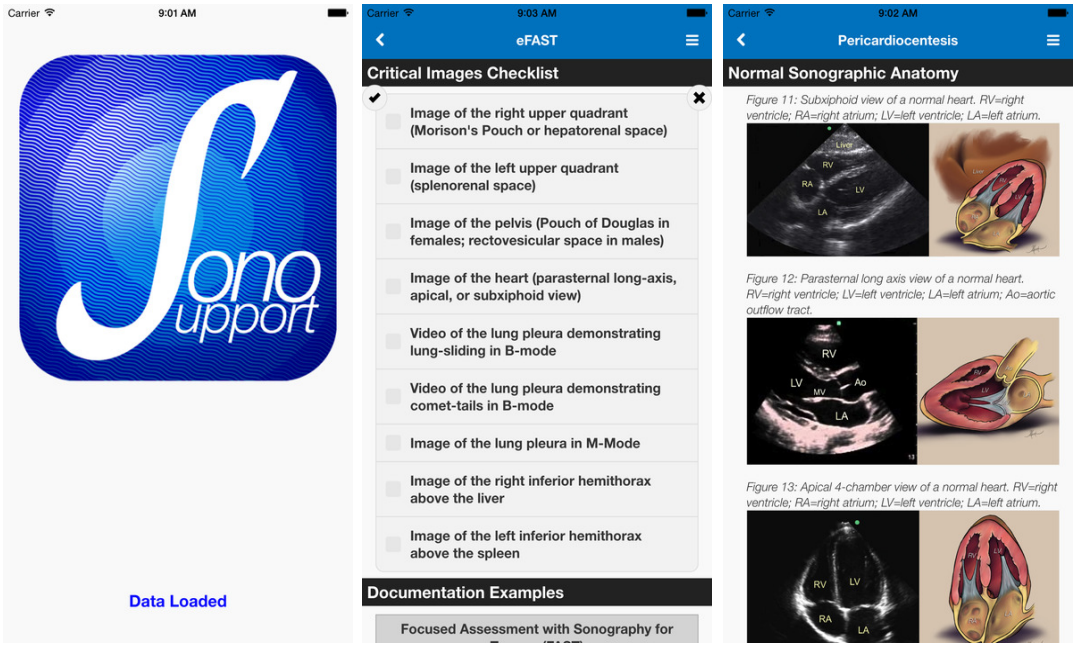

Figura 11 - Screenshots da SonoSupport [19]

Relativamente à *Emergency Ultrasound Handbook*, é uma aplicação gratuita que contém algumas imagens e informações clinicamente úteis com a condicionante de se ter de adquirir a versão "Pro" para se conseguir visualizar os vídeos e a secção das patologias.

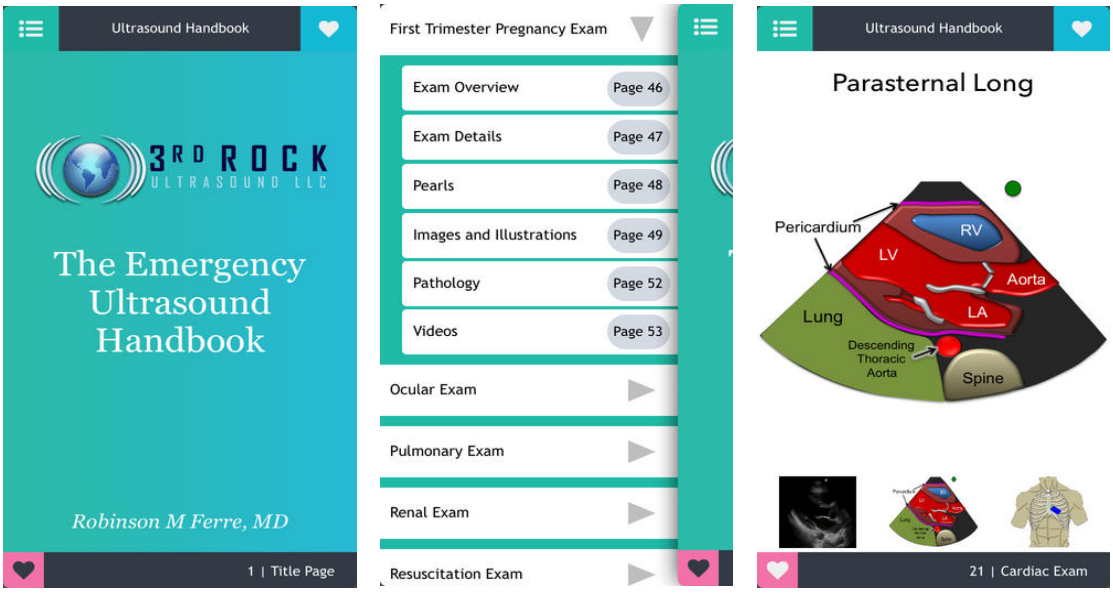

Figura 12 - Screenshots da Emergency Ultrasound Handbook [20]

3.3 Aplicações de recolha de dados

Durante a pesquisa verificou-se que existe pelo menos uma aplicação que permite adquirir dados através de uma sonda conectada a um *smartphone* e que tem o nome de "MobiSante MobiUS SP1" [21]. Acaba por ter algumas condicionantes, tais como, ser compatível apenas com *smartphones* ou *tablets* que operem com o sistema operativo *Windows Phone* [22]. Todo o equipamento é fornecido pela empresa, desde o aparelho até à sonda o que torna o preço de aquisição deste material um pouco elevado e o *smartphone* apenas poderá ser utilizado para este fim, a aquisição das imagens.

Figura 13 - MobiSante MobiUS SP1 System [21]

3.4 Aplicação desenvolvida por K. Foss e sua equipa [23]

É uma aplicação desenvolvida seguindo os princípios de uma aplicação *web* simples (*web-app*) e que pode ser usada em qualquer sistema operativo de *smartphones*. Trata-se de uma aplicação muito simples que exemplifica através de vídeos, a forma de obter a imagem de apenas cinco patologias diferentes e que foram consideradas apropriadas [23]. Aponta-se, no entanto, o facto da aplicação estar escrita em dinamarquês e não existir alternativas para outras linguagens, como um aspeto desvantajoso. Podemos ver através das imagens seguintes, alguns *screenshots* da aplicação a correr num navegador *web*.

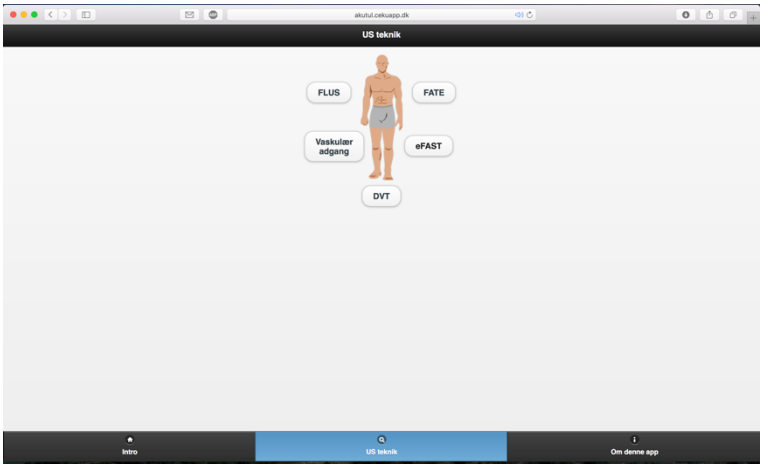

Figura 14 - Screenshot da páginal inicial da aplicação

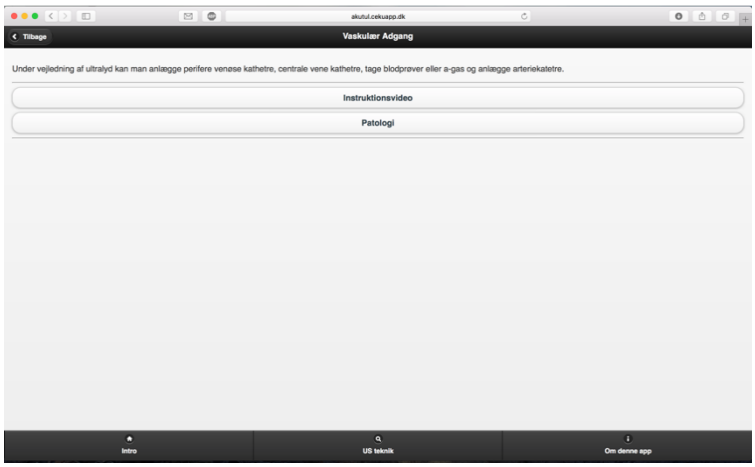

Figura 15 - Screenshot da aplicação após escolha da patologia

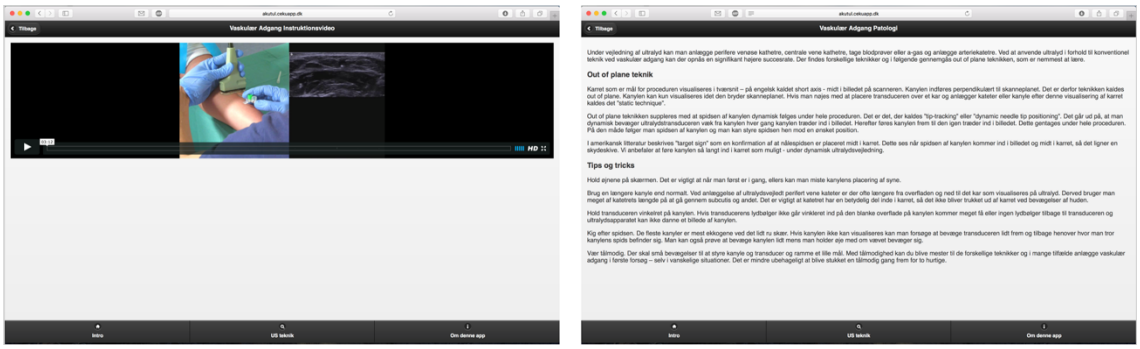

Figura 16 - Screenshots do vídeo exemplificativo e explicação da respetiva patologia

3.5 Aplicação desenvolvida em "Contribuições para a deteção automática de fraturas ósseas em imagens de ultrassom"

Utilizando os métodos de remoção de ruído e os métodos de identificação da linha do osso nas imagens de ultrassom, o Luís em [12], desenvolveu uma aplicação em MATLAB que ajuda o utilizador a identificar uma possível fratura óssea.

Ao inicializar a aplicação, surge uma imagem com algumas instruções sobre o procedimento que o utilizador deve ter (Figura 17), em seguida, e através da imagem de um esqueleto (Figura 18), o utilizador escolhe o osso pretendido, caso a zona selecionada incluir mais do que um osso, será feito um "*zoom*", para que sejam apresentados os ossos dessa área de seleção. Em seguida é pedido para escolher a imagem que pretende que seja analisada pela aplicação, essa imagem deve corresponder ao osso escolhido devido aos diferentes filtros aplicados e aos diferentes parâmetros que é necessário aplicar em cada um. Deve clicar-se em "Abrir" para que a imagem seja carregada e, a aplicação analisa a imagem. Concluída essa análise, é enviada a resposta para o utilizador e apresentada em 3 imagens distintas, em que uma apresenta até três graus de probabilidade de existência de fratura representados através de 3 símbolos diferentes, conforme referido na secção 2.2.6, outra apresenta a linha do osso sobre a imagem original e a terceira imagem, corresponde a uma representação gráfica que representa a soma das intensidades dos pixéis correspondentes a cada coluna da imagem em análise.

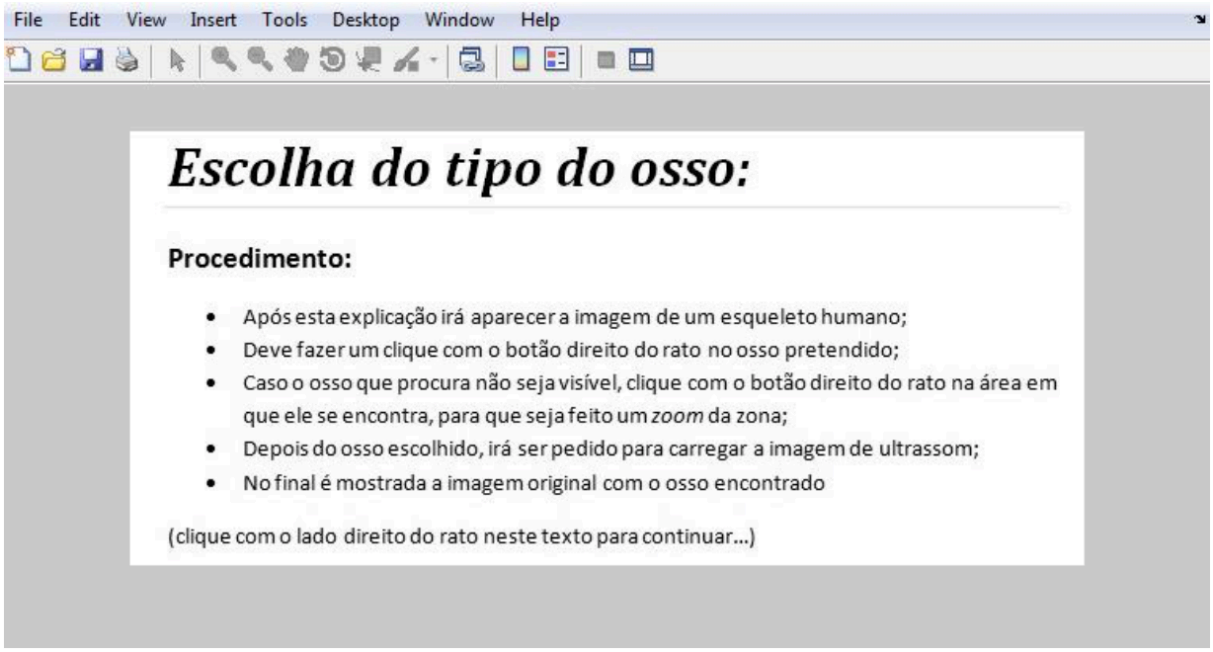

Figura 17 – Janela inicial do programa de aplicação [12]

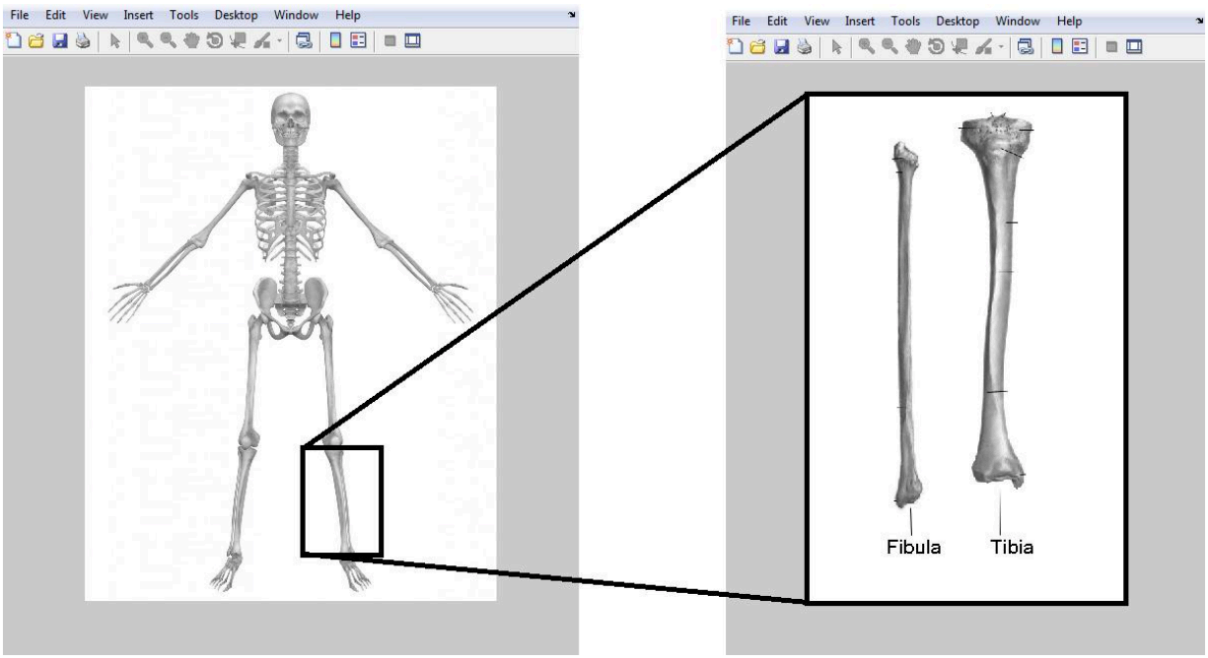

Figura 18 – Exemplo de utilização do zoom na aplicação para diferenciar os ossos para análise [12]

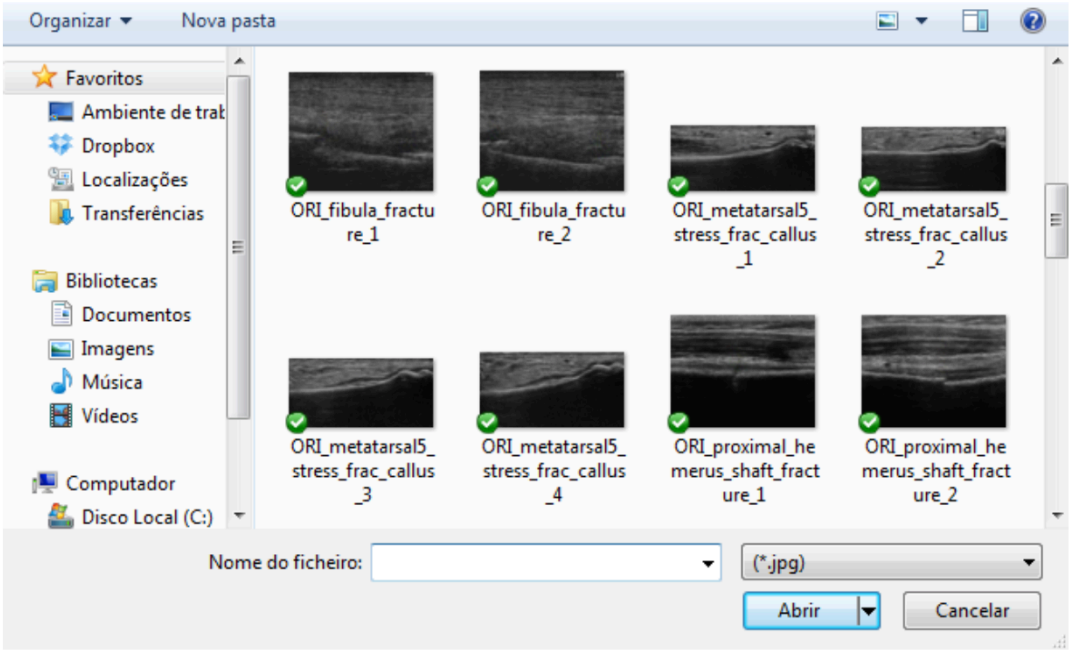

Figura 19 – Visualização da caixa de carregamento da imagem US na aplicação [12]

3.6 *Web frameworks*

O conceito de *web Framework* pode muitas vezes confundir-se com o conceito de biblioteca (*library*), mas são conceitos totalmente diferentes. Numa biblioteca, só se utiliza o

que realmente se necessita, é uma ferramenta que possui um conjunto de funções, classes, procedimentos, etc.

Uma *framework* pode incluir compiladores, bibliotecas, conjuntos de ferramentas e interfaces de programação de aplicativos (APIs) que reúnem todos os diferentes componentes para permitir o desenvolvimento de uma aplicação. Uma *web Framework* é um *software* desenvolvido para apoiar o desenvolvimento de aplicações web, fornecendo uma forma *standard* para a construção e desenvolvimentos das mesmas.

Como tal, a utilização de uma Framework nesta dissertação pode ser uma mais valia tentando interligar a aplicação desenvolvida em [12] com uma linguagem de programação direcionada para a web. Após pesquisa verificou-se que existem linguagens que permitem essa ligação, como por exemplo, Java, C++ e Python. Pretende-se por isso, apresentar brevemente nesta secção 3 *frameworks* web *open source,* para facilitar o acesso a documentação, mais utilizadas em cada uma dessas linguagens de programação: Spring, Wt e Django.

3.6.1 Spring

Figura 20 - Logotipo da framework Spring

A Spring [24] é a *framework* mais utilizada na linguagem JAVA. Foi escrita por Rod Johnson no ano de 2002 o qual a apresentou no seu livro intitulado "*Expert One-on-One: JEE Design and Development*" implementando uma lógica de negócios conhecida como EJBs (Enterprise JavaBeans). No ano seguinte, em 2003, foi iniciado realmente o processo de criação da Framework Spring sem a utilização da lógica acima, EJBs, sendo oficialmente apresentada em 2004 no livro "*Expert One-To-One J2EE Development Without EJB*" [25]. Tornou-se por isso muito popular entre os programadores de Java por se conseguir utilizar como alternativa ou até mesmo para trabalhar em conjunto com o modelo EJB por ser uma *framework* desenvolvida seguindo o padrão de arquitetura MVC (*Model View Controler*).

MVC é um padrão de arquitetura de software que, resumidamente, separa a aplicação em 3 camadas distintas, mas que trabalham em conjunto.

- *Model* é a camada responsável pela manipulação dos dados, que passa pela escrita, leitura e validação dos dados. (Ex.: lê e valida os dados de login de um utilizador para a aplicação.)

- *View*, a camada de interação com o utilizar, ou seja, uma interface que irá exibir os dados através de uma página html ou xml. (Ex.: página onde se insere os dados do login e após a verificação dos mesmos pelo *Model*, será exibida a página da aplicação.)

- O *Controller* é a camada que recebe os pedidos do utilizador e encaminha o *View* e *Model* correto para o utilizador. (Ex.: envia o *View* para que sejam inseridos os dados de login, relaciona com o *Model* para verificar esses dados e envia a *View* após a autentificação ter sido efetuada com sucesso ou não.)

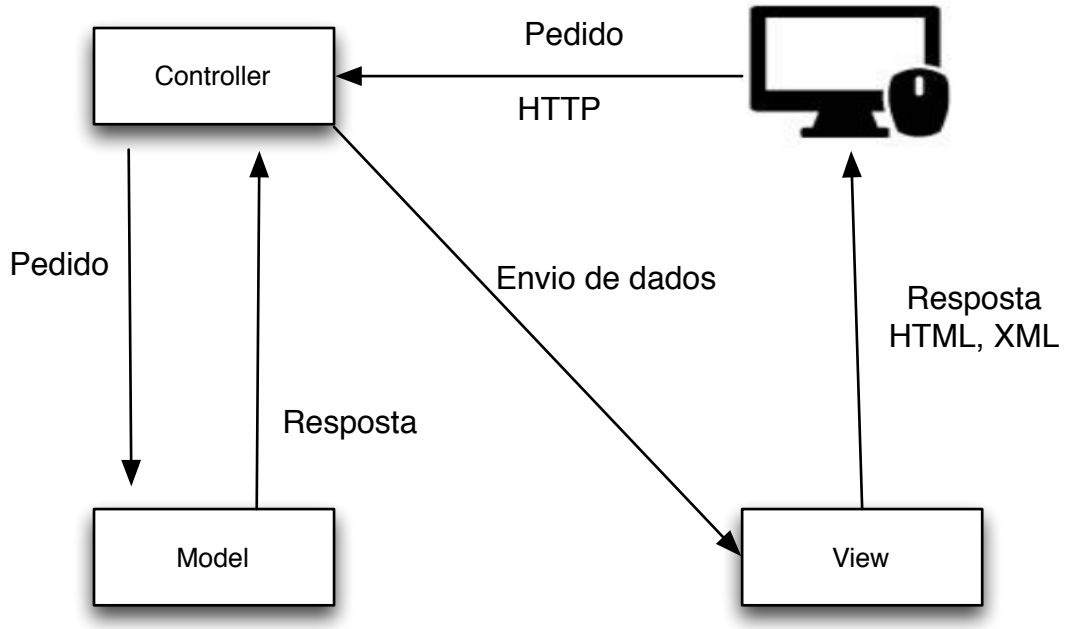

Figura 21 - Diagrama de funcionamento da arquitetura MVC

3.6.2 Wt (Web toolkit)

Wt (lendo-se *witty*) [26] é uma *framework* desenvolvida para a linguagem de programação C++. Permite aos programadores de C++ a criação de aplicações *web* utilizando o estilo de programação utilizado para fazer as GUI nesta linguagem, sendo depois estas aplicações traduzidas pelo Wt para o browser *web*. Utiliza componentes individuais em vez de utilizar o padrão de arquitetura de *software* MVC (*Model View Controller*). Uma aplicação web desenvolvida com o Wt, é escrita apenas numa linguagem, o C++, sendo que as bibliotecas são capazes de gerar o código HTML/XHTML, JavaScript ou AJAX necessário

[27], tornando-se assim na Framework de C++ mais popular entre os programadores desta linguagem de programação.

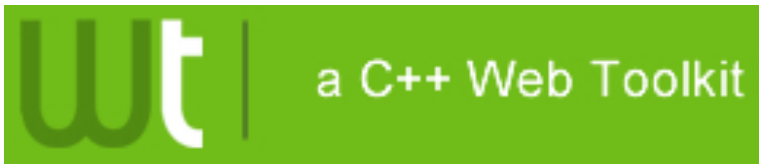

Figura 22 - Logotipo da framework Wt

3.6.3 Django

Django [28] é uma *web framework* escrita na linguagem de programação Python, que foi criada por Jacob Kaplan-Moss, Adrian Holovaty e Simon Willison em 2003 com o intuito de criar um gestor de conteúdos para ajudar o departamento de tecnologias *web* de um jornal da região de Kansas nos Estados Unidos da América. Apenas em 2005 é que foi lançada como *opensource* e com licença BSD (Berkeley Software Distribution), que se trata de isso mesmo, uma licença de código aberto. Segundo a página de apresentação e toda a comunidade do Django, esta Framework ajuda os programadores a desenvolverem aplicações com maior rapidez e com menos código, desenhada para perfecionistas.

Figura 23 - Logotipo da framework Django

Esta *framework* segue a arquitetura padrão MTV (*Model Template View*) e segue o principio DRY (*Don't Repeat Yourself)* que permite aos programadores reutilizarem ao máximo o código já desenvolvido evitando que estes se repitam.[29]

A arquitetura padrão MTV tem as seguintes características principais que a definem e que separa o processo do mesmo em 3 camadas:

- *Model*: Refere-se à camada de acesso à base de dados, que contém toda a informação sobre os dados, a forma de como conseguimos acedê-los, como validá-los e toda a relação entre eles.

- *Template*: Refere-se à camada de apresentação, onde através de um conjunto de páginas HTML é definida a apresentação ao utilizador da página *web*. Aqui são tomadas todas as decisões em relação à interface da aplicação *web*.

- *View*: Refere-se à intitulada camada da lógica de negócios. Nesta camada está presente toda a informação que irá diferenciar os *models* e *templates*, isto é, irá relacionar cada *model* com o *template* correto.

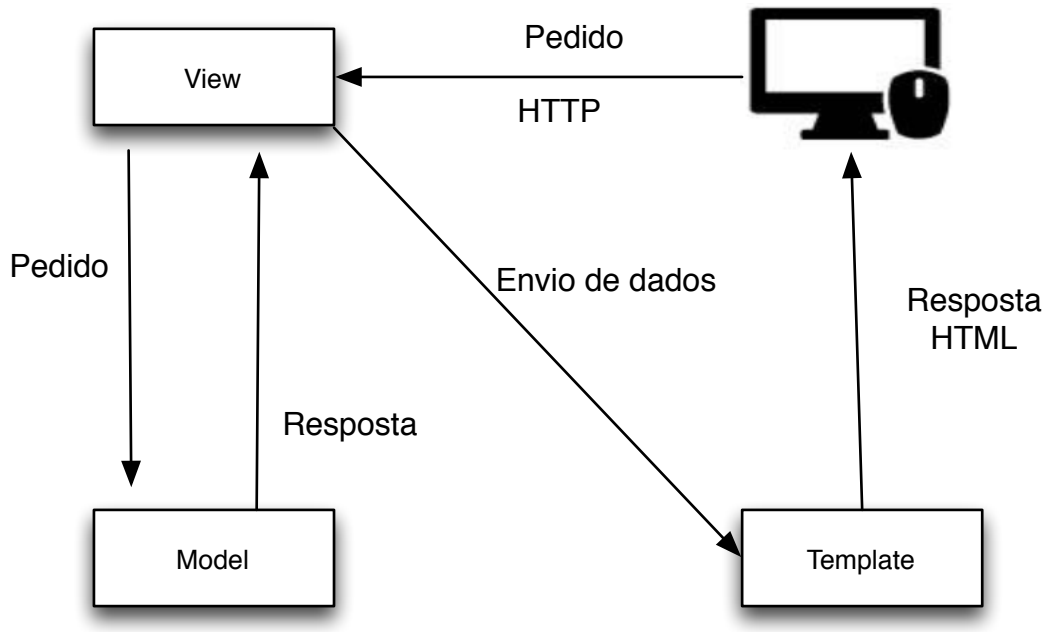

Figura 24 - Diagrama de funcionamento da arquitetura MTV

3.7 Conclusão

Esta dissertação está direcionada para o desenvolvimento de uma aplicação em tempo-real para deteção de fraturas ósseas, como tal, ao se efetuar pesquisa, verificou-se que apesar de o diagnóstico à distância e a telemedicina estarem a ser bastante utilizados, apenas existem aplicações do tipo educacional no que concerne os ultrassons. Para clarificar, apresentaram-se algumas das aplicações existentes no mercado.

Apresentou-se também o sistema de apoio ao utilizador que foi desenvolvido em [12], que consiste num aplicativo desenvolvido em MATLAB, onde são aplicados os filtros de remoção de ruído e os métodos de pesquisa de linha de osso e identificação de fratura óssea apresentados no capítulo 2, com todos os parâmetros já definidos.

Para finalizar este capítulo, fez-se uma breve explicação do conceito Framework web e apresentaram-se 3 *frameworks web,* Spring, Wt e Django, escritas em Java, C++ e Python, respetivamente. Estas são as 3 linguagens de programação mais direcionadas para o desenvolvimento web, que permitem uma ligação ao MATLAB através de um mecanismo que será apresentado no capítulo seguinte, juntamente com a metodologia proposta para o desenvolvimento de uma aplicação em tempo-real de fraturas ósseas em imagens de ultrassom, usando uma destas 3 *frameworks*.

Capítulo 4 – Metodologia proposta para aplicação em tempo-real para deteção de fraturas ósseas em imagens de ultrassom

4.1 Introdução

Neste capítulo é proposto um aplicativo web para deteção de fratura ósseas adaptando a aplicação desenvolvida em [12], em linguagem MATLAB. A aplicação consiste em fazer a deteção de fraturas ósseas em imagens de ultrassom, em que aplica os filtros e métodos de pesquisa referidos no capítulo anterior. Como tal, e para que se conseguisse o aproveitamento do trabalho desenvolvido anteriormente, verificou-se que havia a possibilidade de interligar a aplicação com outras linguagens de programação de forma a se comutar, por exemplo, uma página web com a aplicação através de uma *engine*. A *engine* é apenas um pacote que, após instalação, permite a ligação entre o *MATLAB* e o algoritmo desenvolvido, por exemplo, em JAVA ou *Python*. Com esta ferramenta, pode-se utilizar as potencialidades do *MATLAB*, numa aplicação desenvolvida numa destas linguagens, que neste caso, torna-se bastante vantajoso.

Para ajudar o desenvolvimento da aplicação, a utilização de uma *framework web* pode ser uma mais-valia pelas razões referidas no capitulo anterior, como tal, apresentar-se-á a *framework* escolhida de forma mais detalhada e os métodos utilizados para interligação entre a aplicação desenvolvida em [12] e o aplicativo web aqui proposto.

4.2 Web framewowrk utilizada: Django

Django é um *web application framework* grátis e *open source* escrito na linguagem de programação Python, que simplifica bastante o processo de criação de uma aplicação *web*, como já foi referido no capítulo anterior. A utilização desta *framework* deve-se ao facto de esta ter disponível muita documentação bastante completa e explicada de forma clara, e por ser uma das mais utilizadas para uma linguagem que está em crescente popularidade e por consequência em crescente utilização [30]. Sites como *Instagram*, *Pinterest*, *Mozilla* e o site da *National Geographic* utilizam esta *Framework.*

No capitulo anterior foi referido que esta Framework utiliza a arquitetura padrão MTV esta arquitetura é muito semelhante à arquitetura MVC, mas o *Django* utiliza a sua própria forma de implementação, isto porque, na parte do *Controller*, é tudo tratado pela *framework Django* e toda a ação acontece nos *models*, *views* e *templates*, enquanto que na MVC, o *Controller* controla todo o fluxo de informação entre o *model* e o *view* através de programação lógica que decide se a informação que sai da base de dados passa pelo *model* ou pelo *view*.

4.3 MATLAB engine para Pyhton

O nome correto e que é dado pela MathWorks, é MATLAB Engine API para Python [31], sendo que esta fornece um pacote para o Pyhton chamado "matlab" e que permite que o utilizador possa chamar funções de MATLAB nesta linguagem de programação.

Para que seja possível a sua utilização, é necessária a instalação [32] desse mesmo pacote e do MATLAB. Com esta ferramenta pode-se chamar uma função do MATLAB e esta devolve o resultado diretamente para o programa em Python. Para que se perceba um pouco melhor o funcionamento da *engine* apresenta-se de seguida o exemplo fornecido pela MathWorks:

```
    import matlab.engine
    eng = matlab.engine.start_matlab()
    tf = eng.isprime(37)
    print(tf)
True
```

Neste exemplo podemos ver que é necessário haver o *import*, porque se trata de um pacote, neste caso o pacote "matlab.engine". Em seguida é criado o objeto "eng", uma referência para aceder ao engine do MATLAB, inicializado com a função "start_matlab()"**.**

`"tf"` corresponde ao objeto que chamará a função feita em MATLAB com o nome "isprime" com o respetivo argumento. Finalmente é feito o pedido para nos mostrar a resposta que é dada pela função que neste caso foi "True".

No caso aqui em estudo, ter-se-á mais argumentos de entrada e saída, tal como é demonstrado no exemplo fornecido em [33] que determina o maior denominador comum entre dois números:

```
    import matlab.engine
    eng = matlab.engine.start_matlab()
    t = eng.gcd(100.0,80.0,nargout=3)
    print(t)
(20.0, 1.0, -1.0)
```

Neste exemplo, verifica-se que tudo é igual ao primeiro exemplo com a exceção da inserção dos argumentos de entrada. Nestes, ter-se-á que definir o número de argumentos de saída que se pretende que sejam apresentados.

4.4 Aplicação web proposta

A aplicação proposta nesta dissertação utiliza as ferramentas acima referidas, isto é, a *framework* Django e a MATLAB engine para fazer a ligação entre o sistema de apoio ao utilizador desenvolvido em [12] e a interface no âmbito desta tese desenvolvida.

Nos exemplos apresentados em 3.3, pode-se verificar que a engine espera que se chame apenas uma função com os respetivos argumentos de entrada, como tal, foi necessária uma adaptação do algoritmo do sistema de apoio transformando-o numa só função com os respetivos argumentos de entrada e saída e foi necessário alterar também o método de escolha do tipo de osso. A razão desta adaptação reflete-se no facto de o algoritmo estar preparado e definido para ser apresentado em forma de aplicação desenvolvida em MATLAB, uma GUIDE. Assim, apenas os filtros e métodos aplicados eram funções MATLAB, sendo que o algoritmo onde era efetuada a atribuição de parâmetros e métodos, apenas chamava essas funções.

Após este passo, o desenvolvimento da aplicação passou por duas versões sendo a primeira uma versão mais simplificada e a segunda mais aproximada à interface que era apresentada no sistema de apoio desenvolvido em [12].

4.4.1 Primeira versão

Numa primeira versão, constituindo uma versão de teste, apresenta-se uma interface com um aspeto visual simplificado. Para o utilizador aceder à aplicação tem de efetuar a autenticação com o seu respetivo nome de utilizador e palavra-passe como apresentado na Figura 23. Na figura 24 apresenta-se a página inicial da aplicação, onde se pode ver uma caixa

de texto com as instruções para o utilizador saber como proceder para obter o resultado da imagem que pretende que seja analisada pela aplicação. Nas imagens seguintes mostra-se todo processo através de *printscreens* da aplicação.

Figura 25 - Área de autenticação para aceder à aplicação

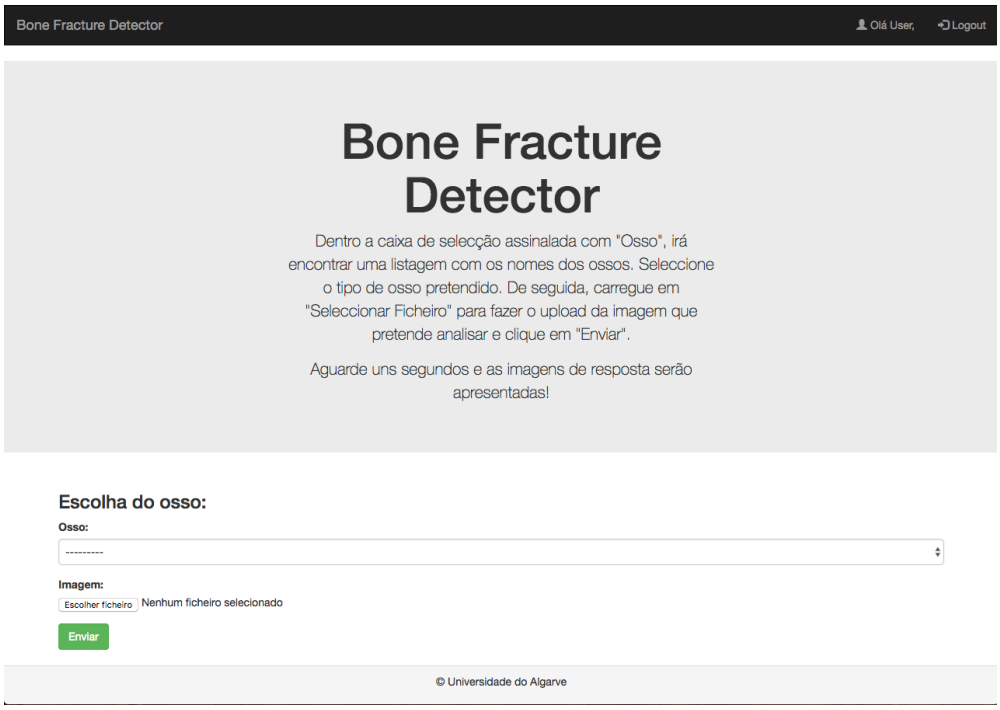

Figura 26 - Página inicial da aplicação versão 1

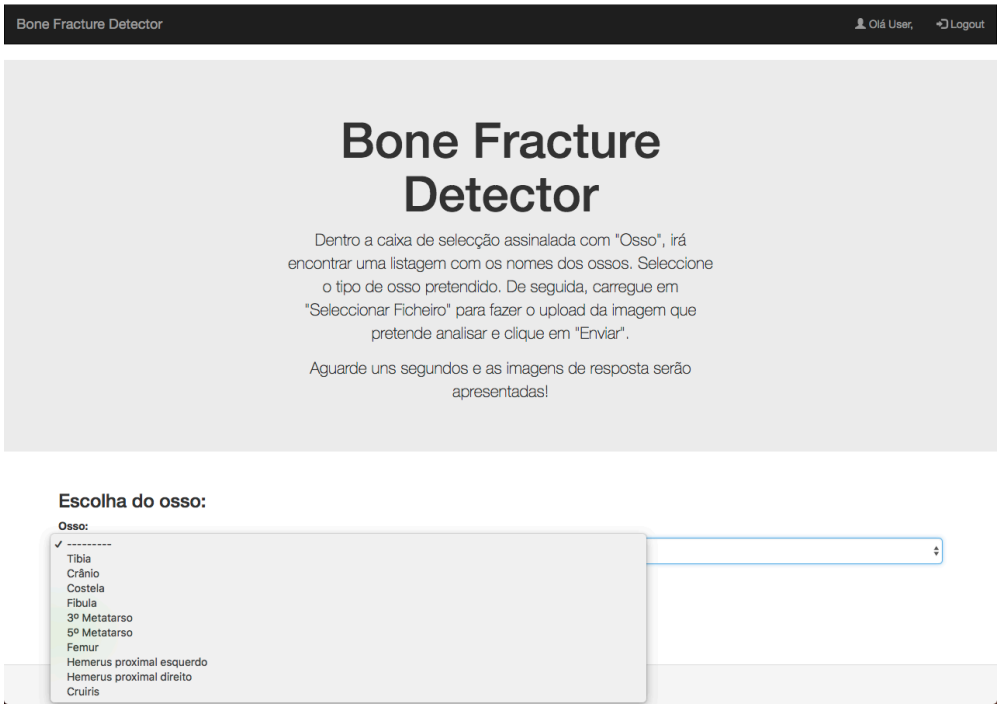

Figura 27 - Escolha do osso pretendido para a análise

Dentro da caixa de seleção encontra-se uma listagem de ossos (Figura 25), em que o utilizador deverá escolher um deles e de acordo com o osso que pretende que seja analisado, para de seguida se proceder à escolha do ficheiro (Figura 26), mais concretamente uma imagem em formato jpeg, que se pretende que seja analisado pelo aplicativo.

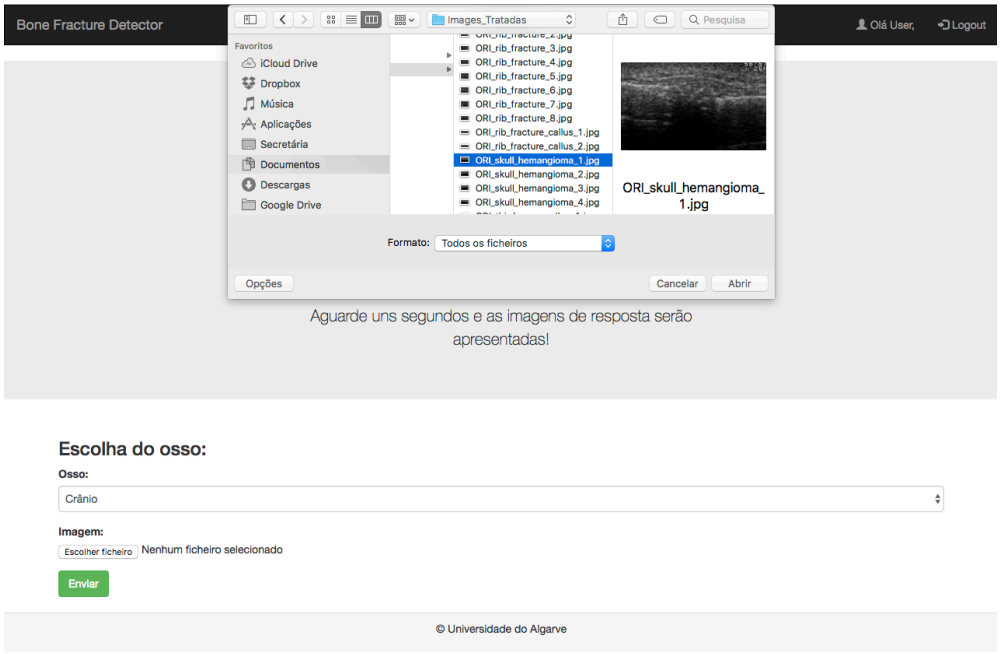

Figura 28 - Seleção da imagem correspondente ao osso escolhido

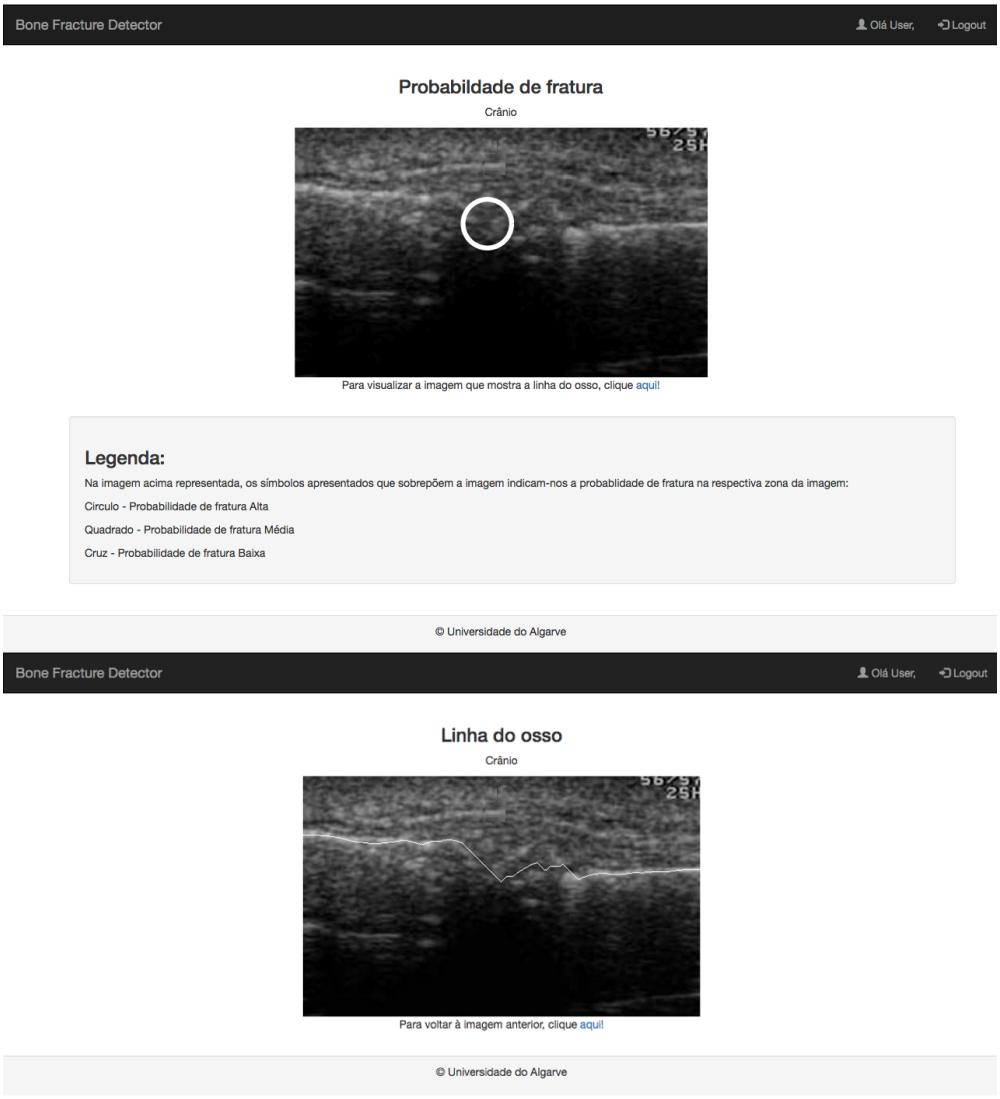

Figura 29 - Output da aplicação

Concluída a análise da imagem por parte da aplicação, mais especificamente, pelo programa em MATLAB, é apresentado o resultado da mesma com a indicação da probabilidade de existência de fratura na imagem analisada de acordo com a tipologia círculo, triângulo ou cruz, referida na secção 2.2.6. O utilizador tem ainda a hipótese de nesta janela de visualização, selecionar a opção de aceder à imagem que apresenta a linha osso.

Caso pretenda voltar ao início e analisar uma nova imagem, o utilizador deve "clicar" no nome da aplicação que se encontra no canto superior esquerdo e sendo conduzido para a pagina inicial do aplicativo, possibilitando assim uma nova análise de imagem ultrassonográfica.

4.4.2 Segunda versão

Numa segunda versão, adotou-se o método de apresentação mais aproximado ao modelo apresentado em [12], isto é, a utilização da imagem de um esqueleto para se proceder à escolha do osso que se pretende que seja analisado. Tal como na primeira versão, o utilizador terá de inserir os seus dados de autenticação para aceder a esta primeira página que agora será exibida com o aspeto visual da figura 28.

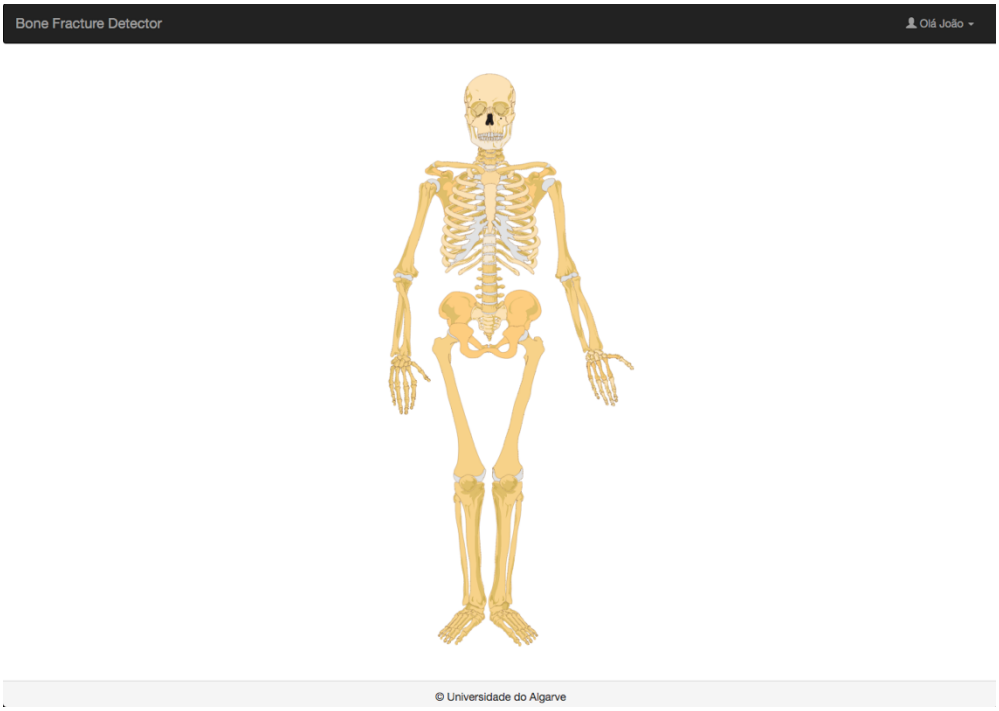

Figura 30 - Página inicial da aplicação

A imagem do esqueleto é uma imagem no formato SVG (*Scalable Vector Graphics*), que representa as imagens em formato vetorial, sendo que basicamente é um mapa XML (*eXtensible Markup Language*) que descreve matematicamente uma figura gráfica bidimensional. A utilização deste formato de imagens em aplicações é bastante comum, para que não haja perda na qualidade da imagem no redimensionamento da mesma, e para se conseguir a utilizar como forma de interação com o utilizador, tal como neste projeto.

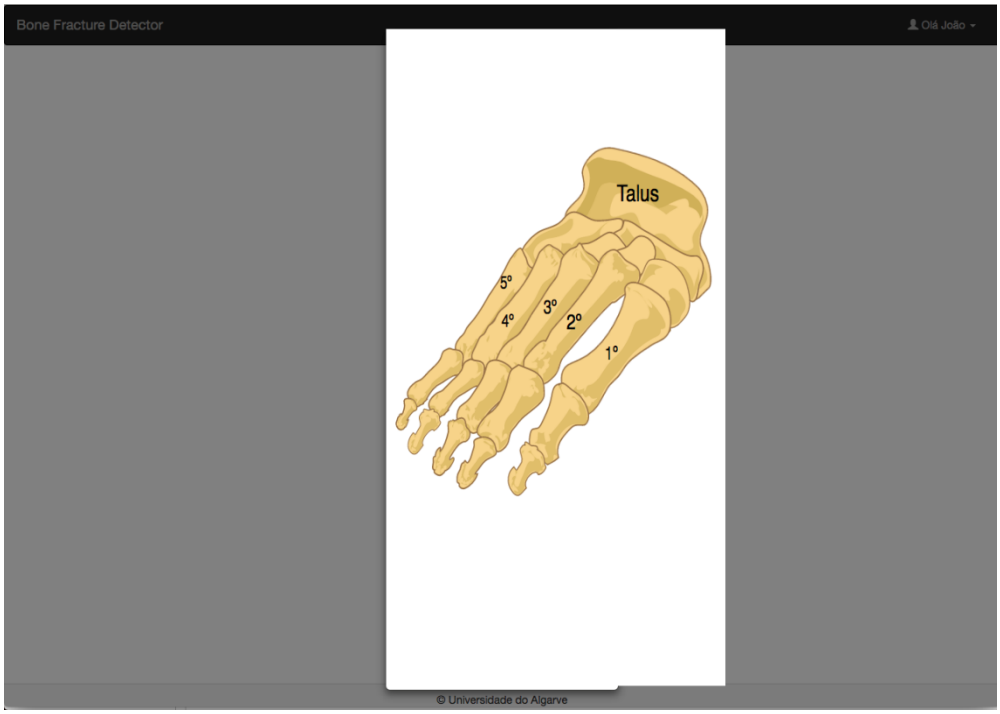

Figura 31 - Imagem ampliada para seleção de ossos no pé

Nesta versão, com a utilização da imagem do esqueleto, a escolha de determinados ossos torna-se mais complicada por existirem ossos mais pequenos, como os ossos dos pés. Como tal, e para facilitar a escolha desses, adotou-se o mesmo método utilizado em [12] com a ampliação da imagem na zona dos respetivos ossos.

Na Figura 31 e Figura 32, podemos ver os exemplos de ampliação de imagem utilizados na aplicação.

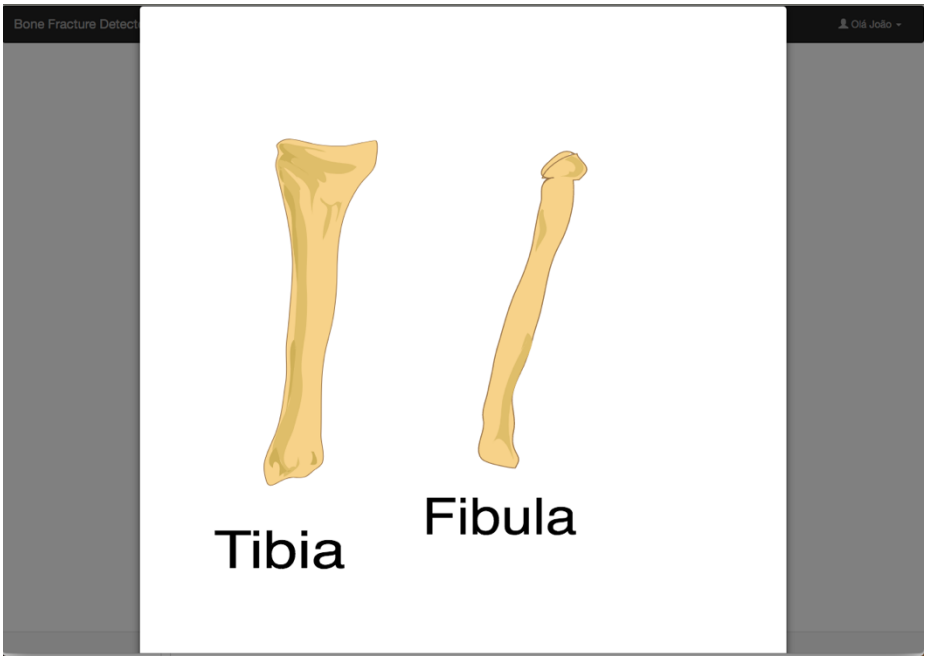

Figura 32 - Imagem ampliada para seleção entre fíbula e tíbia

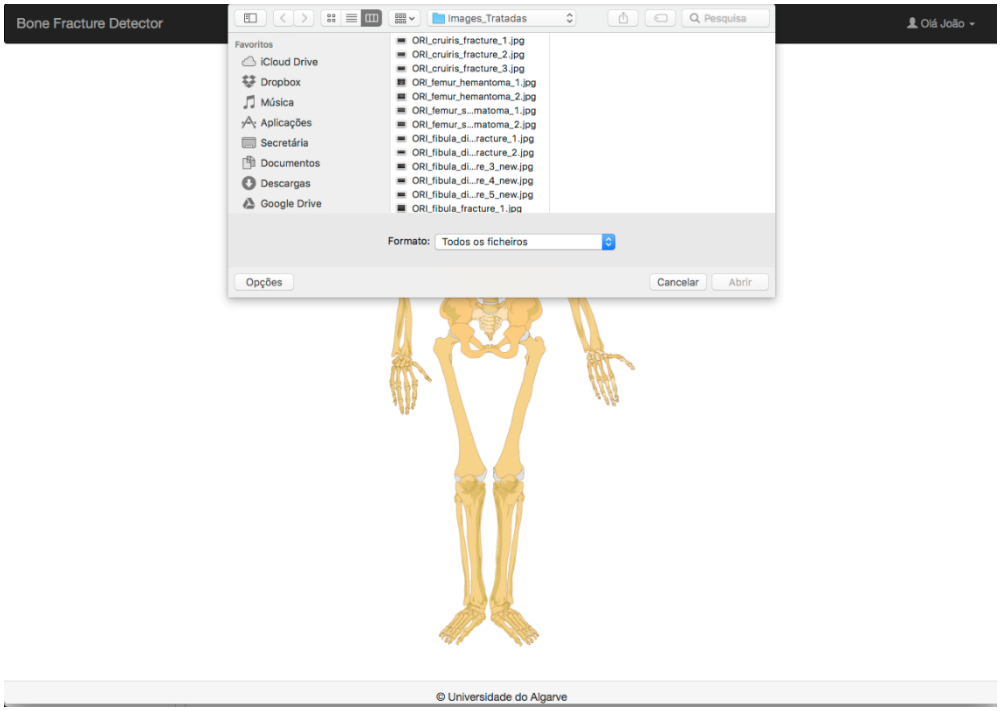

Figura 33 - Seleção do ficheiro para ser enviado para análise

Efetuada a escolha do osso, o pedido de seleção de ficheiro será exibido automaticamente e, posteriormente, será apresentado o resultado da análise o qual será idêntico à figura 27.

Para se voltar ao início e caso se queira analisar uma nova imagem, o processo é o mesmo da versão anterior, ou seja, basta "clicar" no nome da aplicação que se encontra no canto superior esquerdo, e a página inicial será novamente exibida.

Nesta versão está também incluído um espaço onde o utilizador poderá visualizar todas as imagens que foram analisadas e processadas pela aplicação. Pretende-se com este espaço reduzir o risco de perda da informação devido a um encerramento inesperado da aplicação e/ou situações similares que impedem o utilizador de visualizar as imagens com a atenção necessária e facilitar a consulta de outras imagens anteriormente analisadas.

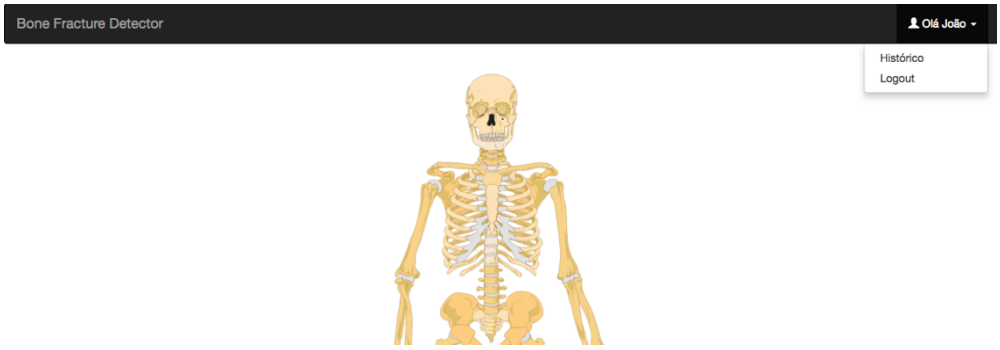

Figura 34 - Pequeno menu do utilizador

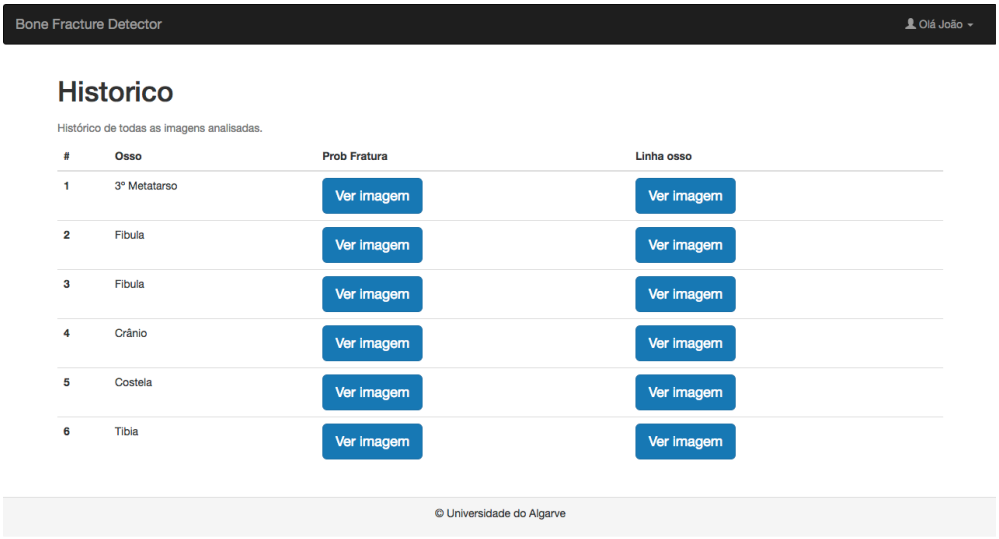

Figura 35 - Visualização do histórico de imagens analisadas

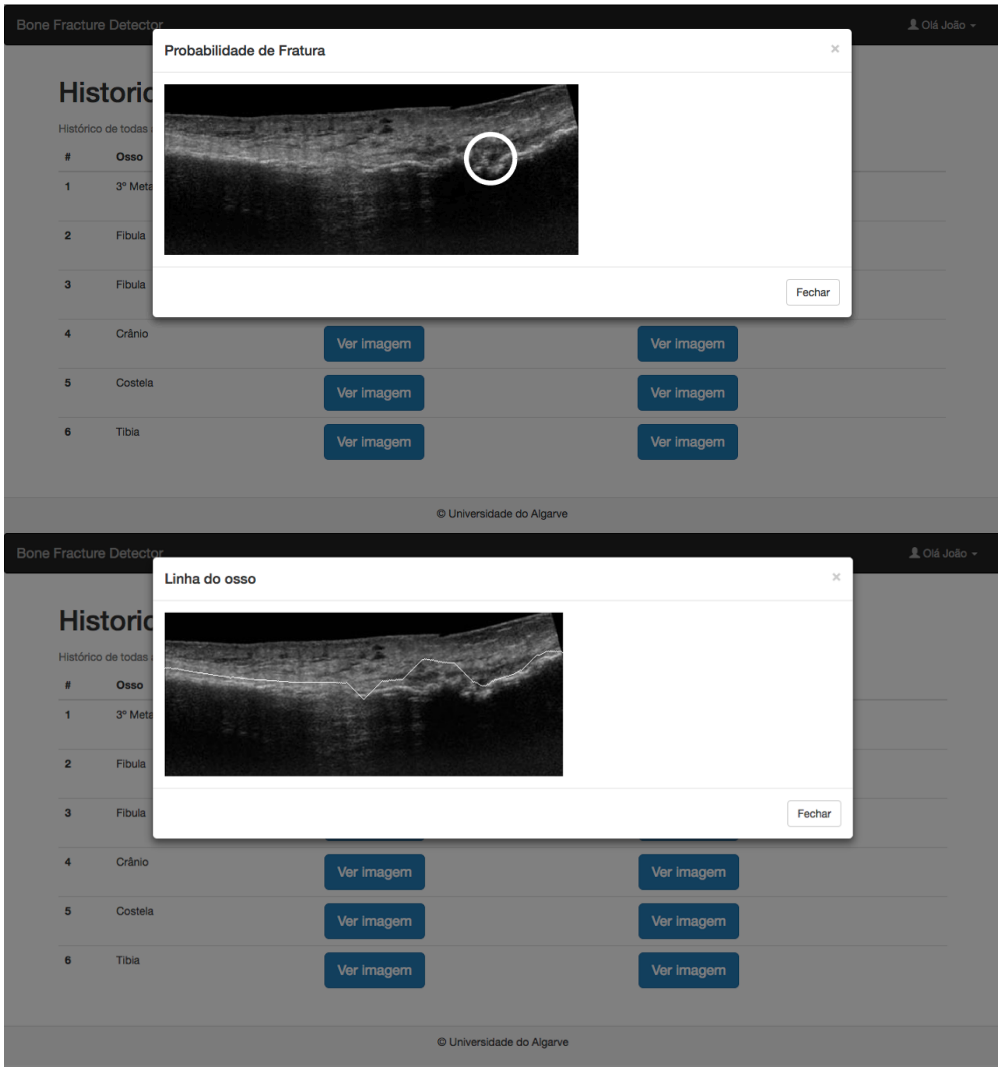

Figura 36 - Apresentação das imagens do histórico

4.5 Visão geral da aplicação

Esta aplicação foi desenvolvida com base num algoritmo trabalhado em [12] e que para facilitar o uso do mesmo por parte do utilizador, foi desenvolvido um sistema de apoio ao utilizador que se pode traduzir numa aplicação ou numa GUIDE, por ter sido desenvolvida em MATLAB. No capítulo anterior apresentou-se os filtros que são utilizados para a redução do ruído *Speckle* (*Mean Filter* e SSR) e os métodos de identificação do osso e possível fratura óssea (*leftright*, *middle* e *3spli+middle*) utilizados nesse algoritmo.

Na aplicação aqui desenvolvida, toda esta parte estará ligada ao servidor e o utilizador nunca terá acesso, ou seja, toda a escolha de parâmetros será feita automaticamente a partir do momento em que é feita a escolha do osso na imagem e a aplicação dos filtros no momento em que a imagem é enviada para o servidor para ser analisada. Na figura 35 podemos visualizar o diagrama esquemático do funcionamento da aplicação aqui desenvolvida.

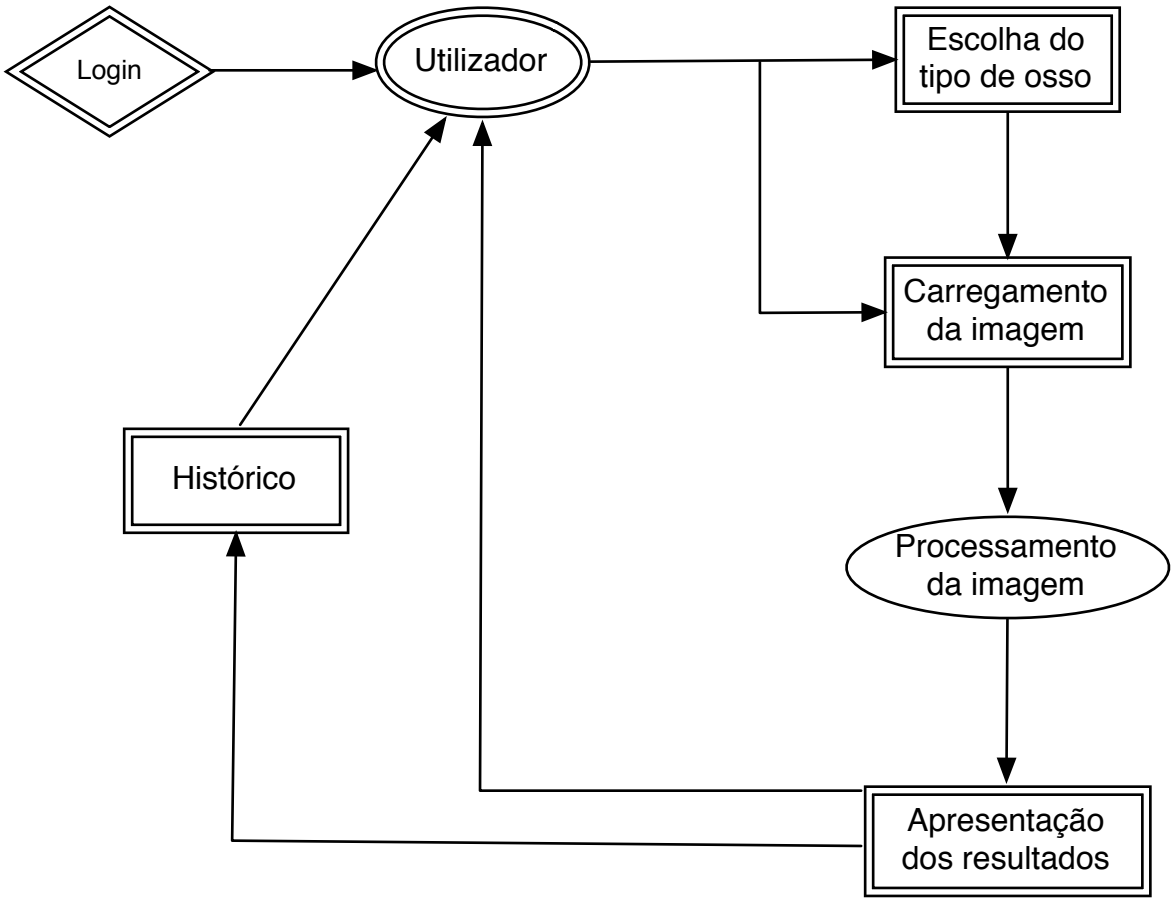

Figura 36 - Diagrama esquemático do funcionamento da aplicação

4.6 Conclusões

Neste capítulo foi apresentado o método proposto para identificação de fraturas ósseas em tempo real de imagens de ultrassom. Foi visto no capítulo anterior que havia a possibilidade de interligar a aplicação em MATLAB desenvolvida em [12], como tal, escolheu-se então a linguagem *Python* para este fim.

A utilização de uma Framework, pode ser sempre uma mais-valia e a utilização das mesmas está em grande crescimento. Para este projeto, escolheu-se a Framework *Django* por ser a mais utilizada nesta linguagem, a documentação disponível estar bastante completa, disponível gratuitamente e por a linguagem *Python* e esta Framework ainda estarem em grande crescimento e procura no mercado de trabalho. A utilização da MATLAB *engine* permitiu então a ligação entre o sistema de apoio ao utilizador que já havia sido desenvolvido, conseguindo assim passar para o desenvolvimento da aplicação deste projeto.

É feita uma apresentação da primeira versão, que serviu um pouco como versão de teste, onde se conseguiu verificar se a resposta da aplicação era a pretendida e os respetivos tempos de resposta que esta levaria para apresentar os resultados. De seguida, é apresentada então a versão final da aplicação, a segunda versão, onde se inseriu uma imagem SVG para interação com o utilizador. Com esta alteração, a escolha do osso passaria a ser feita através da imagem, onde o utilizador clica sobre a imagem no osso pretendido e o pedido para selecionar a imagem pretendida para análise, será feito automaticamente. Ao acrescentar esta imagem para interação, os tempos de resposta seriam novamente um problema. No capítulo seguinte, passar-se-á para a apresentação dos resultados obtidos na análise dos tempos de resposta obtidos no projeto anterior [12] e na aplicação desenvolvida no presente projeto de dissertação.

Capítulo 5 – Procedimento experimental e análise de resultados

5.1 Introdução

Como já referido anteriormente, o algoritmo utilizado para processamento das imagens ultrassonográficas, foi desenvolvido em [12], sendo que houve a necessidade de efetuar uma pequena adaptação do mesmo para que se conseguir utilizar a ferramenta que liga o MATLAB ao algoritmo em Python.

O tempo que a aplicação demora a apresentar os resultados é um detalhe bastante importante nesta tese por estarmos a lidar com aplicativos web e aplicações em tempo-real. Pretende-se por isso, neste capítulo, apresentar uma revisão/análise dos tempos de resposta do sistema de apoio ao utilizador desenvolvido em [12], para se conseguir comparar os resultados obtidos com esta nova metodologia.

5.2 Análise do algoritmo desenvolvido pelo Luís [12]

5.2.1 Imagens utilizadas para análise

O desenvolvimento deste projeto foi desde o primeiro momento baseado no algoritmo e sistema de apoio ao utilizador desenvolvido em [12], e na realização do mesmo foram utilizadas imagens de fraturas ósseas obtidas através de ultrassom para testar os diferentes métodos de identificação de fraturas.

Num total de 44 imagens que foram adquiridas de 2 bases de dados públicas diferentes [34] e [35], existem imagens de vários ossos do corpo, e oito delas apresentam patologias do tecido cortical. Na tabela seguinte, podemos verificar os diferentes tipos de osso, patologias e o número de imagens para cada tipo.

Tipo de osso e patologia	Número de imagens
Fratura de costela com interrupção da cortical	8
Fratura de costela com formação de calo ósseo	2
Hemangioma do crânio com uma irregularidade cortical	4
Hematoma subperiosteal do fémur	2

Hematoma subperiosteal calcificado do fémur	2
Fratura de *stress* no perónio (fíbula) distal	2
Fratura cominutiva do perónio (fíbula) com interrupção da cortical óssea	2
Fratura de *stress* da tíbia distal com calo periosteal	7
Fratura da diáfise proximal do úmero	4
Fratura de *stress* do terceiro metatársico com formação de calo ósseo	4
Fratura de *stress* do quinto metatársico com calo periosteal	4
Fratura da extremidade distal do cruiris com formação de calo ósseo	3
	Total: 44

Tabela 1 - Tipos de osso, patologia e número de imagens [12]

Algumas das imagens continham uma seta a apontar para o local exato onde estava localizada a fratura e uma moldura em torno da imagem, por essa razão, antes de iniciar a pesquisa da fratura, foi necessário um pré-processamento da imagem, eliminando essa seta copiando os valores dos pixéis em seu redor. Todas estas imagens foram tratadas pelo Luís durante a realização da sua tese e aproveitadas aqui para testar o algoritmo utilizando uma nova metodologia.

5.2.2 Tempos de resposta associados aos métodos do algoritmo

O tempo de resposta do sistema de apoio desenvolvido anteriormente, era um ponto crucial para este novo projeto por se tratar de uma aplicação em tempo-real e a resposta ter de ser quase imediata. Em [12] está representada a Tabela 2 que apresenta os tempos médios consumido pelo algoritmo, apenas na redução do ruído, para cada um dos filtros nas 44 imagens existentes.

Filtro	**SSR**	**Mean**	**Median**	**Gamma**
Tempo (milisegundos)	67,45	0,05	91,62	647,8

Tabela 2 - Tempo médio de processamento para tipo de filtro [12]

Com esta informação, apesar de ser um tempo mínimo, consegue-se perceber onde e o porquê de haver maiores ou menores perdas de tempo em diferentes ossos e imagens. Tudo isto está diretamente dependente do tipo de osso que se irá analisar, da quantidade de ruído existente na imagem e ainda do método de identificação de fratura que será aplicado nesse

osso. Como já foi referenciado, estes parâmetros já estão todos definidos para cada tipo de osso sem que o utilizador tenha de se preocupar com a escolha dos mesmos. Através da tabela seguinte, podemos verificar todas as combinações de parâmetros que apresentaram melhores resultados.

Tipo de osso	**Filtro**	**Isline (%)**	**L**	**Tipo de procura**
Costela	*Mean Filter*	15	50	*middle*
Crânio	*SSR*	5	5	*middle*
Fémur	*Mean Filter*	20	50	*3split+middle*
Perónio (Fíbula)	*SSR*	20	20	*3split+middle*
Tíbia	*SSR*	40	50	*leftright*
Diáfise proximal do úmero	*SSR*	10	20	*leftright*
Terceiro metatársico	*Mean Filter*	5	50	*3split+middle*
Quinto metatársico	*SSR*	5	20	*middle*
Cruiris	*Mean Filter*	10	20	*3split+middle*

Tabela 3 - Tipo de ossos e as variáveis a aplicar aos algoritmos de pesquisa de linha de osso [12]

Para a pesquisa da linha osso, existem 3 métodos, como já referido anteriormente no capítulo 2. Para cada um dos métodos também está uma perda de tempo associada, tempo que terá de se somar ao tempo acima referido relativo à redução de ruído. Foi calculado também em [12] o tempo médio de cada um dos algoritmos de pesquisa de linha de osso como podemos ver na Tabela 4.

Tipo de pesquisa	**leftright**	**middle**	**3split+middle**
Tempo (milissegundos)	292,57	283,34	383,45

Tabela 4 - Tempo médio para cada um dos métodos de pesquisa de linha de osso propostos[12]

Ao identificar a linha do osso, o algoritmo irá passar para a identificação de fratura, método que foi também apresentado no capítulo 2, secção 2.2.6. Mais uma vez, para este método foi também calculado o tempo médio de resposta, como se pode ver na Tabela 5.

Número de fraturas	**Uma**	**Duas**	**Três**
Tempo (milissegundos)	2,9	17,5	73

Tabela 5 - Tempo de identificação das fraturas [12]

Neste último processo o resultado também é como esperado, pois quantas mais fraturas existirem, maior será o tempo de resposta.

5.2.3 Resultados obtidos em teste ao sistema de apoio ao utilizador

Antes de começar o desenvolvimento da nova metodologia, achou-se interessante verificar o tempo de resposta que a aplicação iria necessitar para apresentar a resposta ao utilizador. Este teste passou pela utilização do programa MATLAB, em que se utilizou uma "ferramenta" chamada de "Run and Time" para se conseguir exatamente isso, correr a aplicação e verificar os tempos de resposta.

Nas tabelas seguintes, podemos verificar os tempos de resposta do sistema ao aplicar os diferentes métodos e filtros de redução de ruído.

Nota: Os tempos apresentados são para o conjunto completo das imagens de cada tipo de osso. Apenas o pior caso é uma medição individual.

	Nº de imagens	**Mean Filter**	**SSR**	**Pior caso**
Costela	10	0,010 s		0,010 s
Crânio	4		3,247 s	0,745 s
Fémur	4	0,053 s		0,011 s
Perónio (fibula)	4		5,147 s	1,282 s
Tíbia	7		9,058 s	1,550 s
Diáfise proximal do úmero	4		6,207 s	1,248 s
Terceiro metatársico	4	0,010 s		0,010 s
Quinto metatársico	4		3,834 s	1,013 s
Cruiris	3	0,010 s		0,010 s

Tabela 6 - Tempos de "Run and Time" dos filtros de remoção de ruído

A medição dos tempos de análise das imagens foi feita em dois momentos, um com o conjunto total das imagens existentes para cada tipo de osso, e posteriormente, cada imagem individualmente. Na medição dos filtros de remoção de ruído, verificou-se que com a utilização do filtro *Mean*, os tempos de resposta eram praticamente sempre os mesmos, como se pode verificar na Tabela 6. Foi feito o teste várias vezes para despiste, e o resultado obtido foi sempre o mesmo.

	Nº de imagens	**leftright**	**middle**	**3split+middle**	**Pior caso**
Costela	10		11,302 s		1,426 s
Crânio	4		4,637 s		1,145 s
Fémur	4			24,778 s	6,898 s
Perónio (fíbula)	4			22,622 s	5,787 s
Tíbia	7	12,770 s			2,325 s

Diáfise proximal do úmero	4	8,934 s			1,789 s
Terceiro metatársico	4			17,120 s	4,631 s
Quinto metatársico	4		5,307 s		1,524 s
Cruiris	3			10,079 s	4,181 s

Tabela 7 - Tempos de "Run and Time" dos métodos de identificação da linha de osso

Na utilização dos métodos de identificação de fratura, o método 3split+middle é claramente o que mais tempo consome, como já se tinha visto anteriormente.

Nota: Estes testes foram realizados com a utilização da versão 2012b do MATLAB.

5.3 Implementação da metodologia proposta

A implementação deste sistema em tempo-real envolve a comunicação entre servidor e cliente, onde se espera que a comunicação entre ambos tenha muito pouco atraso.

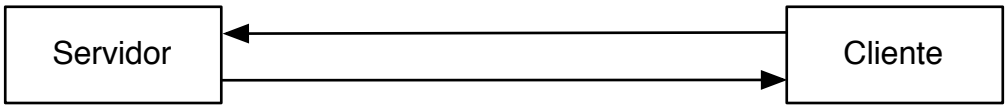

Figura 37 - Diagrama de comunicação entre servidor e cliente

Esta comunicação na primeira versão da aplicação, e ainda durante o desenvolvimento da segunda versão foi o maior obstáculo, sendo que a perda de tempo de resposta do servidor era muito superior ao esperado. Por cada vez que se enviava uma imagem para ser processada e analisada pela aplicação, existia uma perda de tempo que se situava entre os 25 e 30 segundos. Para uma aplicação em tempo-real, este é um resultado bastante problemático.

Inicialmente, abria-se a aplicação e iniciava-se o processo de escolha de osso e envio da imagem para ser analisada pela aplicação. Aqui, o tempo médio de espera pela resposta era cerca de 26 segundos, valor que era explicado pelo facto de cada vez que era enviada uma imagem para ser processada, era necessário inicializar o *engine* do MATLAB, como referido no capítulo 4, secção 4.3.

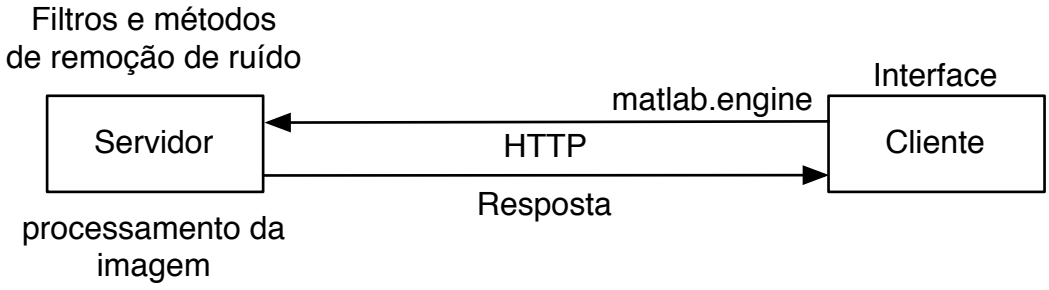

Figura 38 - Diagrama de comunicação na primeira versão da aplicação

Na passagem para a segunda versão da aplicação, ao inserir uma imagem do esqueleto no formato SVG, de forma a se conseguir fazer a seleção do osso ao clicar sobre a imagem, esta perda de tempo tinha de ser altamente melhorada, porque por vezes este tipo de animações ou interações pode levar a que o *browser* leve mais tempo a renderizar a imagem.

Para que não houvesse este consumo exagerado de tempo durante a análise da imagem, o que faria com que o utilizador esperasse bastante para obter uma resposta, quando a deveria ter quase imediatamente, o processo passava por colocar a MATLAB *engine* a ser inicializado, uma só vez, no momento em que o servidor era ligado. Deste modo, apenas se perderia a maioria deste tempo uma só vez, tornando a resposta do servidor, mais rápida.

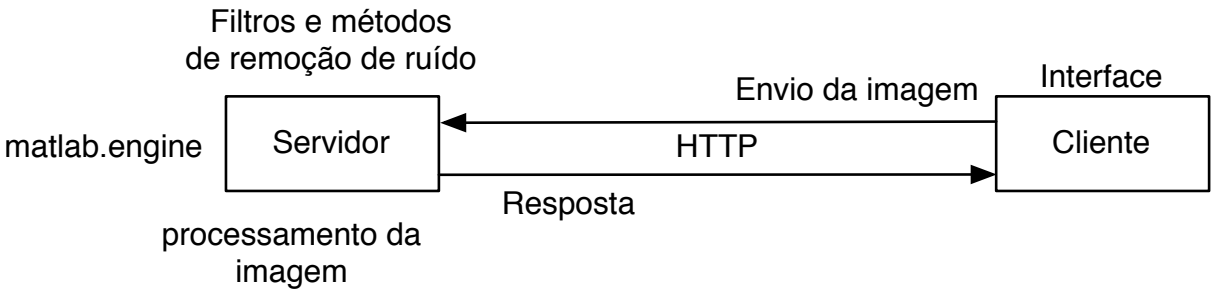

Figura 39 - Diagrama de comunicação pretendido para a segunda versão

Após vários testes, conseguiu-se chegar ao resultado pretendido. Com a MATLAB *engine* a ser ativada no momento em que se ligava o servidor, passou-se a ter a maior parte da perda de tempo apenas a uma vez e quando se liga o servidor. Ou seja, ligamos o servidor e automaticamente a MATLAB *engine* irá ativar, claro que com uma perda de tempo situada entre os 20 e 25 segundos, mas apenas acontecerá uma vez. A partir do momento em que servidor estiver ligado, a ligação entre a aplicação e o MATLAB já está feita, assim, quando o utilizador envia a imagem para ser processada pela aplicação, o tempo médio de espera situa-se entre 3 a 5 segundos, ao utilizar a versão 2014b do MATLAB. Este tempo já envolve o processamento da imagem e a comunicação entre servidor e cliente.

Comparativamente aos resultados obtidos na análise dos tempos do processamento e análise das imagens por parte da aplicação desenvolvida em [12], a diferença situa-se apenas na comunicação entre servidor e cliente.

5.4 Conclusões

Este capítulo iniciou-se com uma apresentação das imagens adquiridas, tratadas e utilizadas anteriormente em [12]. Um conjunto de 44 imagens de 9 tipos de osso diferentes, que foram adquiridas de duas bases de dados públicas diferentes.

Apresentou-se os tempos de resposta obtidos também em [12] no processamento das respetivas imagens para a redução de ruído utilizando 4 filtros de remoção do ruído *Speckle*, sendo que apenas dois desses filtros foram usados no algoritmo por terem apresentado melhores resultados na fase de testes.

Os métodos de identificação da linha do osso em conjunto com a identificação de fratura, já tendo apresentado bons resultados no que diz respeito aos tempos de resposta, e como tal, foram aqui apresentados também. Antes de iniciar o desenvolvimento do algoritmo para a nova aplicação, decidiu-se fazer um novo teste para verificar os tempos de resposta do sistema de apoio ao utilizador que utiliza os métodos de remoção de ruído, identificação da linha de osso e identificação da fratura. Os resultados obtidos nesse teste preliminar são apresentados neste capítulo tendo servido de base para o desenvolvimento do novo algoritmo. Pretendia saber-se a média dos tempos de espera máximos que se podia obter ao ser enviada uma imagem para ser analisada pela aplicação.

Este projeto envolvia a ligação entre servidor e cliente de forma a se conseguir utilizar a aplicação para deteção de fraturas ósseas em imagens ultrassónicas. Este tipo de comunicação por norma, envolve um tempo de resposta que não se espera que seja muito longo, e foi com esse princípio que o desenvolvimento deste projeto partiu. Os resultados obtidos na primeira versão da aplicação foram diferentes dos desejados, tendo-se obtido um intervalo entre 25 e 30 segundos de espera pela resposta. Na segunda versão da aplicação, foram feitas algumas alterações em que uma delas foi crucial para obter o objetivo pretendido. A colocação do mecanismo de comunicação com o MATLAB no lado do servidor, fez com que se perdesse a maioria do tempo no momento da ativação do servidor, fato que pode acontecer apenas uma vez por dia. Com esta alteração, o tempo de resposta da aplicação a apresentar os resultados da análise da imagem reduziu para cerca de 5 segundos em cada análise.

Capítulo 6 – Conclusões e trabalho futuro

6.1 Conclusões

De acordo com os resultados apresentados no capítulo anterior, conclui-se que os objetivos inicialmente propostos para este projeto de dissertação foram alcançados. O algoritmo do aplicativo móvel aqui desenvolvido permite, a identificação de fraturas ósseas nas imagens ultrassonográficas em tempo-real.

O desenvolvimento deste projeto deu continuidade ao estudo anteriormente desenvolvido em [11], [12] ao utilizar os métodos de remoção de ruído do tipo *Speckle* e os algoritmos de identificação da linha de osso e fratura óssea que obtiveram melhores resultados num aplicativo móvel de forma a permitir a deteção das fraturas ósseas em imagens de ultrassom em tempo-real.

Os métodos propostos e desenvolvidos no curso desta dissertação foram apresentados no capítulo 4 e os resultados obtidos na análise do sistema de apoio ao utilizador desenvolvido em [12], comparativamente aos resultados obtidos no aplicativo em tempo-real aqui desenvolvido, são apresentados no capítulo 5.

Ambas as versões da aplicação aqui desenvolvidas, apresentaram os resultados esperados a nível de apresentação das imagens com a identificação da linha de osso e identificação de fratura óssea. A primeira versão serviu para verificar que o tempo de resposta da aplicação era demasiado grande, e a razão passava pela inicialização do *engine* a cada vez que era submetida uma imagem para análise da aplicação. No desenvolvimento da segunda versão da aplicação, este fator foi tido em causa, tendo em conta o uso de uma imagem em formato SVG que poderia influenciar ainda mais o tempo de resposta da aplicação ao apresentar os resultados da análise da imagem. Nesta versão, a *engine* será inicializada no momento em que o servidor iniciar, o que fará com que a maior parte da perda de tempo aconteça apenas no início da sessão de diagnóstico. Esta alteração permitiu que o tempo de resposta da aplicação a cada vez que é enviada uma imagem para análise, passasse do intervalo de 25-30 segundos para um intervalo de tempo situado entre os 4 e 7 segundos. A inicialização do servidor nesta versão consumirá mais tempo, mas acontecerá apenas uma vez por sessão de diagnóstico. O importante será apresentar os resultados do processamento da imagem o mais breve possível, o que se conseguiu com sucesso, comparativamente aos

resultados obtidos na análise preliminar feita ao sistema de apoio ao utilizador, como demonstrado no capítulo anterior.

6.2 Trabalho futuro

Apesar de ter se ter chegado a resultados bastante positivos, existem aspetos do algoritmo que talvez possam ser melhorados.

Com a crescente popularidade e utilização da linguagem Python, a passagem do algoritmo do sistema de apoio ao utilizador desenvolvido em MATLAB para Python pode ser possível. Esta alteração poderá influenciar ainda mais o tempo de resposta da aplicação e com a qual se evitaria a utilização da MATLAB *engine*, que consome bastante tempo na respetiva inicialização.

Referências

[1] M. Mercuri, T. Sheth, e M. K. Natarajan, «Radiation exposure from medical imaging: A silent harm?», *C. Can. Med. Assoc. J.*, p. 413:414, 2011.

[2] E. C. Lin, «Radiation Risk From Medical Imaging», *Mayo Clin. Proc.*, vol. 85, pp. 1142–1146, 2010.

[3] S. X.O., F. Jin, M. S. Linet, W. Zheng, J. Clemens, J. Mills, e Y. T. Gao, «Diagnostic X-ray and ultrasound exposure and risk of childhood cancer.», pp. 531–536, 1994.

[4] S.-T. Feng, M. W.-M. Law, B. Huang, S. Ng, Z.-P. Li, Q.-F. Meng, e P.-L. Khong, «Radiation dose and cancer risk from pediatric CT examinations on 64-slice CT: a phantom study», *Eur. J. Radiol.*, 2010.

[5] C. R. McNeil, J. G. Mcmanus, e S. Mehta, «The accuracy of portable ultrasonography to diagnose fractures in an austere environment», *Prehosp Emerg Care*, 2009.

[6] J. G. Mcmanus, M. J. Morton, C. S. Crystal, T. J. McArthur, J. S. Helphenstine, D. A. Masneri, S. E. Young, e M. A. Miller, «Use of ultrasound to assess acute fracture reduction in emergency care settings», *Am J Disaster Med*, 2008.

[7] K. P. Cross, F. Warkentine, I. K. Kim, e R. I. Paul, «Bedside ultrasound diagnosis of clavicle fractures in the pediatric emergency department», *Acad Emerg Med*, 2010.

[8] R. M. Haralick, K. Shanmugam, e I. Dinstein, «Textural Features for Image Classification», *IEEE Trans. Syst. Man Cybern.*, vol. 3, pp. 610–621, 1973.

[9] T. H. Marshburn, E. Legome, A. Sargsyan, e D. Robinson, «Goal-directed ultrasound in the detection of long-bone fractures», *J. Trauma Inj. Infect. Crit. Care*, pp. 329–332, 2004.

[10] J. Baun, «Sonographic Image Interpretation», em *Cap. 3 de Physical Principles of General and Vascular Sonography*, 2009.

[11] N. Fernandes, «Deteção automática de fraturas ósseas em imagens ultrassónicas», 2013.

[12] L. Nascimento, «Contribuições para a deteção automática de fraturas ósseas em imagens de Ultrassom», 2014.

[13] P. Fish, *Physics and Instrumentation of Diagnostic Medical Ultrasound*. 1990.

[14] K. Bontranger e J. Lampignano, *Tratado de Posicionamento Radiográfico e Anatomia Associada*, 6.ª ed. 2006.

[15] A. Achim, A. Bezerianos, e P. Tsakalides, «Novel Bayesian Multiscale Method for Speckle Removal in Medical Ultrasound Images», *IEEE Trans. Med. Imaging*, vol. "0,

No.8.

[16] K. Karthikeyanm e D. C. Chandrasekar, «Speckle Noise Reduction of Medical Ultrasound Images using Bayeshrink Wavelet Threshold», *Int. J. Comput. Appl.*, vol. 22–No 9.

[17] A. Bullock, R. Dimond, K. Webb, J. Lovatt, W. Hardyman, e M. Stacey, «How a mobile app supports the learning and practice of newly qualified doctors in the UK: aan intervention study», 2015.

[18] R. F. Reardon, J. R. Mateer, e O. J. Ma, «Pocket atlas of emergency ultrasound», 2011. [Em linha]. Disponível em: https://itunes.apple.com/us/app/pocket-atlas-of-emergency-ultrasound/id408558888?mt=8. Verificado a: 28/09/2017

[19] «SonoSupport: a clinical emergency medicine and critical care ultrasound reference tool». Disponível em: https://itunes.apple.com/us/app/sonosupport-clinical-emergency-medicine-critical-care/id638608139?mt=8. Verificado a: 28/09/2017

[20] O. J. Ma, Mateer, «Emergency Ultrasound», 2008. Disponível em: https://itunes.apple.com/us/app/emergency-ultrasound-handbook/id737953236?mt=8&ign-mpt=uo%3D8. Verificado a: 28/09/2017

[21] «MobiSante MobiUS SP1». Disponível em: http://www.mobisante.com/products/product-overview/. Verificado a: 28/09/2017

[22] «MobiSante MobiUS SP1 FAQ's». Disponível em: http://www.mobisante.com/products/faqs/. Verificado a: 28/09/2017

[23] K. Foss, Y. Subhi, R. Aagaard, E. Bessmann, M. Bøtker, O. Graumann, C. Laursen, J. Weile, e T. Todsen, «Developing an emergency ultrasound app - a collaborative project between clinicians from different universities», 2015.

[24] «Spring». Disponível em: https://spring.io. Verificado a: 28/09/2017

[25] H. L. Weissmann, *Vire o jogo com Spring Framework*. .

[26] «Wt framework». Disponível em: https://www.webtoolkit.eu/wt. Verificado a: 28/09/2017

[27] V. Volkman, «Wt: C++ Web Toolkit Library Lets You Write Scripting-Independent Web Apps», 2008.

[28] «Django Project». Disponível em: https://www.djangoproject.com. Verificado a: 28/09/2017

[29] J. B. de Vasconcelos, *Python: Algoritmia e Programação Web*. .

[30] «The Incredible Growth of Python». Disponível em: https://stackoverflow.blog/2017/09/06/incredible-growth-python/#.WbZnZ-

GOfQ0.twitter. Verificado a: 28/09/2017

[31] «MathWorks - Get Started with MATLAB Engine API for Python». Disponível em: https://www.mathworks.com/help/matlab/matlab_external/get-started-with-matlab-engine-for-python.html. Verificado a: 28/07/2017

[32] «Install Matlab Engine API for Python». Disponível em: https://www.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html. Verificado a: 28/09/2017

[33] «Call MATLAB Functions from Python». Disponível em: https://www.mathworks.com/help/matlab/matlab_external/call-matlab-functions-from-python.html. Verficado a: 28/09/2017

[34] «ultrasound-images». Disponível em: http://www.ultrasound-images.com/. Verificado a: 28/09/2017

[35] «ultrasoundcases». Disponível em: http://www.ultrasoundcases.info/. Verificado a: 28/09/2017