Instituto Politécnico de Castelo Branco
Escola Superior de Tecnologia

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA PARA CLASSIFICADORES DE IMAGEM

Ana Sofia Barros Teixeira

Dissertação apresentada ao Instituto Politécnico de Castelo Branco para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Desenvolvimento de *Software* e Sistemas Interativos, realizada sob a orientação científica do Doutor Eurico Lopes, Professor Coordenador da Unidade Técnico-Científica de Informática da Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco.

Agosto 2011

Dedicatória

Aos meus pais, irmãos, amigos e ao Paulo.

Agradecimentos

Agradeço ao meu orientador que me guiou na realização e conclusão deste projeto, com opiniões sempre esclarecedoras e incentivadoras que possibilitaram a conclusão do mesmo.

Agradeço a todos os meus colegas que, de alguma forma, contribuíram para que este projeto fosse realizado e concluído, em especial a Ana por ter estado sempre disponível a ajudar.

Agradeço aos meus pais, irmãos e familiares por todo o carinho, ânimo e por estarem sempre presentes.

Um agradecimento muito especial ao Paulo por todos os incentivos, pelo seu apoio e paciência.

Palavras chave

Imagem, processamento de imagem, interface gráfica, características, descritores, classificação, *data mining*.

Resumo

Olhar para uma determinada imagem ou para um conjunto de imagens é natural a qualquer ser humano, sem que disso se aperceba. Dotar um sistema computacional da capacidade de efetuar o reconhecimento de determinada imagem é um desafio da visão computacional.

Para que o reconhecimento de uma dada imagem seja possível é necessário que primeiro se obtenha essa imagem, depois se for necessário, se melhore a sua qualidade através de algoritmos de pré-processamento e posteriormente se extraiam as características da imagem através da aplicação de um descritor. Desta extração surge um conjunto de características que serão classificadas de forma a obter um resultado final.

O presente estudo visa o desenvolvimento de uma interface gráfica que possibilite de forma modular e sequencial, o processamento de uma determinada imagem, ou conjunto de imagens. De forma a construir uma base de informação, começou por se efetuar uma revisão da literatura do reconhecimento de padrões, nomeadamente dos descritores e classificadores de uma imagem.

O conhecimento, de que uma imagem digital para ser processada teria que passar por diferentes etapas, levou à implementação das principais etapas desse processamento na interface gráfica. A interface desenvolvida apresenta um módulo de pré-processamento da imagem, um módulo de descritores de imagem e um módulo de classificação dessa mesma imagem, de forma a se obter um resultado.

Em função dos dados obtidos considera-se que a existência de uma interface que integre as principais etapas do processamento de imagem é viável e que pode ser um instrumento de trabalho futuro.

Keywords

Image, pre-processing, graphic interface, characteristics, descriptors, classification, *data mining*.

Abstract

For any human being, to look at an image or a group of images is a natural action - people do it without even noticing it. In what concerns computational vision, it is a challenge to equip a computational system with the ability to recognise a certain image.

In order to allow the recognition of an image, we must obtain the image, then enhance its quality through pre-processing algorithms (if needed) and finally extract the characteristics of that image by applying a descriptor. From that description we can obtain a series of characteristics which will be classified in order to reach a final result.

The objective of this essay is to develop a graphic interface which will allow people to process an image or a group of images in a modular and sequential way. In order to build an information basis, I made a thorough review of literature on pattern recognition, mainly on image descriptors and classifiers.

The implementation of the different stages of that processing on the graphic interface was based on the knowledge that, in order to process a digital image, we have to go through different stages. The interface developed has got an image pre-processing module, a module of image descriptors and an image classification module.

The existence of an interface that can incorporate the main stages of image processing is viable, according to the obtained data, and can be a work instrument in the future.

Índice geral

Índice de figuras

Índice de tabelas

Lista de abreviaturas / siglas

ROI	*Region of Interest*
SVM	*Support Vector Machine*
RNA	Rede Neural Artificial
AG	Algoritmos Genéticos
HTD	*Homogeneous Texture Descriptor*
TBD	*Texture Browsing Descriptor*
EHD	*Edge Histogram Descriptor*
MPEG	*Moving Picture Experts Group*
SIFT	*Scale Invariant Feature Transform*
SURF	*Speeded Up Robust Features*
K-NN	*k-Nearest Neighbours*
TP	*True Positive*
FP	Falsos Positivos
FN	Falsos Negativos

Capítulo 1- Introdução

"A habilidade do ser humano em reconhecer e classificar objetos sempre impressionou e continua a impressionar os cientistas... Desde os primórdios da computação, a tarefa de implementar algoritmos emulando essa capacidade humana, tem-se apresentado como a mais intrigante e desafiadora!" (Marques de Sá, 2004).

O aparecimento da computação possibilitou uma evolução a nível tecnológico constante. Um dos grandes avanços é a tentativa de adaptar algumas atitudes e capacidades humanas aos sistemas computacionais.

Qualquer um de nós, ser humano, tem perceção do que nos rodeia. Identificamos constantemente animais, objetos, sons e cheiros através de sinais captados pelos órgãos sensoriais. Para o ser humano estas operações ocorrem de forma natural, sem grande dificuldade e até mesmo sem se aperceber. A interpretação do mundo que nos rodeia, a atribuição de significado às frases que ouvimos, são no entanto operações ainda difíceis de realizar num computador, a não ser que existam hipóteses que restrinjam os objetivos a atingir.

Mais simples do que olhar para um objeto e identificá-lo é escolher, de entre um conjunto de hipóteses, qual se adapta melhor às observações realizadas. Este problema de decisão tem grande utilidade em diversos contextos, como por exemplo em processamento de imagem, reconhecimento da fala ou em diagnósticos médicos. A resolução deste problema passa por métodos específicos e é o objetivo do Reconhecimento de Padrões (Marques, 2005).

Para o reconhecimento de padrões ou objetos, as etapas básicas no processamento de imagens são a aquisição da imagem, realizar um pré-processamento sobre a imagem, segmentar a imagem de forma a excluir partes não necessárias (regiões desfocadas, excesso de luz, zonas escuras, etc.) diminuindo a área da imagem, restando apenas as áreas de interesse e diminuindo o esforço computacional (Gonzalez e Woods, 2008). A extração automática de características visuais mais relevantes da imagem incidirá sobre as áreas de interesse, utilizando descritores de imagem, e a classificação dessas características será realizada através da utilização de classificadores (*neural networks, support vector machines*, etc.) (Dhandra e Hedadi, 2005).

Para se efetuar o processamento de uma imagem, o investigador necessita de obter essa imagem digital. Posteriormente, se pretender, pode utilizar técnicas de pré-processamento para melhorar a qualidade dessa imagem. Em seguida, devem ser extraídas as características da imagem (através de um descritor) para numa fase seguinte se classificar essa mesma imagem.

É neste contexto que surge esta dissertação, procurar compreender as fases do processamento de imagem e colocar numa mesma interface as principais fases, de forma modular e interativa.

1.1. Motivação e Objetivos

O reconhecimento de imagens é por parte do ser humano uma tarefa diária que permite a envolvência com o meio circundante. Cada ser humano reconhece uma imagem através da análise das suas características, no entanto, este reconhecimento é natural e distinto de cada ser humano. Tentar efetuar o reconhecimento de uma imagem através da utilização de sistemas computacionais é um avanço na computação, e atualmente é utilizado nas mais diferentes áreas, como na saúde (tomografia, radiografia, endoscopia, etc.) (New Technologies In Medicine, 2010; Areia, 2008), na biometria (impressão digital, facial, íris, etc.) (Netponto, 2010), entre outros.

O conhecimento prévio das ferramentas de *data mining*, como o Weka e o SAS, possibilitou a descoberta de que através do Weka se poderia efetuar algumas das etapas do processamento de imagem, nomeadamente a classificação, pois este *software* tem já implementado alguns classificadores.

Depois de um estudo sobre as fases do processamento de uma imagem, analisou-se que os investigadores utilizavam ferramentas diferentes para efetuar cada uma das fases desse processamento. Após este estudo, surgiu uma questão que se apresentou como fio condutor de toda esta investigação e que levou à escolha deste tema para investigação. Essa questão de investigação era a seguinte:

Não será possível o desenvolvimento de uma interface gráfica que possibilite, de forma modular, a implementação das principais etapas do processamento de imagem?

Esta questão tornou-se a base desta dissertação e define de forma clara os principais objetivos a desenvolver.

Desta forma o principal objetivo deste estudo era encontrar uma forma de integrar todas as fases pelas quais uma imagem digital passaria até ser classificada, desenvolvendo para isso uma interface gráfica.

No entanto, para que este objetivo fosse concretizado, surgiu a necessidade de definir objetivos específicos que complementassem o principal, sendo eles:

- Efetuar uma revisão bibliográfica do reconhecimento de padrões, especificamente os descritores e classificadores de imagem;
- Definir uma estrutura da arquitetura da aplicação, de forma a apresentar a implementação das principais etapas do processamento de imagem, numa única interface;
- Desenvolver uma interface que permita o pré-processamento de uma dada imagem, ou conjunto de imagens, extrair as características dessa imagem, através de um descritor, proceder à classificação dessas características e obter um resultado;
- Implementação de um descritor para extração das características das imagens, tendo por base a revisão bibliográfica efetuada;

- Implementação de classificadores, através dos quais é possível obter um resultado da classificação e tendo por base a revisão bibliográfica efetuada;
- Conceber uma interface eficaz, interativa que possibilitasse a classificação de qualquer tipo de imagem;
- Interpretar os resultados obtidos;
- Testar a interface desenvolvida através da aplicação de testes de usabilidade, de forma a detetar eventuais problemas na interação entre utilizador e interface e verificar a usabilidade da plataforma.

1.2. Contribuições

Este estudo possibilitou a criação de uma única interface que integra as principais etapas do processamento de imagem. Esta interface é modular, pois possibilita a integração e adição de novas funcionalidades nos módulos de pré-processamento, de extração de características ou mesmo de classificação. Estas novas funcionalidades podem ser desenvolvidas utilizando a *framework* .NET, Java e/ou Matlab.

A interface base foi desenvolvida em C# numa estrutura de *n*-camadas e permite o pré-processamento de uma determinada imagem, a extração das suas características e a sua posterior classificação.

1.3. Organização da dissertação

Esta dissertação está organizada em sete capítulos, ao longo dos quais se apresenta todo o processo de desenvolvimento do presente projeto.

No presente capítulo, introdução, é efetuado um enquadramento ao tema escolhido, referindo a motivação e os objetivos do projeto.

Para melhor compreender conceitos relacionados com o processamento de imagem, é efetuada no segundo capítulo uma revisão bibliográfica. Neste capítulo começa-se por abordar a metodologia do reconhecimento de padrões, nomeadamente a sua divisão em duas partes, a extração de características, através de um descritor, e a classificação. Inicialmente é realizada uma revisão sobre o conceito de descritores de imagens e são apresentados alguns exemplos desses descritores. Seguidamente, é efetuado um estudo sobre o conceito de classificação das características de uma imagem e os seus modos de treino: supervisionado e não supervisionado. A nível de revisão da literatura são ainda apresentados exemplos de classificadores.

No terceiro capítulo é apresentado o esquema geral da interface e a descrição de cada um dos módulos que a compõem. Neste capítulo é também efetuada uma breve apresentação do conceito de *data mining* e são referidas as ferramentas *data mining* mais conhecidas. Esta apresentação tem como objetivo enquadrar a utilização na ferramenta *data mining* Weka neste estudo.

No quarto capítulo é apresentado o esquema de implementação da interface, bem como as ferramentas e linguagens de programação utilizadas para o desenvolvimento da interface e dos

módulos que a constituem. De forma a apresentar a interface gráfica, desenvolvida neste estudo, é efetuada uma descrição da mesma e das funcionalidades implementadas nos respetivos módulos.

No quinto capítulo é efetuado um teste à interface gráfica, sendo apresentado o processamento de cada módulo desenvolvido. É ainda realizada uma interpretação dos resultados obtidos no final do processamento das imagens, nomeadamente da matriz confusão obtida.

No sexto capítulo é efetuada uma avaliação à interface desenvolvida. Desta forma, neste capítulo são apresentados os dados recolhidos da aplicação de testes de usabilidade a um determinado número de participantes. Os dados recolhidos foram analisados e são apresentadas as conclusões dessa análise.

No sétimo capítulo são realizadas as conclusões desta dissertação, as limitações do estudo, sendo ainda apresentadas sugestões de trabalho futuro a desenvolver.

Por fim, é apresentada a bibliografia consultada e os anexos. Dos anexos fazem parte o teste realizado aos participantes com o objetivo de avaliar o seu *feedback* com a interface, o protótipo desenvolvido para esse fim, uma listagem dos filtros existentes na biblioteca AForge.NET e é apresentado o formato de um ficheiro *arff*.

Capítulo 2- Revisão bibliográfica

2.1. Introdução

A aprendizagem computacional (traduzida do termo "*Machine Learning*", conhecida também por Aprendizagem Automática) é uma disciplina da área de Inteligência Artificial que visa o estudo e desenvolvimento de programas computacionais que permitem aos computadores "aprender" com a experiência (Mitchell, 1997). O reconhecimento de padrões é um dos tópicos da aprendizagem computacional que se dedica, fundamentalmente, a desenvolver técnicas automatizadas capazes de aprender a descrever e classificar características significativas perante um conjunto de dados. Marques de Sá (2004) refere que o reconhecimento de padrões é a ciência que trata da classificação e da descrição de objetos.

Um sistema de reconhecimento de padrões pode ser dividido em duas partes: a extração de características e o classificador. Na extração de características é selecionada a informação relevante para a decisão, onde os dados são transformados pelos sensores num conjunto menor de valores, surgindo assim as características (Marques, 2005). Por sua vez estas são utilizadas pelo classificador para escolher a hipótese que descreve o mundo real (Figura 2.1).

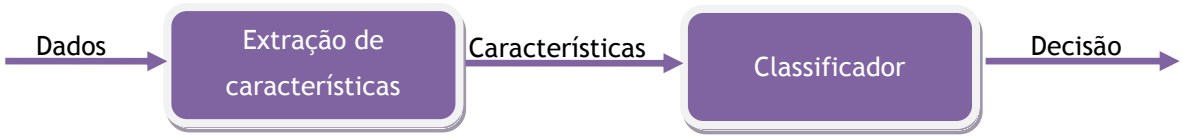


Figura 2.1 - Metodologia do reconhecimento de padrões (Marques, 2005)

O resultado da extração de características representa-se normalmente por um vetor de características. O treino do classificador pode ser realizado em dois modos: o modo supervisionado e não supervisionado.

Ao longo deste capítulo é feita uma pesquisa bibliográfica, onde inicialmente descreve-se o papel dos descritores de imagem, efetuando-se uma análise a alguns desses descritores. Posteriormente, aborda-se o papel da classificação no processamento de imagem e apresentam-se alguns dos classificadores existentes.

2.2. Descritores de imagem

Através dos descritores de imagens é possível obter as descrições das características visuais do conteúdo em imagens ou vídeos, descrevendo as características elementares, tais como: forma geométrica, cor, textura ou movimento, entre outros.

Os descritores possibilitam também a medição de semelhanças/recursos entre duas imagens diferentes. Esses recursos podem ser globais ou locais. A análise global considera a imagem como um todo (Mikolajczyk e Schmid, 2005), enquanto a análise local primeiro segmenta

a imagem em várias regiões de interesse (ROI - *Region Of Interest*), determinando, em seguida, as propriedades e características dessa ROI (Itti, et al., 1998). Essas características podem caracterizar um documento multimédia a nível mundial, por exemplo, uma imagem no interior ou uma ocorrência ao ar livre. Uma ROI é a região definida automaticamente a partir de parâmetros obtidos na própria imagem onde o processamento se encontra centrado.

Existem alguns métodos que são capazes de, além de detetar características (detetor), também descrevem informações relevantes sobre essas características (descritores), de tal forma que seja possível identificá-las, apenas com a informação desses descritores.

O SIFT (Lowe, 1999, 2004), o GLOH (Mikolajczyk e Schmid, 2005), o SURF (Bay, et al., 2008) e o LESH (Sarfraz e Hellwich, 2008) são alguns dos tipos de descritores mais relevantes atualmente. Estes descritores costumam ser invariantes a diversas transformações, como rotação, translação, escala, além de possuírem robustez a ruído e iluminação.

2.2.1. Descritores MPEG-7

A norma MPEG-7, definida pela *Moving Picture Experts Group* (MPEG) (ISO/IEC TR 15938, 2005), define vários descritores de imagens agrupados em cinco categorias: cor, textura, forma, movimento e localização. Todos estes são úteis para pesquisas e consultas de semelhanças.

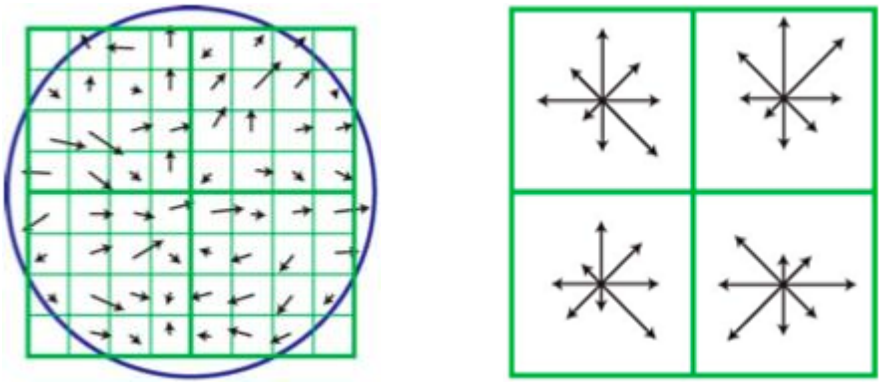
Os descritores MPEG-7 podem caracterizar quatro aspetos essenciais: a cor dominante, o histograma de cor, a seleção do espaço de cor e estrutura da cor (ISO/IEC 15938-3/FCD, 2001). As texturas são caracterizadas por três descritores; *homogeneous texture descriptors* (HTD), *texture browsing descriptors* (TBD), e *edge histogram descriptors* (EHD). A norma MPEG-7 também especifica descritores de forma 2D e 3D (Glowacz et al., 2006). Uma visão geral destas características podem ser encontradas em (Eidenberger, 2003; Manjunath et al., 2001; Ohm, 2001;Yang e Kuo, 1999).

2.2.2. SIFT (Scale invariant feature transform)

O SIFT (Lowe, 1999, 2004) é utilizado para detetar e descrever as características locais das imagens. Foi publicado por David Lowe, em 1999 (Lowe, 1999) e, desde então o SIFT é amplamente utilizado devido aos bons resultados em várias aplicações, incluindo o reconhecimento de objetos, a realidade aumentada ou em sistemas *Content Based Image Retrieval (*CBIR) (Lowe, 2004). Este descritor é invariante à escala, rotação, iluminação, ruído e a pequenas alterações.

Em (Bosch, 2006) são comparados vários descritores na aplicação de classificação de imagens, tendo o descritor SIFT obtido o melhor desempenho.

O método SIFT (Lowe, 2004) é caracterizado por duas fases: (1) deteção de pontos de interesse (*keypoints*) e (2) extração do descritor visual. O objetivo é encontrar *keypoints* que representam locais relevantes da imagem e que se mantenham estáveis em relação às mudanças de escala e rotação. São utilizados filtros de diferenças Gaussianas para detetar estes *keypoints*.



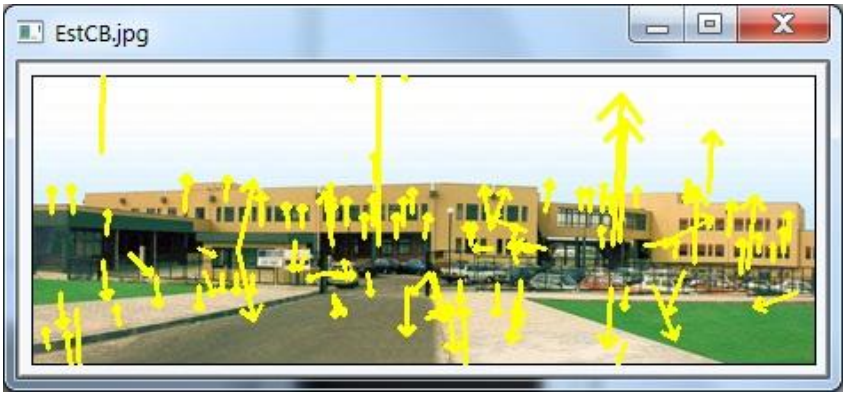
(a) – Gradiente **(b)** – Descritor

Figura 2.2 - Exemplo do descritor SIFT numa região 8x8 (Lowe, 2004)

No primeiro passo, é obtido o gradiente (Figura 2.2 (a)) de cada ponto numa região de 16x16. De seguida para cada bloco de 4x4, é calculado o histograma com 8 direções do gradiente. Cada uma destas regiões é representada por 128 valores (valor obtido através de 16 regiões 4x4 vezes 8 pontos do histograma). O descritor SIFT (Figura 2.2 (b)), devido às propriedades de cada *keypoint*, é caracterizado por ser invariante à iluminação, rotação e escala. Estes descritores apresentam um grau de distinção que permite que funcionem como identificador da imagem em questão (Ballesta, et al., 2007; Bosch, 2006; Lowe, 2004).

A figura seguinte apresenta o resultado de uma aplicação deste método, através de uma simulação efetuada com uma fotografia da Escola Superior de Tecnologia de Castelo Branco.

(a) Imagem original (Escola Superior de Tecnologia de Castelo Branco)



(b) *Output* da Imagem pelo método SIFT

Figura 2.3 - Características detetadas através do detetor SIFT

Na Figura 2.3 (a) é possível visualizar uma imagem da Escola Superior de Tecnologia de Castelo Branco à qual foi aplicado o descritor SIFT, produzindo o resultado apresentado na Figura 2.3 (b). As setas amarelas (vetores) mostram a magnitude e orientação das áreas de interesse detetadas.

2.2.3. GLOH (Gradient Location and Orientation Histogram)

O GLOH (Mikolajczyk e Schmid, 2005) é uma versão mais robusta e especializada do SIFT. Este considera mais regiões espaciais para o histograma. Enquanto o SIFT gera 8 direções de gradiente, o GLOH gera 16, definindo assim um histograma de 272 regiões. O tamanho do descritor é reduzido com PCA (*Principal Component Analysis*) (Jolliffe, 1986), uma técnica usada para reduzir dados multidimensionais a fim de analisá-los. De modo geral, o GLOH apresenta uma maior eficiência na deteção de características em relação ao SIFT, porém os seus resultados não são muito superiores pois requerem uma carga computacional maior (Mikolajczyk e Schmid, 2005). Em cenas com textura ou quando as retas da imagem não são bem definidas, o SIFT tem um melhor desempenho (Mikolajczyk e Schmid, 2005).

Através da Figura 2.4 é possível visualizar uma imagem onde foi aplicado o descritor GLOH.

Figura 2.4 – Aplicação do descritor GLOH em regiões Hessian-Affine (Mikolajczyk e Schmid, 2005)

2.2.4. LESH (Local Energy based Shape Histogram)

LESH (Sarfraz e Hellwich, 2008) é um dos descritores de imagem mais recente em visão computacional.

O descritor LESH é construído sobre um modelo energético local da imagem, através do comportamento da imagem no domínio da frequência *(phase congruency*). Desta forma, o LESH determina as características, acumulando a energia local ao longo de várias orientações, criando histogramas locais das diferentes partes da imagem, e através da concatenação destes gera-se um descritor de dimensão 128, tendo alta tolerância a ruído e iluminação, além de ser invariante à escala (Sarfraz e Hellwich, 2008).

A Figura 2.5 apresenta o modelo energético de uma face frontal e dessa mesma face lateral e mapas de orientação.

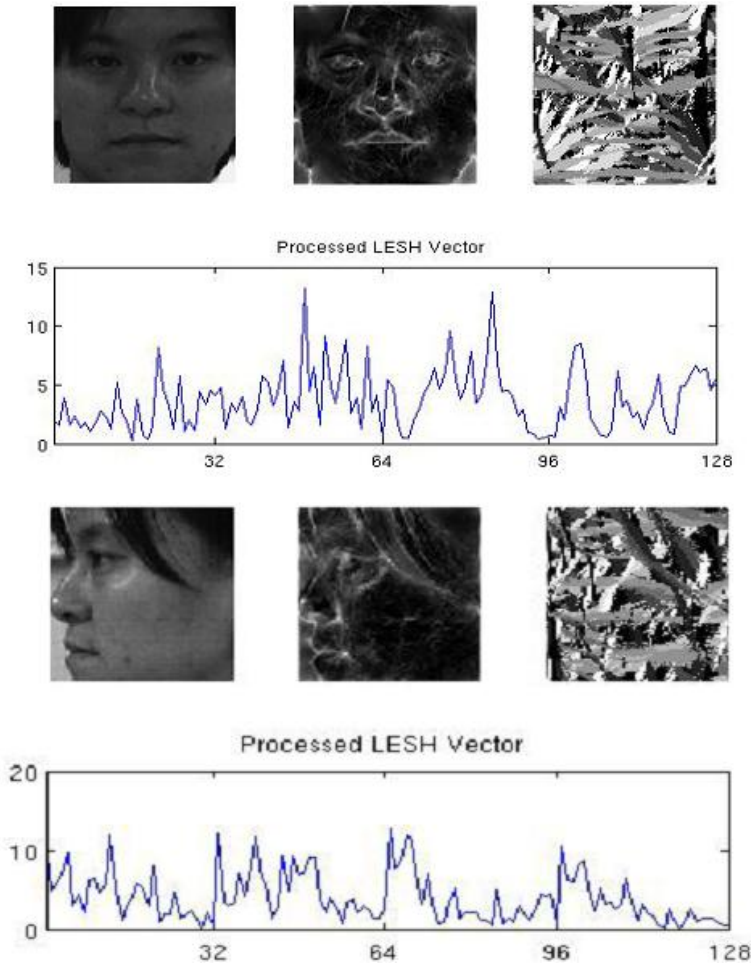


Figura 2.5 - Modelo energético de uma imagem frontal e uma lateral de uma face e vetor de características extraídas pelo LESH (Sarfraz e Hellwich, 2008)

2.2.5. SURF (Speeded Up Robust Features)

O SURF é um descritor invariante à rotação e aos efeitos de escala apresentado recentemente por Bay (Bay, et al., 2008). Este descritor é uma técnica semelhante ao SIFT para a criação de descritores de imagens. Nesta técnica, nem o detetor de pontos, nem o descritor utilizam dados relacionados com a cor. Na deteção de pontos é utilizado o método de imagens integrais (*Integral Images*), que efetua o somatório de valores dentro de uma área retangular (Figura 2.4), reduzindo o tempo de computação (Bay, et al., 2008).

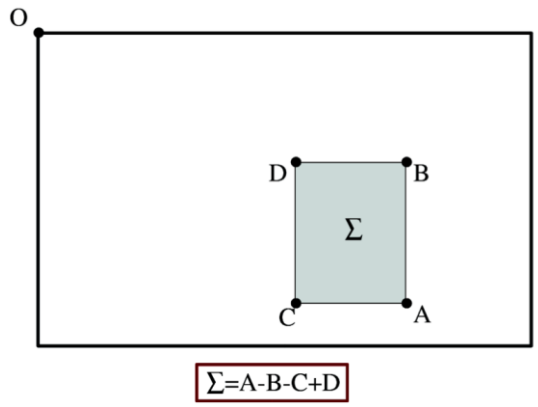


Figura 2.6 - Exemplo de uma imagem integral (Bay, et al., 2008)

O maior avanço do detetor SURF, relativamente aos anteriores, deve-se à velocidade de deteção, permitindo a sua utilização em diversas aplicações, em tempo real (Bay, et al., 2008).

A Figura 2.7 apresenta o resultado da aplicação de um descritor SURF a uma determinada imagem.

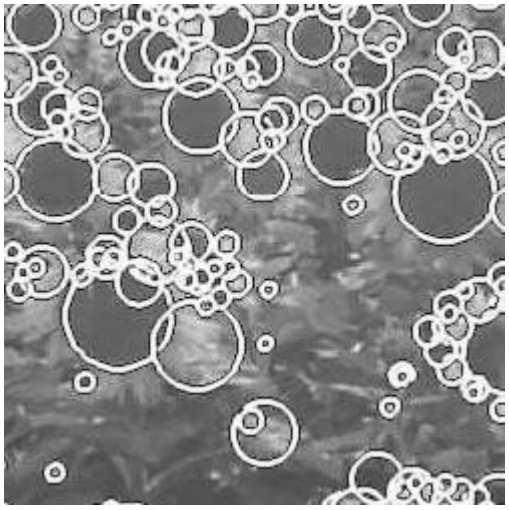


Figura 2.7 - Deteção de pontos de interesse pelo método de SURF (Bay, et al., 2008)

2.3. Classificação das características de uma imagem

Depois de extraídas as características de uma determinada imagem, é necessário proceder à sua classificação. A classificação consiste na associação de um determinado conjunto de características à classe (conjunto de objetos com características comuns) em que se enquadra.

Em processos de visão, sejam eles biológicos ou artificiais, é possível identificar duas partes fundamentais: a segmentação e a classificação (Deco, 2001; Rolls, 2002). Normalmente entende-se a segmentação como o assinalar de uma região de interesse e a classificação como a interpretação da região de interesse. No entanto, olhando para o funcionamento de um sistema de visão e pretendendo atingir soluções sobre imagens naturais, é sempre difícil separar uma da outra visto que para se segmentar uma região é necessário reconhecer pontos de interesse dessa região, logo, é necessário classificar. Por outro lado, para se classificar uma região é necessário deter informação sobre, pelo menos, algumas partes segmentadas dessa região, logo, é necessário segmentar (Deco, 2001; Rolls, 2002).

O método de classificação de imagens pode ser feito de dois modos: modo supervisionado e modo não supervisionado (Marques, 2005).

Os classificadores supervisionados requerem uma aprendizagem/treino dos parâmetros do classificador. Esses métodos são também conhecidos como métodos paramétricos. O classificador é treinado a replicar a decisão correta para todos os padrões de treino. Este treino é comparado à aprendizagem das crianças durante os primeiros anos de vida, onde se incentivam os comportamentos corretos e se corrigem os comportamentos errados (Marques, 2005). Um exemplo deste tipo de classificadores é o SVM (*Support Vector Machine)* (Vapnik, 1995).

Os classificadores não supervisionados ou não paramétricos trabalham diretamente sobre os dados, não requerendo aprendizagem/treino de parâmetros (Marques, 2005). O classificador baseado no vizinho mais próximo (*Nearest-Neighbor - NN)* (Cover e Hart, 1967) é um exemplo deste tipo de classificadores.

Na literatura podemos encontrar diferentes classificadores, como é o caso das redes neurais, sistemas *Fuzzy*, algoritmos genéticos, classificadores de *Bayes*, etc. (Marques, 2005).

2.3.1. Classificador dos K-vizinhos mais próximos (k-NN)

O classificador dos k-vizinhos mais próximos (k-NN) surge da ideia de que objetos que se encontram mais "próximos" no conjunto de atributos (*feature space*) têm mais possibilidades de pertencerem a uma mesma classe. Desta forma, a tarefa de classificação consiste em, para cada novo objeto, determinar a classe dos objetos mais "próximos" e atribuir-lhe a classe predominante. Esta ideia ainda não tem subjacente o valor de *k* (número de vizinhos), no entanto, este tem de ser definido pelo utilizador (Cover e Hart, 1967). É importante destacar que este é um modelo não paramétrico visto que apenas o valor de *k* tem de ser definido antes de começar o processo de aprendizagem. Ou seja, não existe qualquer parâmetro a ser estimado.

A Figura 2.8 apresenta os vizinhos 1, 2 e 3 mais próximos do objeto no centro da circunferência. Este objeto vai ser classificado com base na classe mais votada entre os seus vizinhos.

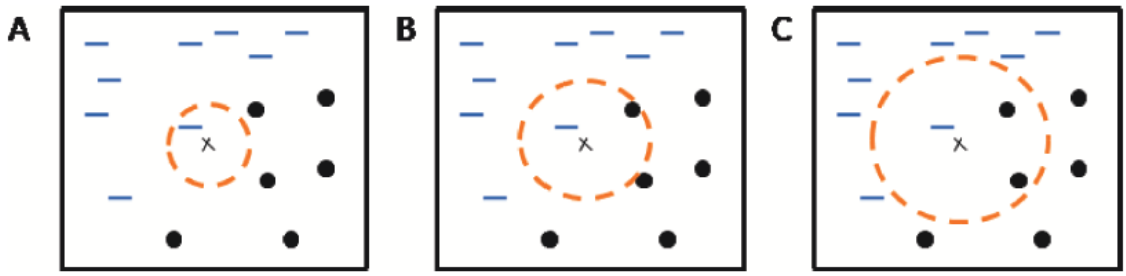


Figura 2.8 - Identificação dos k-vizinhos mais próximos de uma instância: A: k = 1; B: k = 2; C: k = 3 (Pang-Ning, et al., 2006)

No primeiro caso o exemplo será classificado como sendo da classe "−" uma vez que foi definido k = 1 vizinho. Pelo contrário, na última situação, o número de vizinhos mais próximos é k = 3 e dois desses vizinhos pertencem à classe " ● ". Usando o critério de escolher a classe que se encontra em maior número, o objeto será classificado como pertencente à classe " ● ". No caso, de haver empate, como acontece na situação do em que K = 2, escolhe-se aleatoriamente a classe a atribuir ao objeto. O procedimento clássico para evitar a conjuntura anterior é considerar um número ímpar de vizinhos (Sierra, 2000).

De um modo geral, a escolha do melhor valor para k é um problema a averiguar em cada problema de classificação, visto que se k for demasiado pequeno, o classificador dos vizinhos mais próximos pode estar sujeito a um efeito de sobreajustamento motivado pelo ruído no conjunto de treino. Por outro lado, se k é demasiado grande, pode existir o risco de má classificação de uma instância de teste. Isto acontece porque, se o conjunto de vizinhos mais próximos é elevado, há a tendência em abranger exemplos que estejam afastados do objeto a classificar (Sierra, 2000). Alguns estudos empíricos mostram que os melhores resultados obtidos são para k= 3 ou k = 5 (Sierra, 2000).

2.3.2. Neural networks

Embora ainda se ignore muito sobre a forma como o cérebro humano aprende a processar a informação, têm-se desenvolvido modelos que tentam imitar tais habilidades, denominados redes neuronais artificiais. A elaboração destas redes supõe, por um lado, a dedução das características essenciais dos neurónios e suas conexões e, por outro, a implementação do modelo num computador de forma que se possa simular. Os princípios que ainda hoje vigoram sobre as redes neuronais artificiais foram apresentados pela primeira vez por Warren McCullch e Walter Pitts em 1943, e demonstraram que as redes neuronais artificiais podem calcular qualquer função aritmética ou lógica (Hagan et al., 1996).

Normalmente, uma rede neural artificial (RNA) é um processo de treino autoadaptativo capaz de aprender a resolver problemas complexos com base no conhecimento disponível. Um

sistema baseado em RNA simula o funcionamento do cérebro biológico: é composto por elementos de processamento interconectados que simulam os neurónios (Bodri, 2001).

Sendo a aprendizagem uma capacidade que o ser humano desenvolve e coloca em prática diariamente, à medida que os mais variados desafios surgem, não é de estranhar que tais capacidades sejam desejadas nas RNA. Nas RNA o conceito de aprendizagem é baseado na teoria comportamentalista, ou seja, o processo de aprendizagem obedece a um estímulo produzindo uma resposta (Bodri, 2001).

Na área de reconhecimento de padrões, foram desenvolvidas RNA que permitem o reconhecimento de caracteres, a deteção de caracteres errados no sítio errado, a deteção de padrões ou texturas numa linha de montagem não detetáveis ao olho humano, o reconhecimento de voz, deteção de sismos, análise de ecos provenientes de um sonar ou de um radar, entre outros (Adler e Blue, 2002). O reconhecimento visual de imagens é particularmente utilizado na manipulação automática de objetos e correspondência de imagens processadas - imagens ampliadas, comprimidas, transformadas. O reconhecimento visual é utilizado em áreas tão distintas como na identificação e segurança, no reconhecimento de mãos e impressões digitais (Bodri, 2001).

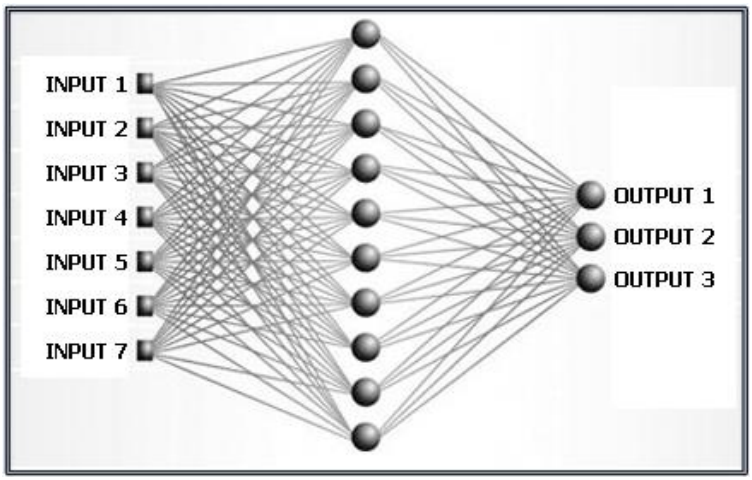


Figura 2.9 - Exemplo de uma rede neuronal artificial (Karthik, 2007)

2.3.3. Fuzzy systems

A lógica *Fuzzy* foi introduzida pelo Lotfi Zadeh em meados dos anos 60 como meio de modelar o conhecimento subjetivo, o qual representa informações linguísticas (Zadeh, 1965). Segundo Zadeh é muito frequente encontrar classes de objetos no mundo real sem contornos bem definidos. Por exemplo a classe dos animais inclui sem dúvidas gatos, cavalos e cães como seus membros, e exclui claramente líquidos e plantas. No entanto, em seres vivos como a estrela-do-mar e as bactérias encontram-se num estado ambíguo respeitante à classe dos animais (Zadeh, 1965). Este tipo de classes não se consegue caracterizar através do senso matemático comum. Podem ser caracterizados através da lógica *Fuzzy* uma vez que esta permite gerir o conceito de parcialmente verdade – lógica multi-valor (Pedrycz e Gomide, 2007).

Um dos exemplos, da maneira de pensar da lógica difusa (Kosko, 1992), remete para o paradoxo clássico do mentiroso de Creta que diz que todos os habitantes de Creta são mentirosos. Se o que ele diz for verdade, então ele é um mentiroso (mas diz a verdade!). Se ele estiver a mentir, então a frase é verdadeira e ele não é mentiroso (mas acabou de dizer uma mentira!). A lógica *Fuzzy*, também conhecida como lógica difusa é uma extensão da lógica booleana que admite valores lógicos intermediários entre o FALSO (0) e o VERDADEIRO (1); por exemplo o valor médio 'TALVEZ' (0,5). Isto significa que um valor lógico difuso é um valor qualquer no intervalo de valores entre 0 e 1 (Pedrycz e Gomide, 2007).

A lógica *Fuzzy* tem encontrado grandes aplicações em áreas como: raciocínio aproximado, robótica, processamento digital de imagem, reconhecimento de padrões, sistemas de controlo, tomada de decisão, medicina (Klir e Folger, 1987; Cox et al., 1998).

2.3.4. Genetic Algorithms

A técnica de otimização por Algoritmos Genéticos (AG), criada por Holland (1975) e desenvolvida e consolidada por Goldberg (1989) tem por base a Teoria da Evolução de Darwin (Coley, 1999; Pham e Karaboga, 2000). O processo de otimização tem início com a geração aleatória de uma população inicial de *n* indivíduos que representam de forma codificada pontos do domínio da função objetivo.

Os algoritmos genéticos, introduzidos por John Holland em 1975 (Coley, 1999; Pham e Karaboga, 2000), são algoritmos de otimização numérica inspirados pelo processo de evolução da natureza e as técnicas de pesquisa direcionadas ao acaso. A aplicação desta técnica de otimização tem sido potenciada desde a sua criação em diferentes áreas, tais como: ciências da computação, bioinformática, nas diversas áreas da engenharia, revelando-se um método heurístico flexível e com sucesso (Coley, 1999; Rothlauf, 2006). Uma das vantagens de um algoritmo genético é a simplificação que eles permitem na formulação e solução de problemas de otimização. Os AG simples normalmente trabalham com descrições de entrada formadas por cadeias de *bits* de tamanho variável, como por exemplo AG usados para Programação Genética. Os AG possuem um paralelismo implícito decorrente da avaliação independente de cada uma dessas cadeias de bits, ou seja, pode-se avaliar a viabilidade de um conjunto de parâmetros para a solução do problema de otimização em questão.

Os AG são algoritmos de pesquisa baseados em mecanismos de seleção natural e na genética natural. Com estes mecanismos os indivíduos com melhores capacidades são normalmente os vencedores num ambiente altamente competitivo. Assim, os AG combinam a sobrevivência dos indivíduos mais aptos com estruturas de *strings*. A informação criada aleatoriamente nestas estruturas é trocada para formar um algoritmo de pesquisa. Em cada geração é criada uma nova população artificial de *strings* desenvolvida a partir de *bits* e/ou conjunto de *bits* das *strings* mais aptas, substituindo a população anterior (Coley, 1999; Rothlauf, 2006).

Os AG tendem vindo a ser aplicados em diferentes áreas como a programação automática, a economia, ecologia, sistemas imunitários, processamento de imagens, entre outros (Rocha e Neves, 1998). No processamento de imagens os AG têm vindo a ser aplicados no alinhamento e análise de imagens (Rocha e Neves, 1998).

2.3.5. Classificador de *Bayes*

Um classificador de *Bayes* é um classificador probabilístico simples baseado na aplicação de teorema de *Bayes* (Gonzalez, 1993). O modelo construído por este classificador é um conjunto de probabilidades. As probabilidades são estimadas pela contagem da frequência de cada valor de característica para as instâncias dos dados de treino (Gonzalez, 1993). Dada uma nova instância, o classificador estima a probabilidade de essa instância pertencer a uma classe específica, baseada no produto das probabilidades condicionais individuais para os valores característicos da instância. O cálculo exato utiliza o teorema de *Bayes* e é por essa razão que o algoritmo é denominado um classificador de *Bayes*. O algoritmo é também denominado de *Naive*, uma vez que todos os atributos são independentes, dado o valor da variável da classe. Apesar deste pressuposto, o algoritmo apresenta um bom desempenho em muitos dos cenários de predição de classes (Gonzalez, 1993).

Em termos simples, um classificador *Naive Bayes* assume que a presença (ou ausência) de uma característica particular de uma classe está relacionada com a presença (ou ausência) de qualquer outra característica. Por exemplo, uma fruta pode ser considerada como uma maçã, se for vermelha, redonda, e cerca de 4 cm de diâmetro. Mesmo que esses recursos dependam uns dos outros ou sobre a existência de outras características, um classificador *Naive Bayes* considera todas essas propriedades para contribuir de forma independente para a probabilidade de que esse fruto possa ser uma maçã (Gonzalez, 1993).

O classificador *Naive Bayes* procura deduzir as probabilidades de uma imagem pertencer a cada uma das classes, ou antes, de encontrar a classe que apresenta maior probabilidade de representar a imagem (George e Pat, 1995). O modelo matemático utilizado durante o treino do classificador *Naive Bayes* consiste numa rede *Bayesiana* na qual todos os atributos são apenas dependentes da classe da imagem, e não possuem relação de dependência entre si.

Dependendo da natureza exata do modelo de probabilidade, os classificadores *Naive Bayes* podem ser treinados de forma muito eficiente num ambiente de aprendizagem supervisionada (Domingos e Pazzan 1997; Gonzalez, 1993).

2.3.6. SVM (Support Vector Machines)

Um caso de um modelo de aprendizagem automática que se inclui na classe dos classificadores supervisionados e que tem recebido muita atenção no meio cientifico é o *Support Vector Machine* (SVM). Este foi originalmente introduzido por Vapnik e os seus colaboradores na década de 90 (Vapnik, 1995). A ideia a ele subjacente, considerando o problema mais básico de

apenas duas classes, é a construção de um hiperplano de margem máxima que separa objetos pertencentes a classes diferentes.

Support Vector Machine (SVM) é um algoritmo de classificação conhecido por ser bem sucedido numa grande variedade de aplicações (Burges, 1998; Kim et al., 2002). Os SVM são uma das abordagens mais populares para modelagem e classificação de dados (Shawe-Taylor e Cristianini, 2004). As suas vantagens incluem a excelente capacidade de generalização, que diz respeito à capacidade de classificar corretamente as amostras que não estão dentro do espaço da característica, usado para o treino.

Resumidamente as SVMs consistem num método para treino de amostras que se baseia num princípio de minimização do risco estrutural, que minimiza, assim, o erro de generalização (Osuna at al., 1997). Dadas duas classes e um conjunto de pontos a essas classes, o SVM determina o hiperplano que separa os pontos de forma a colocar o maior número de pontos da mesma classe do mesmo lado, maximizando a distância de cada classe a esse hiperplano (ver Figura 2.10), sendo consequentemente denominado classificador de margem máxima. Com efeito, uma larga margem entre os valores correspondentes aos pontos dos dois subconjuntos de dados implica um risco de generalização minimizado do classificador (Lovell e Walder, 2006).

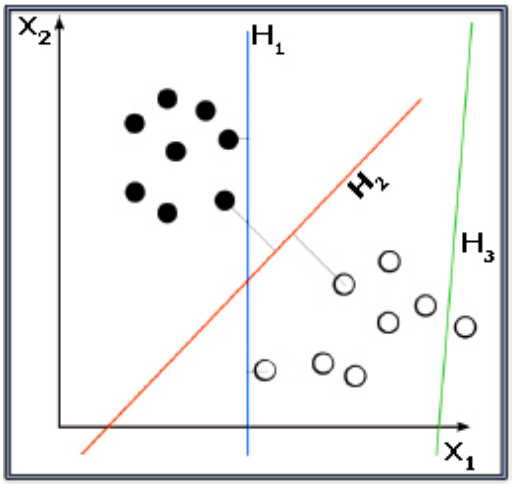


Figura 2.10 - SVM - Margem máxima entre os vetores de dados. (Lu et al., 2007))

As SVMs desenvolvem e aplicam algoritmos que permitem classificar e reconhecer padrões em diversos tipos de dados. Este classificador é utilizado em diversas aplicações, tais como reconhecimento de padrões e de faces, diagnósticos clínicos, supervisão de processos industriais e com relevância para este trabalho, processamento e análise de imagens (Carvalho e Lorena, 2007; Tchangani, 2005).

A utilização de um classificador é efetuada à posterior da extração das características das imagens pertencentes a diferentes classes. Na aprendizagem supervisionada dado um conjunto de exemplos do tipo (x_i, y_i) em que x_i representa um exemplo e y_i a sua classificação, deve-se reproduzir um classificador capaz de prever a classe a que pertencem novos dados, efetuando assim o processo de treino. Através da Figura 2.11 é possível visualizar, de modo genérico este processo.

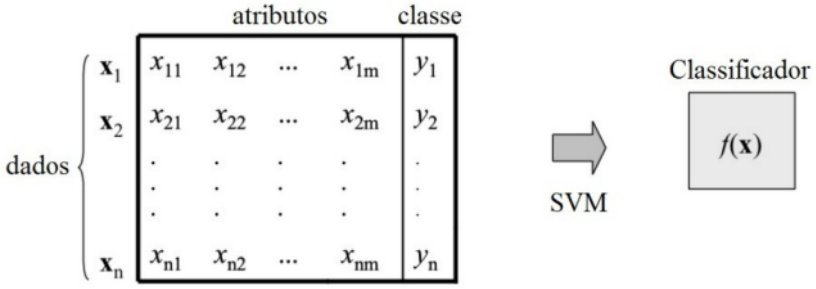


Figura 2.11 – Modelo de um classificador de aprendizagem supervisionada (Carvalho e Lorena, 2007)

2.4. Sumário

O processamento de imagens é uma área em crescimento. Independentemente da área de investigação, é necessário que cada imagem processada possa obter uma classificação final. Para que esta classificação seja possível é essencial que se proceda à extração das características que constituem essa mesma imagem, através de um descritor de imagem. Ao longo deste capítulo foram apresentados alguns dos descritores atualmente relevantes, o SIFT (Lowe, 1999, 2004), o GLOH (Mikolajczyk e Schmid, 2005), o SURF (Bay, et al., 2008), o LESH (Sarfraz e Hellwich, 2008) e o descritor MPEG-7.

Depois de detetadas e extraídas as características das imagens pode-se efetuar a sua classificação. Para que desta classificação se obtenha um resultado é necessário a seleção de um classificador. O método de classificação de imagens pode ser feito de dois modos: escolhendo um classificador supervisionado ou um classificador não supervisionado (Marques, 2005). Um classificador supervisionado requer aprendizagem/treino, do qual é exemplo o classificador SVM, ao passo que um classificador não supervisionado não requer treino/aprendizagem. Ao longo deste capítulo foi efetuado um estudo sobre classificadores e foram apresentados alguns classificadores utilizados no processamento de imagem, como o SVM (Vapnik, 1995), o K-NN (Cover e Hart, 1967), os AG (Holland, 1975), o classificador de *Bayes* (Gonzalez, 1993), as RNA (Hagan et al., 1996) e os sistemas *Fuzzy* (Zadeh, 1965).

Capítulo 3- Arquitetura implementada

3.1. Introdução

Na sequência da pesquisa efetuada no capítulo anterior, referente à deteção e classificação de imagens, pode-se verificar que o processamento de imagem é uma área em expansão. O processamento de imagem pode ser dividido em três etapas principais, como se pode verificar na Figura 3.1:

- **Pré-processamento**: etapa que permite o melhoramento da imagem para futuro processamento;
- **Extração/Deteção**: a extração, a partir da imagem a processar, das características relevantes que existem nessa imagem;
- **Classificação**: a utilização das características para a determinação do resultado.

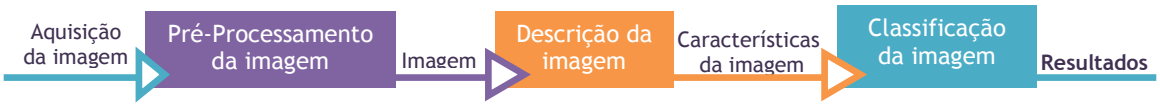


Figura 3.1 - Etapas principais do processamento digital de imagem [Fonte própria]

Neste capítulo é apresentado um esquema geral da interface desenvolvida e são descritos os diferentes módulos nela integrados. Cada um destes módulos é realizado de forma sequencial com vista à obtenção do resultado final.

A classificação de imagens é a última etapa do processamento de imagem e que permite que o utilizador obtenha o resultado desse processamento. Como referido no capítulo anterior, existem diversos classificadores que podem ser implementados, quer com treino prévio ou não. A implementação destes classificadores será efetuada com recurso à ferramenta de *data mining* Weka, pelo que dentro deste capítulo existirá um subcapítulo com o objetivo de enquadrar o conceito *data mining*, apresentar algumas dessas ferramentas atualmente existentes e apresentar a justificação da utilização do *data mining* Weka.

3.2. Esquema geral

De forma a englobar as principais etapas do processamento de uma imagem numa mesma ferramenta, otimizando essas mesmas etapas foi desenvolvido um *software* para esse fim. A ferramenta a desenvolver terá como *input* uma imagem ou um conjunto de imagens, permitirá pré-processamento básico na imagem (RGB, HSL, etc.), efetuará a extração das características dessa imagem, através de um descritor, e posteriormente com base num classificador selecionado obterá o *output* da classificação. O formato do *output* (.txt, .csv, .xml, etc.) deve ser selecionado pelo utilizador, e deve ter a possibilidade de armazenar os resultados para futuras utilizações. O esquema seguinte mostra a arquitetura geral do funcionamento desta interface.

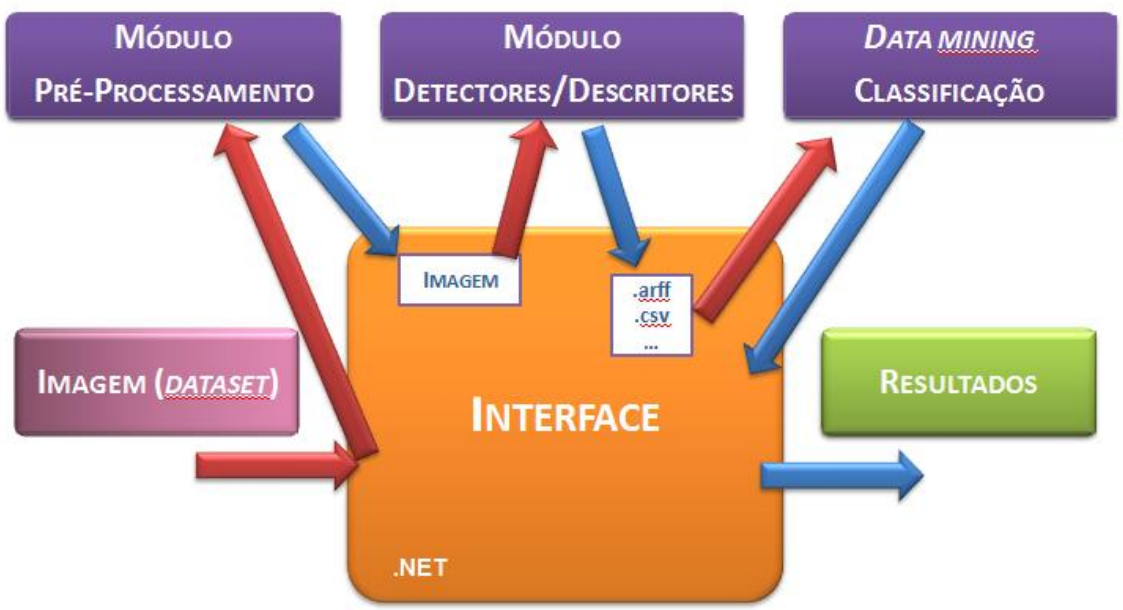


Figura 3.2 - Esquema geral [Fonte própria]

A interface desenvolvida realiza as diferentes etapas de forma sequencial e é modular, uma vez que possibilita a integração e adição de novos módulos sejam de pré-processamento, de extração de características ou mesmo de classificação.

3.3. Descrição dos módulos

A interface desenvolvida integra as principais etapas do processamento de imagem. Através desta interface o utilizador poderá escolher, por exemplo, o método de pré-processamento que pretende para que posteriormente seja realizada a respetiva extração de características, através de um descritor, ex. SIFT. Depois de extraídas as características, estas devem constar num vetor numérico com o formato *.arff (Attribute-Relation File Format)*, para que o utilizador o classifique com recurso à ferramenta de *data mining* WEKA e por fim possa obter o resultado final dessa classificação.

O funcionamento interno da arquitetura concebida divide-se em três módulos principais, como ilustrado na Figura 3.3.

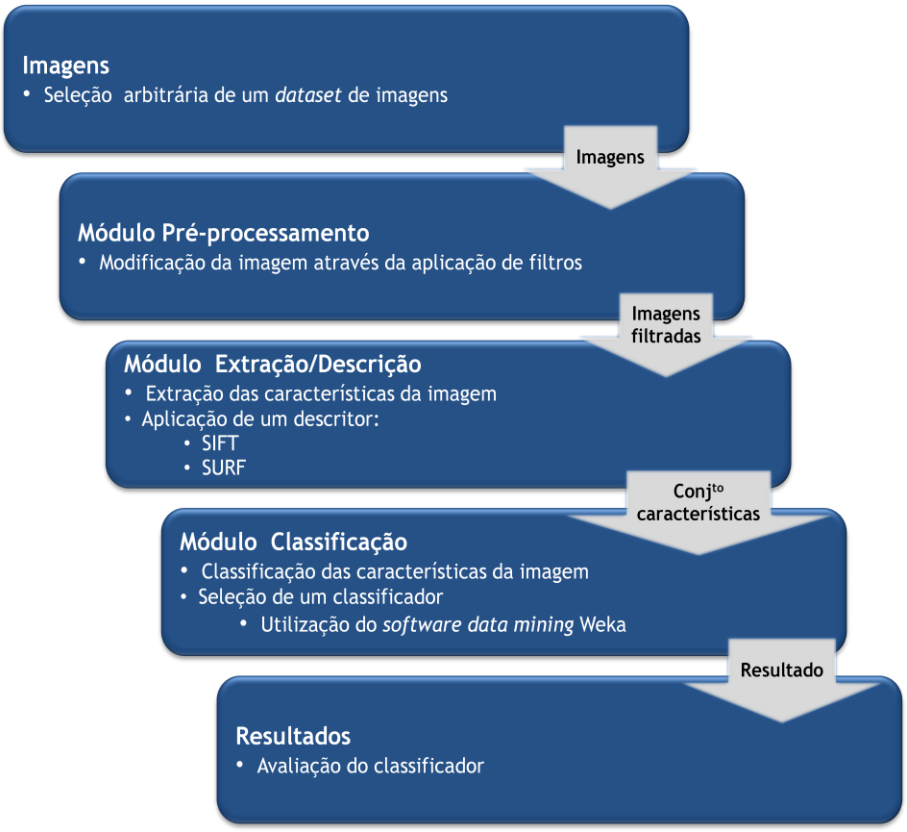


Figura 3.3 – Visão global da interface proposta [Fonte própria]

3.3.1. Módulo de pré-processamento

Após se obter uma imagem digital, o próximo passo no processamento é pré-processar essa mesma imagem, caso o utilizador considere necessário, pois o utilizador pode optar por não efetuar nenhum tipo de pré-processamento sobre a imagem. O pré-processamento tem como objetivo melhorar a imagem, de forma a auxiliar os processos seguintes, através da aplicação de filtros. O pré-processamento envolve técnicas para a remoção de ruído, realce de contrastes e melhoria na nitidez da imagem. Entre essas técnicas encontram-se, por exemplo *Grayscale* (conversão da imagem para tons de cinzento), *RGB colors (*abreviatura do sistema de cores formado por Vermelho (*Red*), Verde (*Green*) e Azul (*Blue)), HSL colors*, etc.

A representação deste módulo pode ser analisada através da Figura 3.4.

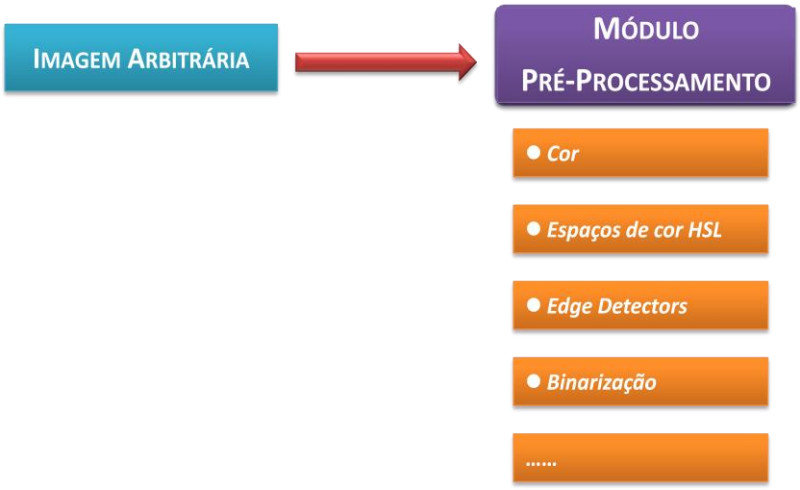


Figura 3.4 - Estrutura do módulo de pré-processamento [Fonte própria]

Através deste módulo o utilizador pode escolher, se pretender, o tipo de pré-processamento que pretende aplicar à imagem que processa.

3.3.2. Módulo de detetores/descritores

O módulo de deteção/descrição é responsável pela extração das características que compõem uma imagem, através da utilização de descritores. O produto final deste módulo é um ficheiro *.arff* com as características da imagem para posterior classificação.

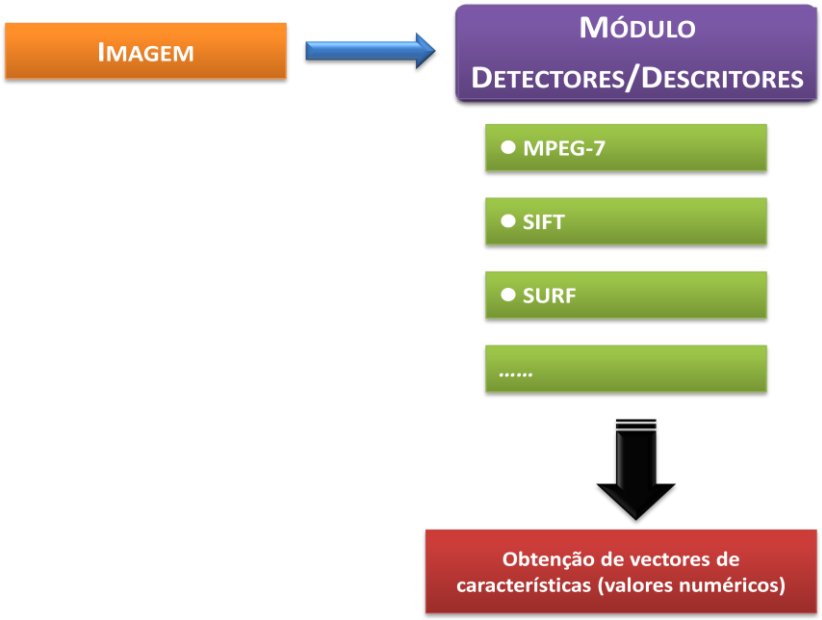


Figura 3.5 - Estrutura do módulo de deteção/descrição [Fonte própria]

3.3.3. Módulo de classificação

O módulo de classificação permite que o utilizador possa obter o resultado do processamento de uma dada imagem, ou de um conjunto de imagens. Para tal recebe um ficheiro com valores numéricos onde se encontram as características da imagem, extraídas através do módulo de descrição/deteção. Este ficheiro é depois classificado através de um classificador escolhido pelo utilizador, de forma a obter-se o resultado final. Este processo encontra-se representado no esquema da Figura 3.6.

A implementação deste módulo foi efetuada com recurso à ferramenta *data mining Weka*, da qual se faz referência no subcapítulo que se segue.

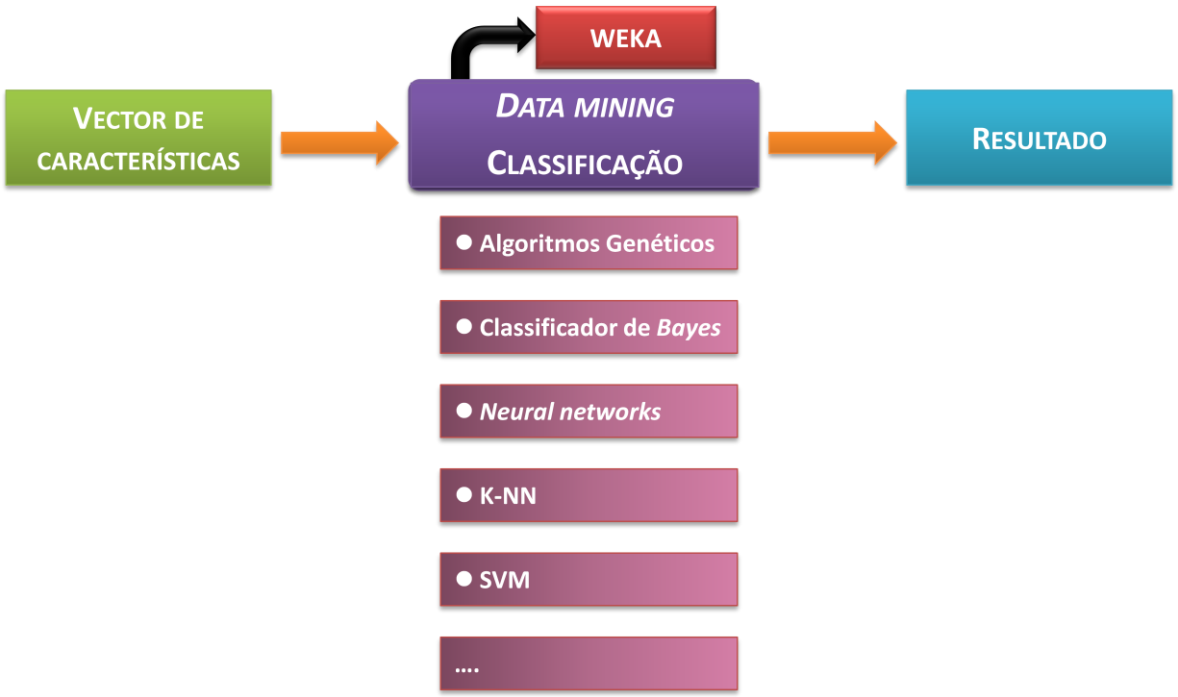


Figura 3.6 – Estrutura do módulo de classificação [Fonte própria]

3.4. Utilização do *data mining* Weka

Os computadores e as tecnologias de informações foram descobertas importantes que permitiram que fosse possível recolher e armazenar grandes quantidades de informação. A dimensão e complexidade da informação que se recolhe pode levar a que não seja possível a extração de todo o conhecimento que essa mesma informação comporta. Com objetivo de solucionar este problema surgiu a área da Descoberta de Conhecimento em Bases de Dados da qual faz parte o *data mining* (Adriaans e Zantinge, 1996) (Fayyad et al., 1996).

Neste subcapítulo esclarece-se o conceito de *data mining* e é efetuado um levantamento e caracterização de ferramentas de *data mining*.

3.4.1. Conceito de *Data Mining*

O crescente desenvolvimento das tecnologias de informação tem permitido o aumento do número de base de dados e consequente necessidade de recurso a análises automáticas de grandes quantidades de informação (Petrovskiy, 2003).

Nos dias de hoje, armazenar e guardar dados é uma realidade, existindo já uma enorme quantidade de dados disponível acerca dos mais variados assuntos (Witten e Eibe, 2005). Mas se existem em grande número e sobre os mais variados assuntos, será que teremos conhecimento concreto dos dados que existem nessas bases de dados? É importante que de um conjunto de dados analisados se consiga obter um sentido, partindo da descoberta de padrões relevantes, obtendo conhecimento desses dados e usando esse conhecimento em situações futuras (Witten e Eibe, 2005).

Segundo Han e Kamber (2006) *data mining* pode ser definido como a extração de padrões de conhecimento incluídos em grandes bases de dados. *Mining* caracteriza o processo de procura de informação relevante num grande volume de informação (Han e Kamber, 2006).

Se para alguns *data mining* é sinónimo de extração de conhecimento de dados (Han e Kamber, 2006), para outros é uma etapa do processo de descoberta de conhecimento em base de dados, o qual é composto por um conjunto de etapas: seleção, pré-processamento, transformação, *data mining*, e finalmente a interpretação (Han e Kamber, 2006) (Adriaans e Zantinge, 1996) (Fayyad et al., 1996).

3.4.2. Ferramentas de *Data Mining* mais utilizadas

Sempre que se desenvolve um *software* é necessário ter em consideração determinados fatores como o domínio de aplicação, em que sistema operativo irá funcionar, a linguagem de programação a utilizar, etc.

Para Santos e Azevedo (2005) uma ferramenta começa por ser caracterizada pela linguagem de programação a utilizar, em seguida pela plataforma do sistema, pela sua portabilidade, o seu estado de desenvolvimento e a sua possibilidade de integração com outras aplicações.

Tendo em conta Goebel e Gruenwald (1999) uma ferramenta pode ser caracterizada em três grupos: características gerais, conectividade com a base de dados e as características de *data mining*. Nas características de *data mining* caracterizam-se tarefas da descoberta tais como: pré-processamento, previsão, classificação, associação, segmentação, visualização e análise exploratória.

King, et al. (2006) definem cinco categorias de características de *software* de *data mining*: capacidade, facilidade de aprendizagem/utilização, interoperabilidade, flexibilidade e a precisão. A capacidade caracteriza e classifica o que uma ferramenta pode fazer, a interoperabilidade caracteriza a possibilidade de integração com outras aplicações e a flexibilidade caracteriza as possibilidades de alteração de parâmetros críticos da ferramenta ao longo do processo.

A Rexer Analytics (2011) efetua anualmente um levantamento sobre a utilização de ferramentas *data mining*. Tendo como base o relatório produzido por esta empresa, algumas conclusões podem ser tomadas relativamente ao grau de satisfação de utilização de ferramentas *data mining* e quais as características importantes na utilização dessas ferramentas *data mining* (Rexer,2010). Para isso são apresentadas de seguida algumas tabelas e gráficos com informação existente no relatório anteriormente referido, sobre utilização de ferramentas *data mining* (Rexer,2010).

A existência de um modelo de expansão/transparência foi considerado importante na escolha da maioria das ferramentas *data mining*. Esta observação foi possível através de uma questão realizada a uma população alvo que englobava empresas, consultores, académicos, instituições governamentais/não governamentais e vendedores, tal como indicado na figura seguinte.

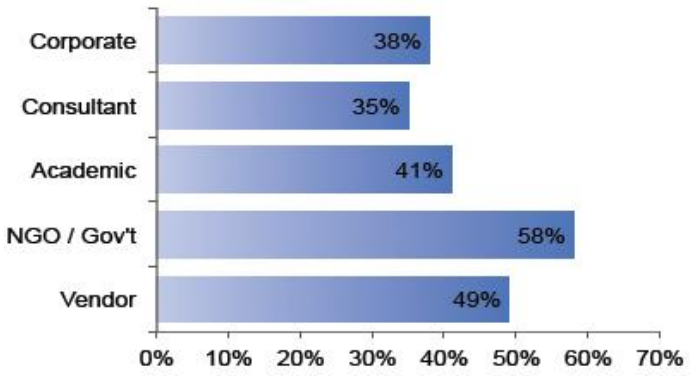


Figura 3.7 – Percentagem de empregadores que consideram importante o módulo de expansão/transparência (Rexer, 2010)

Tendo em conta o tipo de licença do *software* pode-se verificar que neste inquérito foram testados três ferramentas *data mining open source* e cinco comercias.

Tipo de *Software*	IBM SPSS Statistics	IBM SPSS Modeler	Knime	R	SAS	SAS Enterprise Miner	STATISTICA (StatSoft)	Weka
Comercial	x	x			x	x	x	
Open Source			x	x				x

Tabela 3.1 - Caracterização das ferramentas segundo a sua licença

A ferramenta *Weka* foi indicada, pela população alvo, como principal ferramenta *open source* utilizada visto ter um modelo de expansão, tal como se pode verificar na Figura 3.8.

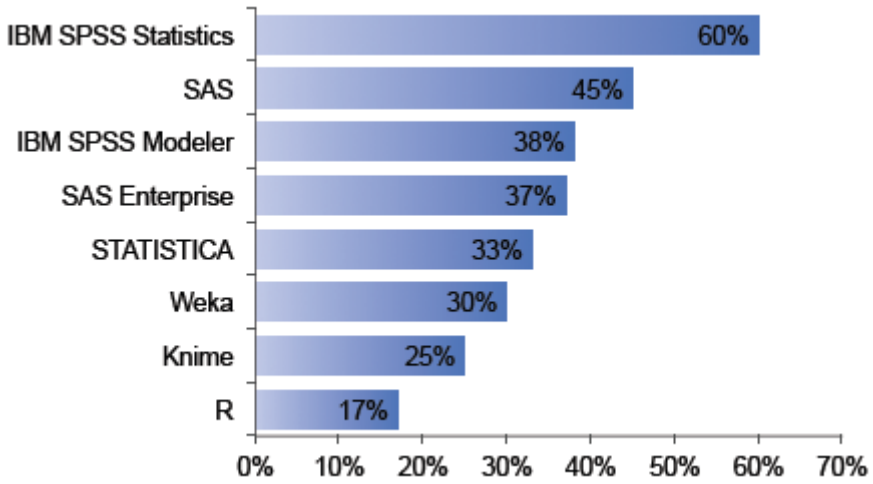


Figura 3.8 - Percentagem da importância de um modelo de expansão como principal ferramenta utilizada (Rexer, 2010)

A população alvo quando questionada sobre qual a ferramenta de *data mining* mais utilizada em 2009, considerou que a ferramenta *open source* R é a mais utilizada (43%) e a ferramenta STATISTICA é a principal ferramenta de *data mining* escolhida (18%). A nível académico tal como a ferramenta R a ferramenta *Weka* é a principal ferramenta utilizada.

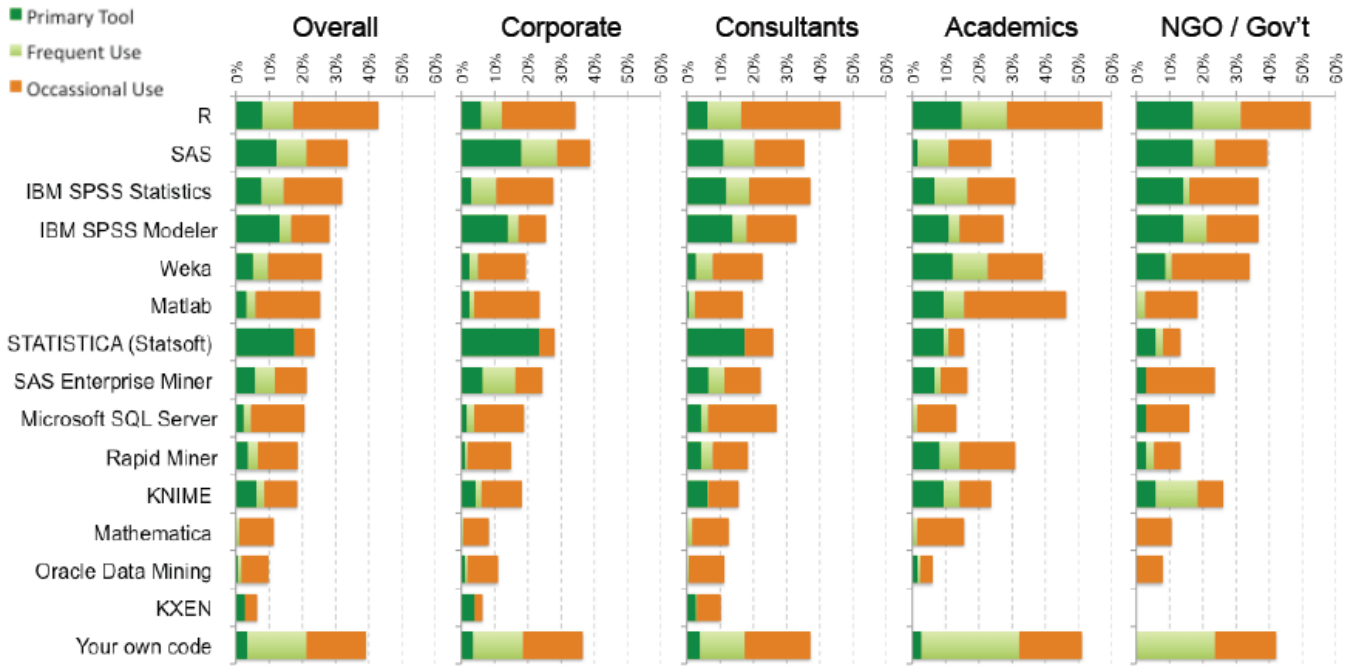


Figura 3.9 – *Data mining software* mais utilizado em 2009 (Rexer, 2010)

Com os vendedores excluídos desta questão, toda a restante população foi questionada sobre o grau de satisfação geral com a utilização de ferramentas *data mining* nos anos 2009 e 2010. O resultado desta questão encontra-se representado na figura seguinte, onde se pode verificar que a nível de ferramentas *open source* a ferramenta R é a que apresenta um grau de satisfação maior, no entanto ninguém se mostrou insatisfeito com a ferramenta *open source Weka* apresentando um grau de satisfação geral razoável.

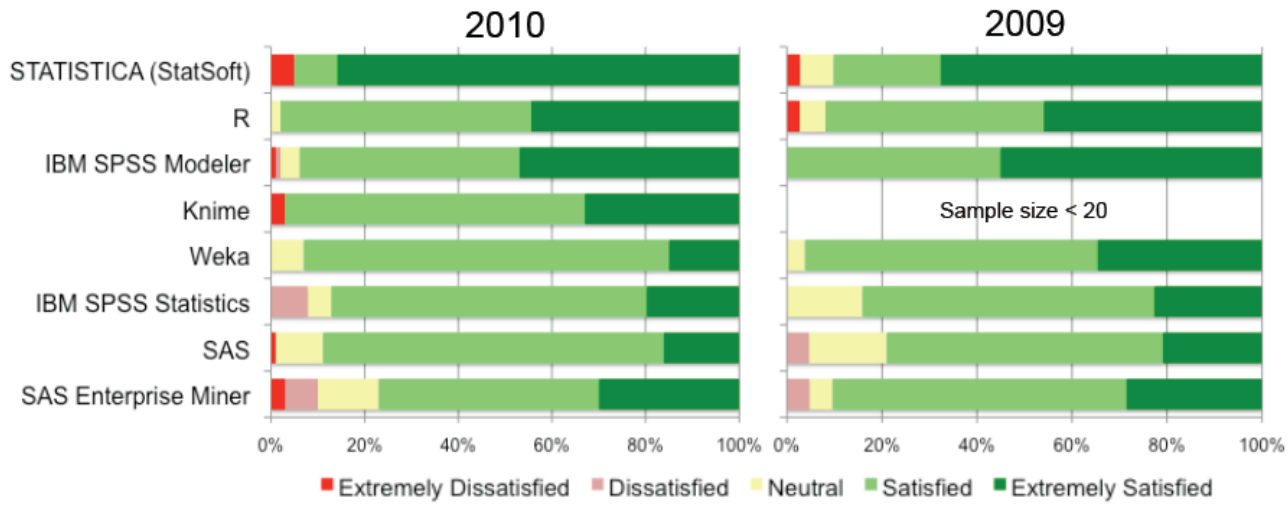


Figura 3.10 – Grau de satisfação geral com a utilização de ferramentas *data mining* (Rexer, 2010)

A população alvo, exceto os vendedores, quando questionada sobre quais os fatores que consideravam mais prioritários na escolha de um *software* de *data mining*, apresentou um conjunto diversificado de fatores prioritários.

	Overall	IBM SPSS Statistics	IBM SPSS Modeler	Knime	R	SAS	SAS Enterprise Miner	STATISTICA (StatSoft)	Weka
Qualidade e precisão de desempenho	**31%**	24%	23%	24%	19%	18%	35%	64%	29%
Habilidade para lidar com grandes conjuntos de dados	**27%**	33%	30%	12%	16%	39%	29%	18%	25%
Variedade de algoritmos disponíveis	**26%**	33%	34%	21%	42%	22%	19%	19%	43%
Custo do *software*	**23%**	26%	21%	36%	33%	21%	23%	6%	29%
Capacidade de manipulação de dados	**22%**	19%	38%	21%	30%	27%	23%	18%	18%
Facilidade de utilização	**20%**	26%	24%	39%	9%	12%	26%	21%	7%
Confiança / Estabilidade do *software*	**18%**	12%	20%	12%	16%	21%	13%	34%	4%
Forte visualização gráfica dos modelos	**15%**	10%	7%	12%	26%	19%	13%	21%	11%
Velocidade	**14%**	5%	7%	9%	23%	13%	10%	21%	18%
O *software* contém uma técnica específica de análise	**13%**	17%	7%	15%	16%	18%	6%	10%	18%
Habilidade para automatizar tarefas repetitivas	**13%**	12%	10%	21%	14%	16%	6%	10%	11%
Qualidade dos gráficos	**12%**	26%	11%	12%	7%	4%	13%	8%	11%
Qualidade da interface com o utilizador	**11%**	7%	15%	12%	2%	6%	13%	13%	14%
Habilidade para modificar as opções do algoritmo	**10%**	7%	8%	9%	23%	7%	23%	5%	18%
Menu de ajuda útil, demos e tutoriais	**4%**	10%	4%	9%	5%	0%	3%	2%	0%

Maior prioridade Menor prioridade

Tabela 3.2 - Frequência de cada fator (Rexer, 2010)

Como se pode analisar através da Tabela 3.2, o *software data mining* Weka tem uma grande variedade de algoritmos disponíveis, apresentando uma frequência de utilização razoável.

3.4.3. Weka

Desenvolvido pela Universidade de Waikato, na Nova Zelândia (Weka, 2010), o *software Waikato Environment for Knowledge Analysis* (WEKA) foi desenvolvido na linguagem de programação Java (orientada aos objetos) e é uma coleção de algoritmos de *machine learning* (Witten e Frank, 2005). O WEKA é um *software open source* construído sob os termos da GNU *General Public License*, que disponibiliza ferramentas para o pré-processamento de dados, classificação, regressão, *clustering*, regras de associação, visualização entre outras características (Witten e Frank, 2005).

O Weka foi um projeto lançado em 1993, onde os algoritmos de aprendizagem estavam disponíveis em vários idiomas e para utilização em diferentes plataformas. É um *software* utilizado em investigação, no ensino e em aplicações que fornece não só um conjunto de algoritmos de aprendizagem, como também a oportunidade dos investigadores implementarem novos algoritmos (Hall, et al., 2009).

Este *software* é implementada em Java, além de conter vários algoritmos de *machine learning* possui ainda ferramentas de classificação e visualização dos dados.

3.4.4. Sumário

Depois de se efetuar uma análise aos diferentes *softwares data mining* existentes pode-se constatar que o *software Weka* continua a ser utilizado pelos investigadores nos seus trabalhos, sendo uma das ferramentas utilizadas a nível académico.

O *software Weka* apresentava ferramentas de classificação já incorporadas, possibilitando a implementação do módulo de classificação de imagens (Witten e Frank, 2005).

O Weka é um *software open source*, permitindo a sua livre utilização e que continua a ser utilizado pelos investigadores, tal como referido anteriormente no relatório produzido pela Rexer Analytics (Rexer, 2010).

O *software* Weka mostrou-se de fácil utilização e apresentou-se como uma ferramenta *data mining* útil e viável para a implementação do módulo de classificação da interface implementada.

3.5. Conclusão

Para se efetuar o processamento de uma imagem, ou de um conjunto de imagens, é necessário que essa imagem passe por vários processos até se obter um resultado final. Desta forma, a interface implementada apresentava uma arquitetura por módulos, onde cada módulo é responsável por cada um dos processos, desde o pré-processamento, passando pela extração de características e por fim efetuando a classificação da imagem.

Todos estes módulos foram implementados de forma sequencial, onde através de uma imagem digital o investigador pode, se assim o pretender, fazer algum tipo de pré-

processamento a essa imagem através do módulo de pré-processamento, de seguida pode-se extrair as características dessa imagem através do módulo de extração/descrição e por fim efetuar a classificação das características extraídas através do módulo de classificação.

A classificação das características extraídas foi um módulo desenvolvido com recurso à utilização da ferramenta de *data mining* Weka. Esta ferramenta já tinha implementado alguns dos classificadores de imagem, fator que levou à sua utilização neste estudo.

O Weka é uma ferramenta que tem como objetivo agregar algoritmos provenientes de diferentes abordagens na subárea da inteligência artificial de dedicada ao estudo da aprendizagem automática. Esta subárea pretende desenvolver algoritmos e técnicas que permitam a um computador "aprender" (obter novo conhecimento) quer indutiva quer dedutivamente.

Capítulo 4- Descrição da interface

4.1. Introdução

A interface implementada possibilita que os utilizadores possam processar uma imagem, ou conjunto de imagens, apenas numa única ferramenta.

De forma a identificar a interface foi-lhe atribuído o nome de ESTLabImg, um diminutivo para EST Laboratório de Imagem, uma vez que se trata de processar imagens digitais, e EST por ser o nome da instituição onde o presente estudo foi desenvolvido.

Como referido anteriormente, para que uma dada imagem possa ser classificada é necessário passar por diferentes etapas. A interface gráfica desenvolvida neste estudo implementa algumas dessas etapas, através do desenvolvimento de três módulos diferentes, sendo eles: o pré-processamento, a extração/descrição e por fim a classificação.

Ao longo deste capítulo é relembrada a questão de investigação e os principais passos necessários à sua resolução. O processamento das imagens é efetuado tendo em conta a análise de uma ou de várias imagens, tendo sido implementados dois métodos para efetuar essa análise. Desta forma é apresentada neste capítulo a estrutura de cada um dos métodos, bem como as ferramentas e linguagens de programação utilizadas na implementação da interface e dos módulos que a constituem. É apresentada também, uma descrição da interface desenvolvida de forma a efetuar o seu enquadramento com o presente estudo.

4.2. Questão de investigação

A interface gráfica desenvolvida tinha como objetivo responder à questão que levou a esta investigação, uma vez que se apresentou como fio condutor do tema deste estudo. Relembrando a questão:

Não será possível o desenvolvimento de uma interface gráfica que possibilite, de forma modular, a implementação das principais etapas do processamento de imagem?

Para que fosse possível responder a esta questão foi necessário definir alguns passos a seguir neste estudo, nomeadamente:

- Pesquisa de informação sobre as etapas do processamento de imagem;
- Seleção de algoritmos relevantes para cada etapa, nomeadamente no pré-processamento, na extração/descrição e na classificação;
- Desenho do esquema geral de interface e a sua divisão em módulos, representativos das etapas do processamento de uma imagem;
- Conceção e desenvolvimento da interface, denominada ESTLabImg;

- Conceção de dois métodos para processar imagem, um onde apenas é processada uma imagem e outro onde é processado um conjunto de imagens e elaboração de fluxogramas representativos desses métodos;
- Implementação dos diferentes módulos;
- Exemplificação da interface aplicada a um *dataset* (conjunto) de imagens;
- Visualização e interpretação dos resultados.

Alguns destes passos já foram desenvolvidos nos capítulos anteriores, os restantes passos serão desenvolvidos ao longo deste e dos capítulos seguintes.

4.3. Esquema implementado

A interface desenvolvida visa o enquadramento de diferentes etapas do processamento de imagem, numa única interface. Desta forma a interface implementa cada etapa através do desenvolvimento de três módulos: o módulo de pré-processamento, o módulo de descrição e o módulo de classificação.

Os módulos desenvolvidos foram implementados nas seguintes linguagens de programação:

- **Módulo de pré-processamento** – utilização da linguagem de programação C# com recurso à *framework* AForge.NET para implementação dos diferentes filtros;
- **Módulo de extração/ descrição** – neste módulo foi implementado o descritor SIFT, referenciado no capítulo dois, na linguagem de programação Visual C++ e o descritor SURF, na linguagem de programação C#;
- **Módulo de classificação** – este módulo foi desenvolvido com recurso à ferramenta de *data mining* Weka. Para tal foi utilizada uma *bridge* para conversão dos ficheiros Weka (Java) em *.dll*.

O esquema da Figura 4.1 descreve a implementação da interface e dos módulos que a constituem.

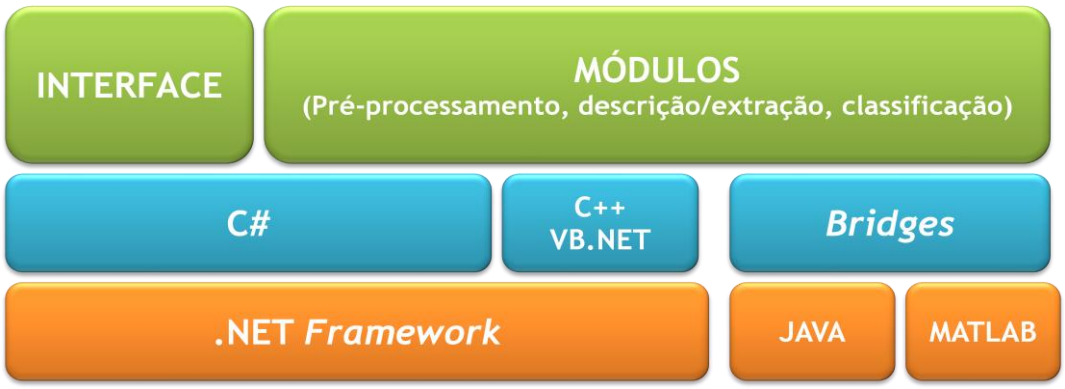


Figura 4.1 – Esquema implementado

Na Figura 4.1 encontra-se na camada base a referência ao Java e ao Matlab, pois nos módulos existentes é possível adicionar código implementado tanto em Java, como em Matlab. A integração deste código na *framework* .NET é necessário recorrer à utilização de uma ferramenta de interligação entre ambas, ou seja, utilização de uma *bridge*, como indicado no tópico 4.3.2.

A interface foi desenvolvida com recurso ao *software Microsoft Visual Studio 2010*, usando a linguagem de programação C# sobre a *framework* .NET.

4.3.1. *Utilização do Microsoft Visual Studio 2010*

Para o desenvolvimento da interface gráfica foi utilizado o *Microsoft Visual Studio 2010*. Este *software* é um ambiente de desenvolvimento integrado (IDE) da Microsoft, que pode ser utilizado para desenvolver aplicações de consola e aplicações gráficas (GUI) assim como aplicações de formulários Windows (*Windows Form Applications*), *Websites*, aplicações Web e serviços *Web*. O *Visual Studio* encontra-se especialmente dedicado à *Framework* .NET, oferecendo suporte em diferentes linguagens de programação como *Visual Basic* (VB), C, C++, C# (C Sharp) e J# (J Sharp) (Loureiro, 2011).

A *Framework* .NET é uma iniciativa da Microsoft que visa uma plataforma única para o desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código gerado para .NET pode ser executado em qualquer dispositivo ou plataforma que possua a *framework*: a plataforma .NET (Loureiro, 2008).

4.3.2. Utilização de *bridges*

Como já referido ao longo deste estudo o Weka é um *software data mining* e a sua implementação é efetuada em Java. Uma vez que a interface implementada foi desenvolvida em .NET foi necessário a utilização de uma *brigde* entre .NET e Weka (Java). Para tal foi utilizada a ferramenta IKVM.

O IKVM é uma implementação do Java para *Microsoft .NET Framework*. Esta implementação inclui (IKVM, 2011):

- Uma máquina virtual Java implementada em .NET;
- Implementação das livrarias de classes do Java em .NET;
- Ferramentas que possibilitem a interoperabilidade entre Java e .NET.

Esta *bridge*, o IKVM, permite a transformação de ficheiros *.class*/*.jar* em ficheiros *.dll*/ *.exe*, ficheiros estes que podem ser referenciados pela plataforma .NET.

A figura seguinte ilustra a implementação do Weka (Java) para .NET.

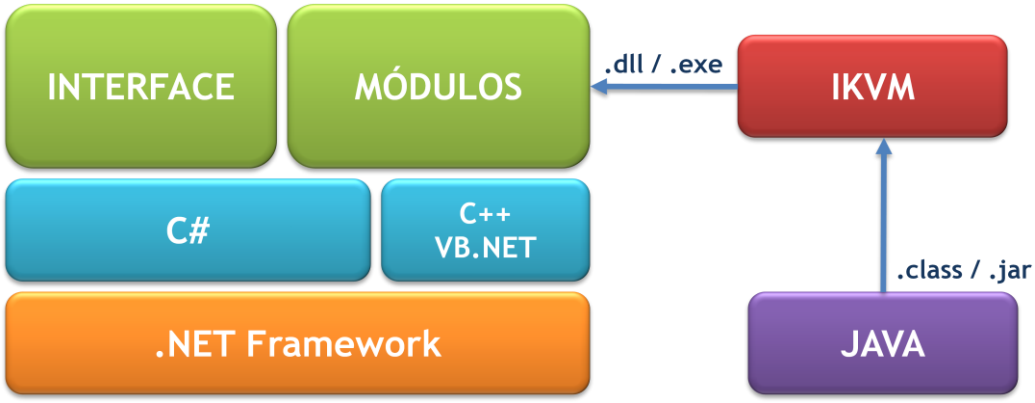


Figura 4.2 - Utilização de implementações Java em .NET

O IKVM não se apresentou como a única *bridge* a utilizar, no entanto o facto de ser uma ferramenta *free* e pela facilidade de conversão foi a adotada para este estudo. Entre as outras opções encontravam-se outras ferramentas, tais como:

- JNBridge (JNBridge, n.d.);
- JBind2.net (JBind2.net, 2004);
- JuggerNET (JuggerNET, 2006);
- J-Integra for .NET (J-Integra for .NET, n.d.).

A interface desenvolvida possibilita a integração com Matlab, sendo para isso necessário a existência de uma *brigde*. Esta *bridge* permitirá a ligação entre Matlab e .NET. Através da Figura 4.3 é possível visualizar um esquema com a implementação de Matlab em .NET, onde são apresentadas três soluções a utilizar como *bridge*:

- *Matlab Builder NE* (Mathworks, n.d.);
- *Matlab .NET interface* (Mathworks, n.d.);
- Soluções *open source* (Ruffaldi, 2003).

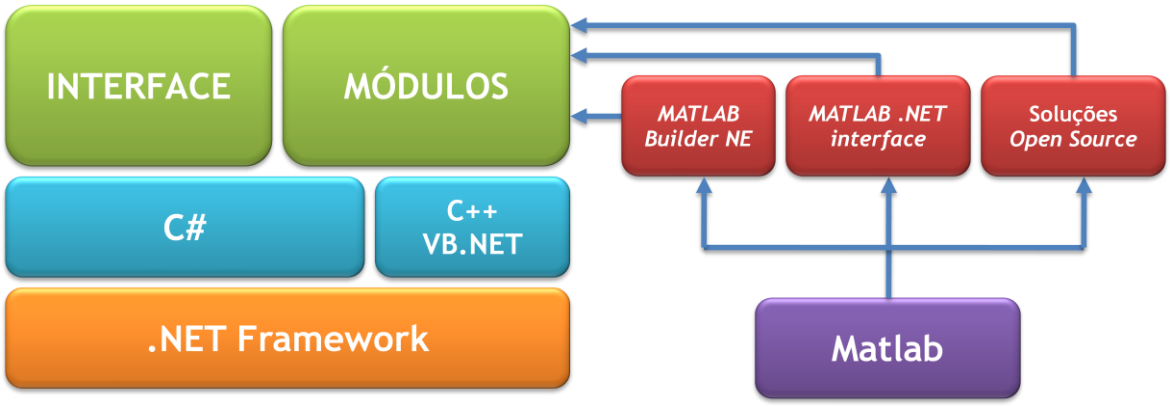


Figura 4.3 - Utilização de implementações Matlab em .NET

As soluções *open source* são disponibilizadas por Ruffaldi (2003) que apresenta três formas de aceder ao Matlab através do .NET:

- *low level* C API;
- DDE (*Dynamic Data Exchange*);
- COM (*.NET COM Interoperability*).

Ruffaldi (2003) apresenta uma livraria que contém acesso direto às matrizes; acesso a ficheiros *Matlab*, armazenando-os em formato binário e validação de expressões. A porção de código seguinte apresenta o acesso às *dll* do Matlab (Ruffaldi, 2003):

```
// Creates a Matrix with the specified dimensions
[DllImport("libmx.dll")]
internal static extern IntPtr mxCreateDoubleMatrix(int n, int m, mxComplexity c);

// Creates a Matrix with the specified dimensions
[DllImport("libmx.dll")]
internal static extern IntPtr mxCreateNumericMatrix(int n, int m, mxClassID classid, mxComplexity c);

// Creates a Multidimensional Matrix with the specific type
[DllImport("libmx.dll")]
internal static extern IntPtr mxCreateNumericArray(int ndim, int[] dims, mxClassID classid, mxComplexity flag);
```

A porção de código que se segue apresenta um exemplo de utilização das funções Matlab no .NET (Ruffaldi, 2003):

```
private void pictureBox1_Click(object sender, System.EventArgs e)
{
    // 1) transform the bitmap into a matrix
    // 2) apply matlab transformation or something
    // 3) show the result as bitmap
    using (EngMATAccess mat = new EngMATAccess())
    {
        Matrix mx = Bitmap2Matrix(bit);
        mat.SetMatrix("aa", mx);
        mat.Evaluate("imshow(aa); aa = imnoise(aa);");
        mx = mat.GetMatrix("aa");
        Matrix2Bitmap(bit, mx);
        MessageBox.Show("Teste");
        pictureBox1.Invalidate();
    }
}
```

4.4. Desenvolvimento da aplicação

Quando se pretende desenvolver uma determinada interface existe ligado a esse desenvolvimento uma preocupação constante: os utilizadores finais. A melhor aplicação pode ter o seu sucesso comprometido se a sua interface não cativar os seus potenciais utilizadores (Cooper et al., 2007).

A ESTLabImg é uma interface que possibilita o processamento de imagens, através da implementação das suas principais etapas. Essas etapas foram desenvolvidas através da implementação de três módulos: o módulo de pré-processamento, o módulo de extração/descrição e o módulo de classificação.

Durante o processo de conceção da interface ESTLabImg implementaram-se dois métodos para realizar o processamento de imagem. Se apenas se pretender a análise de uma imagem, pode-se aplicar um filtro sobre essa imagem e depois aplicar-lhe um determinado descritor (ex. SIFT), ou em alternativa pode-se aplicar um descritor sem que se tenha aplicado qualquer filtro sobre ela, tal como se pode verificar na Figura 4.4.

Figura 4.4 - Análise de apenas uma imagem

O fluxograma que se segue representa o método de análise de apenas uma imagem, como representado na Figura 4.5.

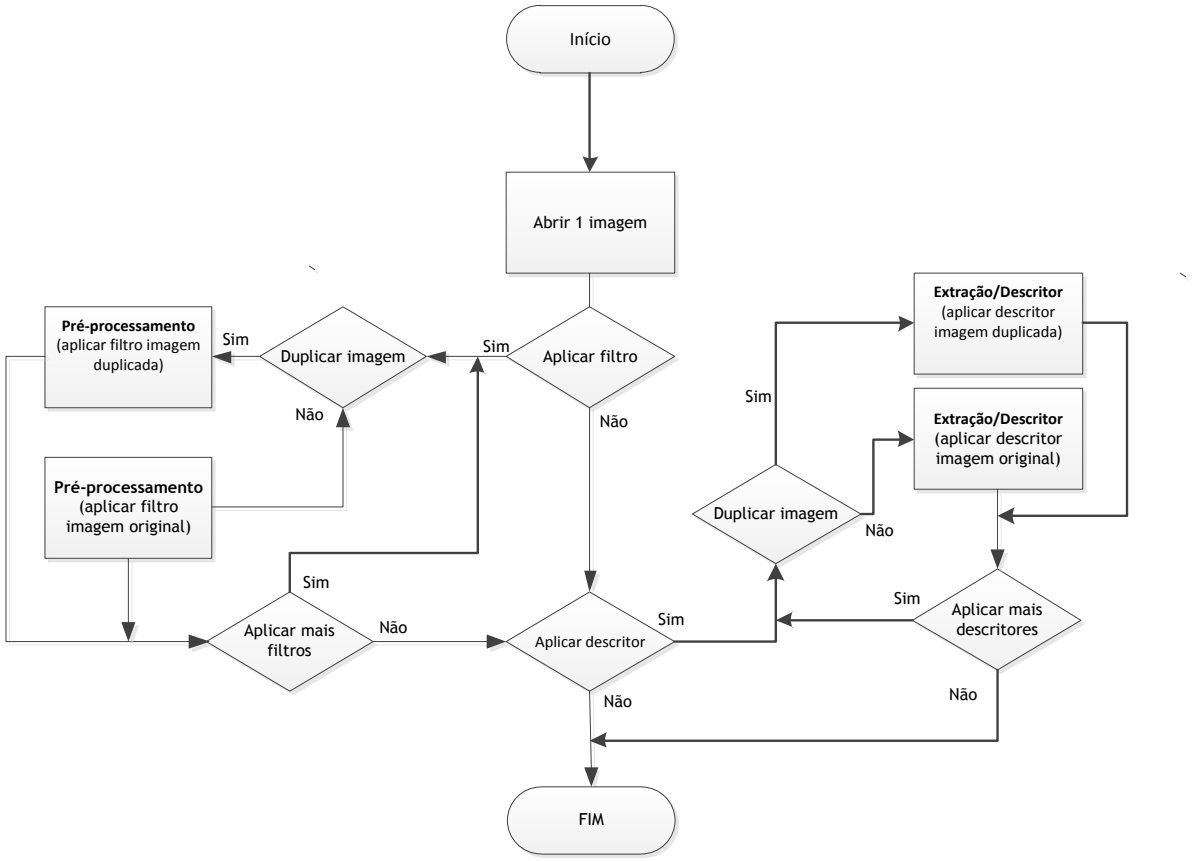


Figura 4.5 - Fluxograma representativo do processamento de uma imagem

Em alternativa pode-se efetuar o processamento de imagens sobre um *dataset* de imagens. Neste caso é aplicado o método de análise às imagens, descrito na Figura 4.6, onde se pode aplicar um ou vários filtros às imagens, posteriormente é aplicado um descritor de forma a extrair as características e por fim selecionar o classificador.

Figura 4.6 - Análise de um *dataset* de imagens

A Figura 4.7 apresenta o fluxograma que descreve o método de análise a um *dataset* de imagens.

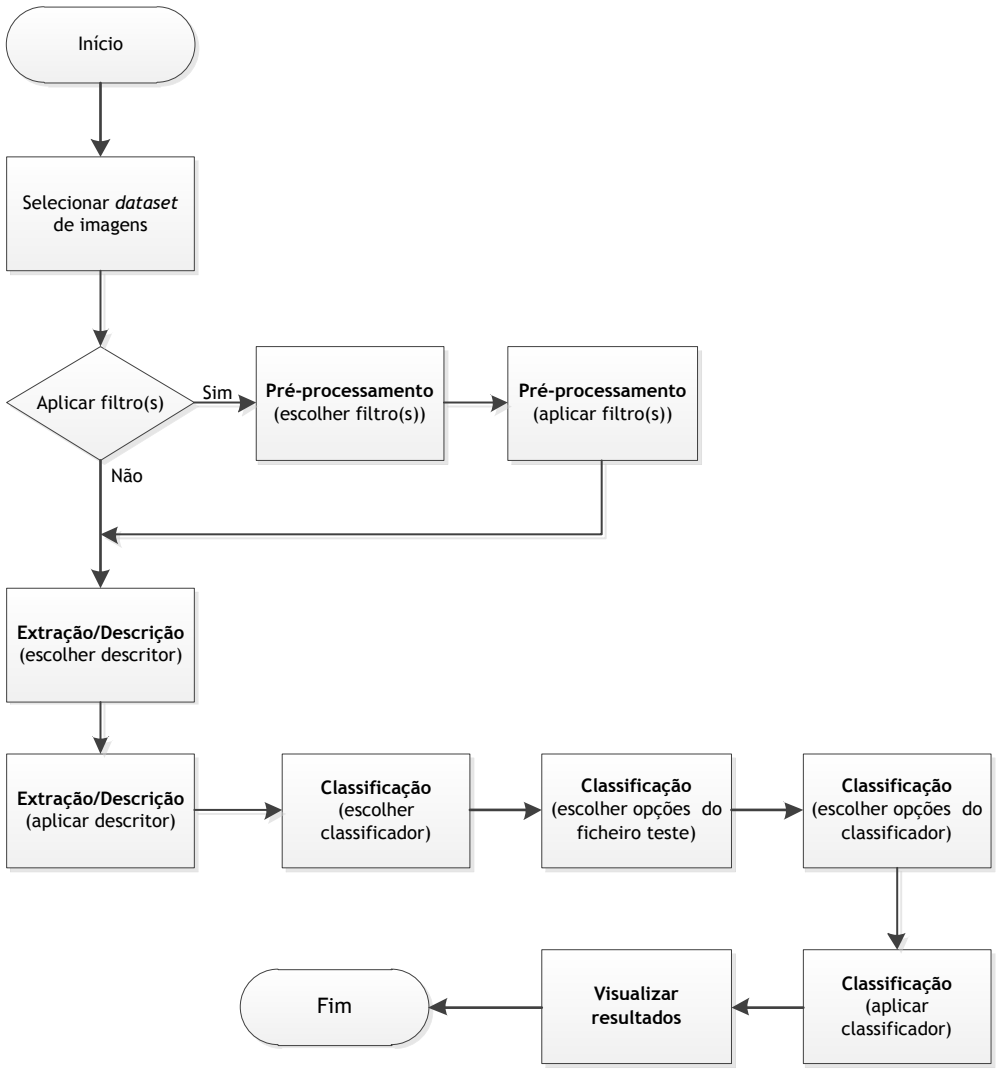


Figura 4.7 – Fluxograma representativo do processamento de um *dataset* de imagens

4.5. Interface implementada

A aplicação final foi estruturada através da implementação de quatro projetos, pertencentes à mesma solução. No *Visual Studio* entende-se por solução a estrutura que engloba um ou mais projetos. Como se pode verificar na Figura 4.8, estes quatro projetos correspondem à interface, ao módulo de pré-processamento, ao módulo de extração/descrição e ao módulo de classificação.

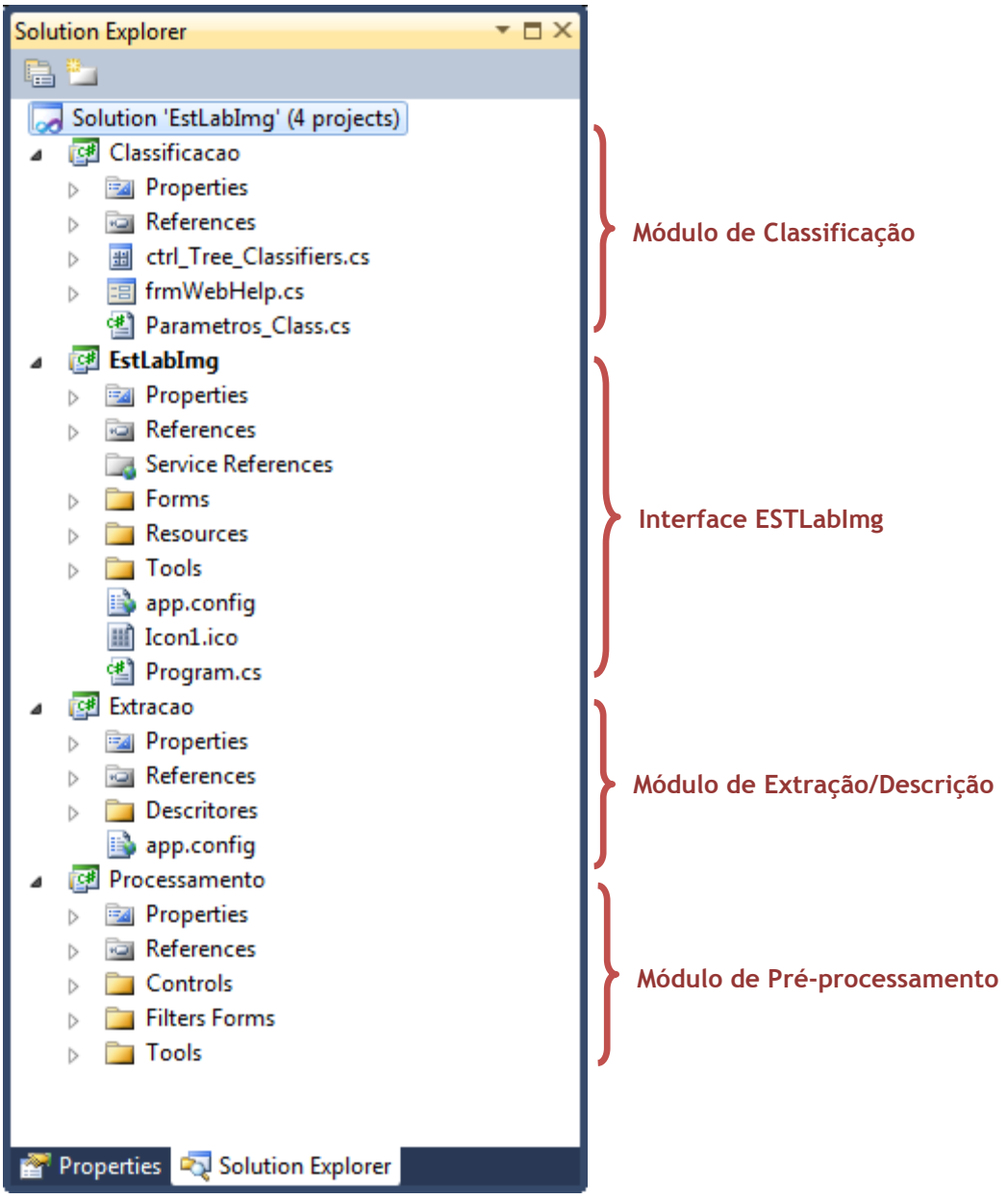


Figura 4.8 - Solução *Visual Studio* com implementação de 4 projetos

O projeto ESTLabImg é o projeto principal, sendo os restantes do tipo *class library*, ou seja, não geram um executável, mas sim uma *dll*. As três *dll* geradas são referenciadas no projeto ESTLabImg, podendo desta forma o projeto principal aceder às funcionalidades dessas *dll* (módulos).

A aplicação final, a ESTLabImg, é apresentada na Figura 4.9, onde se podem visualizar as seguintes funcionalidades:

- Opção de analisar apenas uma imagem;
- Manipulação da imagem, através da implementação de quatro métodos de *zoom*;
- Opção de analisar um *dataset* de imagens;

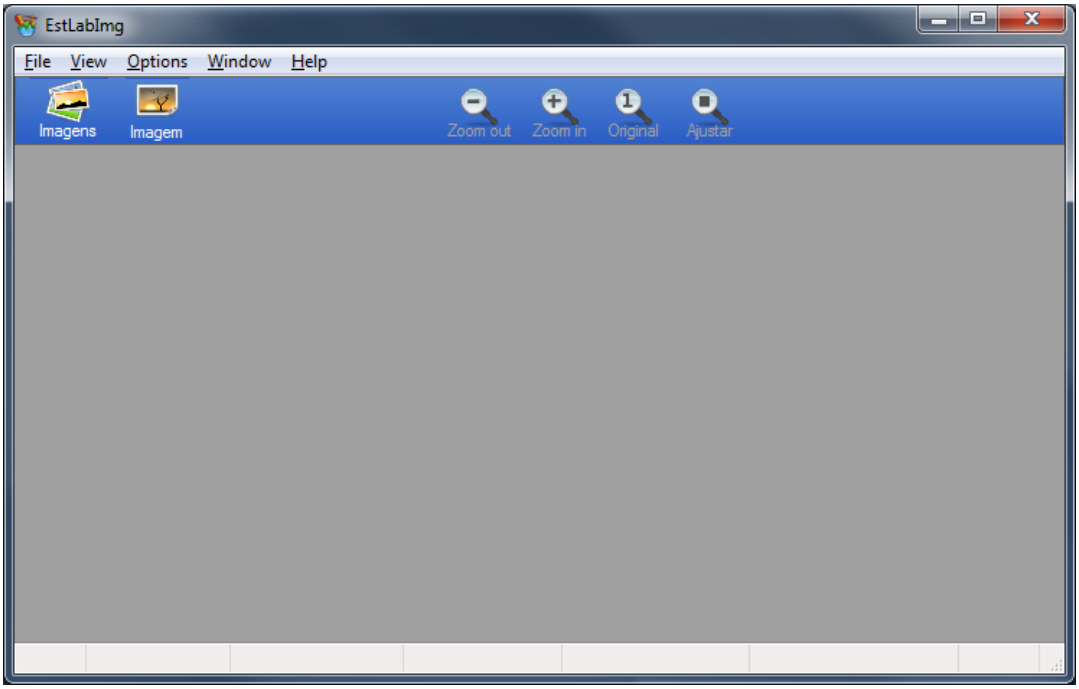


Figura 4.9 – Interface ESTLabImg

Caso se pretenda analisar apenas uma imagem e efetuar a extração das suas características deve-se escolher a opção "Imagem" da ESTLabImg. Na Figura 4.10 é possível visualizar uma imagem sem processamento e o respetivo histograma da imagem que se encontra selecionada, na segunda imagem foi aplicado um filtro (neste caso o *Canny edge detector*) e a terceira imagem mostra a aplicação do descritor SIFT à imagem filtrada.

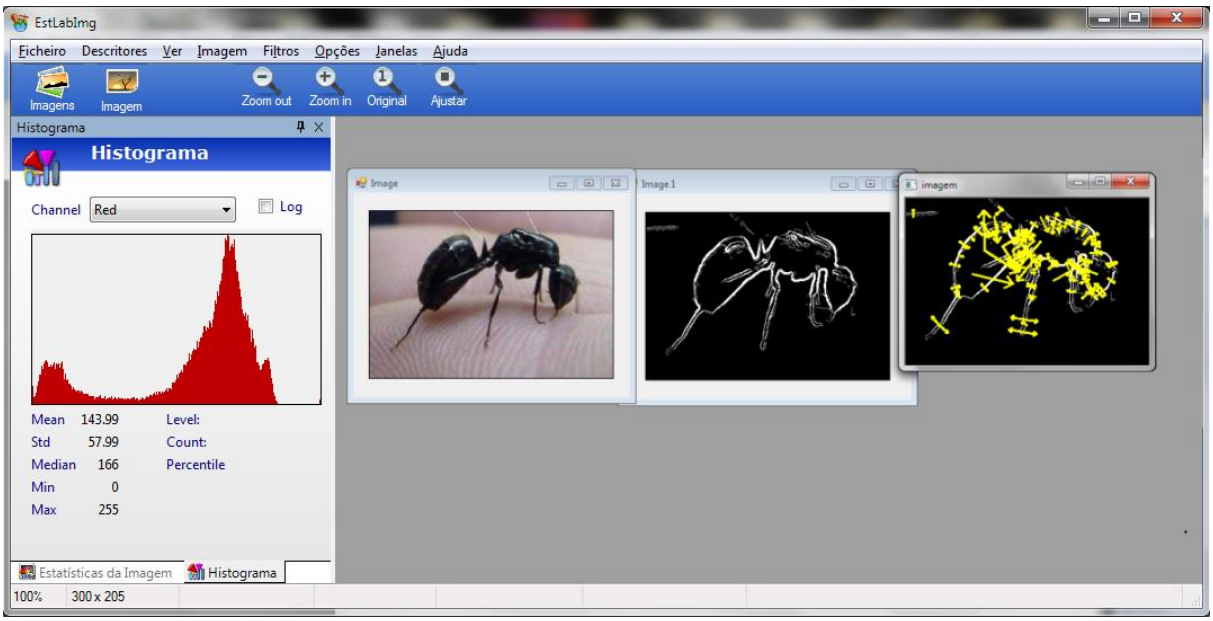


Figura 4.10 - Interface desenvolvida aplicada a uma imagem

No caso de se efetuar o processamento a um *dataset* de imagens deve-se selecionar a opção "Imagens" da ESTLabImg. Esta opção permite a implementação das principais etapas do processamento de imagem, como se pode verificar na imagem da Figura 4.11.

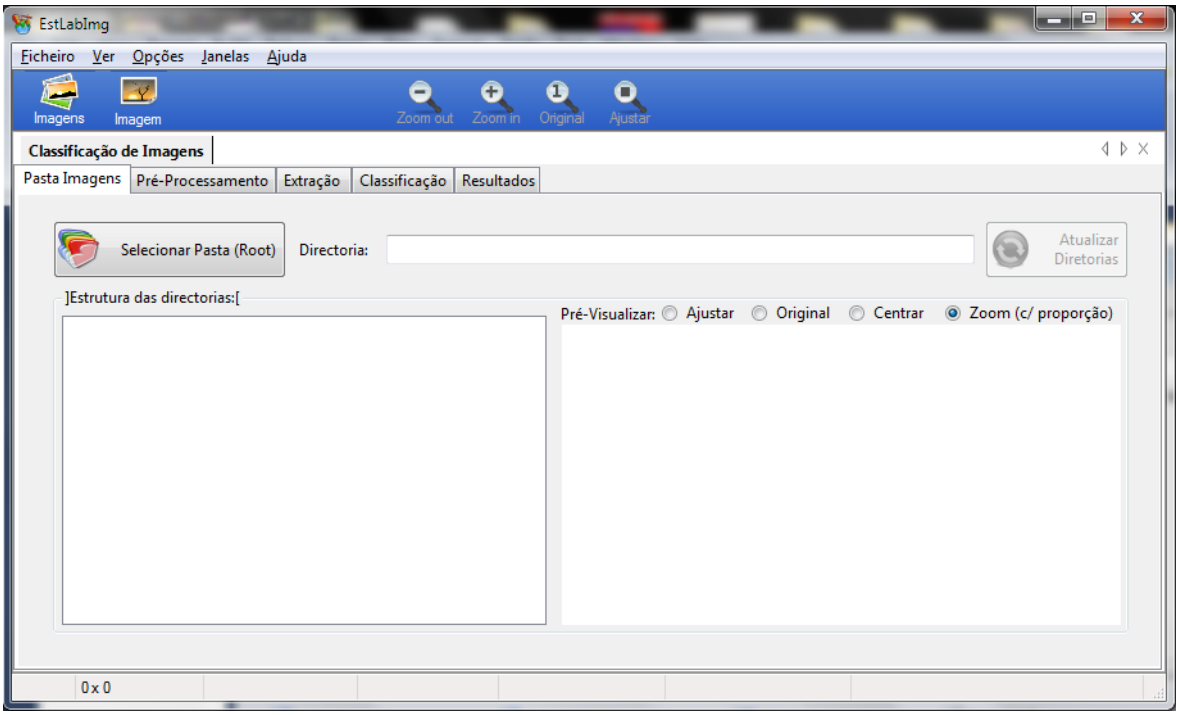


Figura 4.11 - Interface desenvolvida aplicada a um *dataset* de imagens

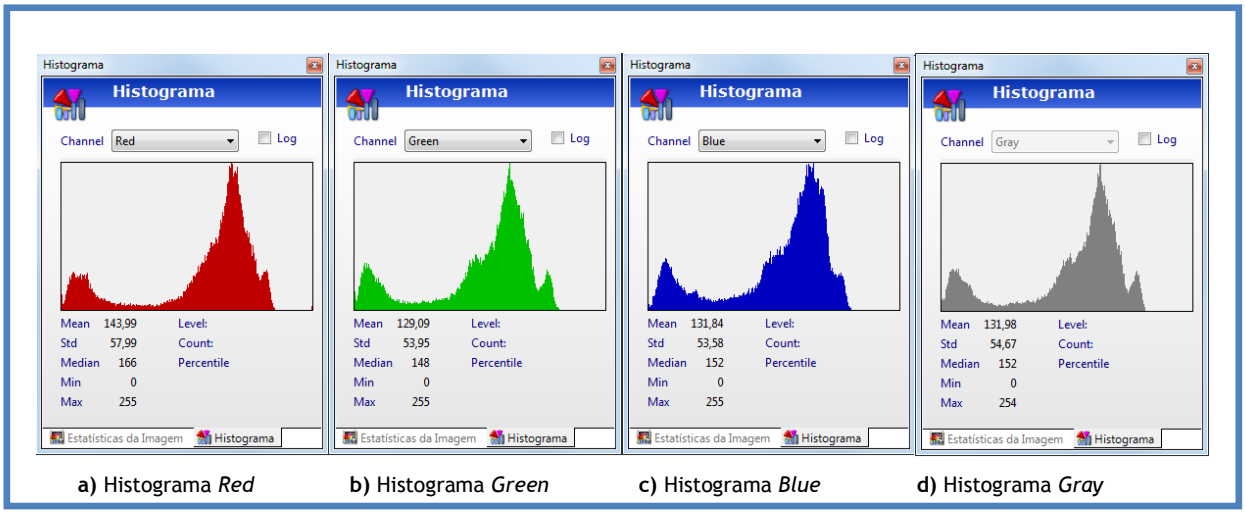
A implementação destes dois métodos de análise de imagem é desenvolvida nos subcapítulos que se seguem.

4.6. Método de análise aplicado a uma imagem

A ESTLabImg permite efetuar o pré-processamento de apenas uma imagem e a extração das suas características através da aplicação de um descritor.

Neste método ao abrir a imagem selecionada é apresentado o histograma RGB (*red, green, blue*) dessa imagem, caso seja a cores. Se a imagem não for a cores aparece o histograma *gray*, como se pode verificar através da Figura 4.12. Um histograma RGB representa a destruição das três cores numa imagem, permitindo a análise destas cores nessa imagem.

O histograma da imagem digital é uma ferramenta útil na etapa de pré-processamento, pois fornece uma visão estatística sobre a distribuição dos *pixels*. A implementação da funcionalidade histograma, nesta aplicação, foi baseada na biblioteca AForge.NET.



a) Histograma *Red* **b)** Histograma *Green* **c)** Histograma *Blue* **d)** Histograma *Gray*

Figura 4.12 - Histograma RGB

Como complemento da análise da imagem foi também implementada a funcionalidade de estatísticas da imagem. Através destas estatísticas é possível identificar o número de total de pixéis, o número de pixéis sem cor preta, o valor máximo, mínimo, médio para cada cor no espaço de cores RGB, entre outros, tal como se pode visualizar na Figura 4.13.

Tal como a funcionalidade histograma, as estatísticas da imagem foram implementadas com base na biblioteca AForge.NET.

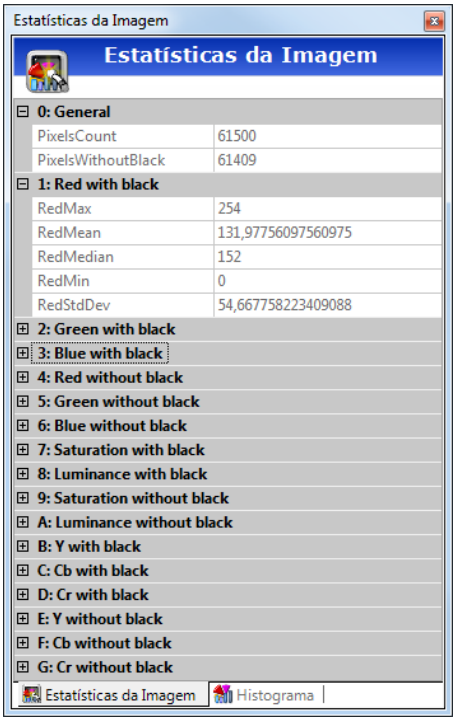


Figura 4.13 - Estatísticas da imagem

Sempre que se abre uma imagem tornam-se visíveis novos menus, o menu descritores, o menu imagem e o menu filtros. Através do menu imagem podem ser efetuadas operações sobre a mesma, tais como o *zoom*, *flip*, *mirror*, entre outras. O menu filtros permite a aplicação de um ou mais filtros à imagem, de forma sequencial. Na Figura 4.14 é apresentada a seleção do filtro *Canny edge detector*.

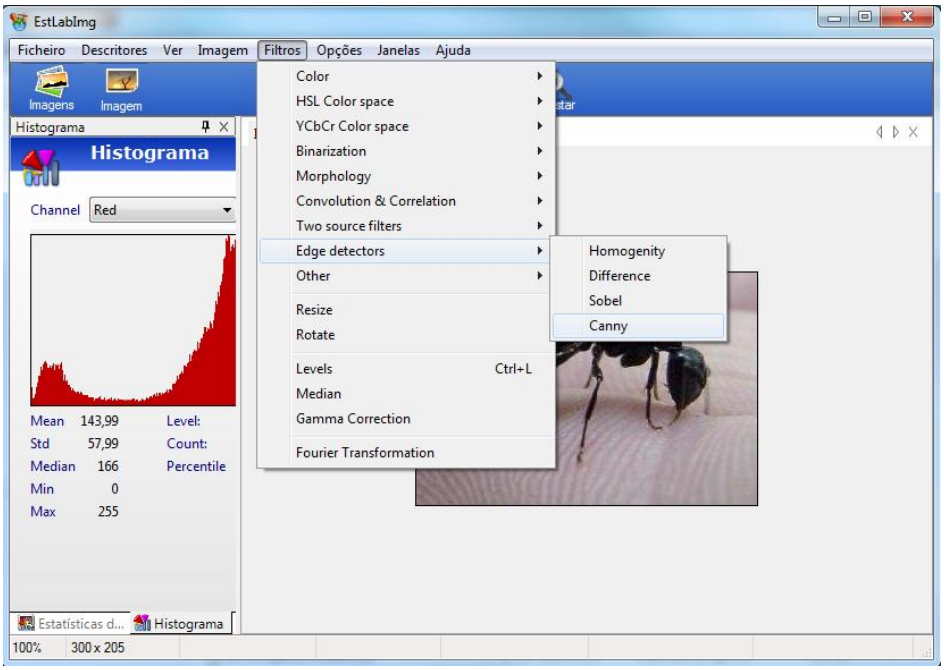


Figura 4.14 - Aplicação do filtro *Canny edge detector*

Através do menu descritores é possível selecionar o descritor que irá extrair as características da imagem, criando uma nova imagem com a identificação das características extraídas. Esta extração pode ser efetuada na imagem original ou na imagem já filtrada. Na Figura 4.15 é possível verificar a aplicação do descritor SURF à imagem e a identificação das suas áreas de interesse.

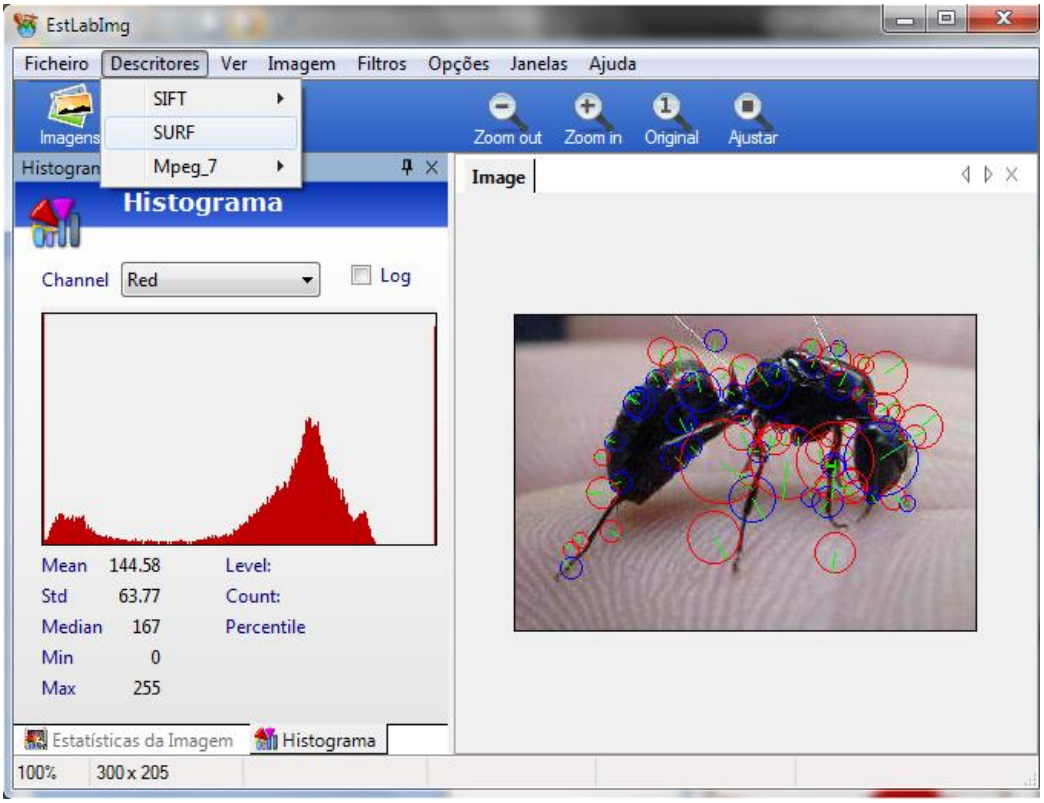


Figura 4.15 - Aplicação do descritor SURF

4.7. Método de análise aplicado a um *dataset* de imagens

Através da ESTLabImg é possível efetuar o processamento de um *dataset*, ou conjunto, de imagens. O *dataset* deverá estar organizado por pastas, onde cada pasta representa uma classe de imagens, como se pode verificar através da estrutura das diretorias representada na Figura 4.16.

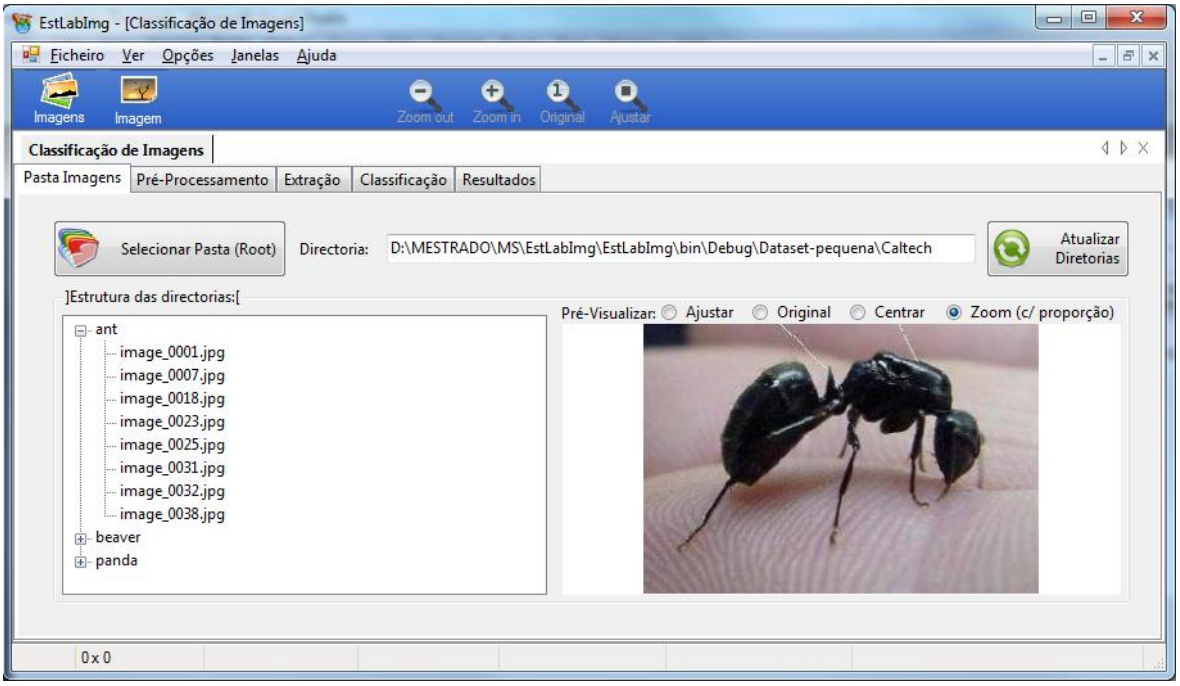


Figura 4.16 - Visualização do *dataset* de imagens em pastas

Como referido anteriormente a aplicação final baseia-se na implementação de três módulos: o de pré-processamento, o de extração/descrição e o de classificação. Nas sessões que se seguem é apresentada a implementação de cada um desses módulos.

4.7.1. Módulo de pré-processamento

O desenvolvimento do módulo de pré-processamento possibilita a implementação de filtros que podem ser aplicados na imagem a processar. A utilização de filtros é uma técnica de processamento de imagem que modifica e/ou melhora a informação contida na imagem, realçando ou removendo características dessa imagem. Nesta interface a implementação dos filtros teve por base a *Framework* AForge.NET.

A AForge.NET é uma *Framework C#* projetada para o desenvolvimento de aplicações que utilizam Inteligência Artificial e Visão Computacional, nomeadamente: processamento de imagem, redes neuronais, algoritmos genéricos, *machine learning*, etc. (AForge, n.d.). É um *software* livre e é constituída por uma vasto número de bibliotecas, entre as quais a AForge.Imaging que contém diferentes rotinas de processamento de imagem, destinadas à aplicação de vários filtros (AForge, n.d.). Os filtros implementados por esta biblioteca podem ser consultados no Anexo 1.

A Figura 4.17 apresenta o módulo de pré-processamento implementado, onde se pode visualizar que foram selecionados e aplicados dois filtros ao *dataset* de imagens, neste caso o filtro *grayscale* e o filtro *sobel*. Através do *status* é possível acompanhar o pré-processamento.

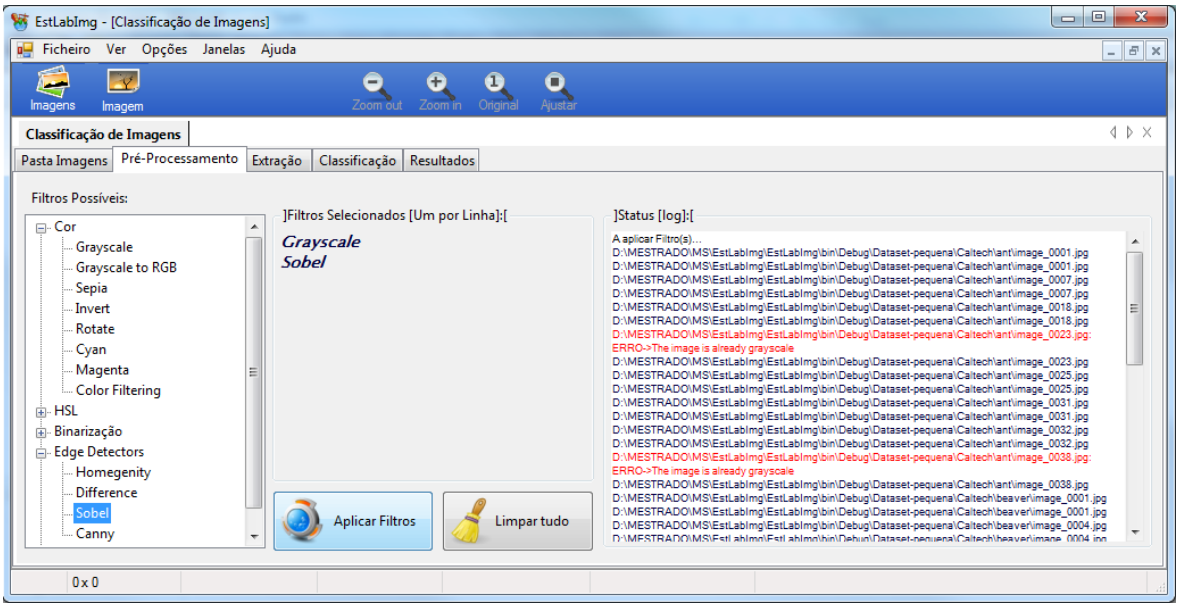


Figura 4.17 - Módulo de pré-processamento

4.7.2. Módulo de extração/descrição

Neste módulo foi implementado o descritor SIFT do qual Lowe (1999) é o autor, como referido no capítulo dois. O código implementado neste módulo é baseado no de Hess (2010), desenvolvido em Visual C++, usando a livraria *open-source* OpenCV.

O OpenCV (2011) (*Open Source Computer Vision Library*) é uma biblioteca multiplataforma originalmente desenvolvida pela Intel em 2000. É uma biblioteca livre ao uso académico e comercial e destina-se ao desenvolvimento de aplicações na área de Visão Computacional. O OpenCV possui módulos de processamento de imagens e vídeo I/O, estrutura de dados, álgebra linear, GUI (Interface Gráfica do Utilizador) entre outros. Esta biblioteca foi desenvolvida nas linguagens de programação C/C++ (Intel, 2001).

Foi também implementado o descritor SURF, igualmente referenciado no capítulo dois. Este descritor foi desenvolvido na linguagem de programação C#, com base na implementação OPENSURF de Evans (2009).

Através da Figura 4.18 é possível visualizar o módulo de extração/descrição, onde se seleciona o descritor pretendido. Esta extração pode ser aplicada ao *dataset* original, ou às imagens onde foi efetuado o pré-processamento. Posteriormente será aplicado o descritor com o objetivo de extrair as características das imagens.

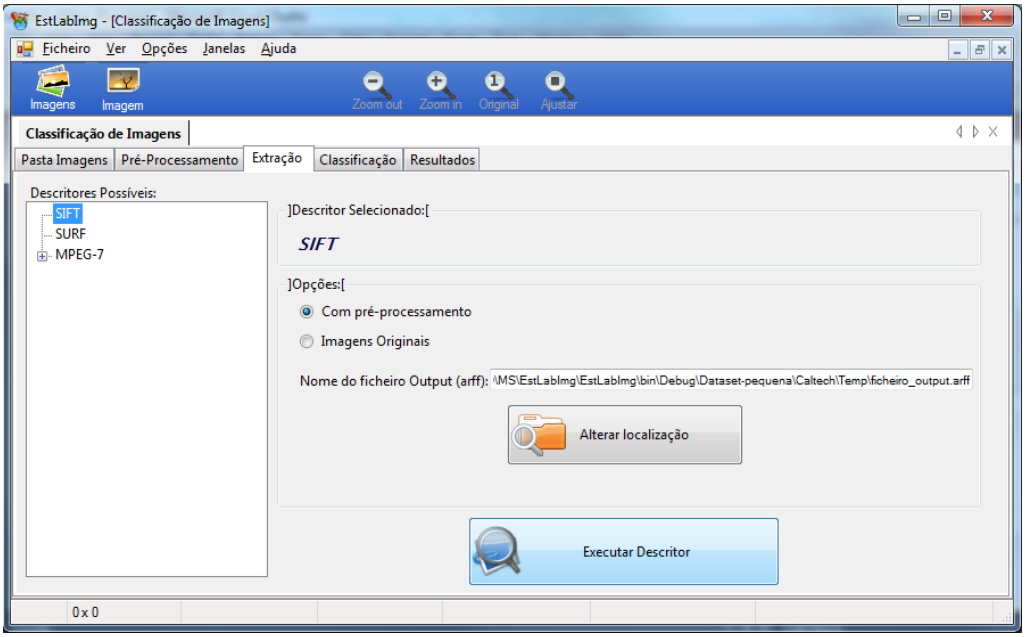


Figura 4.18 - Módulo de extração/descrição

O resultado da aplicação de um descritor às imagens é o conjunto de características obtidas expressas em valores numéricos no formato *arff*, como esquematizado na Figura 4.19. O formato *arff* pode ser consultado no Anexo 2.

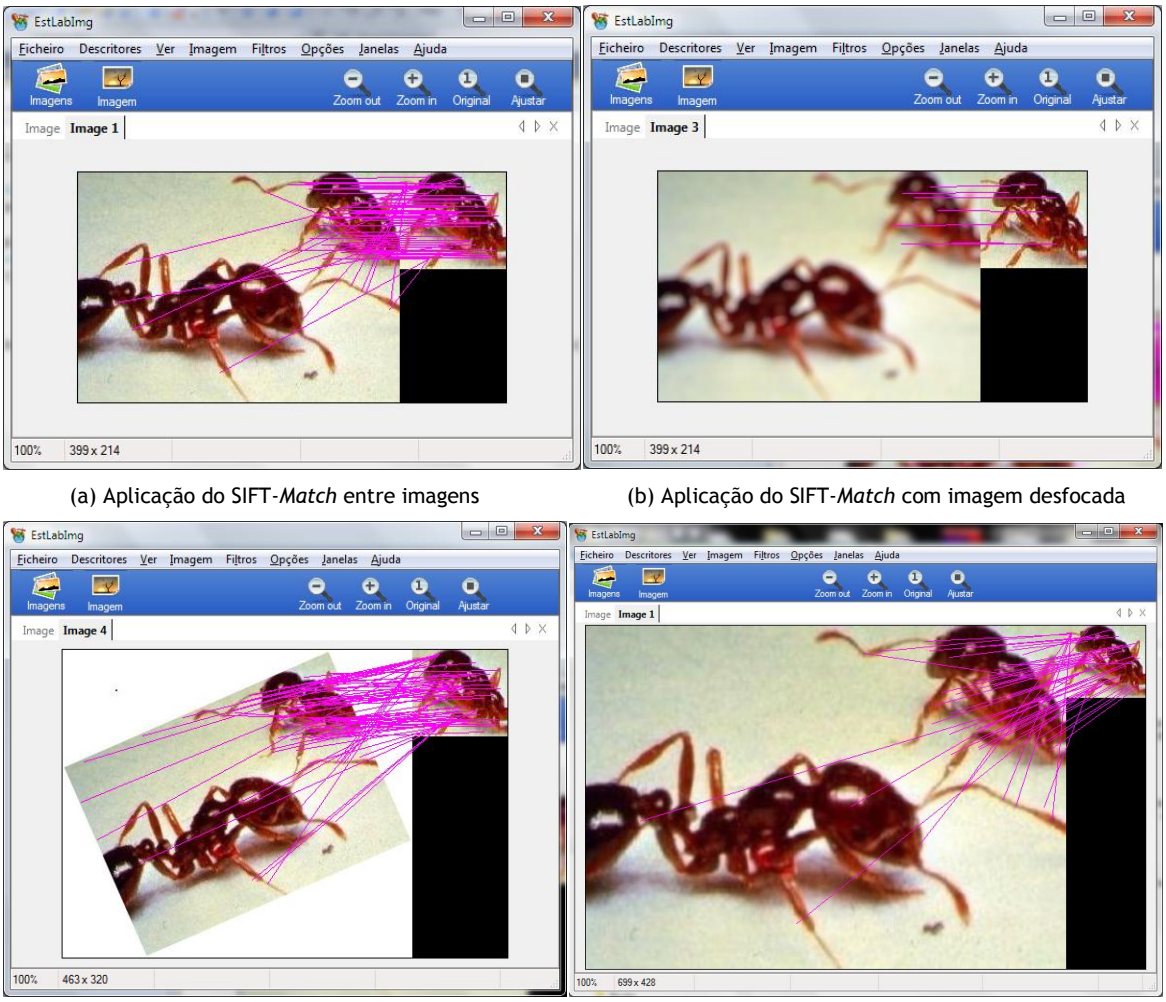
Figura 4.19 - Resultado da aplicação de um descritor

Posteriormente foi também implementada uma funcionalidade do descritor SIFT, o SIFT-*Match*, cujo autor é Lowe (1999), no entanto a implementação neste estudo baseia-se em Hess (2010) e foi desenvolvida na linguagem C++. Na implementação desta funcionalidade são necessárias duas imagens, onde o SIFT-*Match* obtém o vetor de características de ambas as imagens, obtendo correspondência entre elas. A ideia de *match* é extrair as características das duas imagens, e procurar os pontos correspondentes em cada uma.

Através da Figura 4.20 é possível visualizar uma aplicação do SIFT-*Match*, implementado neste estudo em quatro situações diferentes. Na primeira imagem, Figura 4.20 (a), é apresentada a aplicação do SIFT-*Match* a duas imagens, cujo objetivo é procurar as correspondências entre ambas, na segunda imagem, Figura 4.20 (b), é aplicado o SIFT-*Match* às mesmas imagens, no entanto a imagem da esquerda encontra-se desfocada. Nesta aplicação, apesar de não serem identificadas o mesmo número de características encontradas quando aplicado à imagem original, o SIFT-*Match* ainda encontra correspondências mesmo com a imagem desfocadas. Na imagem da Figura 4.20 (c) é apresentada a aplicação do SIFT-*Match* às mesmas imagens, encontrando-se a imagem da esquerda com rotação, onde se verifica que é encontrado um número de características correspondentes significativo. O mesmo acontece com a aplicação

efetuada na Figura 4.20 (d), onde a imagem da esquerda sofreu uma alteração da escala e mesmo assim através do SIFT-*Match* foi encontrado um número de características correspondentes significativo.

Através da aplicação do SIFT-*Match* nestas situações verifica-se que o mesmo é invariante à rotação e a escala, tal como Lowe (2004) tinha referido. Segundo Lowe (2004), o SIFT-*Match* é invariante à escala, rotação, ruído, iluminação e outras pequenas alterações (Lowe, 2004).



(a) Aplicação do SIFT-*Match* entre imagens

(b) Aplicação do SIFT-*Match* com imagem desfocada

(c) Aplicação do SIFT-*Match* com rotação da imagem

(d) Aplicação do SIFT-*Match* com alteração da escala

Figura 4.20 - Extração de características através do SIFT-*Match*

4.7.3. Módulo de classificação

O módulo de classificação foi implementado com recurso ao *software* de *data mining* Weka. A interligação entre este *software*, desenvolvido em Java, e o módulo de classificação, desenvolvido C#, foi possível através do conversor IKVM.

Comando que transforma o weka.jar numa livraria *dll*, de forma a ser referenciado no *Visual Studio:*

```
ikvmc -target:library weka.jar
```

Porção de código demonstrativo do acesso ao *software* Weka e ao Java:

```
/// <summary>
/// Executa o Cross Validation x Folds
/// </summary>
void Executa_Classificação_CrossVal(int Folds)
{
    try
    {
        //Acesso ao WEKA e ao FileReader do Java
        weka.core.Instances insts = new weka.core.Instances(new java.io.FileReader(Ficheiro_arff_Train));
        insts.setClassIndex(insts.numAttributes() - 1);

        AbstractClassifier cls = Classificador_from_string(tb_Classifier.Text);

        (...)
```

Como referido, do resultado do módulo de extração obtém-se um ficheiro *arff* com as características das imagens, ao qual será aplicado um classificador e gerado um modelo de treino, como esquematizado na Figura 4.21.

Figura 4.21 - Aplicação de um classificador

O modelo de treino avalia o ficheiro de teste mediante as seguintes opções:

- **Usar mesmo ficheiro como teste** – esta opção usa como teste o mesmo ficheiro criado para desenvolver o modelo de treino.
- **Usar ficheiro de teste** – esta opção pressupõe a existência de um ficheiro de teste com dados não treinados.
- ***Cross-validation*** – os dados são divididos aleatoriamente em x partes, onde uma dessas x partes será utilizada para testar o modelo. As restantes partes x-1 serão utilizadas como treino. Este processo é repetido x vezes (*folds*) até todas as partes serem treinadas, exatamente 1 vez, e testadas. O resultado final é a média do resultado de cada treino.
- **Dividir em percentagem** – escolhe a percentagem de dados do ficheiro que é utilizada para treino, ficando os restantes para ficheiro de teste.

O classificador pode ser selecionado entre uma lista de classificadores disponíveis, aparecendo as suas opções, por defeito, que podem ser alteradas. A implementação do módulo de classificação é apresentada na Figura 4.22.

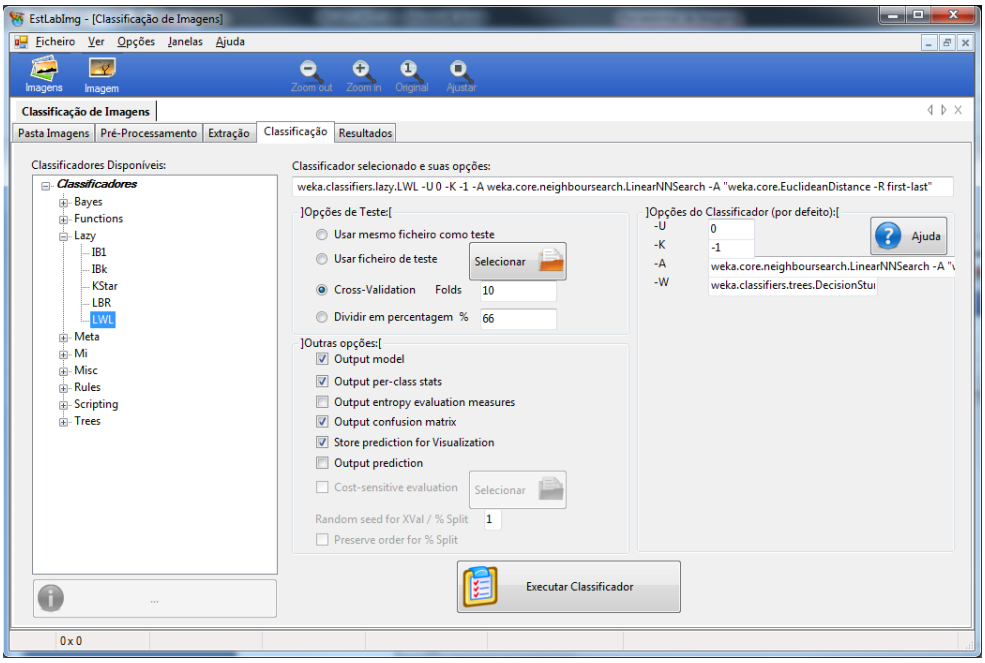


Figura 4.22 - Módulo de classificação

Toda a estrutura do classificador selecionado pode ser consultada através do botão "Ajuda", existente no módulo de classificação. A informação resultante surge sobre forma de *webpage*, como se pode verificar através da Figura 4.23.

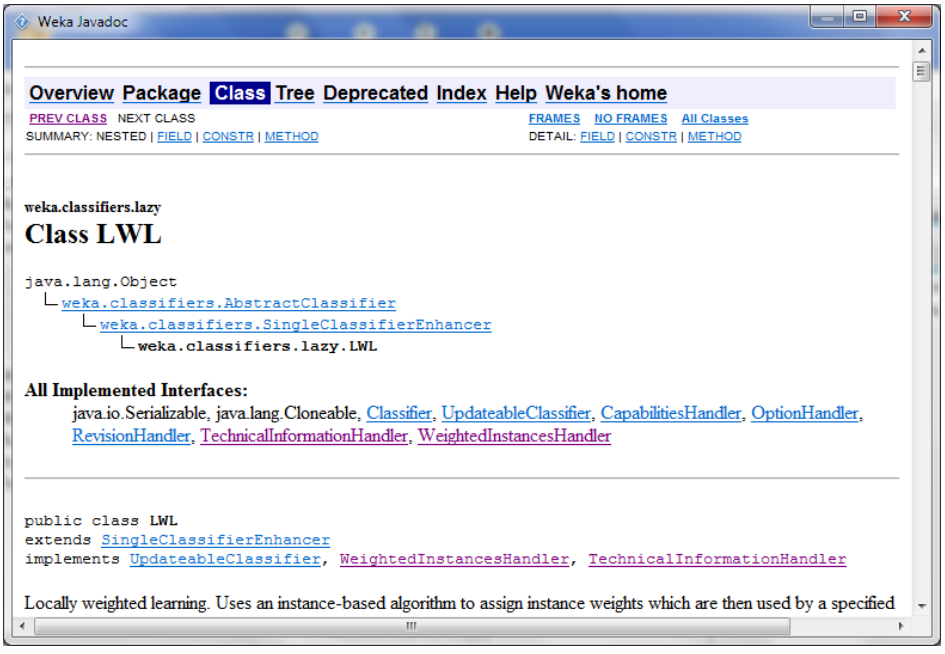


Figura 4.23 - Página com informação sobre o classificador selecionado

Esta página de informação corresponde ao Javadoc relativo ao classificador em questão, encontrando-se na instalação do *software* Weka. Desta forma esta opção só é possível se o *software* Weka se encontrar instalado.

4.7.4. Resultados da classificação

Depois de aplicado um classificador sobre o ficheiro com as características extraídas das imagens é obtido um resultado dessa classificação. Esse resultado aparece num novo separador sendo possível guardar esse resultado em formato *pdf*, *excel* e *word*. A interpretação dos resultados obtidos será detalhada no capítulo 5.

No processo de classificação, a geração de um modelo de treino, dependendo do ficheiro de treino, pode ter um tempo de computação elevado. De forma a evitar esta situação é possível guardar o modelo de treino gerado na classificação para futuras utilizações.

A Figura 4.24 apresenta o separador resultados, onde se pode verificar o resultado de uma classificação.

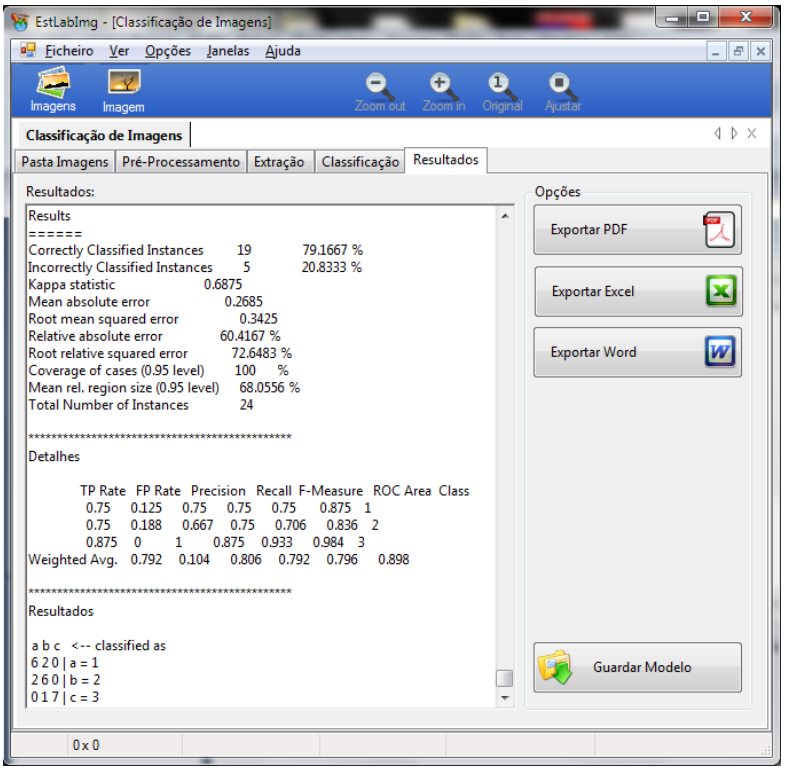


Figura 4.24 - Separador de apresentação dos resultados

4.8. Conclusão

Foram implementados métodos para análise das imagens. Um desses métodos efetua a análise de apenas uma imagem, ao passo que o outro método possibilita a análise de um conjunto de imagens.

Através da análise de apenas uma imagem podem ser aplicados um ou mais filtros a essa imagem de forma sequencial. Posteriormente, é efetuada a extração das características dessa imagem, de forma a selecionar as suas áreas de interesse, através da aplicação de um descritor. Esta extração pode ser aplicada à imagem original ou à imagem filtrada.

A análise de um conjunto de imagens, divididas por classes (pastas), é realizada através da aplicação de três módulos: o módulo de pré-processamento, o módulo de extração/descrição e o módulo de classificação. No módulo de pré-processamento são aplicados sobre as imagens um ou

mais filtros, no módulo de extração/descrição são extraídas as características das imagens, através da aplicação de um descritor e no módulo de classificação é aplicado sobre as características extraídas um classificador. O classificador determina e atribui a classe que melhor se adapta às características.

A interface gráfica desenvolvida neste estudo implementa os métodos acima referidos, possibilitando o processamento de imagens. A aplicação final integra quatro projetos, em que um é a interface principal, a ESTLabImg, é os outros três os módulos já referidos.

A ESTLabImg foi desenvolvida na linguagem C#, tal como o módulo de pré-processamento. Este módulo foi implementado com recurso à *Framework* AForge.NET, nomeadamente à biblioteca AForge.Imaging, responsável pela implementação de vários filtros.

O módulo de extração/descrição foi desenvolvido em C# para implementação do descritor SURF e na linguagem Visual C++ para implementação do descritor SIFT. Na implementação do descritor SIFT foi utilizada a livraria OpenCV e a implementação do descritor SURF teve por base a livraria OPENSURF. Foi também implementada a funcionalidade SIFT –*Match* do descritor SIFT, igualmente desenvolvida em C++.

Por fim o módulo de classificação foi desenvolvido com recurso ao *software data mining* Weka, uma vez que possibilitava a implementação dos classificadores. O *software* Weka foi desenvolvido na linguagem Java sendo por isso necessário a implementação de um conversor entre Java e .NET para que pudesse ser utilizado nesta aplicação. Neste estudo foi utilizado para este fim a *bridge*, ou o conversor, IKVM.

A interface gráfica, ESTLabImg, implementa as principais etapas do processamento de imagem, possibilitando que um determinando conjunto de imagens seja classificado e dessa classificação se obtenha um resultado final.

Capítulo 5- Experiência e interpretação dos resultados

5.1. Introdução

A ESTLabImg implementa as principais etapas do processamento de imagens, desde a aplicação de filtros às imagens (pré-processamento), a extração das características identificativas das imagens (extração/descrição), a classificação dessas características e por fim a apresentação dos resultados da classificação. Desta forma é efetuado, ao longo deste capítulo, um teste à interface ESTLabImg, sendo apresentado o processamento de cada uma das etapas. É igualmente realizada uma interpretação dos resultados obtidos.

As imagens utilizadas para a realização da experiência foram retiradas do *dataset Caltech-101,* disponível *online* (Fei-Fei et al., 2004). Este *dataset* contém 101 categorias de imagens, onde cada categoria possui cerca de 40 a 800 imagens. Na elaboração desta experiência foram escolhidas três destas categorias, onde cada categoria possui imagens a cores, outras em imagens em tons de cinzento (*grayscale*) e imagens desenhadas. A resolução média das imagens é de 300x240. As categorias escolhidas foram:

- **a categoria *ant*** – possui 42 imagens de formigas,
- **a categoria *beaver*** - possui 46 imagens de castores e,
- **a categoria *panda*** - possui 38 imagens de pandas.

5.2. Teste da interface ESTLabImg

Depois de caracterizado o *dataset* de imagens utilizado nesta experiência, fez-se o seu processamento através da interface ESTLabImg.

Inicialmente foi selecionado o *dataset* pretendido. Este *dataset* é dividido por diretorias em que cada diretoria representa uma classe de imagens (ver Figura 5.1).

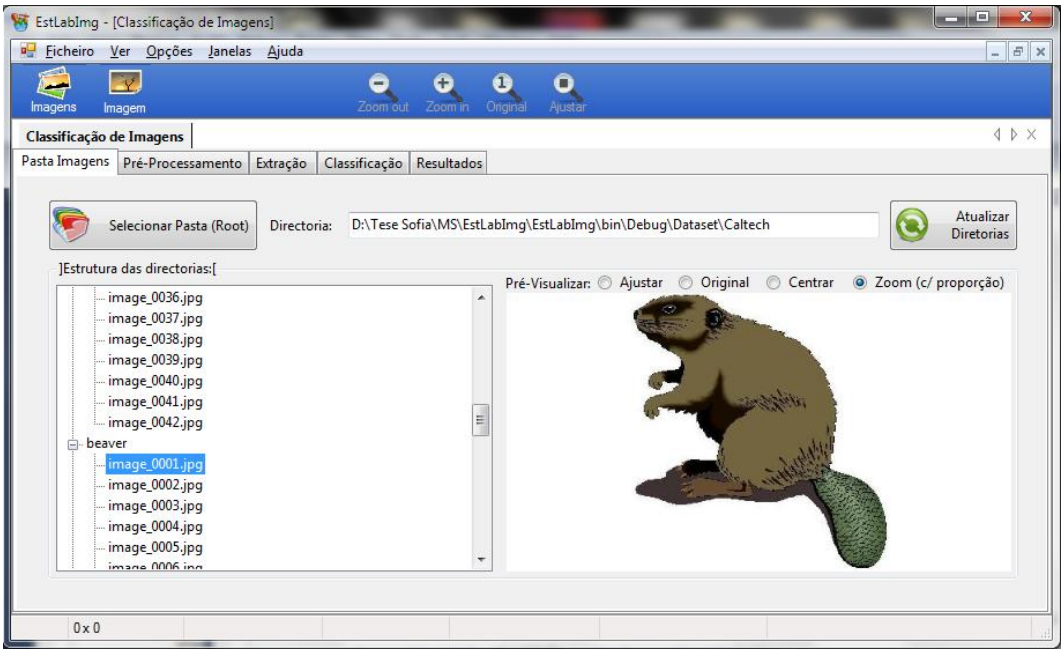


Figura 5.1 – Escolha do *dataset* de imagens para processamento

Depois de selecionadas, as diretorias das imagens, procede-se ao pré-processamento das mesmas, através da aplicação de filtros. Para esta experiência foram selecionados dois filtros aleatoriamente, o filtro *grayscale* e o filtro *difference*. Na Figura 5.2 é possível visualizar a seleção dos filtros e o processo de aplicação desses filtros (através do *status*).

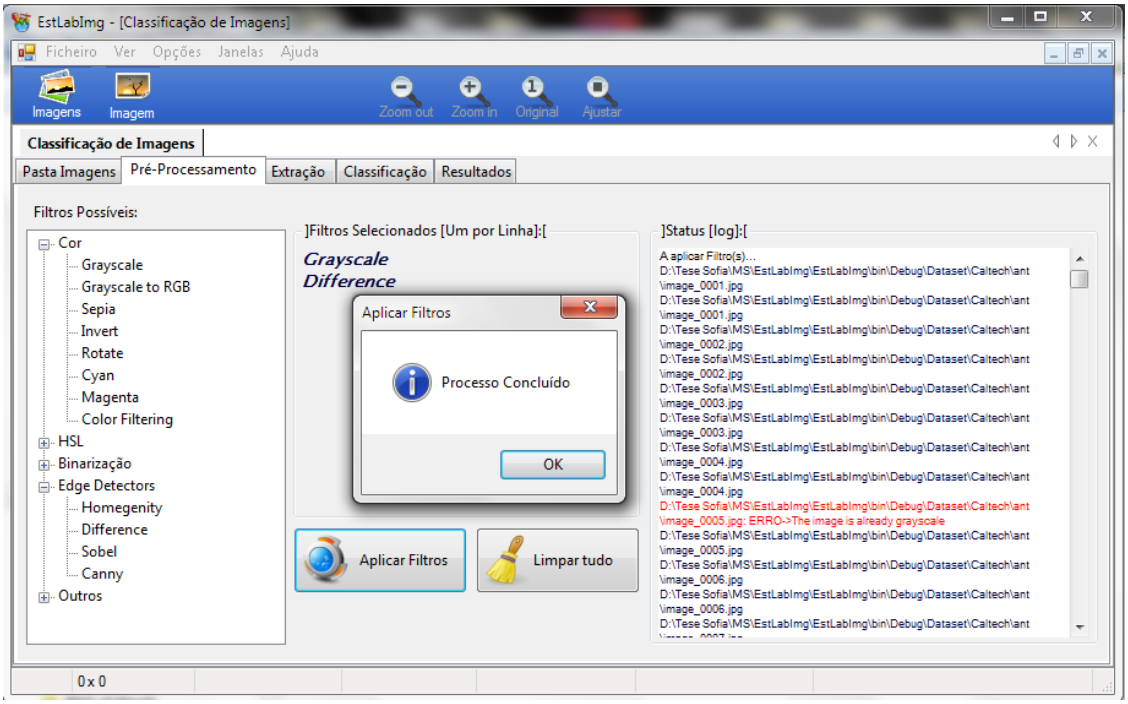


Figura 5.2 - Aplicação de filtros no módulo de pré-processamento

O resultado da aplicação dos filtros ao *dataset* de imagens é a duplicação da estrutura de diretorias, já com os filtros aplicados. Esta duplicação possibilita a consulta das imagens filtradas à posterior.

De forma a extrair as características significativas das imagens foi aplicado o descritor SIFT a cada uma das imagens sobre as quais tinham sido aplicados os filtros. Na Figura 5.3 é apresentada a seleção do descritor SIFT e o seu processo de execução.

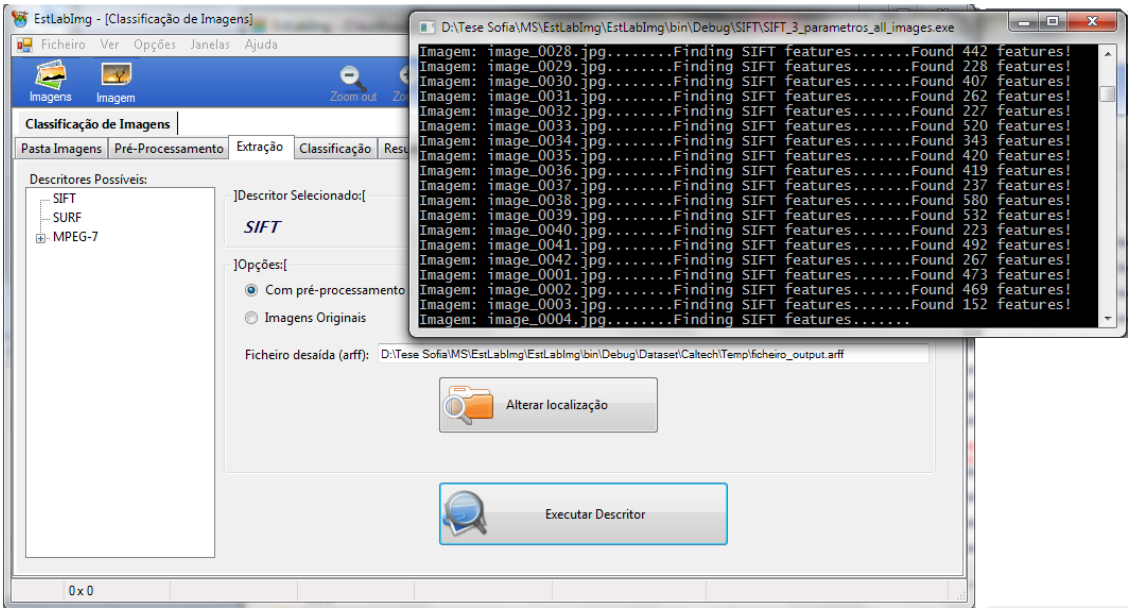


Figura 5.3 - Aplicação do descritor SIFT no módulo de extração/descrição

Do processo de extração é gerado um ficheiro *arff* que irá servir de ficheiro de treino ao processo de classificação. De forma a classificar as características extraídas foi aplicado o classificador SMO (*support vector machine* - SVM) (ver Figura 5.4).

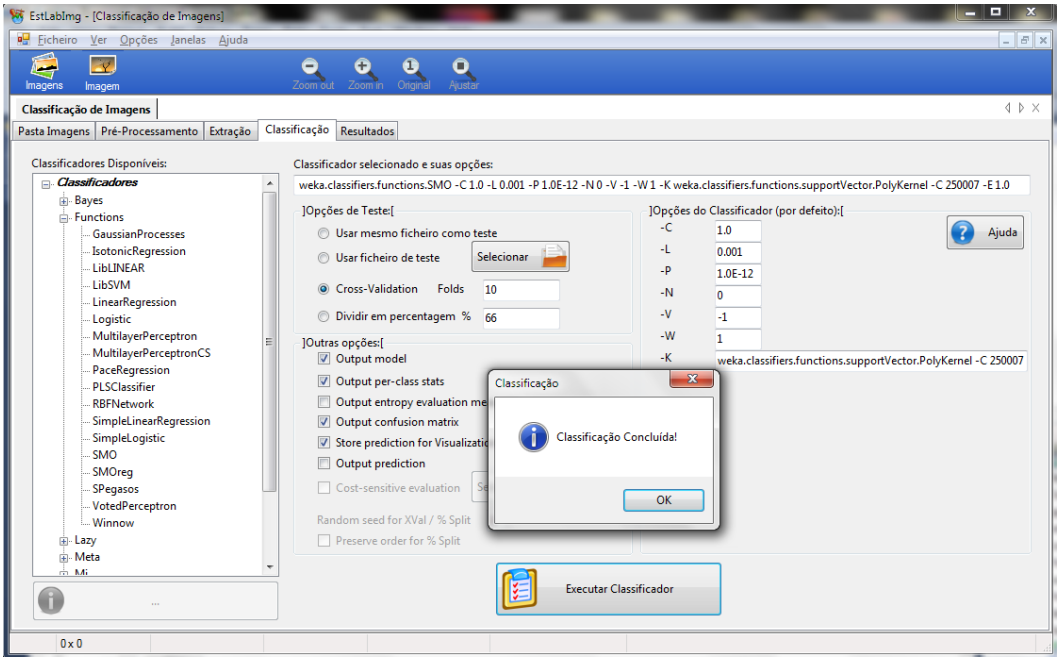


Figura 5.4 - Aplicação do classificador SVM no módulo de classificação

A Figura 5.5 apresenta o separador com o resultado da aplicação do classificador SVM às características extraídas das imagens filtradas.

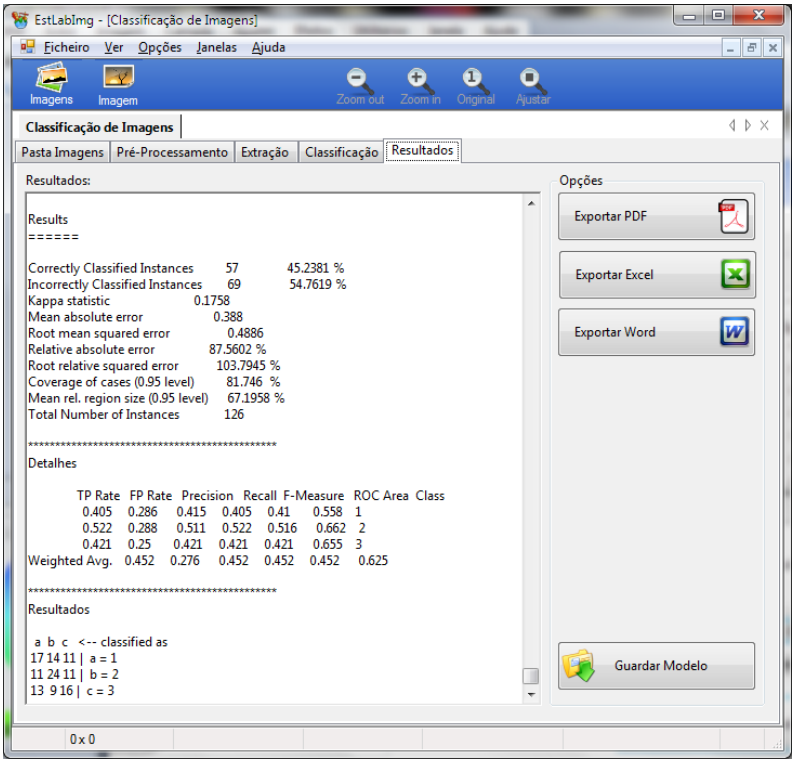


Figura 5.5 - Resultado da aplicação do classificador SVM

5.3. Interpretação dos resultados obtidos

Através dos resultados obtidos é possível avaliar o classificador. Desta avaliação são identificados os seguintes resultados:

- Número de imagens corretamente classificadas *(Correctly classified instances);*
- Número de imagens incorretamente classificadas *(Incorrectly classified instances);*
- Estatísticas *Kappa (Kappa statistic);*
- Número total de imagens classificadas *(Total numbers of instances);*
- Matriz confusão;
    - *TP (True Positive) rate;*
    - *FP (False Positive) rate;*
    - *Precision;*
    - *Recall;*
    - *F-measure*.
- Outros:
    - *Mean absolute error;*
    - *Root mean squared error;*
    - *Relative absolute error;*
    - *Root relative squared error;*
    - *Coverage of cases;*
    - *Mean rel. Region size;*

5.3.1. Matriz confusão

A matriz confusão contém informações relativas a classificações efetuadas através da aplicação de um classificador. O desempenho dos classificadores é frequentemente avaliado através dos dados retirados desta matriz (Kohavi e Provost, 1998).

A matriz confusão de um classificador indica o número de classificações corretas *versus* as previsões efetuadas para cada caso, sobre um conjunto de exemplos T. Nesta matriz as linhas representam os casos reais e as colunas as previsões efetuadas pelo modelo (Kohavi e Provost, 1998). Através da matriz confusão é possível obter informação relativa ao número de imagens corretamente classificadas e incorretamente classificadas, para cada classe. Esta matriz é de AxA, sendo A o número de classes ao qual se aplica o classificador, no caso da experiência efetuada neste estudo existiam três classes de imagens, sendo a matriz confusão de 3x3.

Considerando o resultado obtido na experiência efetuada à ESTLabImg, a matriz confusão obtida é representada na Figura 5.6.

Total linhas	a	b	c		FP	FN
42	17	14	11	a=1	24	25
46	11	24	11	b=2	23	22
38	13	9	16	c=3	22	22
	126	41	47	38		
		Total colunas				

Figura 5.6 - Matriz confusão obtida da experiência à ESTLabImg

Da matriz confusão obtida é possível identificar o número total de imagens corretamente classificadas, neste caso 57, valor este obtido pela soma dos valores da diagonal principal da matriz. Os restantes elementos representam erros de classificação, ou seja, imagens incorretamente classificadas. O FP (falsos positivos) indica o número de imagens incorretamente classificadas na respetiva classe, ao passo que o FN (falsos negativos) indica o número de imagens incorretamente classificadas nas outras classes mas que pertencem à respetiva classe.

Através da matriz confusão é possível obter o rácio *TP rate (rácio verdadeiros positivos)* que relaciona o número de imagens corretamente classificadas da classe x com o número total de imagens classificadas como pertencentes à classe x. Tendo em conta a matriz confusão do exemplo, o *TP rate* da classe 1 seria calculado da seguinte forma: 17/(17+14+11)=0.405.

$$TP\ rate = \frac{imagens\ corretamente\ classificadas\ na\ classe\ x}{total\ imagens\ classificadas\ como\ pertencentes\ à\ classe\ x}$$

O *FP rate* (rácio falsos positivos) relaciona o número total de imagens classificados na classe x, mas que são de outra classe, com a soma de todas as imagens que não pertencem à classe x. Tendo em conta a matriz confusão do exemplo o *TP rate* da classe 1 seria calculado da seguinte forma: (11+13)/(46+38)= 0.286.

$$FP\ rate = \frac{imagens\ classificadas\ numa\ classe\ x, mas\ que\ são\ de\ outras\ classes}{total\ imagens\ que\ não\ pertencem\ à\ classe\ x}$$

A partir da matriz confusão obtém-se também o valor de *precision* (precisão) e do *recall*, que medem a performance do classificador para cada classe, separadamente. Para uma classe x *precision* é o total de imagens corretamente classificadas como x sobre o total de imagens classificadas como x. O *recall* é calculado da mesma forma que o *TP rate*.

$$Precision = \frac{imagens\ corretamente\ classiifcadas\ na\ classe\ x}{total\ imagens\ classificadas\ como\ x}$$

$$Recall = TP\ rate$$

O *F-Measure* combina o *precision* com o *recall* sendo utilizado para comparação entre classificadores.

$$F\text{–}Measure = \frac{2 \times precision \times recall}{precision \times recall}$$

A **matriz confusão de um classificador ideal** é aquela em que todos os elementos fora da diagonal são iguais a zero.

```
 a  b  c  <-- classified as
42  0  0 |  a = 1
 0 46  0 |  b = 2
 0  0 38 |  c = 3
```

Na matriz confusão anterior todas as imagens foram corretamente classificadas, 42 na classe "a", 46 na classe "b" e 38 na classe "c". Nesta situação o classificador não classifica incorretamente nenhuma imagem.

5.4. Detalhes dos resultados da experiência

Tendo em consideração a experiência enunciada na secção 5.2. foram obtidos resultados da aplicação do classificador SMO às imagens filtradas no módulo de pré-processamento. Estes resultados são apresentados na Figura 5.5 onde se pode verificar que:

- Foram classificadas corretamente 57 imagens;
- Foram classificadas incorretamente 69 imagens;
- Através da matriz confusão identificou-se:
    - 17 imagens corretamente classificadas da classe 1, num total de 42 imagens desta classe;
    - 24 imagens corretamente classificadas da classe 2, num total de 46 imagens desta classe;
    - 16 imagens corretamente classificadas da classe 3, num total de 38 imagens desta classe.

Tendo em conta que o modelo de treino apenas classificou corretamente 45,24% das imagens, os resultados não são considerados satisfatórios.

De realçar que os filtros aplicados no módulo de pré-processamento, na experiência, foram selecionados aleatoriamente. De forma a tirar resultados do processamento do *dataset* de imagens sem a aplicação de filtros, efetuou-se um novo processamento do *dataset*, cujo resultado pode ser visualizado na Figura 5.7.

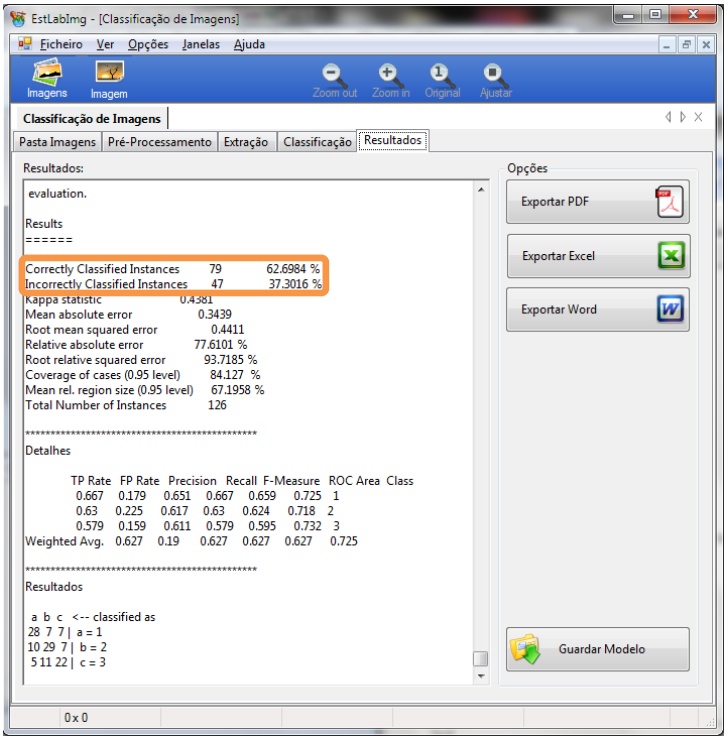


Figura 5.7 - Resultados obtidos sem aplicação de filtros

Sem a aplicação de filtros às imagens processadas verifica-se que o classificador classificou corretamente 62,7% das imagens. Em relação ao processamento com aplicação de filtros, estes resultados foram mais satisfatórios, no entanto ainda longe da solução ótima.

A solução ótima seria aquela em que todas as imagens fossem corretamente classificadas, solução esta representada na Figura 5.8, onde se pode verificar através da matriz confusão que todas as imagens foram corretamente classificadas, em cada classe.

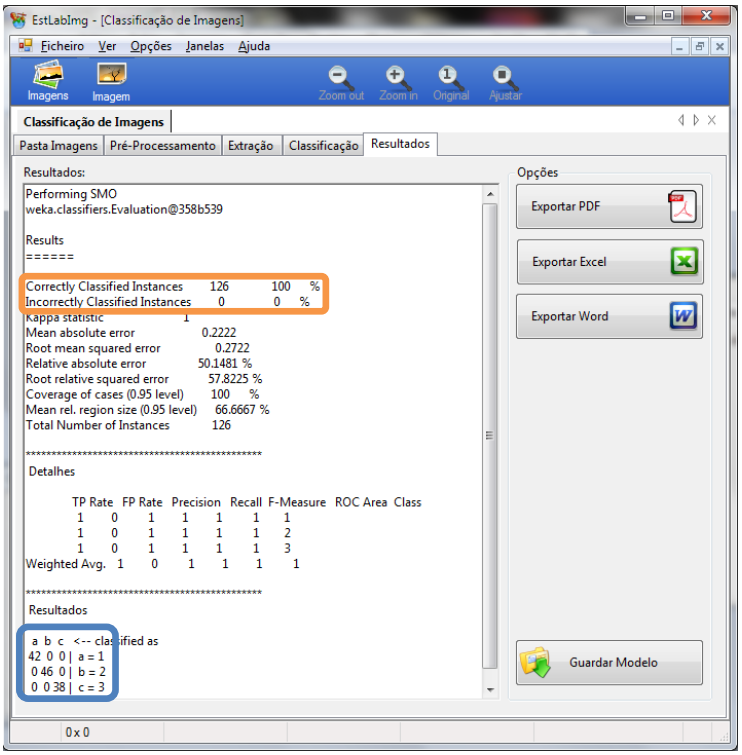


Figura 5.8 - Resultado do processamento com utilização do mesmo ficheiro para teste e treino

O resultado anterior foi alcançado através da seleção da opção de "usar mesmo ficheiro como teste" do módulo de classificação (ver Figura 5.9). Esta opção possibilita que o ficheiro de teste avaliado pelo classificador, seja o mesmo gerado pela extração de características, ou seja, o ficheiro de treino. Como seria de esperar apresenta valores demasiado otimistas, pois está a testar exatamente os mesmos dados treinados.

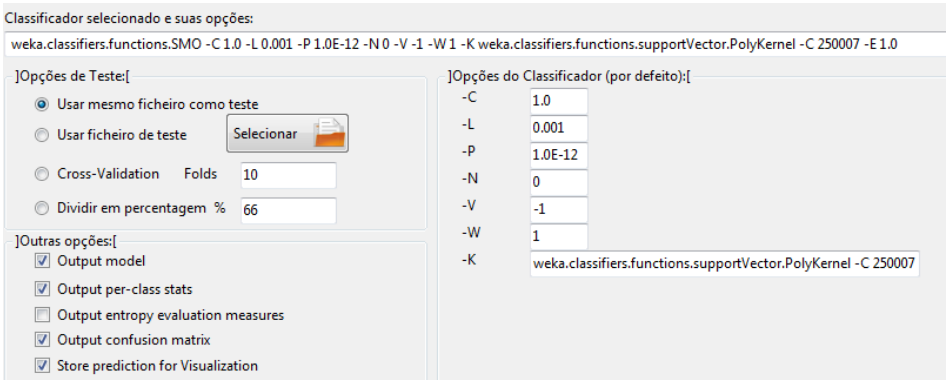


Figura 5.9 - Utilização do mesmo ficheiro para teste e para treino

5.5. Conclusão

A realização da experiência descrita neste capítulo mostra que com a interface ESTLabImg é possível implementar as etapas do processamento de imagens de forma simples, numa única interface.

O *dataset* escolhido para a realização da experiência tinha 126 imagens, separadas por três categorias, ou classes. As imagens apresentavam-se em diferentes formatos, umas a cores, outras a *grayscale* e outras em forma de desenho, tendo o processamento destas imagens demorado 2 minutos e 41 segundos, desde o momento em que foi selecionado o *dataset* até à apresentação dos resultados.

Da análise dos resultados apresentados é possível identificar o número de imagens corretamente classificadas e o número de imagens incorretamente classificadas. A matriz confusão é um dos resultados do qual se tiram mais conclusões, nomeadamente, o número de imagens corretamente classificadas e incorretamente classificadas para cada classe, o número de falsos positivos, o número de falsos negativos, a precisão obtida na classificação, entre outras. Os números de imagens corretamente classificadas são os que se encontram na diagonal principal da matriz, sendo considerada solução ótima da classificação a situação em que todos os valores fora desta diagonal se encontrem a zero.

Verificou-se que o resultado da experiência efetuada não era satisfatório, uma vez que a percentagem de imagens corretamente classificadas era de 45,24%. No entanto, não era objetivo deste estudo a apresentação de uma solução com resultados ótimos, o objetivo era a criação de uma aplicação que permitisse a integração dos passos do processamento de imagem, onde o investigador pudesse escolher os diferentes algoritmos que pretendia investigar em cada um desses passos, procedendo a todo o processamento numa única interface.

Capítulo 6- Usabilidade da interface

6.1. Introdução

Segundo Cooper, et al. (2007, p. 143) os testes de usabilidade determinam se os utilizadores conseguem realizar corretamente as tarefas existentes numa interface. Carvalho (2008) refere que os testes de usabilidade devem ser feitos ao longo do processo de desenvolvimento, mesmo durante a fase de conceção da interface tal como referido por, entre outros, Rubin (1994), Smith e Mayes (1996).

Desta forma, e definida a interface a criar, impunha-se algum tipo de validação da proposta sugerida por parte de possíveis utilizadores. Deste modo, procedeu-se à validação da interface, através da aplicação de um teste, com o objetivo principal de detetar pontos críticos da interface.

O objetivo principal da implementação deste teste prendeu-se com a necessidade de construir uma interface lógica, coerente, interativa, de fácil navegação e que permitisse ao utilizador compreender que o processamento de imagem se divide em diferentes etapas. Desta forma foi pedido a um grupo de cinco participantes que interagisse com a interface, testasse todas as suas funcionalidades e indicasse problemas de compreensão, utilização, navegação entre separadores/*tabs* e apresentasse sugestões de melhoria. Foi também efetuada uma caracterização dos participantes, de forma a analisar o seu nível académico e o seu grau de literacia digital. Esta caracterização permitiria identificar os participantes como possíveis utilizadores.

Normalmente, os resultados destes testes revelam áreas problemáticas, áreas onde os utilizadores revelam dificuldade em compreender as funcionalidades presentes na interface (Cooper, et al., 2007).

Ao longo deste capítulo é efetuada uma análise dos dados recolhidos com a aplicação destes testes.

6.2. Aplicação do teste

O teste aplicado teve por base um protótipo da interface a desenvolver. Este protótipo foi desenvolvido no *Microsoft Visual Studio 2010* e garantia a navegação entre os diferentes separadores/*tabs*, apesar de algumas áreas ou funcionalidades não estarem completamente desenvolvidas ou funcionais, o que não impedia a interação entre participante e interface.

Foi pedido a um conjunto de cinco participantes que explorassem as funcionalidades da interface, através da navegação nos diferentes separadores/*tabs*. Foi também pedido a esses participantes que fossem demonstrando e verbalizando comentários e alterações a efetuar ao protótipo apresentado.

Durante o teste, o acompanhamento foi constante, de modo a fornecer ajuda, caso fosse necessário.

Pretendia-se avaliar a interface relativamente a:

- *Compreensão e facilidade de utilização* – neste parâmetro pretendia-se que o participante interagisse de forma rápida com a interface, compreendendo a navegabilidade de todo o processo. Depois de compreender o funcionamento, pretendia-se avaliar a capacidade de localizar a informação pretendida;
- *Facilidade de memorização* – neste parâmetro pretendia-se que depois de o participante já ter aprendido o funcionamento da interface fosse capaz de se lembrar sem necessidade de efetuar nova aprendizagem;
- *Apresentação da interface* – neste parâmetro pretendia-se avaliar o *layout* da interface, se esta se apresenta de forma lógica e coerente;
- *Navegação entre separadores/tabs* – neste parâmetro pretendia-se descobrir se a forma de navegação entre os diferentes separadores/*tabs* era percetível e estruturada;
- *Registar alterações* – de forma a registar eventuais alterações e/ou problemas, através deste parâmetro o participante poderia efetuar os comentários que considerasse pertinentes.

6.3. Protótipo funcional

Os testes foram desenvolvidos utilizando um protótipo desenvolvido para a realização dos mesmos. Neste protótipo já se encontravam implementadas grande parte das funcionalidades da interface, no entanto nem todas estavam concluídas. Os participantes foram informados que se tratava de um protótipo e que se pretendia avaliar a interação com esse mesmo protótipo.

A realização dos testes foi efetuada num computador portátil, onde se encontrava instalada a aplicação e onde os participantes poderiam interagir com a mesma. Os participantes foram esclarecidos dos objetivos da aplicação do teste, procedendo de seguida à exploração do protótipo e tendo posteriormente realizado um teste onde avaliavam alguns parâmetros, indicados em seguida.

O protótipo realizado nos testes encontra-se disponível no Anexo 3.

6.4. Parâmetros avaliados

A realização de testes de usabilidade teve como principal objetivo perceber como funciona a interface projetada, quando utilizada em situações reais e qual a facilidade de utilização dessa interface. Deste modo, o que se pretendia perceber era se existiam pontos da interface que funcionassem mal, que não fossem percetíveis, que deixassem os utilizadores confusos sem perceberem o objetivo da interface ou partes da mesma.

Para proceder à recolha dos resultados foi criado um formulário no *Google Docs*, onde os participantes depois de conhecerem e interagirem com a interface registavam as suas opiniões relativamente às questões que lhe eram colocadas. Este teste encontra-se disponível no Anexo 4 deste documento.

Para proceder à análise e tratamento dos dados recorreu-se ao Microsoft Excel, onde os dados recolhidos foram sistematizados em tabelas a partir das quais se criaram gráficos adequados à demonstração dos dados.

6.4.1. Caracterização dos participantes

Este teste foi aplicado a um grupo de cinco participantes, caracterizado através de dados recolhidos na primeira parte do teste. Pretendia-se com esta caracterização traçar o perfil geral dos participantes e compreender o seu nível de literacia digital.

O teste aplicado aos participantes pode ser consultado no Anexo 4 deste documento. Os dados recolhidos apresentam-se nas tabelas que se seguem.

Os cinco participantes tinham idade compreendida entre os 32 e 43 anos de idade, sendo a média de idades de 38 anos. A nível de sexo existiram dois participantes do sexo feminino (40 %) e três do sexo masculino (40 %). Relativamente ao nível académico dois dos participantes eram licenciados (40 %), dois eram mestrados (40%) e um era doutorando (20%).

A nível de função/área profissional pode verificar-se que todos trabalham na área da informática, pelo que se pode depreender que a sua literacia digital será de nível elevado, apresentando-se como possíveis utilizadores da interface em desenvolvimento.

	Idade	Sexo	Habilitações literárias	Profissão/ área profissional
Participante 1	42	Masculino	Licenciatura	Engenharia de sistemas
Participante 2	35	Masculino	Mestrado	Design/ multimédia
Participante 3	32	Masculino	Doutorando	Professor ensino superior/ informática
Participante 4	37	Feminino	Licenciatura	Tecnologias de informação
Participante 5	43	Feminino	Mestrado	Professora/ área da informática, programação

Tabela 6.1 - Caracterização geral dos participantes

6.4.2. Compreensão e facilidade de utilização

Depois de explorarem a interface, os participantes procederam à avaliação da mesma relativamente à sua compreensão e utilização. Dos dados recolhidos foi possível observar que 60% dos participantes consideraram a aplicação fácil de utilizar e compreender, 20% considerou a aplicação fácil de compreender, no entanto difícil de utilizar e 20% considerou que demorou algum tempo a compreender o funcionamento da aplicação, tal como pode ser verificado na tabela e no gráfico seguintes.

Compreensão e facilidade de utilização da aplicação desenvolvida	
Resposta 1 - A aplicação é fácil de compreender e de utilizar.	**3**
Resposta 2 - A aplicação é fácil de compreender, no entanto não considero fácil de utilizar.	**1**
Resposta 3 - Demorei algum tempo a compreender o funcionamento da aplicação.	**1**

Tabela 6.2 - Dados recolhidos referentes à compreensão e facilidade de utilização da interface

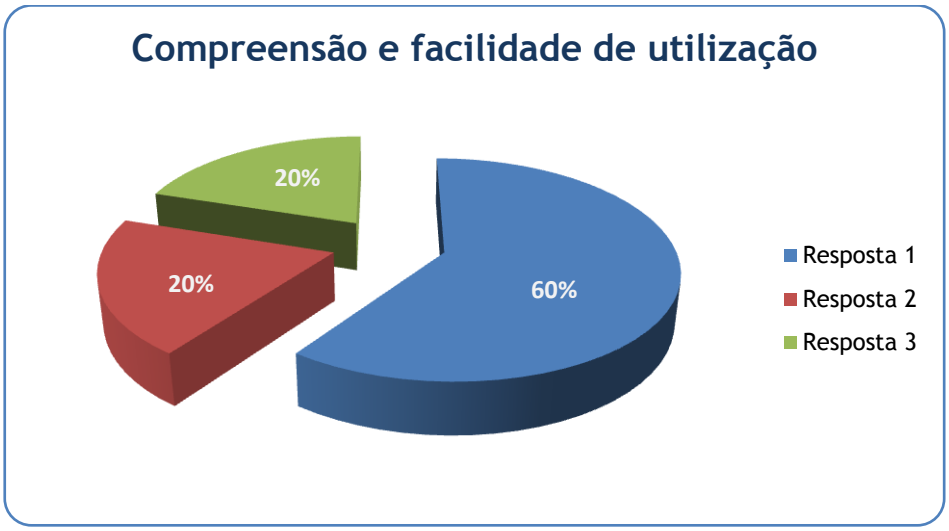


Figura 6.1 - Classificações obtidas pelos participantes relativamente à compreensão e facilidade de utilização da aplicação desenvolvida

6.4.3. Facilidade de memorização

Numa fase posterior à compreensão e utilização da interface pretendeu-se que os participantes referirem se conseguiam se lembrar de todo o processo necessário ao processamento de uma dada imagem. A resposta a esta questão pressupunha que o participante já tivesse aprendido a interface e que fosse capaz de se lembrar mais tarde dos diferentes passos necessários ao processamento de uma imagem, sem ter que fazer nova aprendizagem.

Mediante os resultados obtidos, verificou-se que 80% dos participantes lembrava-se das diferentes fases do processamento de uma imagem, no entanto não consideravam que a aplicação fosse fácil de memorizar, ou seja, possível de utilizar mais tarde sem aprendizagem. Apenas 20% dos participantes considerou que a aplicação era fácil de memorizar e que se lembrava de todos os passos e nenhum dos participantes indicou que tinha esquecido todo o processo que tinha efetuado na primeira aprendizagem, tal como pode ser verificado na tabela e no gráfico seguintes.

Facilidade de memorização do processo de processamento de uma imagem, depois de utilizar a aplicação e obter um resultado.	
Resposta 1 - Esqueci-me de como o processo se efetuou.	**0**
Resposta 2 - Lembro-me do processo de processamento, no entanto não considero de fácil memorização.	**4**
Resposta 3 - O processo é de fácil memorização, lembro-me de todos os passos.	**1**

Tabela 6.3 - Dados recolhidos referentes à capacidade do participante se lembrar de todas as fases da interface

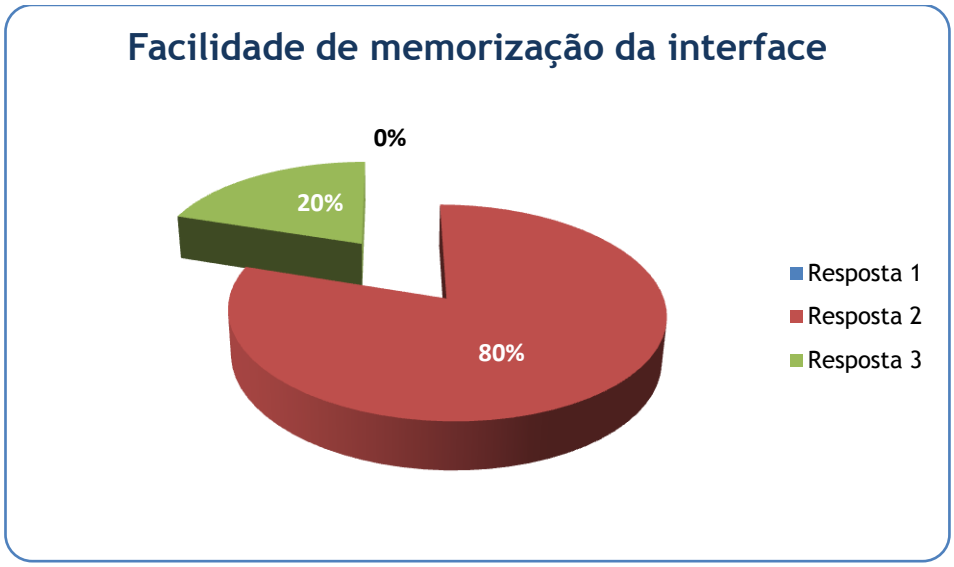


Figura 6.2 - Classificações obtidas pelos participantes relativamente à capacidade de se lembrarem de todas as fases da interface

6.4.4. Apresentação dos separadores/*tabs* da aplicação

De forma a analisar o *feedback* dos participantes relativamente à apresentação da aplicação e à forma como os seus separadores/*tabs* se dispunham, efetuou-se uma questão sobre a forma como a aplicação se apresentava.

Dos dados recolhidos evidenciou-se que todos os participantes (100%) consideraram que os separadores/*tabs* da aplicação se apresentavam de forma lógica, como se pode verificar através da tabela e do gráfico seguintes.

Apresentação dos separadores/*tabs* da aplicação.	
Resposta 1 - Os separadores são apresentados de forma confusa.	**0**
Resposta 2 - Não considero coerente a apresentação de alguns separadores.	**0**
Resposta 3 - Os separadores são apresentados de forma lógica.	**5**

Tabela 6.4 - Dados recolhidos referentes à apresentação dos separadores/*tabs* da aplicação

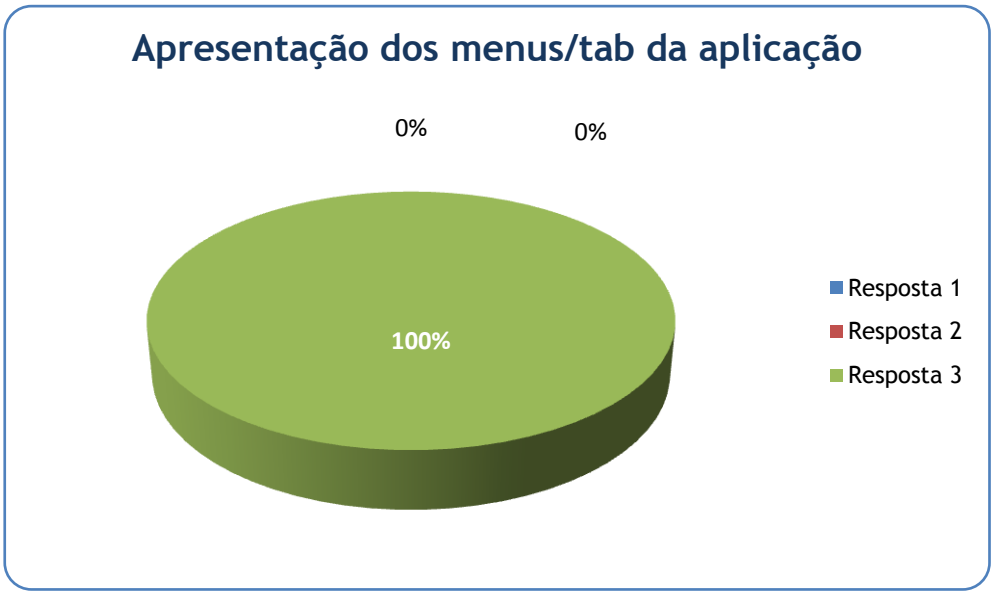


Figura 6.3 - Classificações obtidas pelos participantes relativamente à apresentação dos separadores/*tabs* da aplicação

6.4.5. Navegação entre os separadores/*tabs*

Depois de ter sido apresentada a aplicação, de terem explorado as suas funcionalidades e de terem conhecimento dos passos necessários ao processamento de imagem, os participantes foram questionados quanto à navegação entre os diferentes separadores da aplicação.

Dos dados obtidos foi possível verificar que 80% dos participantes consideraram que era fácil navegar entre os separadores, pois a aplicação encontrava-se bem estruturada e 20% dos participantes consideraram que apesar de ser fácil navegar entre os separadores o conteúdo da aplicação não estava bem estruturado. Nenhum participante considerou que a aplicação não se encontrava bem estruturada tal como se pode analisar através da figura e do gráfico seguintes.

Navegação entre os separadores/*tabs* da aplicação.	
Resposta 1 - A aplicação não está bem estruturada, o que dificulta a navegação.	**0**
Resposta 2 - Apesar de a navegação ser fácil o conteúdo não está bem estruturado.	**1**
Resposta 3 - É fácil de navegar entre os separadores, a aplicação está bem estruturada.	**4**

Tabela 6.5 - Dados recolhidos referentes à navegação entre os separadores/*tabs*

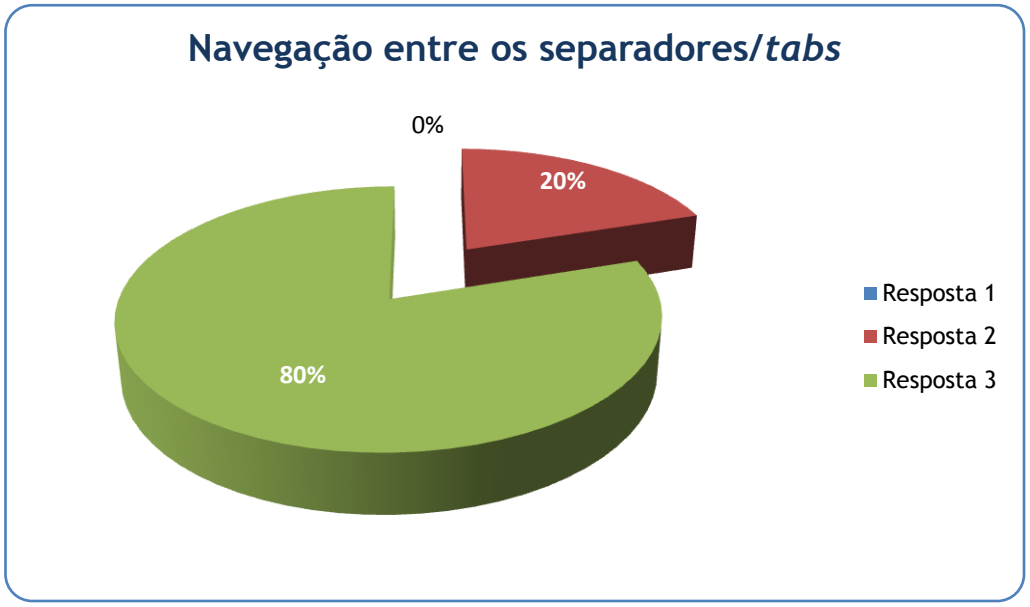


Figura 6.4 - Classificações obtidas pelos participantes relativamente à navegação entre os separadores/*tabs*

6.4.6. Sugestões dos participantes

Além dos dados recolhidos a partir das questões anteriormente referidas, foram também recolhidas todas as sugestões e opiniões dadas pelos participantes. Os dados obtidos foram muito relevantes, já que permitiram identificar a aprendizagem e satisfação por parte dos participantes. Foi igualmente importante pois possibilitou a deteção de eventuais problemas que a aplicação apresentava.

Na tabela seguinte são apresentados os principais problemas e/ou sugestões referidas pelos participantes.

Sugestões dadas pelos participantes
• Falta um aspeto gráfico mais apelativo. • Sendo um processo de vários passos com uma sequência, poderia ter o seguinte: **1.** Sabermos em que passo estamos, tipo passo 1 de x **2.** As opções (passos) que ainda não podemos fazer estarem desabilitados, por ex.: não poder executar o classificador sem primeiro escolher as imagens • Estar tudo desativado e ir ativando as opções conforme as escolhas. • Ter um *help*, em cada *tab* a explicar/ como fazer. • Quando se "carrega imagem" desabilitar o "carrega pasta" e vice-versa. • Pré-visualizar a imagem no 1º *tab*, não sei se fará muito sentido, pois logo a seguir existe o "pré-processamento" com a mesma opção e faz mais sentido estar lá. • No "pré-processamento" e na "classificação" era mais prático podermos "arrastar" o filtro para a caixa de seleção. • Na fase de pré-processamento deveria constar a possibilidade de guardar a imagem já tratada/filtrada para posteriores utilizações. • Em cada fase deveria existir uma explicação sobre a operação a efetuar, para se perceber melhor o que é preciso fazer.

Tabela 6.6 - Sugestões dadas pelos participantes

6.5. Conclusão

A aplicação deste teste tinha como principal objetivo detetar pontos críticos da interface/ferramenta. Este objetivo foi cumprido e útil, já que se detetaram e apontaram vários pontos que posteriormente foram revistos/melhorados na interface.

A análise feita a partir dos dados recolhidos nos testes tentou alcançar conclusões sobre os aspetos a melhorar e sobre a satisfação dos participantes. Desta forma, trataram-se os dados e apresentaram-se sob a forma de gráficos e tabelas, para que esses dados fossem mais facilmente compreendidos e interpretados.

Outro dos objetivos da análise dos dados recolhidos foi o de tentar perceber até que ponto a interface promove a aprendizagem por parte dos utilizadores, ou seja, se conseguem lembrar-se das tarefas efetuadas.

As sugestões apontadas pelos participantes foram muito pertinentes e possibilitaram que alguns problemas fossem retificados, permitindo melhorar a interação do utilizador com a interface e evitar a perda de tempo em futuras alterações.

Capítulo 7- Conclusões

7.1. Conclusão

O presente estudo tinha como principal objetivo o desenvolvimento de uma interface gráfica que possibilitasse a implementação das principais etapas do processamento de uma imagem, de forma sequencial numa única interface.

Para que este objetivo fosse cumprido foi necessária uma base sólida de conhecimento sobre as diferentes etapas do processamento de imagem, para que de seguida se pudesse começar a conceção e desenvolvimento da interface.

A revisão bibliográfica efetuada permitiu o conhecimento e a compreensão do processo de processamento de imagem. O estudo do reconhecimento de padrões, nomeadamente dos descritores e dos classificadores, foi de grande importância para que os módulos de extração/descrição e classificação pudessem ser desenvolvidos.

Concluída a revisão bibliográfica, avançou-se para a conceção de um esquema geral da interface a implementar. Desta conceção verificou-se que a interface iria ser constituída por três módulos diferentes que permitiriam cada um, a implementação de uma etapa do processamento de imagem. Estes módulos foram definidos como módulo de pré-processamento, módulo de extração/descrição e módulo de classificação.

Na implementação do módulo de pré-processamento foram implementados diferentes filtros através da utilização da biblioteca AForge.Imaging da *framework* AForge.NET.

No módulo de extração/descrição foram implementados dois descritores, o descritor SIFT e o descritor SURF. Apesar de nos objetivos inicialmente propostos apenas se tivesse projetado a implementação de um único descritor, com o avanço do trabalho foi possível a implementação de um outro descritor. Deste avanço no trabalho e da revisão bibliográfica efetuada foram implementados os descritores SIFT e SURF por apresentarem resultados satisfatórios. Foi também implementada neste módulo uma funcionalidade que não foi apresentada nos objetivos indicais, a implementação do SIFT-*Match*, que possibilita a comparação entre duas imagens e a identificação dos seus ponto-chave correspondentes.

Durante a conceção do módulo de classificação, e tendo já o conhecimento prévio dos classificadores através da revisão bibliográfica, foi possível identificar que a sua implementação poderia ser efetuada recorrendo à utilização da ferramenta de *data mining* Weka, uma vez que já tinha implementado os classificadores.

Partindo dos princípios estudados foi iniciado o projeto de desenvolvimento da interface gráfica para processamento de imagens.

A interface gráfica foi desenvolvida em *Visual Studio 2010*, na linguagem de programação C#. O módulo de pré-processamento foi desenvolvido em C#, o módulo de extração/descrição foi desenvolvido em Visual C++ e em C# e o módulo de classificação foi desenvolvido com recurso ao Weka, através da utilização de uma *brigde* (IKVM), entre Weka (Java) e .NET. A utilização de diferentes tecnologias contribuiu para o conhecimento das funcionalidades das mesmas.

Foram implementados dois métodos para análise das imagens. Um desses métodos permite a análise de apenas uma imagem, à qual se pode aplicar um ou mais filtros e onde podem ser identificadas as características dessa imagem, através da aplicação de um descritor.

O outro método permite que se analise um *dataset* de imagens dividido por classes (pastas), onde serão aplicados os módulos de pré-processamento, extração/descrição e classificação. Através do módulo de pré-processamento podem ser aplicados um ou mais filtros ao *dataset* de imagens, para de seguida se extraírem as características dessas imagens. Esta extração é efetuada pelo módulo de extração/descrição através da aplicação de um descritor. Depois de extraídas as características das imagens é aplicado um classificador, através do módulo de classificação, que efetua a classificação das mesmas. Desta classificação é apresentado um resultado do processamento do *dataset*.

Os resultados mostram a avaliação do classificador, nomeadamente através da matriz confusão, onde são apresentadas as imagens corretamente e incorretamente classificadas, os falsos positivos e os falsos negativos de todas as classes das imagens.

Estes métodos de análise de imagem implementados contribuem para a concretização do objetivo principal, a criação de uma interface gráfica para classificação de imagens.

De salientar que a aplicação foi desenvolvida numa estrutura de N camadas e de forma modular, permitindo assim adicionar novas funcionalidades aos módulos já existentes sem grande alteração de código, nem alteração nos métodos de análise de imagem. As funcionalidades a adicionar podem ser implementadas em qualquer linguagem de programação da plataforma .NET, bem como Java ou Matlab, estas últimas recorrendo ao uso de *bridges*. Esta característica de multilinguagem de programação é uma mais-valia, pois muitas destas funcionalidades já se encontram implementadas e disponibilizadas para uso. Assim, não é necessária a conversão entre linguagens de programação nem a mudança de aplicação, para concluir um processo de análise de imagens.

Antes de se obter a interface final foi elaborado um protótipo com a finalidade de se efetuar um teste de usabilidade da mesma. Este teste foi aplicado a cinco participantes que depois de lhes ter sido explicado o objetivo e funcionamento da interface puderam interagir livremente com a mesma. Com a aplicação destes testes avaliou-se a interface, nomeadamente em relação à compreensão, facilidade de utilização, apresentação dos separadores/*tabs* e navegação da interface. Os participantes registaram as suas sugestões relativamente à interface o que possibilitou retirar conclusões relativamente aos pontos críticos da interface e das áreas onde poderiam ser efetuados melhoramentos.

De um modo geral considera-se que os objetivos propostos neste projeto foram cumpridos, sendo possível responder à questão inicialmente proposta e que se apresentou como base de investigação para esta dissertação, realçando o seu principal objetivo:

Não será possível o desenvolvimento de uma interface gráfica que possibilite, de forma modular, a implementação das principais etapas do processamento de imagem?

Depois de desenvolvida a interface este objetivo foi cumprido e foi possível a criação de uma interface capaz de efetuar algumas das etapas do processamento de imagem, nomeadamente a etapa do pré-processamento, a etapa da extração/descrição das características e por fim a etapa da classificação dessas características.

Por implementar diferentes etapas do processamento de imagem, desde o seu pré-processamento, passando pela extração das características da imagem e por fim fazendo a classificação dessas características, a interface gráfica poderá servir como base de investigação a investigadores na área do processamento de imagem.

7.2. Limitações do estudo

A aplicação apesar de ser multilinguagem de programação, uma das limitações é o facto de neste momento, para adicionar novas funcionalidades aos módulos existentes, ser necessária nova compilação dos mesmos.

A ferramenta Weka começou a ser desenvolvida em 1993, com uma constante atualização de novas funcionalidades. Esta aplicação não faz uso de todas as funcionalidades do Weka, algumas de bastante relevância como o pré-processamento, não das imagens mas sim das características extraídas, onde faz filtragem de dados. Também não faz uso da implementação de Clusters, nem implementações relativas à apresentação dos dados e/ou resultados graficamente. É possível adicionar o acesso a estas funcionalidades à aplicação, mas que no momento não se encontra implementado graficamente. No entanto, é possível executar todas estas funcionalidades do Weka a partir da aplicação, caso o utilizador tenha conhecimento dos comandos Weka pela linha de comandos.

Relativamente à implementação dos testes de usabilidade considera-se uma limitação ao estudo o reduzido número de participantes e a forma como foram selecionados, o que pode levar a que a amostra representativa do público-alvo seja considerada inválida e as conclusões generalizáveis. No entanto, considera-se que os resultados obtidos com a aplicação destes testes foram bastante expressivos e reveladores dos principais pontos críticos a rever e melhorar na interface, pelo que, e apesar da limitação enunciada, se consideram como relevantes os dados recolhidos.

7.3. Trabalho futuro

Este estudo possibilitou o desenvolvimento de uma única interface gráfica que implementa as principais etapas pelas quais uma imagem passa durante o seu processamento. No entanto, novas funcionalidades podem ser adicionadas à interface desenvolvida, nomeadamente:

- No módulo de pré-processamento o investigador pode aplicar diferentes filtros à imagem. No entanto existem mais filtros dos que se encontram implementados nesta interface, pelo que futuramente poderiam ser implementados novos filtros, ou usar filtros já implementados;
- De forma a serem retiradas partes não necessárias numa imagem, uma sugestão seria a implementação de algoritmos de segmentação;

- Através de um descritor é possível a extração das características de uma imagem. Como referido na revisão bibliográfica deste estudo, existem diferentes descritores que se podem aplicar no processamento de imagem. Como neste estudo se implementou o descritor SIFT e o SURF, uma sugestão futura seria a implementação de mais descritores, pois são essenciais para a classificação;
- O Weka é uma ferramenta *data mining* utilizada neste estudo no módulo de classificação. O Weka integra outras funcionalidades que não foram exploradas nesta interface, desta forma poderiam ser implementadas algumas dessas funcionalidades, nomeadamente o processamento dos dados extraídos, a implementação de *clusters*, entre outros;
- O OpenCV é uma biblioteca multiplataforma utilizada no módulo de extração/descrição para implementação do detetor SIFT. Esta plataforma *open source* possui várias funções que não foram exploradas nesta interface, sendo sugestão futura a exploração e implementação de outras funções do OpenCV na interface apresentada;
- A biblioteca Aforge.NET foi utilizada no módulo de pré-processamento, pois possui vários filtros. No entanto esta biblioteca incorpora mais funcionalidades do processamento de imagens, além da aplicação de filtros, que poderiam ser exploradas e utilizadas na aplicação, nomeadamente nos módulos de extração/descrição e classificação.

A interface foi testada com um *dataset* que se encontra *online*. Uma sugestão seria a replicação de testes já efetuados por investigadores da área e comparação de resultados.

Referências

Adler, J. L. e Blue, V. J., 2002. *A cooperative multi-agent transportation management and route guidance system*. Transportation Research C Vol. 10, pp. 433-454.

Adriaans, P. e Zantinge, D., 1996. *"Data Mining,1 ed."*, Harlow: Addison-Wesley.

AForge.NET Framework. [Online] Disponível em: http://code.google.com/p/aforge e em http://www.aforgenet.com/framework/features [Acedido a 1 de Março 2011].

Areia, M. et al., 2008. *External validation of a classification for methylene blue magnification chromoendoscopy in premalignant gastric lesions. Gastrointestinal Endoscopy*. Gastrointest Endosc.

Ballesta, M. Gil, A. Mozos, O.M. e Reinoso, O., 2007. Local descriptors for visual SLAM, *in 'Workshop on Robotics and Mathematics'*, Portugal:Coimbra.

Bandyopadhyay, S. e Maulik, U., 2002. *Genetic clustering for automatic evolution of clusters and application to image classification*. IEEE pattern recognition, Vol.35, pp.1197-1208.

Bay, H. Ess, A. Tuytelaars, T. e Van Gool, L., 2008. *SURF: Speeded Up Robust Features, Computer Vision and Image Understanding (CVIU)*, Vol. 110, N. 3, pp. 346-359. [Online] Disponível em: ftp://ftp.vision.ee.ethz.ch/publications/articles/eth_biwi_00517.pdf [Acedido a 1 de Novembro 2010].

Bodri, B., 2001. *A neural-network model for earthquake occurrence*. Journal of Geodynamics, Vol. 32, pp. 289-310.

Boiman, O. Shechtman, e E. Irani, M.,2008. In defense of Nearest-Neighbor based image classification, Computer Vision and Pattern Recognition CVPR 2008. *IEEE Conference on Publication,* 23-28 June 2008, pp. 1-8.

Bosch, A. Zisserman, A. e Munoz, X. Scene classification via pLSA. *In Proceedings of the European Conference on Computer Vision*, Graz, Austria, volume 3954 of Lecture Notes in Computer Science, Springer, 2006, pp. 517–530.

C#, [Online] Disponível em: http://msdn.microsoft.com/en-us/vcsharp/aa336706, [Acedido a 10 de Fevereiro 2011].

Carvalho, Ana Amélia Amorim, 2008. *Testes de Usabilidade: exigência supérflua ou necessidade?*, [Online] Disponível em: http://www.lits.dei.uminho.pt/tu.pdf [Acedido a 10 de Março 2011].

Carvalho, A. C. P. L. F. Lorena, A. C., 2007. *Uma introdução às Support Vector Machines*. Revista de Informática Teórica e Aplicada. v. 14, pp. 43-67.

Coley, A. D., 1999. *An Introduction to Genetic Algorithms for Scientists and Engineers*. Singapore: World Scientific, pp.188.

Cooper, A. Reimann, R. e Cronin, D., 2007. *About Face 3 - The Essentials of Interaction Design*, New York: John Wiley and Sons.

Cover, T. M. e Hart, P. E., 1967. *Nearest Neighbor Pattern Classification*, IEEE Transactions on Information Theory, vol. IT-13, Nº1, pp. 21-27.

Cox, E. et al., 1998. The Fuzzy Systems Handbook. 2. ed. Orlando, FL, USA: Academic Press.

Deco, G. e Zihl, J., Fevereiro 2001. *"Top-down selective visual attention: A neurodynamical approach*" Visual Cognition, vol. 8, nº 1, pp. 119-140.

Demuth, H. e Beale, M., 1998. *Neural Network Toolbox*. Massachusetts: The MathWorks, Inc., Natick.

Dhandra, B. e Hedadi, R., 2005. *Classification of abnormal endoscopic images using RGB Color and Morphological watershed segmentation*. International Conference on Cognition and Recognition - ICCR 2005. Mysore, India, 22-23 December.

Domingos, P. e Pazzan, M., 1997. *"On the optimality of the simple Bayesian classifier under zero-one loss"*. Machine Learning, 29:103–137.

Eidenberger, H., 2003. How good are the visual MPEG-7 features? In Proceedings SPIE Visual Communication, *Image Processing Conference*, Vol. 5150, pp. 476–488.

Evan, Christopher, 2009. Notes on the OpenSURF Library. [Online] Disponível em: http://www.chrisevansdev.com/computer-vision-opensurf.html, [Acedido a 10 de Abril 2011].

Fayyad, U. Shapiro, G. e Smyth, P.,1996. "From Data Mining to Knowledge Discovery in Databases", *AI Magazine*.

Fei-Fei, L. Fergus, R. e Perona P., 2004. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. IEEE. CVPR 2004, *Workshop on Generative-Model Based Vision*.

George, H. John e Pat, Langley, 1995. Estimating Continuous Distributions in Bayesian Classifiers. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. pp. 338-345.

Glowacz, A. Grega, M. Leszczuk, M. e Romaniak, P., 2006. Detecting panoramic image overlaps with MPEG-7 descriptors, *Proceedings of the international conference on signals and electronic systems (ICSES '06)*, Poland:Łódź.

Goebel M. e Gruenwald L., Junho 1999. "*A Survey of Data Mining and Knowledge Discovery Software Tools*", ACM SIGKDD, Volume 1.

Gonzalez, R. C., e Woods, R. E., 1993. *Digital Image Processing*, Addison-Wesley Publishing Company.

Hagan, M.T. Demuth, H. B. e Beale, M., 1996. *Neural Network Design*. PWS Publishing Company.

Hall, M. et al., 2009. *The WEKA data mining software: an update*. SIGKDD Explorations, 11(1):10-18, [Online] Disponível em: http://www.cs.waikato.ac.nz/~ml/publications.html [Acedido a 23 de Janeiro 2011].

Hess, Rob, 2010. [Online] Disponível em: http://blogs.oregonstate.edu/hess/code/sift/ [Acedido a 12 de Janeiro de 2011].

IKVM, 2011. [Online] Disponível em: http://www.ikvm.net/ [Acedido a 10 de Abril 2011].

Intel Corporation, Copyright © 1999-2001. *"Open Source Computer Vision Library"* Reference Manual, Disponível em: http://software.intel.com/sites/oss/pdfs/OpenCVreferencemanual.pdf. [Acedido a 10 de Março 2011].

ISO/IEC 15938-3/FCD, 2001. *Information Technology - Multimedia Content Description Interface - Part 3: Visual*. Singapura MPEG Meeting.

ISO/IEC Standard TR 15938, 2005. *Information technology-multimedia content description interface*.

Itti, L. Koch, C. e Niebur, E., 1998. *A model of saliency-based visual attention for rapid scene analysis*. IEEE TPAMI, 20(11): pp. 1254–1259.

J-Integra for .NET, n.d. [Online] Disponível em: http://j-integra.intrinsyc.com/net.asp [Acedido a 5 de Fevereiro 2011].

JBind2.net, 2004. [Online] Disponível em: http://www.j2dotnet.com/ [Acedido a 5 de Fevereiro 2011].

JNbrigde, n.d. [Online] Disponível em: http://www.jnbridge.com/ [Acedido a 5 de Fevereiro 2011].

Jolliffe, I. T., 1986. *Principal Component Analysis*. Springer-Verlag, pp. 487.

JuggerNET, 2006. Solutions for Language Integration. [Online] Disponível em: http://codemesh.com/index.html [Acedido a 5 de Fevereiro 2011].

Karthik, 2007. [Online] Disponível em: http://karthik3685.wordpress.com/2007/11/03/nose-picking-using-neural-networks) [Acedido a 15 de Dezembro 2010].

KDnuggets. [Online] Disponível em: http://www.kdnuggets.com [Acedido a 1 de Fevereiro 2011].

Kim, K. I. Jung K. Park, S. H. e Kim, H. J., 2002. *Support Vector Machines for texture classification*. IEEE Trans, PAMI.

King, M. et al., 2006. *"Evaluation of Fourteen Desktop Data Mining Tools"*, [Online] Disponível em: http://www.datamininglab.com/pubs/smc98_king_elder.pdf [Acedido a 10 de Fevereiro 2011].

Kittler, J., 1998. *Combining Classifiers: A Theoretical Framework*. Pattern Analysis Applic.

Kittler, J. Hatef, M. Duin R. W. e Matas, J., 1998. *On Combining Classifiers*. IEEE Trans, PAMI.

Klir, G. Folger, T., 1987. Fuzzy sets: uncertainty, and Information. 1. ed. Upper Saddle River, NJ, USA: Prentice-Hall.

Kohavi, R. e Provost, F., 1998. Glossary of Terms. Machine Learning, 30(2-3), 271-274.

Kosko, B., 1992. "*Neural networks and fuzzy systems: a dynamical systems approach to machine intelligence*", Prentice-Hall International, pp. 4.

Loureiro, Henrique, 2011. *"C# 4.0 com Visual Studio 2010"*. Editora FCA.

Loureiro, Henrique, 2008. *"Visual Basic 2008"*. Editora FCA.

Lovell, B. e Walder, C., 2006. *"Support Vector Machines for Bussiness Applications"*, Business Applications and Computational Intelligence, Idea Group Publishers.

Lowe, D., 1999. Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision*, Corfu, Greece, vol. 2, pp. 1150–1157.

Lowe, D., 2004. *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision, vol. 60 (2), pp. 91–110.

Lu, H. Huang, Y. Chen, Y. e Yang, D., 2007. Real-time facial expression recognition based on pixelpattern-based texture feature. *In Proc. Electronic Letters*, pp. 916–918.

Manjunath, B. Ohm, J. R. Vasudevan, V. V. e Yamada, A., 2001. Color and texture descriptors. *IEEE Trans Circuits and Systems for Video Technology*, vol.11, nº6, pp. 703–715.

Marques de Sá, J. P., 2004. Introdução à Programação. Faculdade de Engenharia do Porto. [Online] Disponível em: http://paginas.fe.up.pt/~jmsa/recpad/index.htm [Acedido a 22 de Julho de 2011].

Marques, Jorge S., 2005. *Reconhecimento de Padrões: métodos estatísticos e neuronais*. Lisboa: IST Press, 2ª edição.

Mathworks, n.d., MATLAB Builder NE for Microsoft .NET Framework, [Online] Disponível em: http://www.mathworks.com/products/netbuilder [Acedido a 5 de Abril 2011].

Mathworks, n.d., Using .NET from MATLAB: An Overview, [Online] Disponível em: http://www.mathworks.com/help/techdoc/matlab_external/brpb5k6-1.html [Acedido a 5 de Abril 2011].

Mitchell, T., 1997. *Machine Learning*. McGraw-Hill.

Mikolajczyk, K. e Schmid, C., 2005. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis e Machine Intelligence*, 27(10), pp. 1615–1630.

New Technologies In Medicine, 2010. [Online] Disponível em: http://mezocore.wordpress.com [Acedido a 18 de Janeiro de 2011].

Netponto, Tecnologias Biométricas, 2010. [Online] Disponível em: http://www.netponto.com [Acedido a 18 de Janeiro de 2011].

Ohm, J. R., 2001. The MPEG-7 visual description framework-concepts, accuracy and applications, *In CAIP 2001*, nº 2124 in LNCS, pp. 2–10.

OpenCV, 2011. [Online] Disponível em: http://opencv.willowgarage.com/wiki/ [Acedido a 12 de Janeiro de 2011].

Osuna, E. Freud, R. e Girosi, F., 1997. Training support vector machines: an application to face detection. *In Proc. Computer Vision and Pattern Recognition*, pp. 130–136.

Pang-Ning T. Steinbach, M. e Kumar, V., 2006. *Introduction to data mining*, Adison-Wesley.

Pedrycz, W. e Gomide, F., 2007. *"Fuzzy Systems Engineering : Toward Human-Centric Computing"*; Wiley/IEEE Press.

Petrovskiy, M. I., 2003. *"Outlier Detection Algorithms in Data Mining Systems"*, Programming and Computer Software, Vol. 29, Nº 4, pp. 228-237.

Pham, D.T. e Karaboga, D., 2000. *Intelligent Optimisation Techniques*. London: Springer Great Britain, pp.261.

Remco, B. et al., 2010. WEKA-experiences with a java open-source project. *Journal of Machine Learning Research*, 11:2533-254, [Online] Disponível em: http://www.cs.waikato.ac.nz/~ml/publications.html [Acedido a 23 de Janeiro 2011]

Rexer Analytics, 2011. Analytic and CRM consulting, [Online] Disponível em: http://www.rexeranalytics.com [Acedido a 15 de Março 2011].

Rexer, K., dataminersurvey@rexeranalytics.com, 2011. *Rexer Analytics' 4th Annual Data Miner Survey Summary*. [email] Mensagem enviada para Paulo Alves (palves@ipcb.pt) Data de envio: 10 de Abril de 2011.

Rish, I., 2001. *"An empirical study of the naive Bayes classifier"*. IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence.

Rocha, M. Neves J., 1998. *Uma aproximação à resolução do problema do caixeiro viajante via programação genética*, Departamento de Informática, Universidade do Minho, Braga: Portugal.

Rolls, E. T. e Deco, G., 2002, *Computational neuroscience of vision*. Oxford England: Oxford University Press.

Rothlauf, F., 2006. *Representations for Genetic and Evolutionary Algorithms*, Netherlands: Springer, pp.314.

Rubin, J., 1994. *Handbook of Usability Testing*, New York: John Wiley and Sons.

Ruffaldi, Emanuele, 2003. 1.2.3 ways of integrating MATLAB with the .NET, [Online] Disponível em: http://www.codeproject.com/KB/dotnet/matlabeng.aspx [Acedido a 7 de Abril 2011].

Santos, M. F. Azevedo, C., *"Data Mining, Descoberta de Conhecimento em Bases de Dados"*, FCA – Editora de Informática, 2005.

Sarfraz, M. S. e Hellwich, O., 2008. Head pose estimation in face recognition across pose scenarios. *In International conference on Computer Vision Theory and Applications*, pp. 235–242.

SAS, [Online] Disponível em: http://www.sas.com [Acedido a 15 de Março 2011].

Shawe-Taylor, J. e Cristianini, N., 2004. *Kernel Methods for Pattern Analysis*. Cambridge: Cambridge University Press.

Sierra B., 2002. *Aportaciones metodologicas a la Clasificacion Supervisada*, Tese de Doutoramento. Espanha: Universidade del País Vasco.

Smith, C. e Mayes T., 1996. *Telematics Applications for Education and Training: Usability Guide*. Comission of the European Communities, DGXIII Project.

Tchangani, Ayeley P., 2005. *Support Vector Machines: A Tool for Pattern Recognition and Classification*. Studies in Informatics & Control Journal 14: 2. 99 – 110.

The Data Mine. *Data Mining Software, Tools and Applications*. [Online] Disponível em: http://www.the-data-mine.com/bin/view/Software/DataMiningSoftware [Acedido a 1 de Fevereiro 2011]

Vapnik V.N., 1995. *The Nature of Statistical Learning Theory*. New York: Springer.

Visual Studio 2010, [Online] Disponível em: http://www.microsoft.com/visualstudio/pt-br/visual-studio-2010-launch [Acedido a 4 de Setembro 2010].

Webb, G. e Pazzani, M., 1998. *Adjusted Probability Naive Bayesian Induction*. Australia: 11th Autralian Joint Conference on Artificial Intelligence.

Weka. *Weka Machine Learning Project.*, 2010. [Online] Disponível em: http://www.cs.waikato.ac.nz/~ml/index.html [Acedido a 4 de Julho 2010].

Witten, Ian H e Eibe, Frank, 2005. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. 2nd ed. St. Louis: Morgan Kaufmann.

Yang, Z. e Kuo, C., 1999. Survey on image content analysis, indexing, and retrieval techniques and status report of MPEG-7, *Tamkang Journal of Science and Engineering*, vol. 3, nº2, pp. 101-118.

Zadeh, Lotfi A., 1965. *Fuzzy Sets. Information Control*, vol.8.

Anexos

A existência destes anexos prende-se com o facto de terem sido documentos de apoio ao trabalho desenvolvido. Os anexos apresentados são os seguintes:

- **Anexo 1** – Lista de filtros implementados pela biblioteca AForge.Imaging, integrada na *framework* AForge.NET.
- **Anexo 2** – Formato *arff*.
- **Anexo 3** – Protótipo desenvolvido para a elaboração do teste do anexo 4. De referir que o anexo 3 também se encontra em formato digital.
- **Anexo 4** – Teste de usabilidade da interface.

Anexo 1 – Biblioteca AForge.Imaging

A AForge.Imaging é uma das bibliotecas integradas na *framework* AForge.NET que contém diferentes rotinas de processamento de imagem, que se destinam na aplicação dos seguintes filtros (AForge, n.d):

- *Linear color correction filters* (correção RGB/HSL/YCbCr, correção de brilho/ contraste/saturação);
- *Nonlinear color correction filters (*equalização de histograma*)*;
- *Image re-coloring filters* (*grayscale* – tons de cinzento, sépia, rotação de canais, inversão);
- *Pixel filtering by color* (espaços de cor: RGB, HSL, YCbCr);
- *Color channels manipulations* (espaços de cor: RGB and YCbCr);
- *Binarization filters* (*threshold, threshold with carry, ordered dithering, Bayer dithering, Floyd-Steinberg dithering, Burkes dithering, Jarvis-Judice-Ninke dithering, Sierra dithering, Stucki dithering*);
- *Adaptive binarization (*estatística de imagens*, iterative thresholding, Otsu thresholding);*
- *Adaptive local thresholding;*
- *Mathematical morphology filters (*erosão, dilatação, abertura, encerramento*, top hat, bottom hat, hit-and-miss*);
- *Edge detectors* (homogeneidade, diferença, *sobel, canny*);
- *Blobs processing* (*counting, extraction, filtering, connected component labeling*);
- *Filling holes* em imagens binárias;
- Corner detectors (Moravec, Susan);
- *Resize and rotation (nearest neighbor, bilinear, bicubic);*
- *Transformation to/from polar coordinates;*
- *Hough transformation (line and circle transformations);*
- *Exhaustive template and block matchin;*
- *Image color statistics (RGB, HSL, YCbCr) and vertical/horizontal statistics (RGB);*
- *Smoothing filters (Median, Mean, Conservative Smoothing, Adaptive Smoothing);*
- *Texture generators (clouds, marble, wood, labyrinth, textile);*
- *Texture filters (texturing, merging, filtering);*
- *Noise generators (additive, salt-and-papper);*
- *Flat Field Illumination correction, Simple skeletonization, Shrink, Canvas crop/fill/move, mirroring, Bayer filter;*
- *Fourier transformation (low-pass and hi-pass filters);*
- Entre outros.

Anexo 2 – Formato *arff*

ARFF files have two distinct sections. The first section is the Header information, which is followed the Data information. The Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types. An example header on the standard IRIS dataset looks like this:

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
@RELATION iris

@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

The Data of the ARFF file looks like the following:

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Lines that begin with a % are comments. The @RELATION, @ATTRIBUTE and @DATA declarations are case insensitive.

The ARFF Header Section

The ARFF Header section of the file contains the relation declaration and attribute declarations.

The @relation Declaration

The relation name is defined as the first line in the ARFF file. The format is:

```
@relation<relation-name>
```

where<relation-name> is a string. The string must be quoted if the name includes spaces.

The @attribute Declarations

Attribute declarations take the form of an ordered sequence of @attribute statements. Each attribute in the data set has its own @attribute statement, which uniquely defines the name of that attribute and it's data type. The order the attributes are declared indicates the column position in the data section of the file. For example, if an attribute is the third one declared then Weka expects that all that attributes values will be found in the third comma delimited column.

The format for the @attribute statement is:

@attribute<attribute-name><datatype> where the <attribute-name> must start with an alphabetic character. If spaces are to be included in the name then the entire name must be quoted.

The <datatype> can be any of the four types currently (version 3.2.1) supported by Weka:

numeric

<nominal-specification>

string

date [<date-format>]

where<nominal-specification> and <date-format> are defined below. The keywords numeric, string and date are case insensitive.

Numeric attributes

Numeric attributes can be real or integer numbers.

Nominal attributes

Nominal values are defined by providing an <nominal-specification> listing the possible values: {<nominal-name1>, <nominal-name2>, <nominal-name3>, ...}

For example, the class value of the Iris dataset can be defined as follows:

@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

Values that contain spaces must be quoted.

String attributes

String attributes allow us to create attributes containing arbitrary textual values. This is very useful in text-mining applications, as we can create datasets with string attributes, then write Weka Filters to manipulate strings (like StringToWordVectorFilter). String attributes are declared as follows:

@ATTRIBUTE LCC string

Date attributes

Date attribute declarations take the form:

@attribute<name> date [<date-format>] where<name> is the name for the attribute and <date-format> is an optional string specifying how date values should be parsed and printed (this is the same format used by SimpleDateFormat). The default format string accepts the ISO-8601 combined date and time format: "yyyy-MM-dd'T'HH:mm:ss".

Dates must be specified in the data section as the corresponding string representations of the date/time (see example below).

ARFF Data Section

The ARFF Data section of the file contains the data declaration line and the actual instance lines.

The @data Declaration

The @data declaration is a single line denoting the start of the data segment in the file. The format is:

```
@data
```

The instance data

Each instance is represented on a single line, with carriage returns denoting the end of the instance. Attribute values for each instance are delimited by commas. They must appear in the order that they were declared in the header section (i.e. the data corresponding to the nth @attribute declaration is always the nth field of the attribute). Missing values are represented by a single question mark, as in:

```
@data
4.4,?,1.5,?,Iris-setosa
```

Values of string and nominal attributes are case sensitive, and any that contain space must be quoted, as follows:

```
@relationLCCvsLCSH
@attribute LCC string
@attribute LCSH string
@data
AG5,   'Encyclopedias and dictionaries.;Twentieth century.'
AS262, 'Science -- Soviet Union -- History.'
AE5,   'Encyclopedias and dictionaries.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Phases.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Tables.'
```

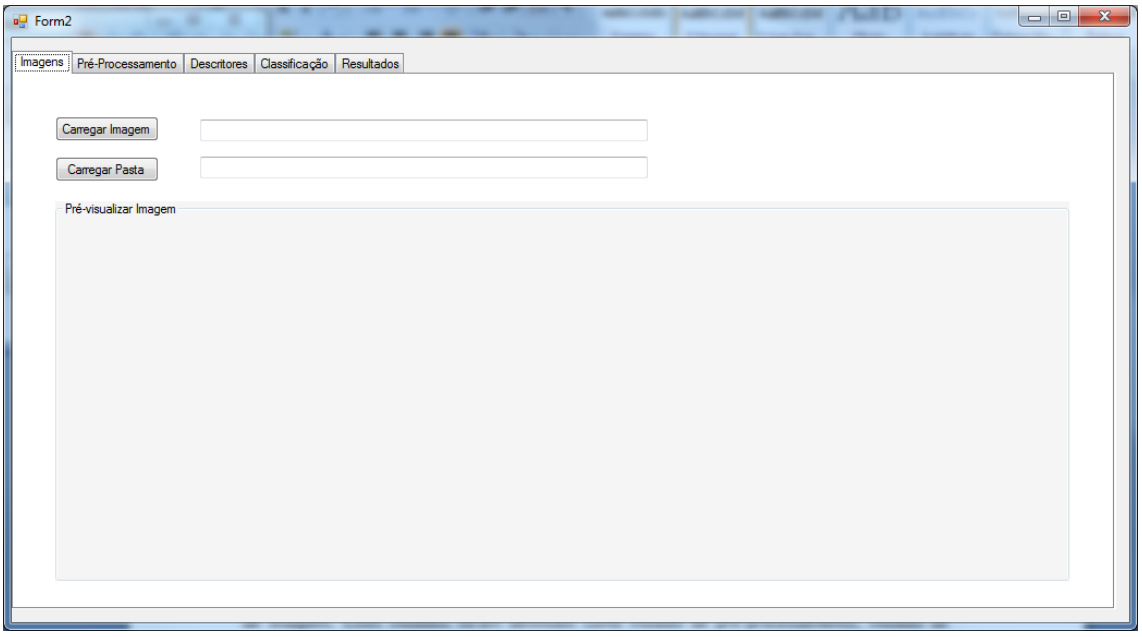
Dates must be specified in the data section using the string representation specified in the attribute declaration. For example:

```
@RELATION Timestamps
@ATTRIBUTE timestamp DATE "yyyy-MM-ddHH:mm:ss"
```

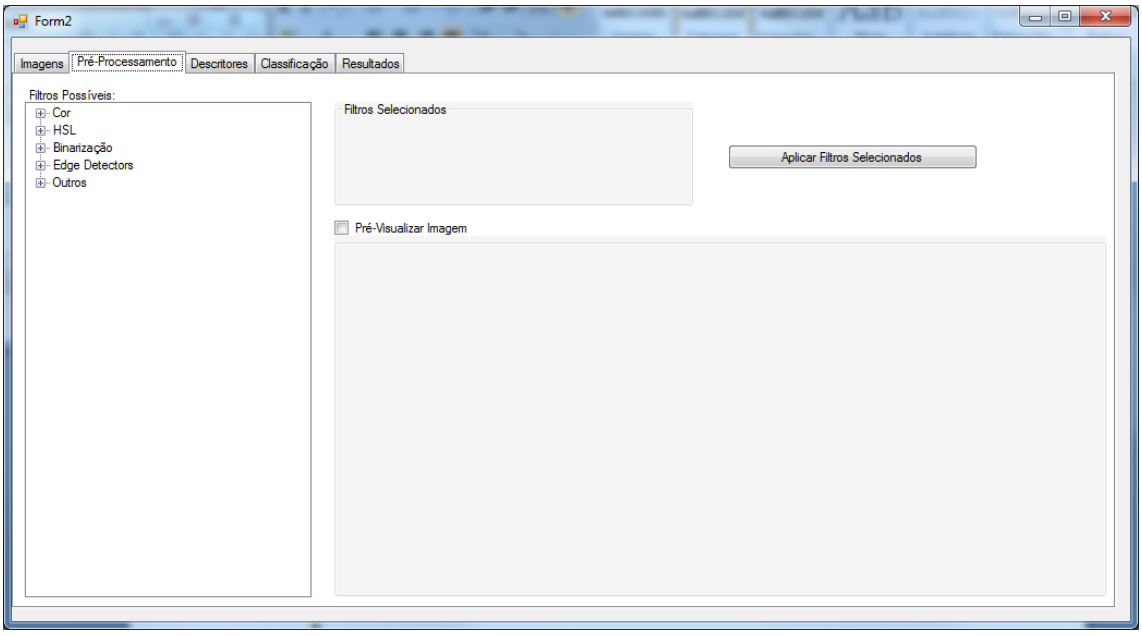
Anexo 3 – Protótipo desenvolvido

Para a aplicação dos testes de usabilidade foi apresentado aos participantes do teste um protótipo da aplicação a desenvolver.

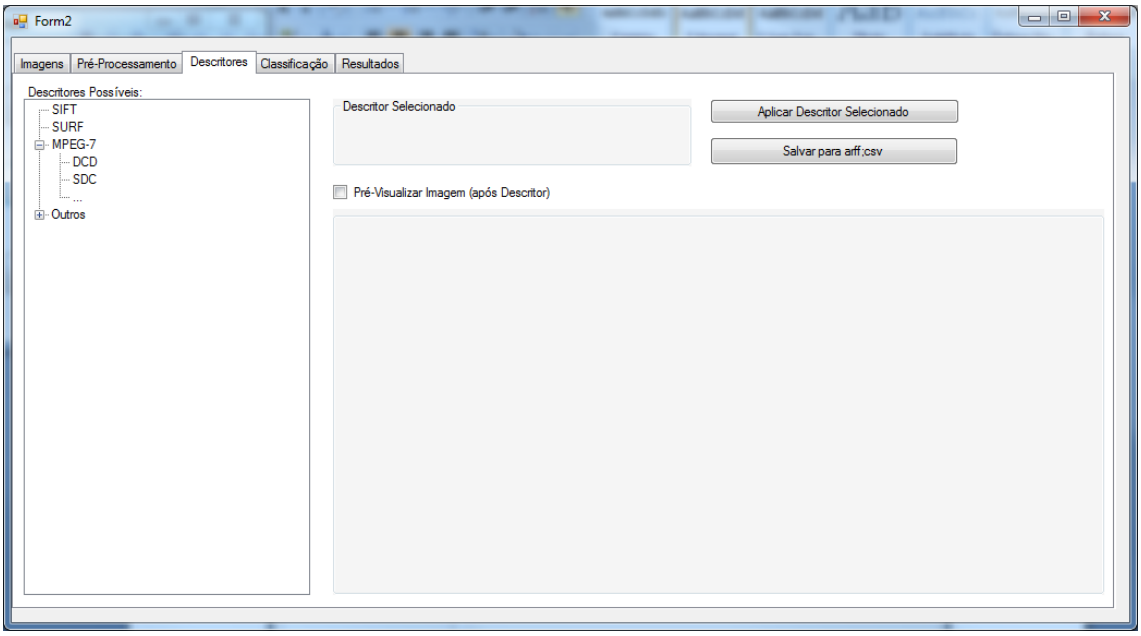
O protótipo implementava grande parte das funcionalidades da aplicação final, sendo constituída por três módulos: o módulo de pré-processamento, o módulo de descritores e o módulo de classificação. Estes módulos são implementados através de separadores/*tabs*, como se pode verificar nas figuras que se seguem. A primeira figura permite que se escolha uma imagem para processamento, ou um conjunto de imagens.



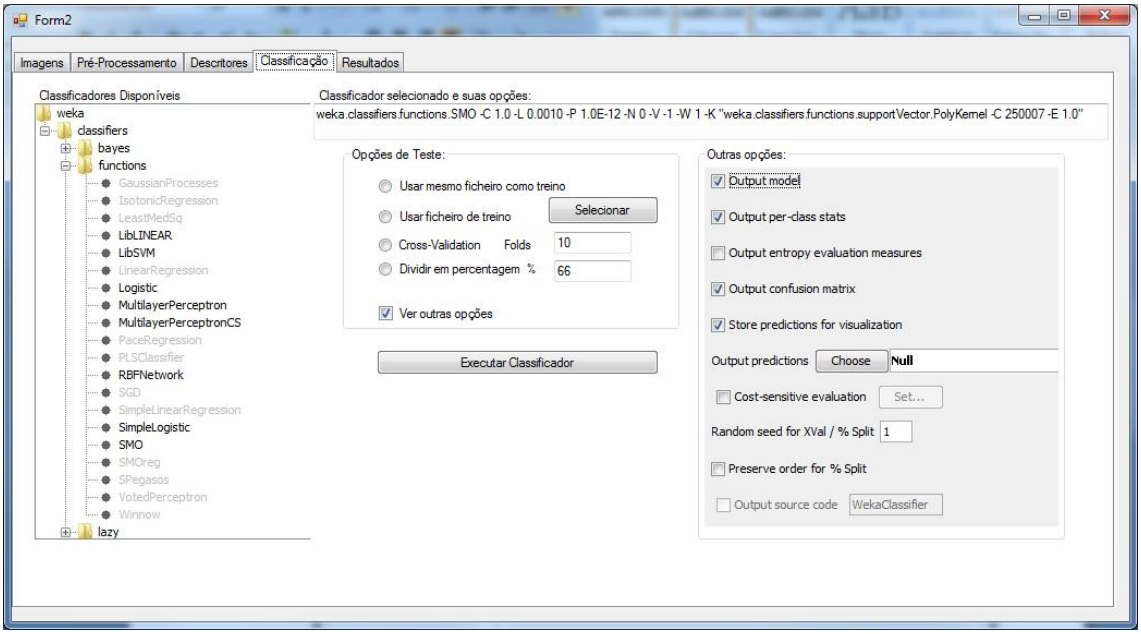
Caso se pretenda aplicar um filtro às imagens deve-se efetuar o seu pré-processamento, através da escolha e posterior aplicação do filtro pretendido.



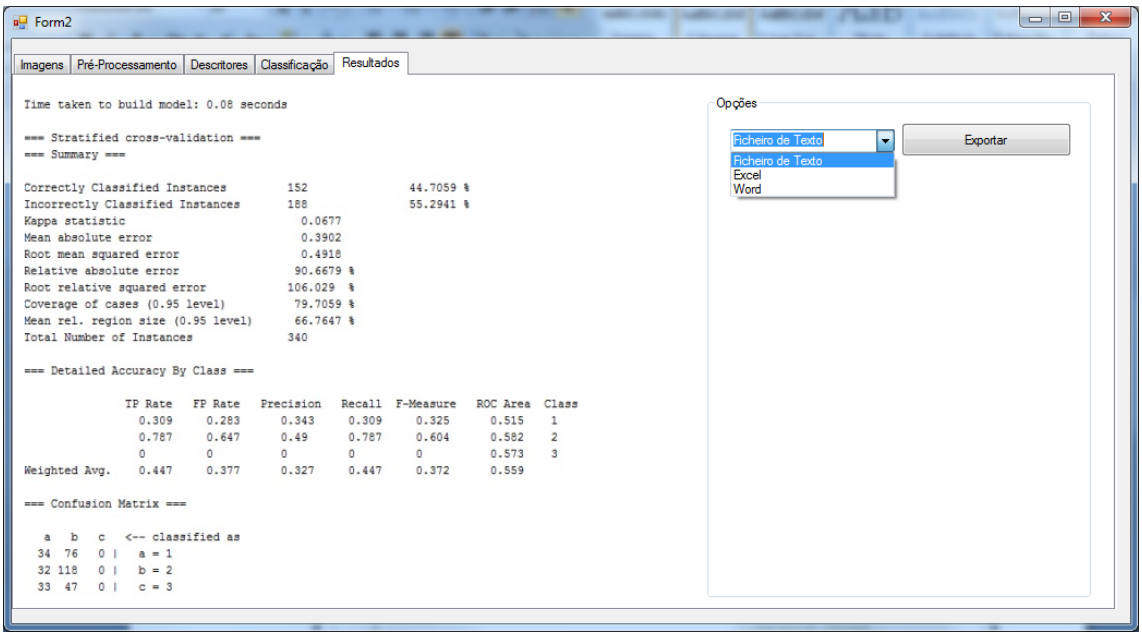
A extração das características de uma imagem é realizada através da escolha e posterior aplicação de um descritor. Esta extração pode ser efetuada a imagens pré-processadas ou à imagem original.



Depois de extraídas as características das imagens para um ficheiro deve-se escolher e posterior aplicar um classificador, de forma a efetuar a classificação dessas características.



Depois de aplicado o classificador o resultado final é apresentado num novo separador/*tab*, com o formato da figura seguinte.


Form2
Imagens
Pré-Processamento
Descritores
Classificação
Resultados
Time taken to build model: 0.08 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 152 44.7059 %
Incorrectly Classified Instances 188 55.2941 %
Kappa statistic 0.0677
Mean absolute error 0.3902
Root mean squared error 0.4918
Relative absolute error 90.6679 %
Root relative squared error 106.029 %
Coverage of cases (0.95 level) 79.7059 %
Mean rel. region size (0.95 level) 66.7647 %
Total Number of Instances 340
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
0.309 0.283 0.343 0.309 0.325 0.515 1
0.787 0.647 0.49 0.787 0.604 0.582 2
0 0 0 0 0 0.573 3
Weighted Avg. 0.447 0.377 0.327 0.447 0.372 0.559
=== Confusion Matrix ===
a b c <-- classified as
34 76 0 | a = 1
32 118 0 | b = 2
33 47 0 | c = 3
Opções
Ficheiro de Texto
Ficheiro de Texto
Excel
Word
Exportar

Anexo 4 - Teste de usabilidade

O presente teste insere-se numa tese de mestrado e tem como objetivo analisar e avaliar a usabilidade de uma ferramenta que otimize os passos no processamento de imagem, e a execução de código em diferentes linguagens (Java, Matlab, etc.).

De forma a efetuar a caracterização dos participantes indique:

1.	Idade.

2.	Sexo.

Feminino. ☐

Masculino. ☐

3.	Profissão/ área profissional.

4.	Habilitações literárias.

Ensino secundário. ☐

Bacharelato. ☐

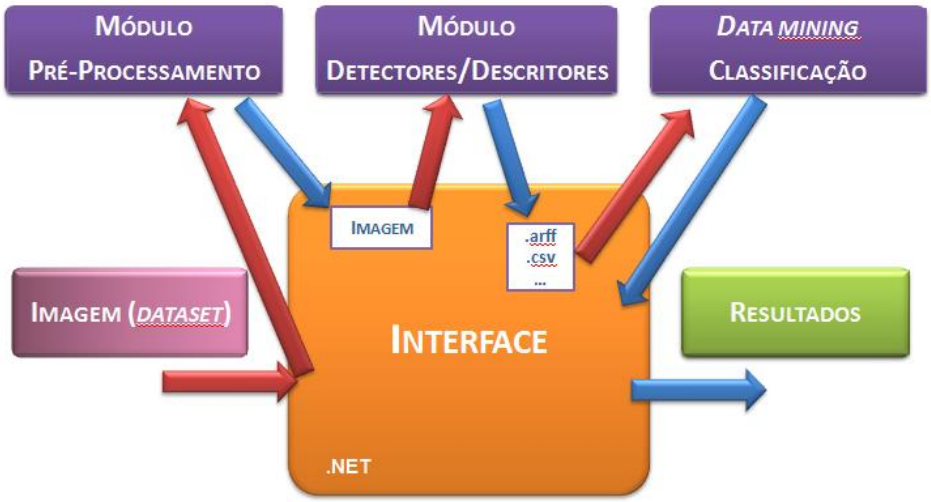
Licenciatura. ☐

Mestrado. ☐

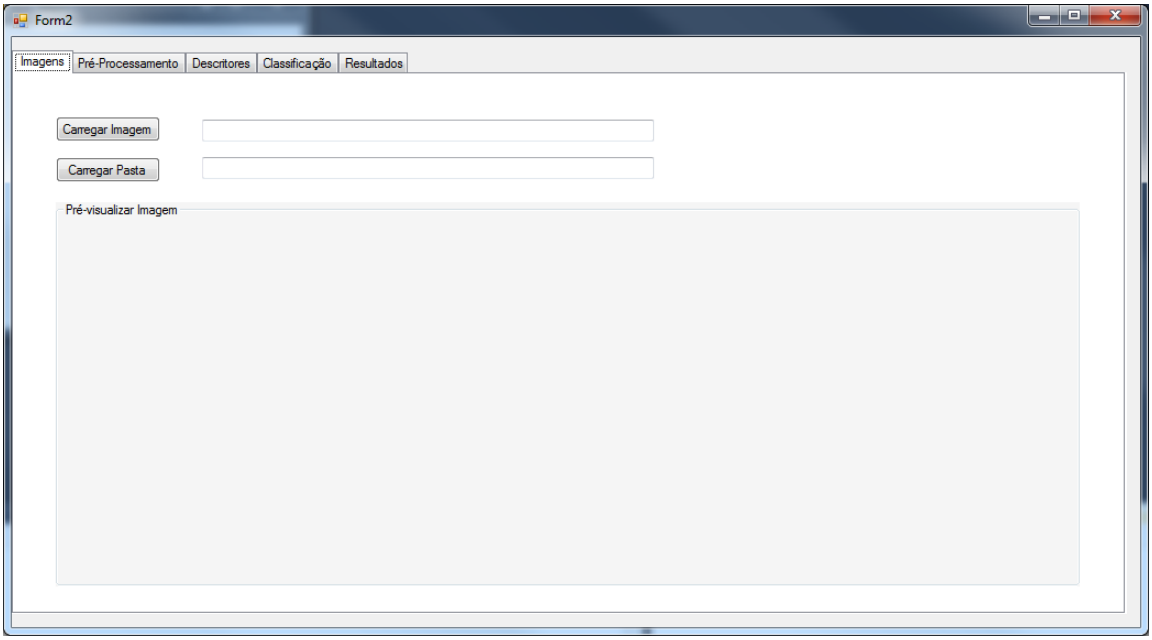
Doutoramento. ☐

Outro. ☐ Qual? _____

O processamento de imagem é um processo que comporta diferentes etapas. Pretende-se o desenvolvimento de uma aplicação que implemente de forma sequencial, numa única interface, as principais etapas do processamento de imagem. Essas etapas foram desenvolvidas através da implementação de módulos: o módulo de pré-processamento, o módulo de extração/descrição e o módulo de classificação. A aplicação deste teste deverá ter como *input* um conjunto de imagens, permitir algum pré-processamento básico na imagem (através da aplicação de um filtro), efectuar a extracção das características dessa imagem (através da aplicação de um descritor) e com base no classificador pretendido ter como *output* o resultado da classificação. O esquema seguinte mostra de uma forma geral o funcionamento desta ferramenta.



O protótipo utilizado para este teste tem a seguinte interface:



Cada separador *tab* da interface corresponde à implementação de uma etapa do processamento de imagem.

Tendo como base o protótipo da aplicação a desenvolver e depois de testar as suas funcionalidades, indique como classifica cada uma das seguintes questões:

5.	Compreensão e facilidade de utilização da aplicação desenvolvida.

Demorei algum tempo a compreender o funcionamento da aplicação. ☐

A aplicação é fácil de compreender, no entanto não considero fácil de utilizar. ☐

A aplicação é fácil de compreender e de utilizar. ☐

6.	Facilidade de memorização dos passos do processamento de uma imagem, depois de utilizar a aplicação e obter um resultado.

Esqueci-me de como o processo se efetuou. ☐

Lembro-me do processo de processamento, no entanto não considero de fácil memorização. ☐

O processo é de fácil memorização, lembro-me de todos os passos. ☐

7.	Apresentação dos separadores/*tabs* da aplicação.

Os separadores são apresentados de forma confusa. ☐

Não considero coerente a apresentação de alguns separadores ☐

Os separadores são apresentados de forma lógica. ☐

8.	Navegação entre os separadores/*tabs* relativamente à estrutura da aplicação.

A aplicação não está bem estruturada, o que dificulta a navegação. ☐

Apesar de a navegação ser fácil o conteúdo não está bem estruturado. ☐

É fácil de navegar entre os separadores, a aplicação está bem estruturada. ☐

9.	Quais as alterações que efetuaria à aplicação apresentada.