# Deteção Automática de Conteúdos Explícitos em Sequências de Vídeo

Victor Emanuel Teixeira Fernandes


Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

18 de Julho de 2014

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º Ano, do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Victor Emanuel Teixeira Fernandes, Nº 1090464, 1090464@isep.ipp.pt

Orientação científica: Paula Maria Marques Moura Gomes Viana, pmv@isep.ipp.pt

Empresa: MOG Technologies, S.A.

Supervisão: Alexandre Ulisses, alexandre.ulisses@mog-technologies.com
Supervisão: Pedro Ferreira, pedro.ferreira@mog-technologies.com




Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

18 de Julho de 2014

Esta página foi intencionalmente deixada em branco.

*"Make things as simple as possible,*
*but not simpler."*

Albert Einstein

Esta página foi intencionalmente deixada em branco.

# Agradecimentos

Em primeiro lugar, agradeço ao Eng.º Alexandre Ulisses e ao Eng.º Pedro Ferreira pela sua orientação e apoio. Agradeço à empresa MOG Technologies, S.A. e aos seus colaboradores, pela oportunidade da realização deste projeto em ambiente profissional. Agradeço ao Pedro Santos, responsável pelo desenvolvimento do módulo de controlo de qualidade, pelo seu apoio ao longo da fase de implementação do projeto no produto.

Agradeço também à minha orientadora, Prof. Paula Viana, por toda a ajuda, sugestões e disponibilidade que foram inexcedíveis ao longo do desenvolvimento deste projeto.

Agradeço a toda a minha família, amigos e colegas pela sua amizade, companheirismo e pelo apoio e força que me deram durante o meu percurso académico.

Finalmente, agradeço de forma muito especial aos meus pais e à minha irmã, que sempre acreditaram em mim e sempre me ajudaram em todas as situações ao longo destes últimos cinco anos.

A todos, o meu sincero obrigado.

Victor Fernandes

Esta página foi intencionalmente deixada em branco.

# Resumo

Nos últimos anos, o fácil acesso em termos de custos, ferramentas de produção, edição e distribuição de conteúdos audiovisuais, contribuíram para o aumento exponencial da produção diária deste tipo de conteúdos. Neste paradigma de superabundância de conteúdos multimédia existe uma grande percentagem de sequências de vídeo que contém material explícito, sendo necessário existir um controlo mais rigoroso, de modo a não ser facilmente acessível a menores.

O conceito de conteúdo explícito pode ser caraterizado de diferentes formas, tendo o trabalho descrito neste documento incidido sobre a deteção automática de nudez feminina presente em sequências de vídeo. Este processo de deteção e classificação automática de material para adultos pode constituir uma ferramenta importante na gestão de um canal de televisão. Diariamente podem ser recebidas centenas de horas de material sendo impraticável a implementação de um processo manual de controlo de qualidade.

A solução criada no contexto desta dissertação foi estudada e desenvolvida em torno de um produto especifico ligado à área do *broadcasting*. Este produto é o mxfSPEEDRAIL F1000, sendo este uma solução da empresa MOG Technologies.

O objetivo principal do projeto é o desenvolvimento de uma biblioteca em C++, acessível durante o processo de *ingest*, que permita, através de uma análise baseada em funcionalidades de visão computacional, detetar e sinalizar na *metadata* do sinal, quais as *frames* que potencialmente apresentam conteúdo explícito.

A solução desenvolvida utiliza um conjunto de técnicas do estado da arte adaptadas ao problema a tratar. Nestas incluem-se algoritmos para realizar a segmentação de pele e deteção de objetos em imagens. Por fim é efetuada uma análise critica à solução desenvolvida no âmbito desta dissertação de modo a que em futuros desenvolvimentos esta seja melhorada a nível do consumo de recursos durante a análise e a nível da sua taxa de sucesso.

**Palavras-Chave**

MOG Technologies, OpenCV, *Machine Learning*, Controlo de Qualidade Automático, Deteção de conteúdo explícito, Segmentação de Pele, Deteção de objetos em imagens.

Esta página foi intencionalmente deixada em branco.

# Abstract

In the last few years, we have been experiencing an exponential increase in the delivery of audiovisual content, fostered by new technology that enables diminishing production, editing and distribution costs. In this scenario of super abundance of audiovisual content, a large percentage of it may contain explicit content and this should be subject to a closer analysis, so that minors cannot access it easily. Although the explicit content can be characterized in several acts, this dissertation only dealt with the automatic detection of female nudity present in video sequences.

The solution developed in this dissertation was researched and developed around a specific product connected to the broadcasting market. This product is mxfSPEEDRAIL F1000, an ingest solution built at MOG Technologies.

The main objective of this project was the development of a C++ library to be used during the ingest process and that enabled, using computer vision approaches, to detect and label frames that potentially contain explicit content.

State of the art techniques, adapted to the problem under analysis, have been used to perform skin segmentation and object detection in images. A detailed analysis of the solution's performance was also done in order to implement further improvements in future releases.

**Keywords**

MOG Technologies, OpenCV, Machine Learning, Automatic Quality Control, Explicit Content Detection, Skin Segmentation, Object detection in images.

Esta página foi intencionalmente deixada em branco.

# Conteúdo

# Lista de Figuras

Esta página foi intencionalmente deixada em branco.

# Lista de Tabelas

Esta página foi intencionalmente deixada em branco.

# Acrónimos

| | |
|---|---|
| **API** | Application Programming Interface |
| **BSD** | Berkeley Software Distribution |
| **CPU** | Central Processing Unit |
| **DLL** | Dynamic-link library |
| **FTK** | Forensic Toolkit |
| **HMM** | Hidden Markov Models |
| **HSV** | Hue, Saturation, Value |
| **JPEG** | Joint Photographic Experts Group |
| **LBP** | Local Binary Patterns |
| **LGPL** | GNU Lesser General Public License |
| **LUT** | Look up Table |
| **MOG** | Media, Objects and Gadgets |
| **MOG QCM** | MOG Quality Control Manager |
| **MOG QC ECD** | MOG Quality Control Explicit Content Detector |
| **MLP** | MultiLayer Perceptron |
| **MPEG** | Moving Picture Experts Group |
| **MXF** | Material Exchange Format |
| **OpenCV** | Open Source Computer Vision Library |
| **SDK** | Software Development Kit |
| **SimpleCV** | Simple Computer Vision |
| **SMPTE** | Society of Motion Picture & Television Engineers |
| **SOM** | Self-Organizing Map |
| **SVM** | Support Vector Machine |
| **XML** | eXtensible Markup Language |

Esta página foi intencionalmente deixada em branco.

# 1 Introdução

Desde a sua criação no século XX, a televisão sofreu grandes modificações. A crescente inovação tecnológica a que estamos expostos na sociedade atual modificou totalmente a forma de funcionamento desta tecnologia. Estas modificações não foram só observadas no aspeto funcional deste equipamento, mas também na forma de produzir e difundir de conteúdo multimédia. Uma das maiores mudanças observou-se no processo de produção, com a evolução de sistemas baseados em cassete para sistemas baseados em ficheiros, otimizando assim todo o processo. Associado a esta grande inovação, assim como à proliferação das redes de comunicações e ao fácil acesso a meios de produção, edição e distribuição de conteúdos audiovisuais, verificou-se um aumento exponencial do conteúdo multimédia produzido e processado diariamente.

Uma quantidade não desprezável desses conteúdos contém uma determinada percentagem de imagens que incluem violência física, atos sexuais, nudez, insultos verbais, etc., exigindo assim um controlo de acesso mais rigoroso. No âmbito desta dissertação apenas serão consideradas questões relacionadas com o controlo de acesso a conteúdos explícitos que apresentem imagens de nudez feminina, sendo o principal objetivo facilitar a classificação automática do mesmo em sequências de vídeo, usando as funcionalidades oferecidas pelas técnicas da visão computacional.

Dada a quantidade de informação que diariamente é recebida por um canal de televisão, podendo chegar a centenas de horas de vídeo, o controlo manual é praticamente impossível de ser realizado, sendo por isso de extrema importância encontrar processos automáticos para identificar e sinalizar os conteúdos para adultos.

Desta forma, a MOG Technologies S.A., pretende atualizar o seu *software* com

funcionalidades que preencham esta lacuna observada neste nicho de mercado.

## 1.1 Contextualização

O projeto desenvolvido no contexto desta dissertação foi proposto pela empresa portuguesa MOG Technologies S.A. líder mundial em produtos e soluções para ambientes de pós-produção e difusão de conteúdos audiovisuais baseados em sistemas de ficheiros. A principal missão da MOG é assegurar a sua posição no mercado mundial como fornecedor de soluções centralizadas de *ingest*[1] e de ferramentas de desenvolvimento do formato Material Exchange Format (MXF) [1]. A principal característica dos seus produtos é a sua capacidade de converter qualquer tipo de formato de vídeo recebido num formato comum, conseguindo assim fornecer a interoperabilidade entre equipamentos em cenários de *broadcasting* e edição de vídeo em tempo real.

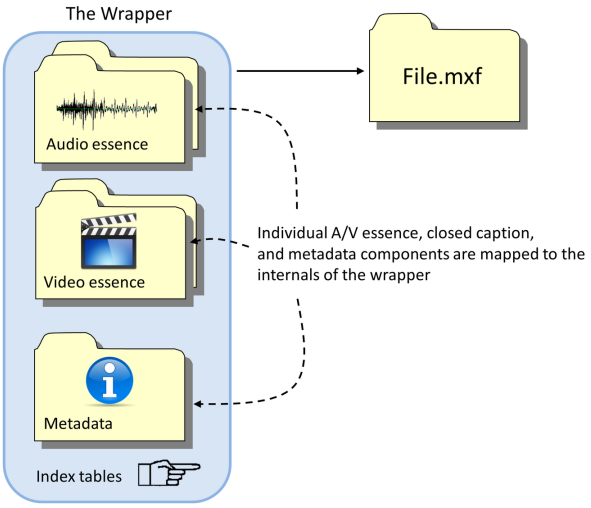


**Figura 1.1:** *Conteúdo de um ficheiro MXF [1]*

O MXF é um tipo de formato aberto, normalizado pela Society of Motion Picture & Television Engineers (SMPTE) que tem como principal objetivo promover a interoperabilidade entre as diferentes entidades do *workflow* de um operador de televisão, através da definição de um *standard* para o formato dos ficheiros utilizados. Este formato não influencia o tipo de codificação de vídeo ou áudio proveniente do ficheiro original, funcionado apenas como um *wrapper* (em português, empacotador) que é capaz de transportar dados referente ao aúdio, video e metadados (Figura 1.1).

O fato do MXF ter sido criado em parceria com a indústria de *broadcasting* tornou-o numa tecnologia muito eficiente e indispensável para este tipo de mercado [1, 2].

[1] O termo *ingest* no contexto de *broadcasting* refere-se ao processo de transferência de ficheiros para programas de edição de vídeo digital ou para sistemas de armazenamento de ficheiros.

Os produtos criados pela MOG Technologies S.A. são agrupados na família mxfSPEEDRAIL, sendo cada um dos produtos recomendado para processos distintos do *workflow* em que estão inseridos. De acordo com as necessidade dos clientes, estes podem escolher o produto da família mxfSPEEDRAIL que mais se adequa às suas necessidades.

No contexto desta dissertação a solução a ser desenvolvida é vocacionada para o produto mxfSPEEDRAIL F1000. O mxfSPEEDRAIL F1000 tem como entrada ficheiros de vídeo digitais que posteriormente podem ser transferidos para editores, dispositivos de armazenamento, pastas partilhadas ou servidores, preservando os metadados do ficheiro [3]. As suas principais características são:

- **Edição de vídeo em tempo real** - permite o uso de *software* de edição de vídeo enquanto o ficheiro está no processo *ingest*, reduzindo assim o tempo de espera do utilizador;
- **Capacidades de transcodificação** - devido a estas capacidades, o produto consegue promover a interoperabilidade entre diferentes equipamentos utilizados no *workflow* de pós-produção, pois é capaz de processar diferentes tipos de ficheiros de vídeo, homogeneizando o fluxo de informação;
- **Edição de vídeo** - utiliza processos simples de edição de ficheiro, tais como agregação de diferentes ficheiros e introdução de cortes;
- ***Ingest* em simultâneo** - permite processar vários ficheiros de vídeo em simultâneo de modo a conseguir minimizar o tempo de esperar associado a este processo;
- **Regras automáticas** - é possível configurar o sistema para que este automaticamente processe a informação e armazene a mesma numa localização pré-determinada;
- **Acesso remoto** - permite o controlo do equipamento remotamente, disponibilizando uma interface *web*, através de qualquer *browser* compatível com o Adobe Flash.

O objetivo principal desta dissertação é o desenvolvimento de uma biblioteca, que possa ser integrada no produto mxfSPEEDRAIL F1000, capaz de analisar o conteúdo de sequências de vídeo de forma automática. O principal interesse desta funcionalidade é a simplificação do processo de classificação do conteúdo multimédia assistindo o utilizador humano na sua tarefa de identificação e classificação de conteúdo explícito, permitindo assim reduzir significativamente o tempo necessário para classificar as centenas de horas de vídeo que chegam diariamente às estações de televisão.

Neste contexto é importante realçar a importância deste projeto para a MOG Technologies S.A., pois constitui a parte central de uma nova funcionalidade que esta empresa pretende implementar no seu produto. É também a primeira funcionalidade desenvolvida nesta empresa, utilizando técnicas de visão computacional, podendo assim ser considerado um projeto inovador que implica novos desafios e pode acrescentar um novo valor nos produtos desta empresa.

## 1.2 Objetivos

O objetivo principal deste projeto é a criação de uma biblioteca em C++, que seja acessível durante o processo de *ingest* dos conteúdos. Esta biblioteca funcionará como um *plugin* que será incorporado num módulo de controlo de qualidade. Este *plugin* irá analisar o conteúdo presente nas sequências de vídeo. Durante essa análise é necessário que o *plugin* consiga efetuar a deteção das imagens que potencialmente apresentam conteúdo explícito e *a posteriori* sinalizar as mesmas na *metadata* do ficheiro. Porém, ao longo deste projeto foi necessário estabelecer um conjunto de objetivos intermédios de modo a conseguir desenvolver o projeto de forma sequencial. Estes objetivos foram:

- Estudo das diferentes técnicas de segmentação e deteção de objetos em imagens, e seleção da que mais se adequa ao contexto desta dissertação;
- Criação de um mecanismo de deteção de conteúdo explícito em imagens ou em *frames* de sequências de vídeo;
- Integração da funcionalidade desenvolvida no produto mxfSPEEDRAIL F1000 da MOG Technologies, S.A.;
- Análise do consumo de recursos por parte do *plugin* desenvolvido, de modo a garantir o menor consumo de recursos possível, para que esta funcionalidade não influencie o tempo de resposta do mxfSPEEDRAIL F1000 durante o processo de *ingest* de conteúdos multimédia;

## 1.3 Estrutura do relatório

Este relatório está dividido em seis capítulos principais. Neste primeiro capítulo faz-se uma introdução ao projeto, focando a sua contextualização e os seus principais objetivos, apresentando a estrutura deste relatório.

O segundo capítulo é inteiramente dedicado à análise do estado da arte, onde as diferentes tecnologias envolvidas neste projeto são apresentadas e comparadas, de

modo a conseguir optar pela tecnologia que se enquadra melhor no contexto deste projeto.

No terceiro capítulo são apresentadas diversas *frameworks* associadas às funcionalidades de visão computacional. Uma vez que cada uma apresenta características distintas, este capítulo termina com uma análise comparativa entre as *frameworks*, de modo a ser possível optar pela que melhor se adapta às necessidades associadas ao desenvolvimento da solução descrita nesta dissertação.

O quarto capítulo aborda a implementação prática do projeto e as consequentes melhorias que tiverem que ser efetuadas, de modo a conseguir otimizar os resultados obtidos.

O quinto capítulo incide na integração do módulo desenvolvido no produto mxfSPEEDRAIL F1000 e na análise de recursos e tempo despendido na classificação das *frames* com conteúdo explícito.

Por fim, no último capítulo apresenta-se um resumo de todo o trabalho e discute-se o grau de satisfação em função dos objetivos implementados face aos inicialmente propostos. Também neste último capítulo são apresentadas as melhorias e futuras implementações que este projeto poderá vir a sofrer.

Esta página foi intencionalmente deixada em branco.

# 2

# Técnicas de análise e classificação automática de conteúdos em imagens

Associado ao crescimento do uso dos meios de comunicação, está o aumento da quantidade de conteúdo multimédia que é produzido diariamente. Verificar individualmente o conteúdo de cada ficheiro que é recebido nas estações televisivas, torna-se um enorme desafio para as equipas responsáveis pela classificação do material. O facto da televisão digital ser um meio de comunicação com uma grande adesão a nível mundial, torna necessária a deteção e censura de conteúdo explícito devido a razões éticas e sociais [4].

Apesar de existir um enorme número de trabalhos científicos relacionados com este tema, atualmente não existe nenhum produto comercial que consiga efetuar a deteção de conteúdo explícito no mercado do audiovisual profissional. Este facto deve-se à grande complexidade dos algoritmos que são normalmente utilizados para este fim, os quais são baseados em técnicas de visão por computador recorrendo a abordagens de inteligência artificial, tal como *machine learning*.A isto, acresce o elevado grau de dificuldade que estes algoritmos têm em obter bons resultados de classificação para todo o tipo de conteúdo. Isto é, não é trivial conseguir obter algoritmos genéricos que conduzem a taxas de sucesso satisfatórias qualquer que seja o conteúdo de vídeo em análise.

Inicialmente serão introduzidos conceitos de *Machine Learning* e segmentação de imagens. A temática de segmentação de imagens será aprofundada de modo a serem exploradas as diversas técnicas de segmentação de regiões de pele.

Após esta introdução serão descritas as várias metodologias utilizadas para a

deteção de objetos, que são essenciais para compreender o mecanismo de deteção de conteúdo explícito em imagens.

Existem diferentes metodologias para efetuar a deteção de conteúdo explícito. Nestas estão incluídas soluções que envolvem a utilização de técnicas de segmentação de pele e de deteção de objetos. As tecnologias são introduzidas neste capítulo, de forma a facilitarem a compreensão do seu funcionamento e dos conceitos associadas às mesmas.

## 2.1 *Machine Learning*

O termo *machine learning* refere-se a um campo da inteligência artificial cujo objetivo é converter dados em informação. Para conseguir efetuar esta transformação, os algoritmos de *machine learning* extraem padrões dos dados, de modo a estarem aptos para efetuar previsões e decisões segundo um determinado padrão. As funcionalidades fornecidas por esta tecnologia são utilizadas em vastas áreas, como por exemplo: visão computacional, motores de busca, sistemas de recomendação, economia, reconhecimento de texto, recomendação de publicidade, entre outras [5].

Devido à sua grande expansão, atualmente existem diferentes algoritmos de *machine learning*. Na tabela 2.1, estão descritos os principais tipos de aprendizagem associados com esta tecnologia.

**Tabela 2.1:** *Descrição dos diferentes tipos de aprendizagem utilizados em machine learning*

| **Tipo de Aprendizagem** | **Descrição** |
|---|---|
| Aprendizagem supervisionada | Ao longo do seu funcionamento, este mecanismo de aprendizagem baseia-se num conjunto de dados com entradas e saídas já conhecidas. Assim, é construido um modelo que gera um conjunto de previsões mais precisas quando introduzidos novos dados, otimizando assim o processo de classificação [6]. |
| Aprendizagem não supervisionada | Esta abordagem tenta organizar os dados em diferentes classes, de modo a que os dados contidos nessas consigam ser correlacionados entre si. Para efetuar a classificação este tipo de aprendizagem, tenta encontrar um padrão de modo a conseguir classificar corretamente a informação. O modelo de decisão é criado com base na análise das propriedades das saídas [7]. |
| Aprendizagem por reforço | Tal como um ser humano aprende com a experiência, este algoritmo desenvolve um modelo de acordo com a interação que este realiza com os dados. A este tipo de aprendizagem está associado um agente que efetua diversas interações com os dados e analisa os resultados de cada experiência. Em cada experiência este método associa uma recompensa ou uma penalização. Com base nestas ações é gerado um modelo de classificação, onde é maximizada a diferença entre as recompensas e as penalizações [8]. |

Embora exista um elevado número de algoritmos de *machine learning*, apenas serão detalhados os algoritmos mais importantes para o desenvolvimento da solução descrita nesta dissertação: o Support Vector Machine (SVM) e o *AdaBoost*. Estes algoritmos são bastante utilizados na área de visão computacional, devido à sua rápida resposta de classificação, sendo o seu foco a deteção de classes de objetos em imagens.

### 2.1.1 Máquinas de Vetores de Suporte (SVM)

O algoritmo SVM foi criado por Vladimir Vapnik em 1995, sendo utilizado em diversos tipos de aplicações. O método de aprendizagem associado a este algoritmo enquadra-se nos métodos de aprendizagem supervisionada. Durante o processo de aprendizagem, são criados classificadores lineares e não estatísticos. Este algoritmo representa os dados como pontos num espaço dimensional.

Para ser possível distinguir os dados em duas classes distintas, é definido um hiperplano. O hiperplano é o limite que separa e distingue, todos os pontos de uma classe dos pontos de outra classe. O cálculo do hiperplano ótimo baseia-se na largura de margem. Esta margem é definida com base nos vetores de suporte, consistindo na largura máxima entre os vetores de suporte das diferentes classe e o hiperplano. Os vetores de suporte são os pontos dos dados que estão mais próximos do hiperplano (ver figura 2.1) [9].

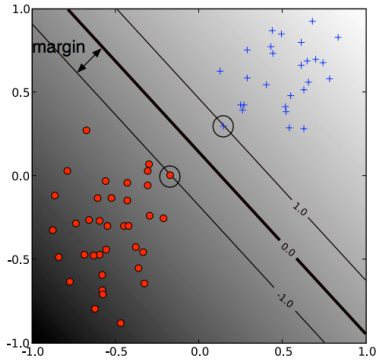


**Figura 2.1:** *Classificação linear, os dados identificados por círculos representam os vetores de suporte [10]*

O objetivo principal deste algoritmo centra-se na definição de um hiperplano ótimo, de modo a que este consiga separar as diferentes classes de dados com a maior margem possível.

No entanto, nem sempre é possível efetuar uma separação linear entre as diferentes classes de dados. Nos casos em que se torna difícil gerar uma separação linear,

utilizam-se as funções de *kernel* para que seja efetuada a transformação para um domínio onde seja possível a definição de um hiperplano linear que minimize os erros de classificação. Este processo está representado na figura 2.2 [9, 10].

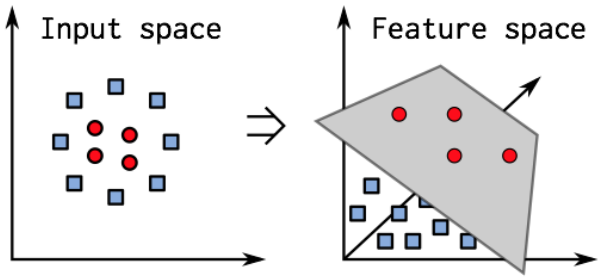


**Figura 2.2:** *Transformação de um domínio não linear para um domínio linear utilizando as funções de kernel [11]*

Mesmo com as transformações, por vezes é necessário otimizar a definição do hiperplano podendo esta definição ser efetuada recorrendo a diferentes métodos. Um dos métodos mais utilizado para a resolução deste problema é o *kernel* polinomial ou uma função de base radial.

O algoritmo SVM é ideal para classificar grandes volumes de dados, devido à rapidez do seu processo de classificação.

A principal desvantagem deste algoritmo é o facto de este apenas conseguir classificar duas classes de dados, quando a maior parte dos problemas reais apresentam diversas classes. Utilizando o algoritmo SVM é necessário, durante o processo de treino, dividir as múltiplas classes em conjuntos de dois de modo a ser possível ultrapassar esta limitação [9, 10, 11].

### 2.1.2 *AdaBoost*

O algoritmo de *Adaptative Boosting (AdaBoost)* foi criado por Y.Freud e R. Shapire em 1995, de modo a melhorar os problemas e as dificuldades apresentadas pelo algoritmo *Boosting* [13]. A principal característica, que distingue este algoritmo dos restantes, é o facto de este criar um classificador forte com base num conjunto de classificadores fracos [13, 12].

Durante o processo de criação do classificador forte, são atribuídos diferentes pesos aos classificadores fracos. O classificador forte *H(x)* (classificador final) é descrito pela seguinte equação, onde $\alpha$ representa peso atribuído a cada classificador fraco *h(x)*:

$$H(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + ... + \alpha_n h_n(x) \tag{2.1}$$

Como é possível visualizar na figura 2.3 (onde o parâmetro $t$, representa o número

de classificadores fracos), a classificação torna-se muito mais exata quanto maior for o número de classificadores fracos utilizados no processo de treino do classificador. O facto de serem atribuídos diferentes pesos aos classificadores tornam o resultado da classificação mais exato, estando associado ao mesmo uma pequena percentagem de erro [13, 14]. Teoricamente, o erro do processo de aprendizagem deste algoritmo tende para zero, se o número de classificadores fracos for infinito.

O algoritmo *AdaBoost* apresenta a vantagem de não necessitar do um conhecimento prévio do tipo de dados que necessita de classificar. O facto do algoritmo funcionar de forma adaptativa, torna possível melhorar a classificação em cada etapa de treino. Porém também apresenta a desvantagem do processo de treino ser bastante demorado e depender da qualidade dos dados utilizados durante o treino [15].

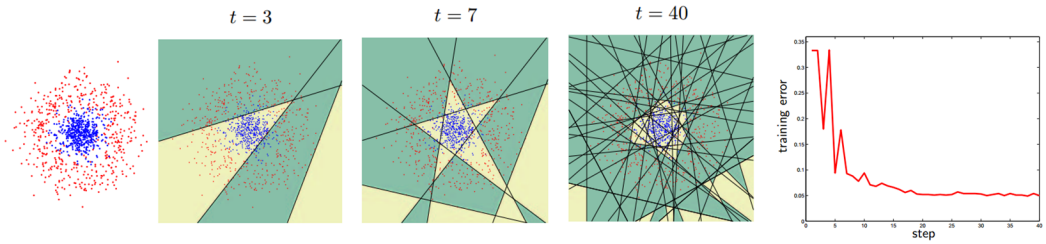


**Figura 2.3:** *Erro de classificação em relação ao número de classificadores fracos. Informação adaptada de [14] associada com o artigo [16]*

Este algoritmo ganhou uma maior importância depois de ser incorporado num dos métodos mais utilizados atualmente para efetuar a deteção de faces em imagens em tempo real [17].

## 2.2 Segmentação de Pele

A segmentação permite realizar a decomposição de uma imagem em regiões ou objetos. Geralmente, quando se pretende identificar um objeto com determinadas características em imagens, utiliza-se como primeiro passo a segmentação. Este processo permite diminuir a região em que o sistema deve efetuar a pesquisa [18, 19].

Embora para um ser humano a tarefa de detetar diferentes regiões e objetos em imagens seja muito intuitiva, a mesma não é de fácil execução para sistemas computacionais, pois envolve muito processamento e cálculos intermédios. Devido à importância que representa para a visão computacional, a segmentação é uma das funcionalidades mais estudadas nesta área. Ao longo dos anos esta tem sido muito desenvolvida, dando origem ao aparecimento de diversas técnicas de segmentação, cada uma com uma aplicação distinta de acordo com a situação.

Os algoritmos de segmentação baseiam-se nas propriedades dos valores de intensidade da imagem. Assim, estes algoritmos pode ser agrupados da seguinte forma

[18]:

- **Segmentação baseada na descontinuidade** - a divisão da imagem é baseada em mudanças de intensidade muito acentuadas;

- **Segmentação baseada na semelhança** - a divisão da imagem é baseada em regiões que apresentam semelhanças.

Embora existam diversas técnicas de segmentação em imagens, no contexto desta dissertação, a técnica de *Thresholding* foi a mais indicada para efetuar a segmentação de pele. Esta técnica é uma das mais simples e uma das mais utilizadas. Este algoritmo separa regiões através das variações de intensidade presentes nas imagens. Geralmente é um bom método para eliminar a informação de fundo de uma imagem [20]. O processo de *thresholding* está associado a uma imagem de escala de cinzentos (*f(x,y)*), onde é definido um valor máximo de variação, o denominado *threshold*. Quando o valor é igual ou superior ao valor de *threshold* (T), o pixel toma o valor 1 (um), quando o valor é menor em relação ao limite (*threshold*) o pixel toma o valor 0 (zero).

$$g(x,y) = \begin{cases} 1, & f(x,y) \geq T \\ 0, & f(x,y) < T \end{cases} \tag{2.2}$$

Assim numa imagem em tons de cinza ou a cores, o resultado do processo de *thresholding* resume-se a uma imagem binária (*g(x,y)*). Como é possível verificar na figura 2.4, depois de utilizando o método de *thresholding* para realizar a segmentação de pele é obtida uma imagem binária onde a região a branco representa a cor associada à pele humana e a região preta apresenta-se como a região onde não existem pixeis associados à informação de cor de pele. Dentro desta técnica de segmentação existe o *thresholding* adaptativo, onde o valor de *threshold* é adaptado através de iterações, de modo a tornar o processo de segmentação o mais robusto e eficaz possível [19].

**Figura 2.4:** *Exemplo de segmentação de pele (a imagem da esquerda representa a fotografia original, a imagem da direita representa o resultado da segmentação de pele)*

A segmentação de imagens pode ser efetuada através do uso de outros algoritmos. Um algoritmo muito utilizado é a deteção de contornos que pode ser efetuada através do cálculo da 1ª e 2ª derivada, ou através da utilização das transformadas de Hough. Outro algoritmo utilizado para o processo de segmentação é o crescimento de regiões (em inglês, *region growing*), baseia-se na agregação de pixeis que apresentam semelhanças entre pixeis presentes na sua vizinhança, formando assim regiões que partilham as mesmas propriedades [18].

O problema da segmentação de pele é colocado em diversas áreas tecnológicas. Na publicação [21], o autor apresenta uma solução de apoio à avaliação do resultado estético do tratamento conservador do cancro da mama, e parte do *software* desenvolvido na sua publicação baseia-se em segmentação de pele. Outra área de aplicação desta tecnologia é a área de segurança, onde a segmentação de pele é utilizada para efetuar a deteção e o acompanhamento de pessoas e faces em imagens [22].

Na área de automação, a segmentação de pele está principalmente associada ao reconhecimento de humanos e de gestos, sendo assim utilizada para melhorar o processo de interação e o controlo de robôs, tornando assim a comunicação entre estes mais fácil e intuitiva para os humanos.

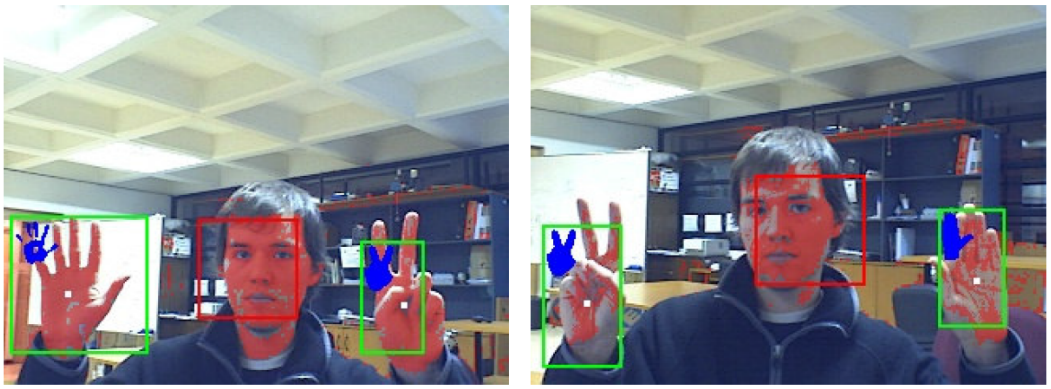

**Figura 2.5:** *Exemplo de reconhecimento de gestos usando a segmentação de pele [23]*

O facto de a pele humana apresentar uma gama de cores muito diversificadas, torna a sua deteção num grande desafio para a área da visão computacional. Embora o seu principal problema seja a grande variedade de cores, existem outros problemas que dificultam a deteção de pele em imagens.

### 2.2.1 Problemas associados aos algoritmos de deteção de pele

No artigo [24] os autores descrevem que a cor da pele é influenciada por dois fatores, sendo estes a cor do sangue (predominância da cor vermelha) e a cor da melanina (variação de cores entre o amarelo e o castanho).

Embora sejam estas as principais causas que condicionam a cor de pele, existem

outros fatores que tornam a segmentação da pele, numa tarefa de difícil execução. Estes são [25]:

- **Iluminação** - a iluminação tem uma forte influência na cor da pele, pois a variação da intensidade e da cor da luz ambiente produzem características inconstantes na cor da pele, como está representado na figura 2.6. Outro factor associado à iluminação é a existência de sombras que alteram a cor da pele;

**Figura 2.6:** *Exemplos de situações em que a luminosidade influencia a informação da cor de pele*

- **Etnia** - o facto de existirem vários grupos de pessoas com etnias diferentes, torna-se um problema para a segmentação de pele, pois estes apresentam diferentes cores de pele (p.e. Africanos, Asiáticos e Caucasianos), traduzindo-se numa maior a gama de cores associadas à pele humana;

- **Características individuais** - fatores como a idade, o sexo e regiões do corpo também afetam a cor característica da pele;

- **Características da câmara** - a cor da pele pode ser influenciada pelo sensor do equipamento que capta a informação e traduz essa informação numa imagem digital. Mesmo sobre as mesmas condições de iluminação, diferentes câmaras podem apresentar cores diferentes para representar a pele da mesma pessoa;

- **Movimentação do objeto** - quando existe movimentação de um objeto numa imagem este causa sempre um certo desfoque nas cores vizinhas, o que pode influenciar o processo de segmentação;

- **Outros fatores** - existem fatores que influenciam a cor da pele na imagem, como a cor de fundo da imagem, maquilhagem, etc.

Embora já existam classificadores de pele robustos, é expectável que, ao processo de segmentação esteja sempre associado um erro devido à enorme complexidade desta tarefa e à existência de vários fatores incontroláveis.

Uma possível solução para minimizar os problemas associados ao processo de segmentação de pele seria a utilização de câmaras infravermelhas. No entanto, devido

ao seu elevado custo, essa solução apenas é utilizada em situações muito específicas, tais como nas áreas da biomédica [25] ou da segurança. Na tentativa de ultrapassar alguns dos problemas descritos, optou-se por realizar estudos de implementação de classificadores de pele humana em diferentes espaços de cor. Os resultados são apresentados na secção seguinte.

### 2.2.2 Espaços de cor utilizados na segmentação de pele

A eficiência do processo de segmentação está relacionada com o espaço de cor utilizado, não devendo a escolha ser efetuada de forma aleatória. A escolha deve ser efetuada de modo a reduzir falsas deteções e conseguir diferenciar todos os tons de pele. Para tal deve ser dada preferência aos que reúnam as seguintes características [26]:

- **Separação da componente de iluminação das componentes de cor**, eliminando a informação de iluminação o processo de segmentação é mais preciso;
- **Deve concentrar todas as variações de cor de pele em apenas uma área reservada desse espaço de cor**.

Embora ao processo de segmentação de pele estejam associados alguns espaços de cor onde a componente de iluminação é separada das restantes componentes de crominância, são poucos aqueles que conseguem concentrar todas as variações da tonalidade de pele numa única região. Os espaços de cor que apresentam as características ideais para a segmentação de pele são o YCbCr e o HSV, sendo estes os mais utilizados neste processo [26].

O espaço de cor YCbCr está incluído no grupo utilizado para a transmissão de sinais de televisão (*broadcasting*), onde também está incluído o espaço de cor YUV. Nestes dois espaços de cor a imagem é decomposta em 3 canais diferentes, sendo o canal Y associado à informação de luminância (intensidade da luz) e os canais Cb,Cr e U,V associados à crominância (informação da cor).

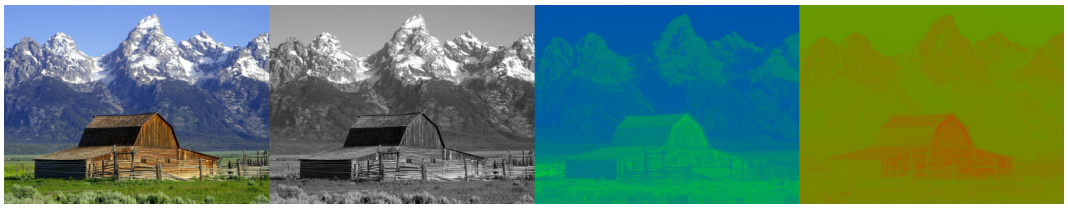

**Figura 2.7:** *Diferentes canais compõem uma imagem que utiliza o espaço de cor YUV [27]*

O espaço de cor HSV (Hue, Saturation, Value) fornece uma representação mais próxima daquela que é utilizada pelo sistema de visão humano para interpretar a cor, tornando mais intuitiva a sua manipulação. O canal H representa a matiz

ou tonalidade, o canal S representa o grau de pureza da tonalidade (p.e. : rosa *versus* vermelho) e o canal V representa a intensidade da luz. O facto de conter a informação relativa à luminosidade separada da informação de cor e apresentar uma transformação linear para o espaço de cor RGB, tornou-o num dos mais utilizados em aplicações de processamento de imagem.

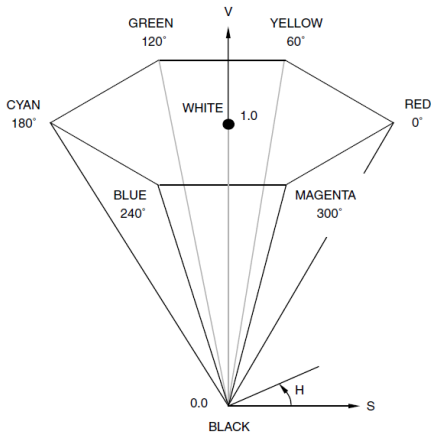


**Figura 2.8:** *Espaço de cor HSV [27]*

### 2.2.3 Técnicas de Classificação de Pele

Embora a escolha do espaço de cor, de certa forma, determine a eficiência do processo de segmentação, esta não é suficiente para colmatar as dificuldades na identificação das regiões de pele em imagens. Desta forma, serão apresentados métodos que são capazes de realizar a classificação de grupos de pixeis como regiões de pele ou de não pele. Algumas das técnicas mais utilizadas para realizar a classificação de pele são [21, 25, 28]:

- **Definição de uma região num determinado espaço de cor** - esta metodologia passa por designar um conjunto de limites no espaço de cor utilizado, de modo a formar duas regiões: *(i)* uma que contenha cores associadas à informação de pele; *(ii)* outra que contenha informação não relacionada com pele. Estes limites variam de acordo com o espaço de cor utilizado para o processo de classificação. O facto deste método ser simples de implementar, apresentar uma rápida execução, não necessitar de um consumo exaustivo de recursos computacionais e apresentar bons resultados, torna-o num método de classificação de pele bastante utilizado;

- **Utilização de uma Look up Table (LUT) normalizada** - utilizando um conjunto de imagens de teste contendo informação de pele, é criado um histograma para cada componente do espaço de cor. Após a normalização dos

histogramas são armazenadas em memória todas as combinações que apresentam informação de pele, sobre a forma de uma LUT. Geralmente os espaço de cor são compostos por três componentes, tendo assim a LUT três dimensões. Porém em espaços de cor que apresentam a separação da informação de luminosidade e da informação de cor é usual eliminar o canal associado à informação de luminosidade, visto que apenas é necessária a informação de cor. Desta forma a informação associada à pele na LUT sofre uma redução para duas dimensões;

- **Classificador Bayesiano** - este classificador baseia-se na utilização do método de Bayes:

$$P(pele|x) = \frac{P(x|pele).P(pele)}{P(x|pele).P(pele) + P(x|não - pele).P(não - pele)} \tag{2.3}$$

onde $x$ representa a cor do pixel. Este método também pode ser utilizado na sua forma simplificada:

$$\frac{P(x|pele)}{P(x|não - pele)} \geq \Phi \tag{2.4}$$

onde $\Phi$ traduz o limiar de decisão do classificador. Neste método é utilizado um histograma de duas ou três dimensões para representar a distribuição de tons de pele num determinado espaço de cor. Em relação aos outros métodos, este e o classificador Gaussiano utilizam probabilidades, o que confere aos mesmos uma maior imunidade ao ruído;

- **Classificador Gaussiano** - este método normaliza um histograma de duas dimensões (a componente de iluminação é eliminada) com base na distribuição gaussiana:

$$P(c|pele) = \frac{1}{2\pi|\sum s|^{1/2}} \cdot e^{-\frac{1}{2}(c-\mu_s)^T \sum s^{-1}(c-\mu_s)} \tag{2.5}$$

onde $c$ representa o vetor de cor, e $\mu_s$ e $\sum s$ são os parâmetros associados à distribuição gaussiana (vetor médio e matriz de variância, respetivamente). Este tipo de classificação pode ser utilizada de forma simples, ou através do uso de várias distribuições gaussianas;

- **Utilização de algoritmos de *machine learning*** - a classificação de dados como classes de pele e não pele é um problema que pode ser resolvido com a utilização de algoritmos de *machine learning*. Associados a este tema estão dois algoritmos de *machine learning*, sendo estes: o Self-Organizing Map (SOM) e MultiLayer Perceptron (MLP). O algoritmo SOM (em português, Mapas auto-organizados) é um algoritmo de *machine learning* que apresenta um método de aprendizagem não supervisionada, sendo necessário identificar quais as saídas

do conjunto de dados estão relacionadas com a informação de pele. O algoritmo MLP (em português, Perceptrão multicamada) está associado a um método de aprendizagem supervisionado. Os resultados associados à utilização de algoritmos de *machine learning* têm a vantagem de serem invariáveis de acordo com o espaço de cor utilizado.

Atualmente para efetuar a classificação de pele em imagens, os sistemas tendem a utilizar métodos simples e de rápida execução, como é o caso do método baseado na definição de uma região no espaço de cor.

## 2.3 Deteção de Objetos

A deteção de objetos é uma das áreas associada à visão computacional responsável pela análise visual da imagem de modo a conseguir identificar objetos. Os algoritmos de deteção de objetos utilizam mecanismos de *machine learning*, de maneira a conseguirem reconhecer diferentes instâncias de uma categoria de objetos. Esta funcionalidade da visão computacional abrange uma vasta gama de áreas tecnológicas sendo aplicada em diversos cenários do quotidiano, que vão desde: segurança rodoviária [29, 30, 31], segurança [32, 33], monitorização de sistemas de vídeo-vigilância [34, 35], rastreio de doenças [21], automação [36], deteção de faces [17, 37, 38], entre outras.

No âmbito da solução a desenvolver, a escolha do algoritmo de deteção de objetos, foi determinada principalmente pelas duas seguintes características desejáveis: velocidade de deteção e grau de precisão. Os métodos de deteção de objetos em imagens que mais se adaptaram a esta necessidade foram: o mecanismo de funcionalidades Haar-like associada ao algoritmo de *machine learning Adaboost* (*framework* de Viola-Jones) e Local Binary Patterns (LBP).

**Figura 2.9:** *Deteção de sinais de trânsito recorrendo à framework de Viola-Jones [29]*

### 2.3.1 *Framework* de Viola-Jones

*Viola* e *Jones* [17] descrevem um mecanismo que garante o reconhecimento de objetos em imagens em tempo real e com um elevado grau de sucesso. Este método é capaz de reconhecer qualquer tipo de objeto, sendo no trabalho [17] utilizado para efetuar a deteção de faces.

Para efetuar a deteção, esta *framework* baseia-se no uso das seguintes componentes:*(i)* Cálculo da imagem integral, de modo a tornar o processo de deteção mais rápido; *(ii)* Características *Haar-like Features*; *(iii)* Utilização do algoritmo de *machine learning Adaboost*; *(iv)* Uso de classificadores em cascata.

Neste mecanismo, a deteção de objetos é efetuada através do uso de características simples. Estas características são utilizadas de modo a reduzir o tempo associado ao processo de deteção de objetos, pois o número de características associadas a um objeto é inferior ao número de pixeis do objeto. Os três tipos de características utilizados são: *(i)* características retangulares de dois retângulos (figura 2.10 A e B) *(ii)* características retangulares de três retângulos (figura 2.10 C); *(iii)* características retangulares de quatro retângulos (figura 2.10 D).

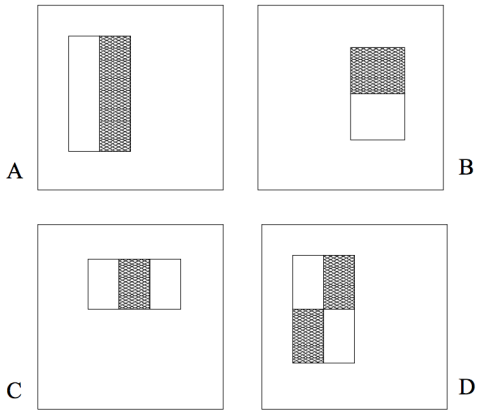


**Figura 2.10:** *Características Haar-like features [17]*

Estas características possuem o formato de retângulos e são denominados por características "Haar-like Features", estas são consideradas necessárias para decompor as imagens em informação relevante para definir a forma de um objeto [17, 37].

Cada característica contem o valor resultante da subtração do somatório dos retângulos brancos com o somatório dos retângulos pretos [17, 37]. Para melhorar a eficiência deste método, os autores do artigo [39] utilizaram as características *Haar-like features* na diagonal, originando assim novos tipos de características (figura 2.11). Com esta modificação foi possível reduzir o número de falsos positivos associados ao processo de deteção de objetos para cerca de 10%.

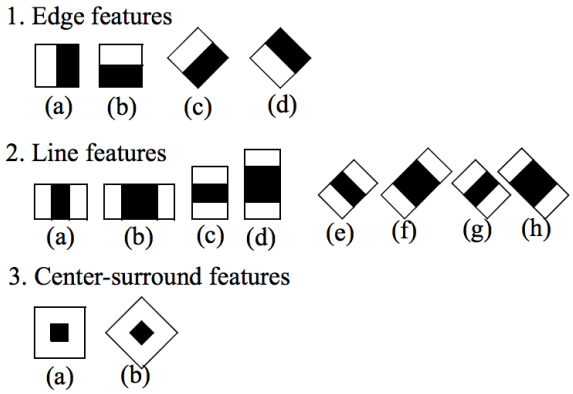


**Figura 2.11:** *Extensão das características Haar-like features propostas por Lienhart e Maydt [39]*

De modo a tornar o processamento das características retangulares mais rápido e eficiente, esta *framework* utiliza uma representação intermédia da imagem original, denominada por imagem integral. Na imagem integral, cada ponto (x,y) é obtido através da soma de pixeis da imagem original. Assim, para cada ponto da imagem integral verifica-se a formula 2.6, onde *ii(x,y)* representa a imagem integral e *i(x',y')* representa a imagem original.

$$ii(x,y) = \sum_{x \leq x', y \leq y'} i(x', y') \tag{2.6}$$

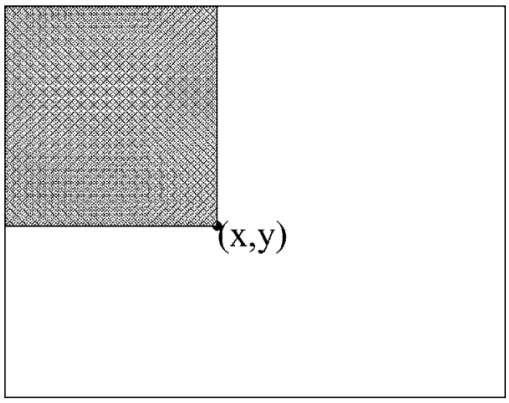


**Figura 2.12:** *Exemplo do cálculo da imagem integral no ponto x,y [37]*

A soma de pixeis associados à região D (figura 2.13) pode ser calculada através do uso de quatro posições distintas. O valor da posição 1 é obtido através a soma de todos os pixeis associados à região A. Da mesma forma o valor da posição 2 é obtido através da soma dos pixeis das regiões A e B. Na posição 3, a soma de pixeis é obtida através da adição dos pixeis associados às regiões A e C. Por fim, o valor

da posição 4 é obtido através da operação A+B+C+D (onde cada letra representa a região descrita na figura 2.13). Assim, desta forma é possível calcular a soma de pixeis na região D, utilizando a seguinte operação, onde os números representam as posições mencionadas anteriormente (4+1)-(2+3) [37].

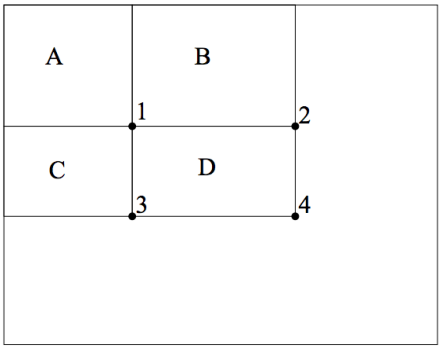


**Figura 2.13:** *Cálculo de uma determinada região da imagem utilizando a imagem integral [17]*

Para a criação de classificadores, a *framework* de Viola-Jones utiliza o algoritmo de *machine learning AdaBoost*. Embora o processo de deteção de objetos seja rápido, este mecanismos apresenta a desvantagem de o processo de treino do classificador ser demasiado moroso. A taxa de falsos positivos associada ao classificador pode ser calculada através da seguinte equação:

$$F = \prod_{i=1}^{K} f_i \tag{2.7}$$

onde $F$ representa a taxa de falsos positivos, $K$representa o número de classificadores fracos utilizados na cascata e finalmente $f_i$ representa a taxa de falsos positivos de cada classificador fraco. Para calcular a taxa de sucesso de deteção utiliza-se a seguinte equação:

$$D = \prod_{i=1}^{K} d_i \tag{2.8}$$

Tal como a equação anterior, $K$representa o número de classificadores fracos utilizados, $D$ representa a taxa de sucesso na deteção e por fim $d_i$ representa a taxa de sucesso de cada classificador fraco.

O processo de procura do objeto na imagem é efetuado por uma janela de deteção. Esta janela percorre inúmeras vezes a imagem, com diferentes escalas. Para percorrer a imagem, cada janela de deteção é deslocada segundo um determinado intervalo de pixeis na horizontal e depois na vertical. O dimensionamento da janela de procura, deve ser feito sempre com especial atenção às dimensões utilizadas durante o processo

de treino da cascata de classificadores (sendo necessário que as dimensões da janela de procura sejam sempre iguais ou maiores em relação às dimensões utilizadas no treino) [17].

A cascata de classificadores é composta por várias etapas, sendo cada uma delas associadas a um classificador fraco. Esta propriedade torna a classificação mais robusta, diminuindo sucessivamente o número de falsos positivos em cada etapa. Na figura 2.14 está apresentado o funcionamento da cascata de classificadores. Sempre que numa das etapas do processo de deteção o resultado da classificação seja determinado como uma classe que não pertence ao objeto, o processo de deteção nessa janela termina, poupando assim recursos computacionais e prosseguindo para a janela de deteção seguinte [17].

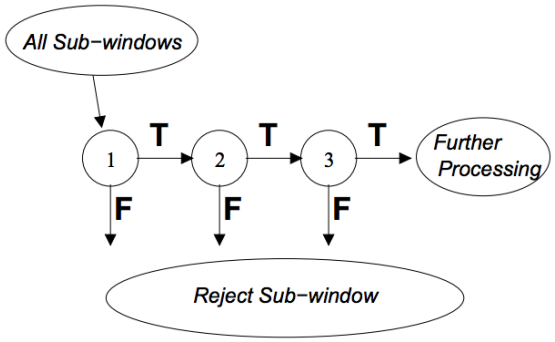


**Figura 2.14:** *Etapas associadas ao processo de classificação, utilizando o método de cascata de classificadores [17]*

Embora esta técnica de deteção de objetos seja referenciada na deteção de faces em imagens, esta pode ser treinada para efetuar deteções dos mais diversos objetos, tendo sempre associado uma grande taxa de sucesso no processo de deteção.

### 2.3.2 Local Binary Patterns

O desenvolvimento do método LBP proposto em [40] foi baseado no método proposto em [17] por Viola-Jones. O que distingue este método é a utilização de características LBP, ao invés de características *Haar-like Features*, tornando o processo de deteção mais rápido. O LBP é um mecanismo que apresenta um bom desempenho na identificação de textura e formas, sendo possível utiliza-lo na deteção de objetos [38, 41, 42].

O operador LBP identifica os pixeis da imagem através de um cálculo, como está especificado na figura 2.15. Este cálculo é efetuado em cada pixel com base nos pixeis vizinhos. É utilizada uma matriz 3x3 em que o pixel a ser identificado pelo operador LBP está localizado no centro da matriz. O cálculo associado à identificação do pixel passa por um processo de *threshold*, ou seja, quando os valores dos pixeis vizinhos

são iguais ou superiores aos valores do pixel a ser identificado, estes tomam o valor 1, caso contrário tomam o valor 0. Depois do processo de *threshold*, ao pixel central é associado a um número binário e é efetuada a conversão de base 2 para base 10 [42].

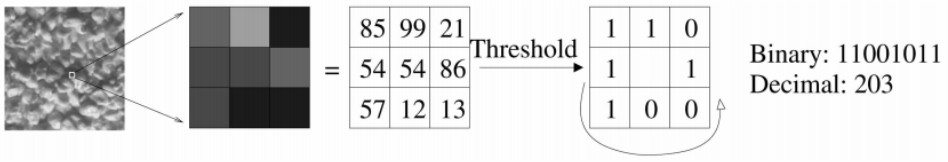


**Figura 2.15:** *Exemplo do cálculo operador LBP [43]*

De forma a conseguir representar o objeto, este método subdivide cada imagem em pequenas regiões independentes, e em cada uma destas regiões são analisados e obtidos os histogramas com base no cálculo do operador LBP. Após a análise de todas estas sub-regiões os histogramas são concatenados num único histograma. que vai ser capaz de descrever a textura e a forma global do objeto a ser identificado [41].

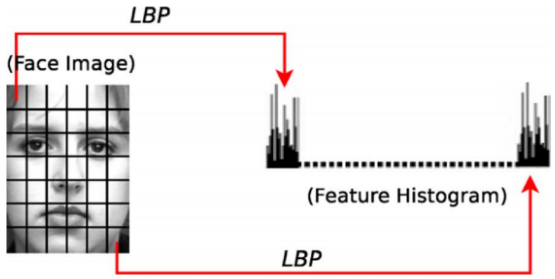


**Figura 2.16:** *Processo de concatenação dos histogramas das sub-regiões da imagem [41]*

Tal como na *framework* de Viola-Jones, é utilizado o algoritmo *AdaBoost* para a criação do classificador e uma cascata de classificadores no processo de deteção.

No artigo [42] este método foi aplicado para efetuar a deteção de faces em imagens, mas tal como a *framework* de Viola-Jones (Haar), este pode ser treinado e aplicado em qualquer tipo de objetos. Em relação à *framework* de Viola-Jones, esta metodologia é mais rápida apresentando, no entanto, resultados inferiores na classificação de objetos.

## 2.4 Deteção de conteúdo explícito

Como é possível verificar na figura 2.17, nos últimos anos existiu um aumento significativo na publicação de trabalhos científicos relacionados com a deteção de conteúdo explícito (nudez), o que demonstra que existe cada vez mais uma maior

procura e necessidade de conseguir desenvolver mecanismos eficientes. A figura 2.17 também transmite a importância que este tema tem tido nos últimos anos perante a comunidade científica.

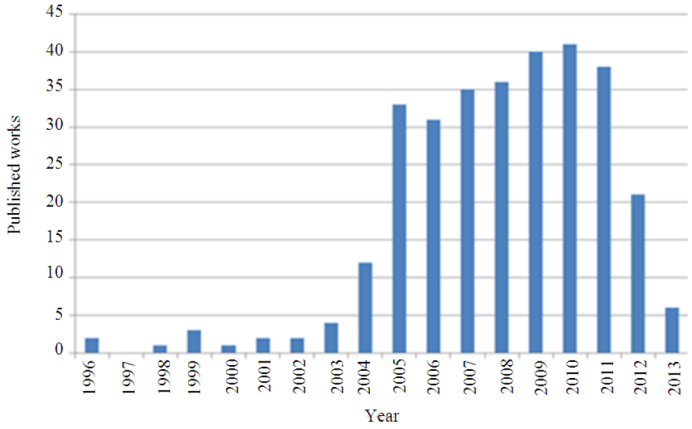


**Figura 2.17:** *Trabalhos científicos relacionados com deteção de conteúdo explícito publicados entre 1996-2013 [4]*

Atualmente existe um número reduzido de mecanismos que permitem efetuar a deteção de material pornográfico em sequências de vídeo. A maior parte das soluções de deteção de conteúdo explicito são desenvolvidas para a análise de imagens. Por este motivo, neste capítulo, serão apresentadas de forma muito resumida as diferentes técnicas que podem ser utilizadas em imagens.

A deteção de conteúdo explícito em imagens pode ser efetuada recorrendo a uma das seguintes abordagens [44]:

- Análise da imagem através da informação associada ao ficheiro (*metadata*);
- Análise da informação visual presente na imagem.

No âmbito desta dissertação foi necessário implementar técnicas baseadas em visão computacional que analisam a informação visual. Os trabalhos publicados nesta área utilizam diversas abordagens que podem ser agrupadas em : *(i)* métodos baseados na informação de cor; *(ii)* métodos baseados na informação de formas ou objetos nas imagens *(iii)* métodos baseados na informação de cor, formas e textura (não têm uma grande relevância).

Na maioria das imagens com conteúdos explícitos existe uma percentagem elevada de pele, tendo os autores no artigo [44] realizado uma análise detalhada sobre este aspeto. Neste trabalho foi realizado um estudo em relação ao parâmetro *hue* do espaço de cor HSV utilizando 100000 imagens que apresentavam conteúdo explícito e 100000 imagens aleatórias, sem qualquer tipo de conteúdo explícito (figura 2.18).

A análise efetuada permitiu concluir que para o espaço de cor HSV, imagens com conteúdo explicito apresentavam uma predominância de informação de cor na região

das cores vermelha e laranja.

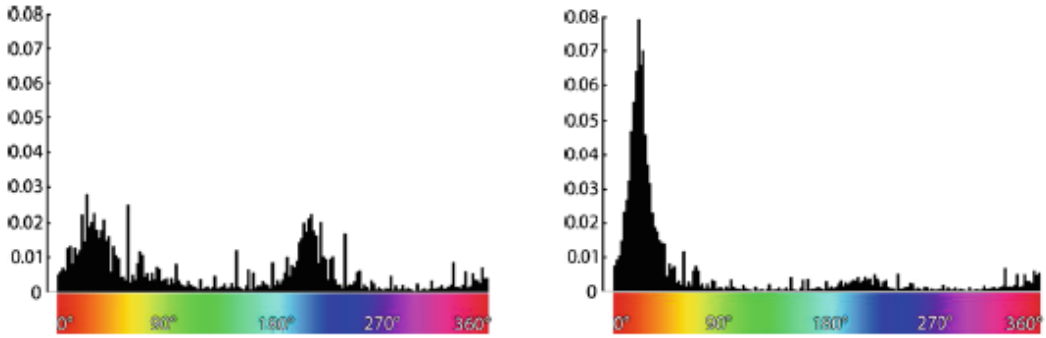


**Figura 2.18:** *Distribuição do valor associado ao parâmetro Hue do espaço de cor HSV em imagens normais (a) e em imagens com conteúdo explícito (b) [44]*

É comum em soluções que utilizam a informação de cor, ser realizado um cálculo da percentagem de pele existente na imagem analisada de maneira a classificar o tipo de conteúdo. Este mecanismo está presente nos artigos [26, 45, 46].

Os métodos de deteção de conteúdo explicito, baseando-se na extração de informação das formas da imagem, centram-se no facto de diferentes imagens apresentarem formas associadas com o conteúdo para adultos [44]. Os mecanismos baseados neste método, foram desenvolvidos com o intuito de diminuir as falsas deteções associadas aos erros que ocorrem no processo de segmentação de pele, e com o facto da eficiência dos métodos baseados na informação de cor estarem fortemente relacionados com o espaço de cor utilizado.

Dentro deste método é possível dividir as soluções em diferentes grupos: soluções baseadas no contorno, nos segmentos de cor, nas formas geométricas [24], momentos e características da versão 7 da norma Moving Picture Experts Group (MPEG). Na maior parte das vezes, para a criação dos classificadores presentes nestes métodos, são utilizados mecanismos de *machine learning* como o *boosting*, o SVM, entre outros [44].

No artigo [44] estão descritas várias soluções, porém a solução apresentada no artigo [47] onde se realiza a classificação do tipo de conteúdo através da verificação da existência de mamilos, foi fundamental para o desenvolvimento da solução descrita nesta dissertação.

A literatura nesta área não apresenta, normalmente, estudos em relação ao consumo de recursos das diversas soluções, nem do tempo que cada deteção demora, tornando assim impossível comparar métodos em relação à sua eficiência e consumo de recursos computacionais.

Segundo *C. Ries* e *R. Lienhart* [44], é possível afirmar que as soluções que se baseiam apenas na segmentação de pele são mais eficientes nas questões de tempo

de resposta e de consumo de recursos, quando comparadas com as soluções que utilizam algoritmos de deteção de objetos e outros mecanismos. De uma forma geral, a deteção de conteúdo explicito é considerada como uma tarefa de difícil execução, sendo assim complicado desenvolver soluções que consigam reunir todos os tipos de situações associadas ao conteúdo explícito.

### 2.4.1 Problemas associados ao processo de deteção de conteúdo explícito

O processo de deteção de conteúdo explícito em imagens pode ser influenciado por aspetos aleatórios, que não são previsíveis durante o processo de deteção. Desta forma, os mecanismos utilizados para efetuar a deteção de material para adulto estão condicionado por certos parâmetros. Alguns destes parâmetros são:

- **Percentagem de pele** - embora esta prática seja muito utilizada para deteção de conteúdo explícito, a existência de outros objetos com cores semelhantes pode levantar problemas quanto à exatidão dos algoritmos. A figura 2.19 ilustra algumas situações que podem conduzir à falha deste método. À primeira imagem está associada uma elevada percentagem de pele segmentada sem qualquer tipo de conteúdo explícito (Falso Positivo), enquanto que na segunda imagem a região de pele é reduzida e nesta está presente conteúdo explícito (Falso Negativo).

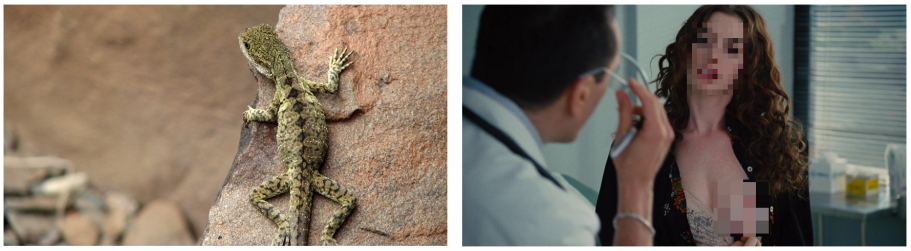

**Figura 2.19:** *Exemplos de falhas da deteção de conteúdo explícito baseado na percentagem de pele*

- **Tons de cinzento** - caso a classificação de conteúdo explícito se baseie apenas no processo de segmentação de pele utilizando a informação da cor, surgem problemas quando as imagens analisadas não contêm informação de cor (Figura 2.20).

**Figura 2.20:** *Imagens em tons de cinzento impossíveis de segmentar usando a informação de crominância*

### 2.4.2 Soluções comerciais/*Open Source*

A deteção de conteúdo explícito em imagens e vídeo, é uma área em desenvolvimento. O facto de não existir um método que garanta a correta sinalização de conteúdo explícito, torna o desenvolvimento de soluções comerciais muito reservada e quase inexistente.

A Porn Detection Stick da empresa Paraben Corporation, analisa todos os ficheiros multimédia presente no computador ou discos amovíveis e efetua um relatório detalhado dos ficheiros que apresentam conteúdo explícito. Segundo os dados presentes no sítio *web*, esta solução apresenta uma eficiência de 99% a identificar e eliminar qualquer conteúdo pornográfico do computador. Este *software* apenas está disponível para o sistema operativo Windows [48].

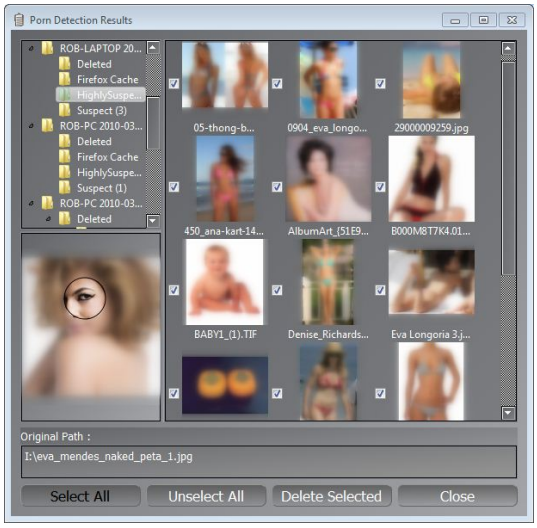


**Figura 2.21:** Software *Porn Detection Stick [48]*

Os autores do artigo [46], no qual desenvolveram um método de deteção de conteúdo explícito baseado na segmentação de regiões de pele, efetuaram uma análise comparativa entre a solução descrita no artigo e o programa Porn Detection Stick e

verificaram que este *software* registou uma eficiência de 89,7%, com 6,8% de falsos positivos. O método desenvolvido neste artigo conseguiu identificar corretamente 88,8% das imagens e apenas teve 5% de falsos positivos. Assim os autores, verificaram que os resultados da solução comercial Porn Detection Stick, não se afastam muito dos resultados do método proposto no artigo em questão.

O *software* Forensic Toolkit (FTK) atualmente encontra-se na quinta versão e é desenvolvido pela empresa AccessData. Este programa regista todas as interações que o utilizador realiza no computador. O software foi desenvolvido para empresas que necessitem de efetuar o controlo sobre ações que os seus colaboradores realizam enquanto utilizam o computador da empresa. Uma das funcionalidades presente neste programa é a deteção de conteúdo explícito através da análise dos ficheiros presentes no computador e no histórico de pesquisas.

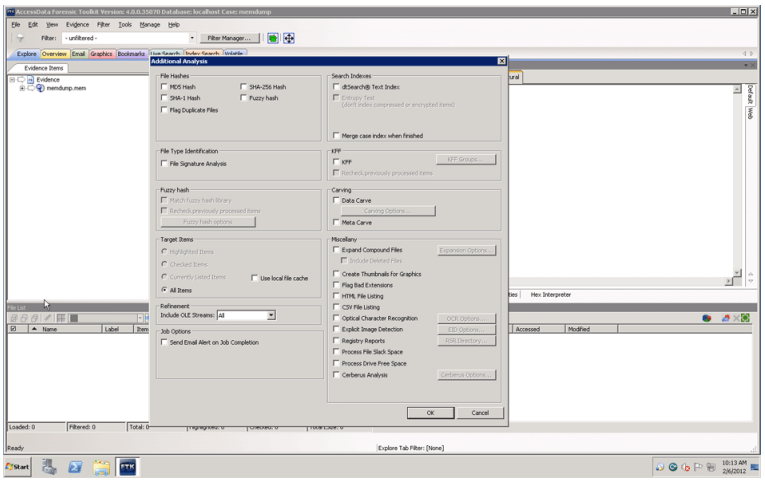

**Figura 2.22:** Software *FTK [49]*

O *software* PNWatch encontra-se na versão Beta e o seu desenvolvimento está estagnado desde 2005. O PNWatch foi desenvolvido pela Yang's Scientific Research Institute e tem como principal objetivo, revolucionar a sinalização de conteúdo explícito em relação as soluções já existentes. Este *software* pretende conseguir eliminar ao máximo as falsas deteções que ocorrem utilizando os métodos e soluções existentes, tanto a nível comercial como a nível científico [50].

Para o desenvolvimento deste *software*, esta empresa equacionou que os humanos apresentam diferentes tons de pele e o conteúdo explícito também é encontrado em imagens em tons de cinzento.

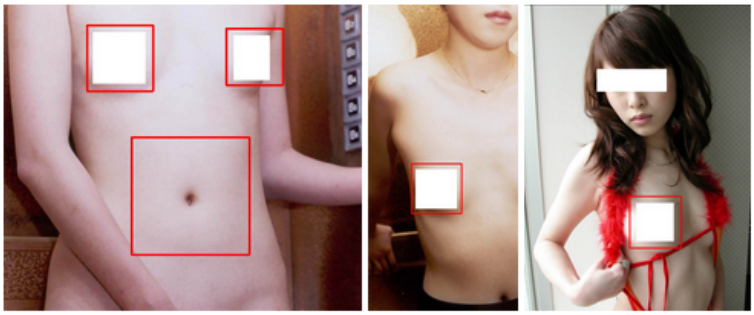

**Figura 2.23:** *Resultados do processo de deteção de conteúdo explícito usando o* software *PNWatch [50]*

Provavelmente este é a solução mais completa de deteção automática de conteúdo explícito em imagens e vídeos. Pois este contêm diversas funcionalidades que conseguem efetuar a identificação de órgãos sexuais (pénis, ânus, vulva, seios e região púbica), posições e atos sexuais (sexo oral, entre outros).

Atualmente existem poucos mecanismos de deteção de conteúdo explicito em páginas *web*. Embora continue em fase de desenvolvimento, Patrick Wied desenvolveu um *script* em JavaScript que permite detetar nudez em fotografias e mais recentemente em vídeos. Patrick Wied, desenvolveu o *script* `Nude.js` com o objetivos de criar uma ferramenta que consiga tornar a *Internet* um meio onde fosse possível controlar a exposição deste tipo de conteúdo. A filosofia de funcionamento deste *script* baseia-se na análise de todas as imagens presentes numa página *web* e em bloquear apenas as imagens que apresentem conteúdo explícito. Segundo os dados divulgados na página pessoal do autor, este *script* na sua primeira versão teve uma eficiência de deteção que rondava os 60% [51].

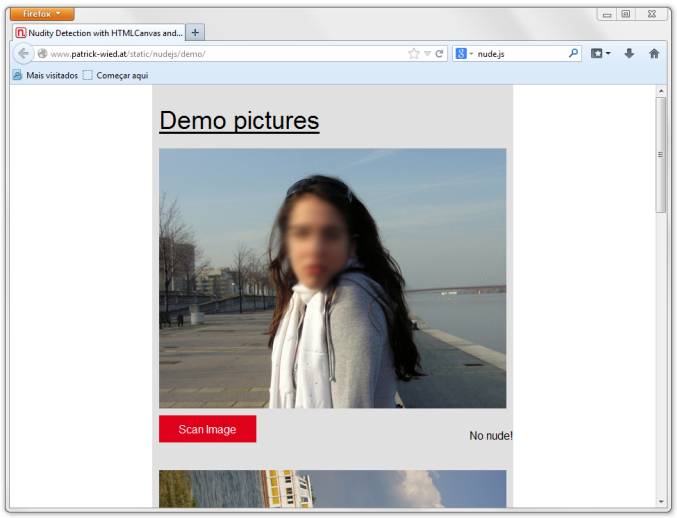


**Figura 2.24:** *Exemplo de funcionamento do* script `nude.js`

São muito poucas as soluções comercias que efetuam deteção automática de conteúdo explícito. As soluções existentes normalmente baseiam-se em software utilizado em empresas para monitorizar o tipo de conteúdo que os seus trabalhadores armazenam e visualizam no seu computador, sendo a deteção de conteúdo explicito apenas um complemento associado a este tipo de *software*. Soluções como o *script* `nude.js` e o PNWatch, são ideais para a filtrar conteúdo explicito em páginas *web*. No contexto de soluções aplicadas a sistemas de *broadcasting*, não existe qualquer *software* que realize a deteção de conteúdo explícito.

# 3

# Frameworks de visão computacional

Este capítulo será utilizado para apresentar as *frameworks* de visão computacional mais importantes e utilizadas atualmente. Inicialmente, será efetuada uma apresentação e descrição das mesmas, de forma a destacar quais são as suas principais características, particularidades e onde estas podem ser aplicadas.

Como conclusão deste capítulo, será efetuada uma análise crítica em relação aos pontos abordados durante a apresentação das *frameworks*, de modo a determinar qual será a que melhor se adapta às necessidades do desenvolvimento da solução a ser integrada do produto mxfSPEEDRAIL F1000.

## 3.1 MATLAB

O MATLAB é um *software* no qual está integrada uma linguagem de programação de alto nível baseada em Java e C++. Este *software* é bastante conhecido por ser utilizado em centros de investigação e faculdades, devido às suas enormes potencialidades matemáticas.

O *software* MATLAB é desenvolvido pela empresa The MathWorks, e contém diversas *toolboxs* ideais para a simulação e desenho de sistemas [52].

As funcionalidades de visão computacional, presentes neste *software*, estão principalmente associadas a três bibliotecas [53]:

- **Biblioteca `Computer Vision System Toolbox`** - inclui funcionalidades associadas à deteção e reconhecimentos de formas, deteção de objetos, seguimento e estimativa de movimento, calibração de câmaras, visão estereoscópica,

entre outras funcionalidades;

- **Biblioteca `Image Processing Toolbox`** - a esta biblioteca estão associadas funcionalidades básicas de processamento de imagem, como por exemplo: cálculo de histogramas das imagens, filtros para ruído, etc.;
- **Biblioteca `Statistics Toolbox`** - nesta biblioteca estão presentes as funções que permite a utilização de algoritmos de *machine learning*, que eventualmente, podem ser aplicados no reconhecimento de objetos em imagens;

Bibliotecas como a `Image Acquisition Toolbox`, `MATLAB Coder` e `Parallel Computing Toolbox`, são necessárias para a criação de aplicações que utilizem as funcionalidades de visão computacional, recorrendo às propriedades deste *software*.

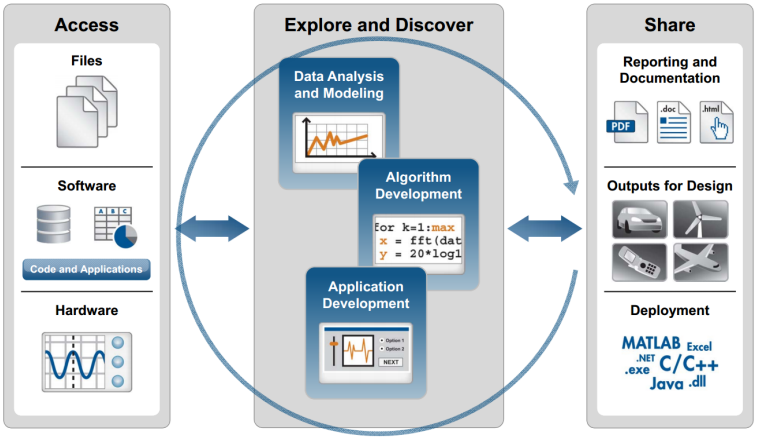


**Figura 3.1:** *Desenvolvimento de soluções com o software MATLAB [53]*

A linguagem de programação utilizada neste programa, foi criada de forma a facilitar a criação de aplicações e o uso de funcionalidades matemáticas. Esta encontra-se bem documentada e apresenta-se como uma forma rápida de criação de protótipos.

Outra das vantagens associadas ao MATLAB, é o facto de este incluir métodos e funcionalidades que permitem a manipulação da informação associada à maior parte dos ficheiros de vídeo e imagem.

## 3.2 OpenCV

O OpenCV é uma biblioteca *open source* que foi desenvolvida ao abrigo da licença de Berkeley Software Distribution (BSD) podendo por isso ser utilizada no contexto académico e comercial, não existindo qualquer problema com a comercialização de *software* que utiliza as funcionalidades oferecidas por esta biblioteca. O OpenCV foi desenvolvido com o principal objetivo de tornar o processamento computacional mais eficiente (visto que a visão computacional tem um consumo excessivo de

recursos computacionais) e conseguir tornar a execução das aplicações de visão computacional em tempo-real. Esta biblioteca é escrita em C e C++ e é suportada em vários sistemas operativos, tais como: Windows, Linux, Mac OS, iOS e Android. O OpenCV teve uma grande disseminação devido ao facto de estar associada a bibliotecas para várias linguagens de programação, como: C++, C, Phyton, Matlab, Java e entre outras [20, 54].

Na figura 3.2 estão representas as quatro principais componentes desta biblioteca [54].

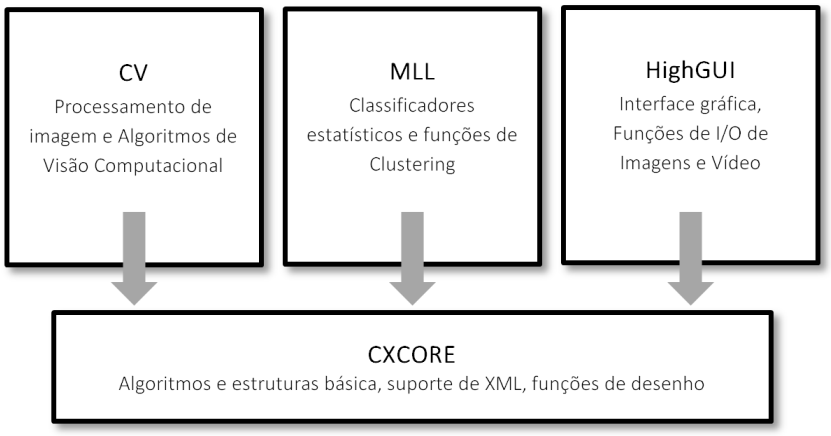


**Figura 3.2:** *Estrutura básica do OpenCV [54]*

Cada uma das componentes representadas na figura 3.2 contém funcionalidades distintas, que tornam o OpenCV numa biblioteca de visão computacional muito robusta e diversificada. As características de cada uma das componentes são:

- **CV** - Contém as funcionalidades relacionadas com o processamento de imagem, seguimento, reconhecimento de padrões e mecanismos de calibração de câmaras;

- **MLL** - Nesta componente estão integradas as funcionalidades de *Machine Learning*, incluindo assim classificadores estatísticos (utilizados para agruparem os dados com base em amostras) e ferramentas de *clustering* (utilizados para agruparem os dados de acordo com semelhanças);

- **HighGUI** - Inclui rotinas e funções de *input/output*, utilizadas principalmente para armazenar e processar vídeos e imagens;

- **CXCORE** - Contém as estruturas de dados utilizadas nas funções de alto nível desta biblioteca. Inclui as funções que processam a álgebra aplicada nas matrizes, a conversão entre espaços de cor, gestão da memória utilizada, entre outras funcionalidades de baixo nível.

O OpenCV contém a biblioteca `CvAux`, que implementa funcionalidades de deteção de faces utilizando Hidden Markov Models (HMM) e outros algoritmos experimentais. No entanto, esta biblioteca tem vindo a ser descontinuada [54].

## 3.3 libCVD

O libCVD é uma *framework* escrita em C++. O desenvolvimento desta biblioteca foi centralizado na adaptação de funcionalidades de visão computacional a ambientes onde é necessário ter um controlo rigoroso em relação ao consumo de memória. Tem como principal objetivo facilitar a implementação e o acesso à informação presente nas imagens e vídeos a serem analisadas pelas funções desta biblioteca. As bibliotecas desta *framework* implementam diversas funções que permitem efetuar a conversão de espaço de cor , operações matemáticas e processamento de imagem. O módulo que engloba as funcionalidades de visão computacional apenas alberga as funcionalidades de morfologia (dilatação e erosão), aplicação de filtros, deteções de cantos, etc. Esta *framework* é compatível nos sistemas operativos Windows, Linux e Mac OS [55]. O libCVD está publicado ao abrigo da licença GNU Lesser General Public License (LGPL), o que possibilita a sua integração em produtos comerciais, não sendo no entanto permitidas alterações nas funções incluídas nesta *framework*.

## 3.4 FastCV

Esta *framework* disponibiliza várias bibliotecas de visão computacional otimizadas para dispositivos móveis. O FastCV é vocacionado para aplicações de realidade aumentada, reconhecimento de texto, deteção e reconhecimento de faces, reconhecimento de gestões e seguimento de objetos. Foi criada pela empresa Qualcomm (empresa que desenvolve processadores para *smartphones*), tendo sido especialmente desenhada para dispositivos móveis que utilizam o sistema Android. A empresa assegura que em novas versões desta *framework* irão ser adicionadas compatibilidades com os sistemas operativos iOS e Windows Phone. Embora esta *framework* não tenha qualquer custo para o utilizador, é necessário aceitar os termos e condições que a empresa apresenta durante a instalação do Software Development Kit (SDK) associado ao FastCV [56].

## 3.5 SimpleCV

O Simple Computer Vision (SimpleCV) é uma *framework* com licença *Open Source*, e o seu principal objetivo é tornar o desenvolvimento de *software* que envolva funções de processamento de imagem, o mais fácil possível. É baseada nas

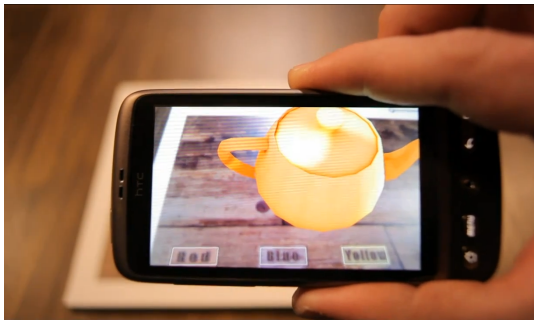

**Figura 3.3:** *Funcionalidades de realidade aumentada implementadas num dispositivo móvel usando a framework FastCV [56]*

bibliotecas do Open Source Computer Vision Library (OpenCV), fornecendo assim o acesso a funcionalidades de visão computacional, sem que o programador necessite de ter conhecimentos prévios de formatos de ficheiros, espaços de cor, profundidade de cor (bit depht), reconhecimento de objetos e outros parâmetros muito importantes no campo da visão computacional [57].

O SimpleCV apenas pode ser utilizado com a linguagem de programação Pyhton. Esta *framework* tem manifestado um grande crescimento na sua comunidade de utilizadores. Este crescimento deve-se ao facto de ser baseado em Pyhton (esta linguagem apresenta um elevado numero de utilizadores), e ao facto de simplificar a utilização de funcionalidades de visão computacional [57, 58].

## 3.6 Vision Blocks

Vision Blocks é um projeto criado pelo MIT Media Lab para tornar a visão computacional acessível a qualquer pessoa sem conhecimento prévio de conceitos de visão computacional e de programação. A ideia para a criação desta ferramenta baseou-se num outro projeto do MIT Media Lab, o projeto Scratch. O Scratch é uma linguagem de programação visual que se baseiam em blocos de instruções, de modo a introduzir conceitos de programação a crianças. O Vision Blocks apenas está disponível através no seu sítio *web*, onde é possível experimentar as mais diversas funcionalidades [59].

## 3.7 *Framework* a utilizar

No âmbito desta dissertação, é estritamente necessário que as funcionalidades sejam codificadas na linguagem de programação C++, o que limita o número de *frameworks* compatíveis com o desenvolvimento do *plugin*. As únicas *frameworks* compatíveis com este tipo de linguagem são: o OpenCV, a libCVD e o FastCV.

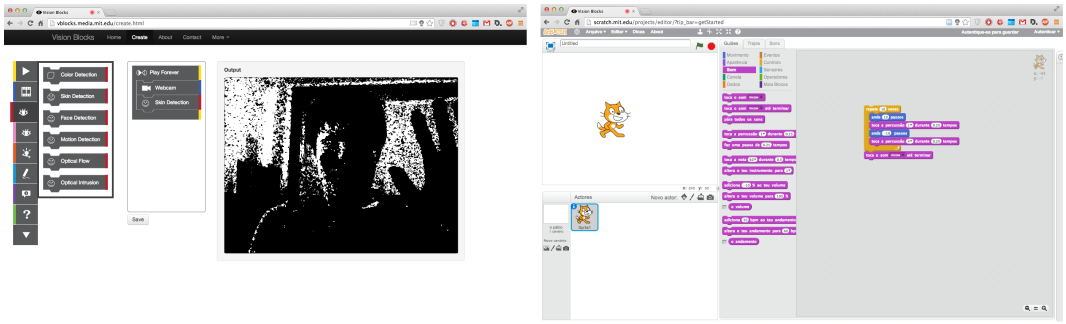

**Figura 3.4:** *Ferramenta Vision Blocks (lado esquerdo) e a ferramenta Scratch (lado direito)*

As frameworks libCVD e FastCV, foram criadas para serem implementadas em situações e contextos muito específicos, o que as tornam em *frameworks* com algumas lacunas e sem funcionalidades que são muito importantes para o desenvolvimento desta dissertação, não apresentando por exemplo mecanismos de *Machine Learning*.

Tanto o SimpleCV como o Vision Blocks são excelentes opções para programadores que pretendam criar um protótipo ou ter uma visão geral das capacidades que a visão computacional oferece. O Vision Blocks contém muito poucas funcionalidades de visão computacional e apenas tem compatibilidade com ambientes *web*. O facto de o Vision Blocks não se adequar com o desenvolvimento de aplicações mais robustas torna esta *framework* muito incompatível com o projeto a desenvolver.

As *frameworks* mais completas, são o OpenCV, MATLAB e SimpleCV, pois contêm várias funções de visão computacional e de *Machine Learning* (uma função bastante importante para o desenvolvimento do *plugin* de deteção de conteúdo explícito). Apesar do SimpleCV e o MATLAB simplificarem o desenvolvimento de aplicações e serem muito completos tornou-os numa solução viável para o desenvolvimento do *plugin*. Devido ao facto destas serem apenas disponível para linguagens de programação diferentes de C++, tornou estas *frameworks* incompatíveis com o desenvolvimento do *plugin*.

A *framework* que mais se adequou às necessidades foi o OpenCV, que apresenta uma elevada portabilidade entre sistemas e linguagens de programação. O OpenCV é uma *framework* que está muito bem documentada e apresenta um elevado nível de robustez, podendo ser implementada em soluções comerciais.

# 4

# Desenvolvimento do *plugin* de deteção de conteúdo explícito

Neste capítulo serão abordados aspetos relacionados com o desenvolvimento do *plugin* que efetua a deteção automática de conteúdo explícito.

O primeiro ponto a ser abordado neste capítulo é a descrição geral do problema e qual será rumo utilizado durante o desenvolvimento de uma solução para o mesmo. No segundo ponto serão explicados os mecanismos utilizados para o desenvolvimento da solução, de modo a ser possível para o leitor entender a utilização de certos parâmetros. Os subcapítulos seguintes, serão utilizados para efetuar a comparação dos resultados obtidos com diferentes métodos de classificação de conteúdo explicito, apresentando uma análise crítica em relação às limitações da solução desenvolvida.

Por fim, no última secção será abordada a arquitetura do protótipo do *plugin* a ser integrado no mxfSPEEDRAIL F1000.

## 4.1 Análise do Problema

A deteção automática de conteúdo explícito nas *frames* de uma sequência de vídeo apresenta-se como grande desafio computacional.

O *plugin* a ser desenvolvido vai integrar parte do sistema de controlo de qualidade dos ficheiros que são submetidos ao processo de *ingest* no produto mxfSPEEDRAIL F1000. Para que o *plugin* consiga detetar conteúdo explícito, este deve conter métodos para detetar pele, seres humanos e órgãos sexuais, pois estes três elementos estão quase sempre presentes numa *frame* que contém nudez feminina.

Para a avaliação do módulo a desenvolver deverão ser considerados 2 tipos de falhas: *(i)* Falsos Positivos, que ocorrem quando o *plugin* deteta conteúdo explícito e este não existe na *frame*; *(ii)* Falsos Negativos, quando o *plugin* não deteta qualquer conteúdo explícito e este está presente na *frame*.

Associado ao processo a implementar, pode ser considerada a existência de parâmetros controlados (aqueles para os quais existem diversas alternativas que podem afetar os resultados e que podem ser escolhidos de forma a obter o melhor resultado) e parâmetros não controlados (aspetos intrínsecos à imagem e sobre os quais o programador não tem qualquer tipo de controlo).

### 4.1.1 Diagrama de blocos

De forma a minimizar os problemas associados a cada uma das abordagens descritas, a solução proposta nesta dissertação é baseada na combinação de três mecanismos aplicados a cada uma das imagens que compõem a sequência de vídeo em análise. A figura 4.1 resume as diferentes etapas que correspondem à aplicação sequencial de diferentes técnicas: *(i)* Segmentação de pele; *(ii)* Deteção de faces; *(iii)* Deteção de seios.

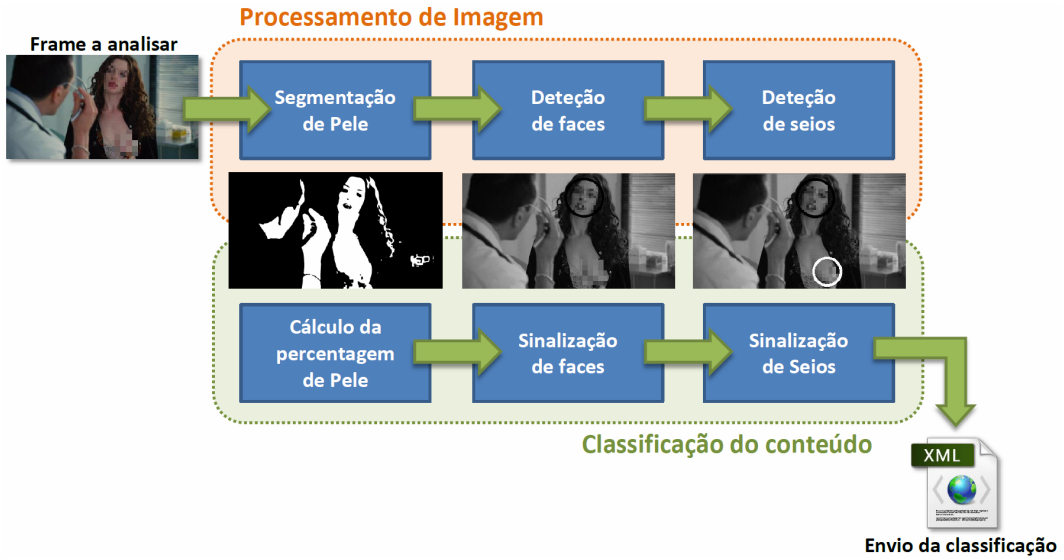


**Figura 4.1:** *Fluxo de informação da solução encontrada*

A primeira etapa implementa uma das abordagens mais comuns para deteção de conteúdo explicito - segmentação de pele. Desta, resulta uma imagem binária para a qual é possível calcular a percentagem de pele presente em cada frame (equação 4.1). Caso o resultado obtido na cálculo da percentagem de pele tenha valores superiores a 50%, é considerado que esta contém conteúdo explícito.

$$Pele(\%) = \frac{N.^o\ de\ pixeis\ de\ pele}{N.^o\ total\ de\ pixeis\ da\ frame} \times 100 \tag{4.1}$$

As etapas seguintes têm como objetivo, reduzir algumas limitações e existência de falsos positivos associados às técnicas baseadas na segmentação de pele. Assim, ao garantir a existência de humanos e de seios, garante-se uma maior precisão dos resultados.

A decisão final do módulo de deteção é o resultado da análise dos valores obtidos em cada uma das etapas tal como ilustrado na Tabela 4.1.

Associado a cada etapa é definido um parâmetro booleano cujo valor muda de `false` para `true` se na etapa em questão é detetado conteúdo explicito.

**Tabela 4.1:** *Diferentes situações de classificação de conteúdo explícito.*

| Situação | Etapa 1 | Etapa 2 | Etapa 3 | Classificação |
|---|---|---|---|---|
| **1** | Verdadeiro | Falso | Falso | *Frame* sem conteúdo explícito |
| **2** | Verdadeiro | Verdadeiro | Falso | *Frame* poderá conter conteúdo explícito |
| **3** | Verdadeiro | Falso | Verdadeiro | *Frame* contém conteúdo explícito |
| **4** | Verdadeiro | Verdadeiro | Verdadeiro | *Frame* contém conteúdo explícito |
| **5** | Falso | Falso | Falso | *Frame* sem conteúdo explícito |
| **6** | Falso | Verdadeiro | Falso | *Frame* sem conteúdo explícito |
| **7** | Falso | Falso | Verdadeiro | *Frame* poderá conter conteúdo explícito |
| **8** | Falso | Verdadeiro | Verdadeiro | *Frame* contém conteúdo explícito |

O desenvolvimento da solução será dividido em dois processos distintos. O primeiro processo será descrito neste capítulo, que está associados ao desenvolvimento do prototipo do *plugin* a integrar no produto. Desta forma, será necessário abordar os aspetos associados à segmentação de pele e ao desenvolvimento do classificador de seios em imagens.

O segundo processo, está associado a nível de alterações e adaptações que foram necessária para a integração do *plugin* no produto, este será descrito no capítulo 6.

## 4.2 Módulo de segmentação de pele

Como foi enunciado em capítulos anteriores, a escolha do espaço de cor influência bastante a eficácia do processo de segmentação de pele. As opções mais comuns são os espaços de cor YUV e HSV.

No entanto, uma vez que se pretende integrar o módulo de deteção nos produtos da MOG Technologioes, e que no *workflow* dos mesmos é apenas admitido o modelo YUV, o projeto ficou condicionado a esta hipótese.

No artigo [26] é feita uma descrição das diferentes técnicas que são utilizadas para efetuar a segmentação de pele. Um dos tópicos abordados é o estudo dos diversos espaços de cor utilizados para este processo. Na figura 4.2 é possível visualizar os resultados associados ao espaço de cor YCbCr, onde está representada a região que inclui todas as variações da cor de pele associada às diferentes etnias. Os resultados presentes na figura 4.2, foram obtidos utilizando os canais de crominância.

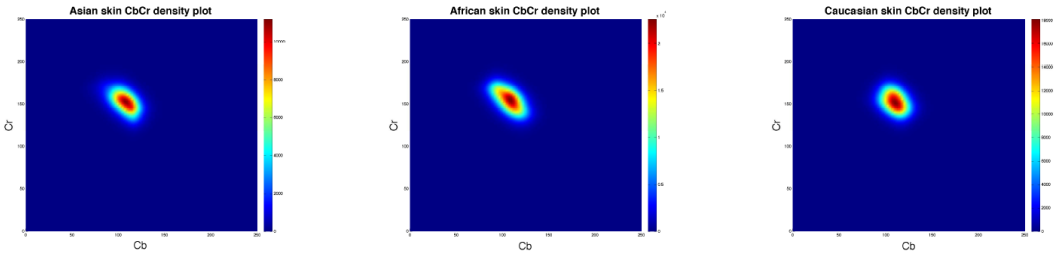


**Figura 4.2:** *Estudo da distribuição da cor de pele através interseção dos histogramas dos canais Cr e Cb do espaço de cor YCbCr [26]*

Para confirmar a veracidade dos resultados descritos no artigo [26], foi realizado um estudo com 100 imagens. Cada uma das imagens utilizadas neste teste continha diferentes condições de iluminação, e apenas apresentavam informação de pele (Figura 4.3).

Das 100 imagens utilizadas, 50 destas eram referentes ao tom de pele caucasiana e as outras 50 ao tom de pele africana. Tal como no artigo [26], foi feita a interseção dos histogramas dos canais Cr e Cb de todas as imagens, tendo-se no fim normalizado os histogramas recorrendo às capacidades da distribuição gaussiana.

Após este estudo foi possível verificar que os resultados obtidos (Figura 4.3) se aproximaram muito dos valores referidos em [26] (Figura 4.2). Depois desta confirmação, foi possível concluir que estes valores podem ser utilizados no contexto da solução a desenvolver.

Depois da verificação da propriedade associada ao espaço de cor YCbCr, foi necessário obter os limites que definem a região que contém a informação de pele, para que seja possível efetuar o *threshold* em cada canal de crominância.

Para tal, foram utilizados os valores presentes no artigo [46]. Neste artigo, os

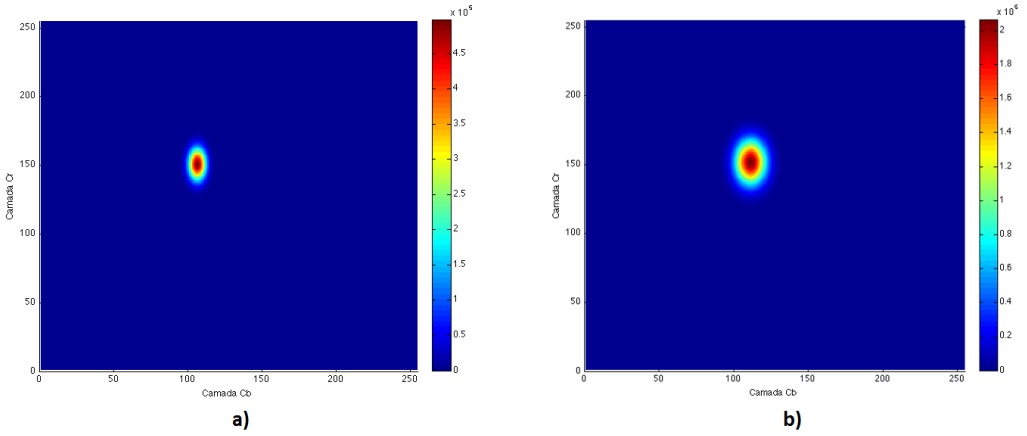


**Figura 4.3:** *Interseção dos histogramas dos canais Cr e Cb a) Africanos b) Caucasianos*

autores efetuaram testes a três valores de *threshold* de forma iterativa, de modo a verificar qual destes apresentava os melhores resultados. Os valores associados ao 3º *threshold* foram os que obtiverem os melhores resultados, sendo os mesmos representados pelo seguinte intervalo $80 \leq \text{Cb} \leq 120$ e $133 \leq \text{Cr} \leq 173$[1] (Figura 4.4).

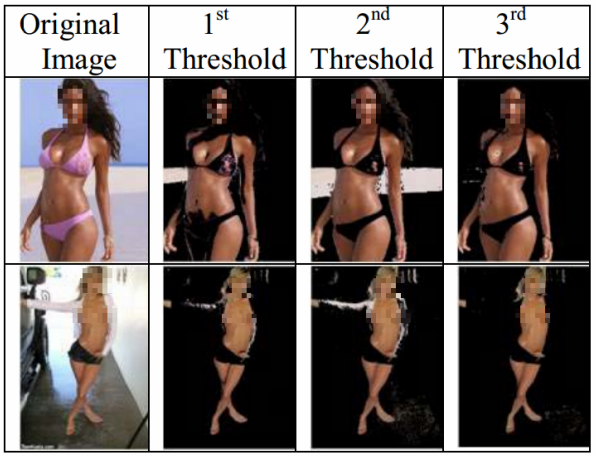


**Figura 4.4:** *Resultados da segmentação de pele usando diferentes intervalos de Threshold [46]*

Como existem poucas diferenças entre os espaços de cor YUV e YCbCr, os valores de *threshold* podem ser utilizados sem introduzirem muitos erros nos resultados de segmentação de pele.

Para ser possível utilizar o módulo de segmentação de pele no *plugin*, foi necessário adaptar o mesmo às funcionalidades do OpenCV. Desta forma, foi criado o método `skinClassifier` e o seu funcionamento está representado na figura 4.5.

[1] Os valores das camadas Cb e Cr variam entre 0 e 255

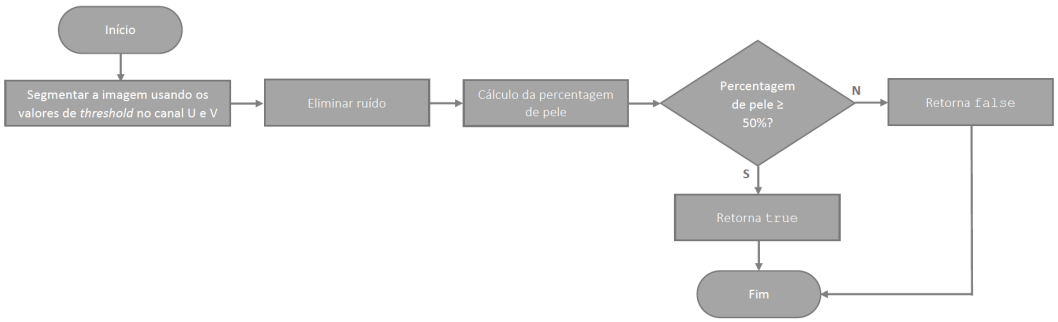


**Figura 4.5:** *Primeira etapa de classificação de conteúdo explícito (método* `skinClassifier`*)*

Inicialmente a frame a ser analisada é segmentada utilizando os limites de *threshold* descritos anteriormente. Após o processo de segmentação é aplicado um filtro, de modo a eliminar o ruído. Por fim é calculada a percentagem de pele presente na imagem. Caso o valor obtido no cálculo seja igual ou superior a 50% é retornada uma variável com o valor booleano `true`, caso contrário o valor retornado será `false`.

### 4.2.1 Estudo da segmentação de pele em diferentes formatos de *chroma subsampling*

O facto de diferentes sequências de vídeo apresentarem distintas razões de compressão nos canais de crominância das *frames*, motivou a realização de um estudo de como diferentes formatos de *chroma subsampling* afetam o processo de segmentação de pele.

Para realizar este estudo, foi necessário criar uma sequência de vídeo que tivesse as seguintes características : *(i)* 50% da frame deveria conter informação de pele; *(ii)* os outros 50% da frame deveria conter informação que não pudesse ser interpretada como pele. Para tal, foi necessário capturar algumas fotografias, sem qualquer tipo de compressão de modo a que estas não introduzissem erros durante o processo de conversão da sequência de vídeo para diferentes formatos de *chroma subsampling*. Para este estudo apenas foram utilizadas fotografias com informação associada ao tom de pele caucasiano.

Este teste apenas envolveu os três formatos de *chroma subsampling* mais importantes no fluxo de dados existentes nos produtos da MOG Technologies (formatos 4:2:2, 4:2:0 e 4:1:1). Na tabela 4.2 estão apresentados os resultados obtidos no teste realizado, sendo possível verificar um aumento do erro associado ao processo de segmentação de pele à medida que aumenta a compressão dos canais de crominância.

A solução apresentada nesta tese permite reduzir o impacto deste problema ao realizar um conjunto de etapas sequenciais que incluem, para além da segmentação

**Tabela 4.2:** *Resultados da percentagem de pele calculada no processo de segmentação de pele utilizando diferentes formatos de chroma subsampling*

| **Formato de chroma subsampling** | **4:2:2** | **4:2:0** | **4:1:1** |
|---|---|---|---|
| Percentagem de Pele | 49,53% | 49,22% | 46,92% |
| Erro | 0,47% | 0,78% | 3,08% |

de pele, a deteção de objetos nas imagens.

## 4.3 Módulo de deteção de faces e seios

A deteção de objetos nas *frames* é uma das etapas mais importantes para a solução a desenvolver. Tendo-se optado pela utilização do OpenCV e dado este disponibilizar as abordagens de Viola-Jones e LBP, estes módulos fazem uso das funcionalidades oferecidas por estes algoritmos.

O OpenCV utiliza ficheiros eXtensible Markup Language (XML) de forma a representar a cascata de classificadores associados à *framework* Viola-Jones e ao LBP. Esta biblioteca disponibiliza um conjunto de classificadores treinados, que podem ser utilizados para a deteção de faces em imagens. Na solução a desenvolver estes poderão ser utilizados na etapa associada à deteção de humanos na imagem.

No conjunto de ficheiros associados aos classificadores de faces, o ficheiro que tem melhores resultados é o `haarcascade_frontalface_alt2.xml`, portanto este ficheiro será utilizado para o processo de deteção de faces [54].

Inicialmente, foi estipulado efetuar o processo de deteção de faces apenas na posição frontal, porém como o processo de deteção de objetos associado ao método LBP é rápido, optou-se por utilizar este método para pesquisar faces de perfil, aumentando assim flexibilidade em relação à deteção de diferentes posições da face. No OpenCV, também está incluído um classificador associado ao método LBP (ficheiro `lbpcascade_profileface.xml`), utilizado para efetuar deteção de faces de perfil. Este será utilizado na solução para aumentar o número de deteções de faces independentemente da posição das mesmas.

Não existindo um classificador específico para a deteção de seios, foi necessário criar um. Para tal utilizou-se um *dataset* contendo 1400 imagens positivas (figura 4.6) e 2000 imagens negativas, obtidas de forma aleatória de um arquivo de imagens.

Após o processo de treino do classificador de seios foi gerado um ficheiro XML, sendo este utilizado no processo de deteção de seios nas imagens.

Associadas ao módulo de deteção de faces e seios, estão a segunda e a terceira etapas de classificação de conteúdo explícito. O funcionamento destas duas

**Figura 4.6:** *Amostras positivas do dataset utilizado para a criação do classificador de seios frontais em imagens*

etapas é muito parecido, estando descritas na figura 4.7 e figura 4.8. O método `faceDetector`, está associado ao processo de deteção de faces. Quando é detetada uma face durante a primeira pesquisa na frame utilizando o método LBP, este termina retornando `true`, caso contrário é efetuada uma nova pesquisa utilizando a framework de Viola-Jones, tal como no processo anterior é retornado `true` caso exista uma instância do objeto, caso contrário é retornado `false`.

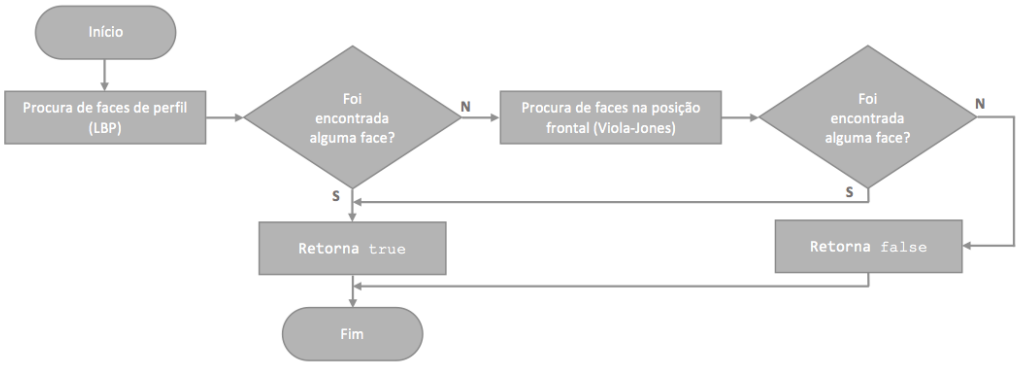


**Figura 4.7:** *Segunda etapa de classificação de conteúdo explícito*

Por fim, na terceira e última etapa de classificação, é efetuada a procura de seios na imagem. Tal como o processo anterior é utilizado o método associado à *framework* de Viola-Jones.

Para adicionar este método ao plugin, foi criado método `breastDetector`. Caso este detete um objeto associado à classe seios é retornado o valor `true`, e em caso de não ser detetado qualquer objeto associado a esta classe é retornado `false`.

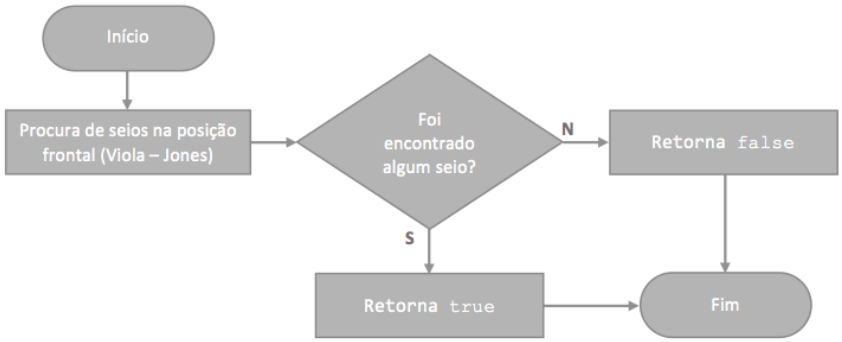


**Figura 4.8:** *Terceira etapa de classificação de conteúdo explícito*

## 4.4 Comparação de resultados

Para efetuar os testes ao *plugin* desenvolvido utilizou-se um *dataset* com 400 imagens das quais 200 apresentam conteúdo explícito, e as outras 200 estão associadas a conteúdo não explicito. Neste *dataset* foram efetuados testes utilizando quatro métodos, de modo a justificar o uso da tabela de classificação. O primeiro método baseia-se na deteção de conteúdo explícito utilizando apenas o processo de segmentação de pele e o cálculo de percentagem da mesma na imagem, um dos métodos mais utilizados, na literatura, para este fim. O segundo método é composto por duas etapas, onde a primeira etapa é baseada no método anterior com o acréscimo da deteção de face, de modo a diminuir os erros que existem no método anterior. O terceiro método baseia-se na utilização do mecanismo anterior, com a adição da deteção de seios de modo a tornar a deteção de conteúdo explícito mais precisa. Por fim, no último teste ao *dataset* é utilizado o método desenvolvido no âmbito desta dissertação, onde a classificação do conteúdo da sequência de vídeo se baseia na utilização de uma tabela que relaciona todas as etapas. Deste modo é possível obter uma deteção de conteúdo para adultos muito mais robusta e exata.

**Tabela 4.3:** *Eficiência dos diversos métodos de classificação de conteúdo explícito*

| **Métodos de Classificação** | **Eficiência da Deteção** | **Falsos Positivos** | **Falsos Negativos** |
|---|---|---|---|
| Segmentação de Pele | 77,5 % | 6,75% | 15,75 % |
| Segmentação de Pele e Deteção de Faces | 74,5 % | 1,25% | 24,25 % |
| Segmentação de Pele e Deteção de Faces e Seios | 74,5 % | 0% | 25,5 % |
| Método Proposto | 94,25 % | 2,5% | 3,25 % |

Após os testes, verificou-se que o segundo e o terceiro método, apenas diminuí-

ram a percentagem de falsos positivos em relação ao primeiro. A obtenção destes resultados deve-se à adição do processo de verificação da existência de humanos na imagem, o que explica a redução dos falsos positivos. O aumento dos falsos negativos está associado às falhas do processo de deteção de faces e seios.

Por fim, a solução proposta (baseada na classificação utilizando a tabela 4.1) foi bastante precisa, tendo uma eficiência de deteção igual a 94,25 %. Em relação aos outros métodos utilizados nos testes efetuados a este *dataset*, o método desenvolvido apresentou melhores resultados. O método desenvolvido apresenta um baixo número de falsas deteções, o que leva a afirmar que este método é uma excelente alternativa em relações aos anteriores.

## 4.5 Limitações da solução

Embora no sub-capítulo anterior, tenha sido feita uma análise crítica em relação aos resultados obtidos, esta não incidiu sobre algumas das limitações que o sistema apresentou durante a fase de testes efetuada à solução desenvolvida no âmbito desta dissertação.

As limitações associadas ao processo de segmentação já eram conhecidas, pois em capítulos anteriores foram abordadas certas temáticas que demonstraram que o processo de segmentação de pele é um processo bastante condicionado por inúmeros fatores. Devido a estas limitações, optou-se pelo uso de um sistema de classificação baseado em diferentes etapas.

Porém também foram observadas algumas limitações nas etapas associadas ao processo de deteção de objetos na *frame*. A principal limitação observada neste processo foi o tamanho da janela de procura do objeto na *frame*. Embora seja improvável existirem faces ou seios com a dimensão inferiores a regiões de 20x20 pixeis, este aspeto não deixa de ser uma limitação na solução.

No processo de deteção de faces, o classificador usado para detetar faces na posição frontal demonstrou ser bastante exato. Já o classificador associado à deteção de faces de perfil, demonstrou ter uma maior taxa de falsas deteções em relação ao classificador de faces frontais. Portanto, uma das limitações da solução é o facto de esta não estar preparada para detetar todas as faces presentes na imagem, embora detete a maior parte das mesmas.

Para efetuar a classificação de seios, foram treinados diferentes classificadores e apenas foi utilizado aquele que obteve melhores resultados. O classificador utilizado no *dataset* da fase de testes obteve uma taxa de 92,75% de sucesso no processo de deteções, com 3,25% de falsos positivos e 5% de falsos negativos, o que demonstrou estar apto para ser utilizado.

Como é possível verificar na figura 4.9 a este classificador estão associadas limi-

tações no processo de deteção. Nas duas primeiras imagens é possível observar que o classificador apenas efetua a deteção na posição frontal, embora o classificador tenha sido treinado para detetar seios nessa posição, esta não deixa de ser uma limitação da solução. As restantes imagens estão relacionadas com falsos positivos. Existem zonas do corpo onde são detetados falsos positivos muito frequentemente, regiões como a zona umbilical, certas zonas do rosto e por vezes cotovelos e joelhos.

Uma das explicações lógicas para este acontecimento, é o facto destas regiões apresentarem algumas semelhanças a nível da forma e das condições de iluminação, em relação às imagens utilizadas durante o processo de criação do classificador de seios.

**Figura 4.9:** *Limitações detetadas durante o processo de deteção de seios nas frames*

## 4.6 Módulos Auxiliares

Nesta secção são descritos os módulos que foram utilizados para emular o funcionamento do *plugin* no produto.

Para ser possível integrar a solução desenvolvida no produto mxfSPEEDRAIL F1000 foi necessário criar o *plugin* `MOG Quality Control Explicit Content Detector (MOG QC ECD)` que contém todas as etapas associadas à classificação do tipo de conteúdo presente nas *frames* analisadas. Após a classificação, o *plugin* deve ser capaz de comunicar o resultado ao módulo de controlo de qualidade (denominado por `MOG Quality Control Manager (MOG QCM)`) de modo a ser possível notificar o utilizador na interface gráfica do produto.

Para tornar o desenvolvimento do protótipo do *plugin* o mais próximo da realidade, foi necessário criar um simulador do módulo de controlo de qualidade. Sendo este essencial para tornar mais simples o processo de adaptação do *plugin* ao produto.

### 4.6.1 Arquitetura do protótipo do *plugin*

O facto de o desenvolvimento do módulo `MOG QCM` decorrer em simultâneo com o desenvolvimento do *plugin*, tornou necessária a criação de um emulador do sistema.

O diagrama de blocos associado a este protótipo está representado na figura 4.10. Este diagrama permite descrever o funcionamento do simulador do módulo de controlo de qualidade (ou gestor de *plugins*) e o *plugin* `MOG QC ECD`. Cada bloco

(simulador e *plugin*) pode ser decomposto em quatro blocos secundários que representam como é processada a informação. Os blocos principais do *plugin* incluem:

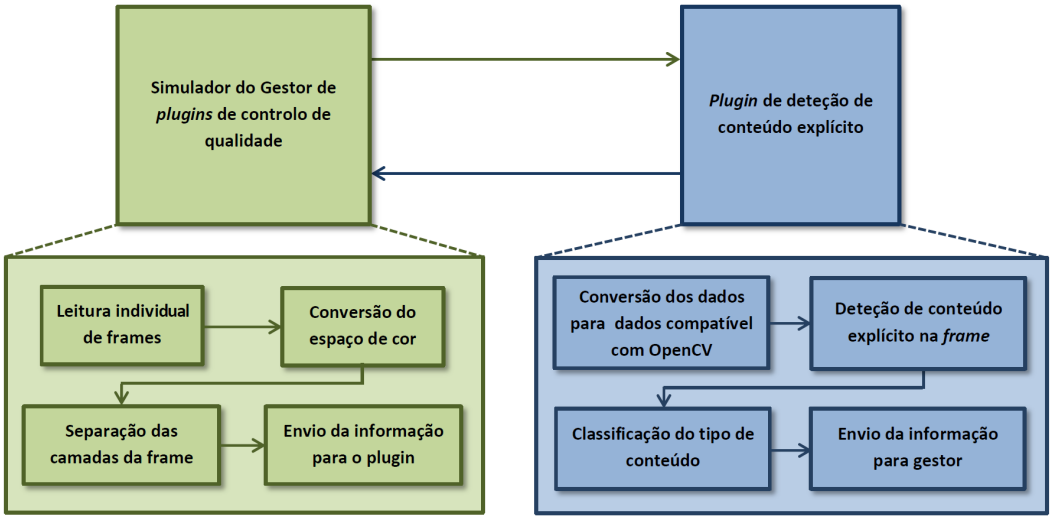


**Figura 4.10:** *Arquitetura do protótipo do plugin e comunicação com o simulador*

- **Conversão dos dados** - este bloco tem como principal objetivo adaptar da informação resultante do processo de *ingest* para estruturas de dados compatíveis com da biblioteca OpenCV;
- **Deteção de conteúdo explícito** - neste bloco vão estar incluídos os módulos de segmentação de pele, e deteção de faces e seios;
- **Classificação do conteúdo** - de acordo com os resultados obtidos durante a análise do conteúdo da *frame*, este bloco vai classificar o tipo de conteúdo, recorrendo á tabela de classificação;
- **Comunicação da classificação** - após a classificação da *frame* o resultado é comunicado para o simulador do módulo de controlo de qualidade.

### 4.6.2 Simulador do módulo `MOG QCM`

Na figura 4.11 está representado, de uma forma resumida, o funcionamento do módulo `MOG QCM` no contexto do processo de *ingest*.

A comunicação entre o módulo de controlo de qualidade (`MOG QCM`) e o *plugin* `MOG QC ECD` é efetuada através do uso de um protocolo. Este protocolo envolve o seguinte fluxo de informação: *(i)* o módulo de controlo de qualidade envia a informação referente aos canais que compõem a *frame*, juntamente com parâmetros associados ao tipo de formato em que a sequência de vídeo está codificada, e as

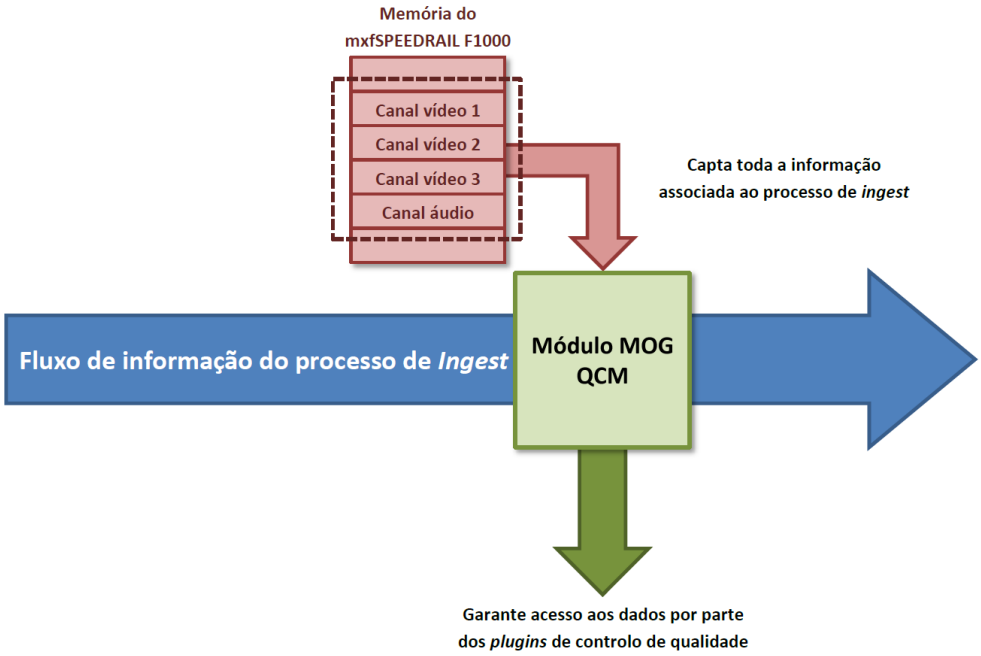


**Figura 4.11:** *Enquadramento do módulo de controlo de qualidade no fluxo de informação associado ao processo de ingest*

dimensões da mesma; *(ii)* o *plugin* envia o resultado da análise da *frames* numa mensagem codificada em XML.

O envio de informação referente aos três canais que constituem a imagem da *frame* são feitos sobre a forma de três *arrays* de caracteres, e cada *frame* é processada de forma individual.

De modo a simular este processo, foi necessário obter a informação individual de cada uma das *frames* de uma sequência de vídeo. Para este efeito, foi utilizado o *software* VLC Player de modo a obter as imagens associadas a cada *frame* do ficheiro de vídeo e armazenar cada uma num ficheiro Joint Photographic Experts Group (JPEG).

Depois de capturadas todas as imagens foi necessário criar um ficheiro de texto que contém a localização (*path*) de cada uma imagens capturadas. Na figura 4.12 é representado este processo.

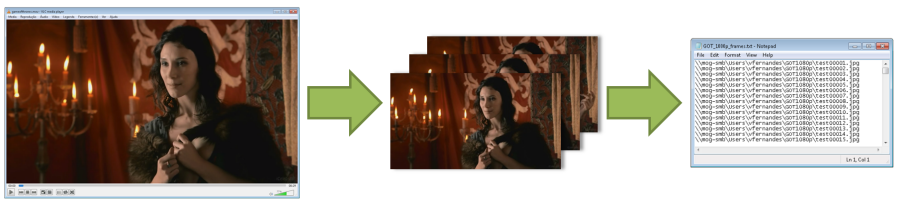

**Figura 4.12:** *Descrição do processo de extração das imagens referentes à sequência de vídeo a ser analisada*

O funcionamento do programa que simula o módulo de controlo de qualidade do mxfSPEEDRAIL está descrito no fluxograma representado na figura 4.13.

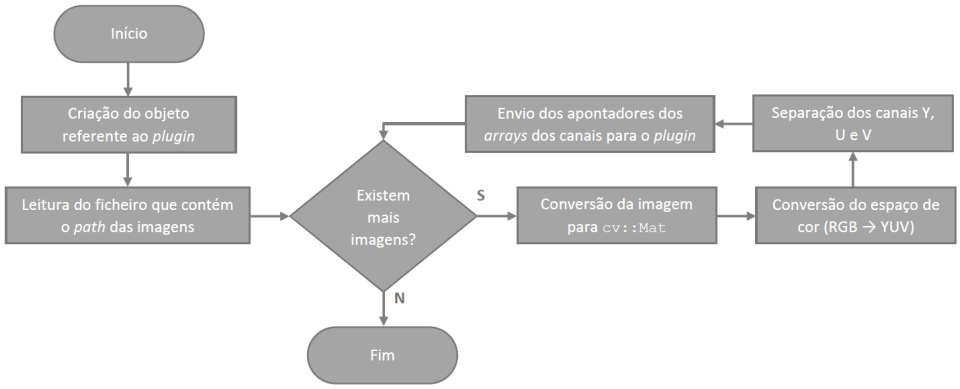


**Figura 4.13:** *Fluxograma que representa o funcionamento do simulador do módulo de controlo de qualidade*

Na figura 4.14 é possível verificar o funcionamento do simulador do módulo de controlo de qualidade.

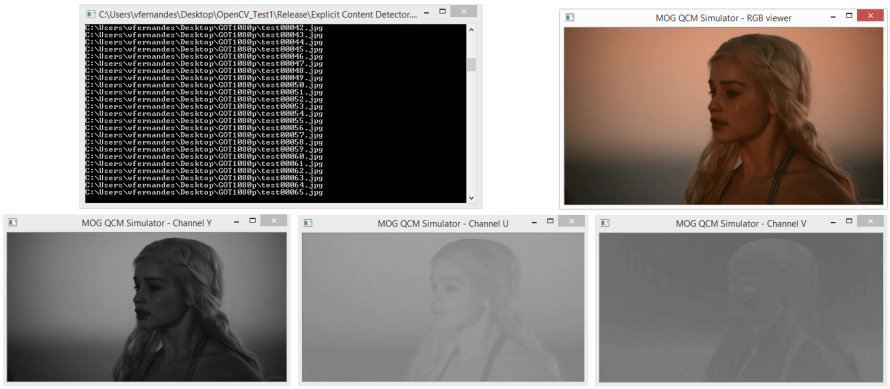


**Figura 4.14:** *Funcionamento do simulador do módulo* `MOG QCM`

### 4.6.3 Protótipo do *plugin* `MOG QC ECD`

A figura 4.15 descreve o funcionamento do *plugin* `MOG QC ECD`. Quando é iniciada a comunicação com módulo de controlo de qualidade, no *plugin* é executado o método `initialize`, onde são feitas todas as inicializações associadas ao funcionamento do *plugin*.

O método `explicitContentAnalyser` é onde são executados todos os métodos associados ao processo de análise e classificação do tipo de conteúdo.

É utilizado o método `createMat` de modo adaptar a informação ao OpenCV e neste estão implementados mecanismos de exceções para ser possível verificar a

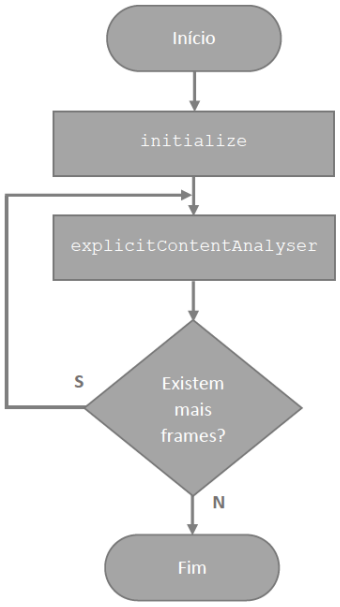


**Figura 4.15:** *Fluxograma associado ao funcionamento do plugin* `MOG QC ECD`

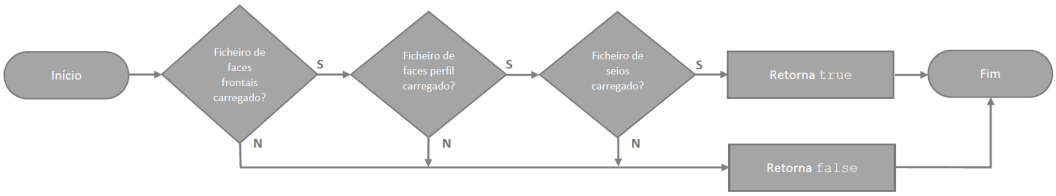


**Figura 4.16:** *Funcionamento do método* `initialize` *pertencente ao plugin* `MOG QC ECD`

ocorrência de qualquer erro associado a este processo.

A primeira etapa a ser executada é a segmentação de pele (associada ao método `skinClassifier`). Após a execução deste método, efetua-se o redimensionamento do canal Y (o processo de deteção de objetos é associado a imagens em tons de cinzento, desta forma a utilização do canal Y é feita de forma aproveitar os recursos disponíveis).

A razão pela qual se realiza o redimensionamento do canal Y, está associada com a diminuição da carga de processamento relacionado com a etapa de deteção de faces e seios. Este aspeto é muito importante pois permite manter a coerência e o tempo despendido no processo de deteção de objetos, para qualquer ficheiros de vídeo, independentemente da resolução do ficheiro 720p, 1080p ou até mesmo 4k.

Desta forma, a largura definida para o processo redimensionamento da imagem é igual a 380 pixeis. Utilizando a equação 4.2 é possível calcular a dimensão da altura, mantendo assim o *aspect ratio* da imagem original.

$$altura\ redimensionamento = \frac{largura\ definida \times altura\ original}{largura\ original} \tag{4.2}$$

Após o redimensionamento, são executados os métodos de deteção de faces, deteção de seios e por fim é efetuada a classificação da *frame*.

A classificação é realizada com base nos resultados obtidos em cada uma das etapas associadas aos processo de deteção de conteúdo explícito.

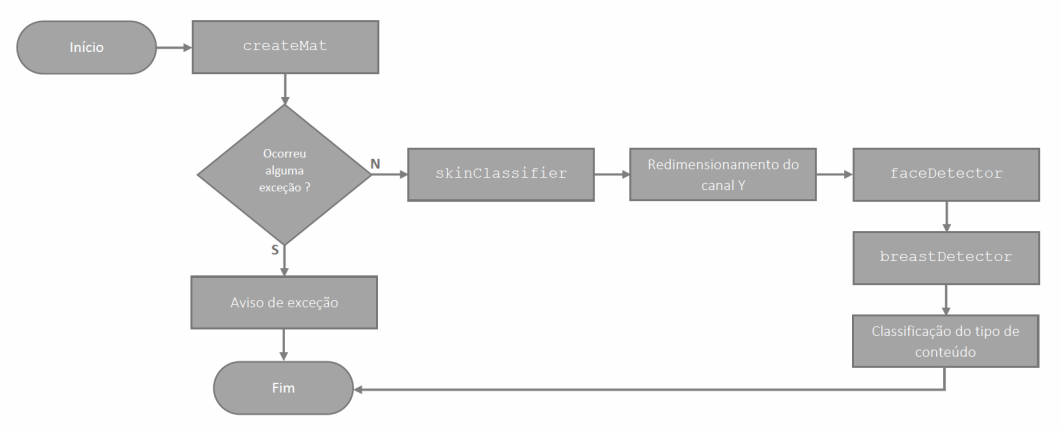


**Figura 4.17:** *Funcionamento do método* `explicitContentAnalyser`

# 5

# Integração do *plugin* no mxfSPEEDRAIL F1000

Neste capítulo são abordadas as etapas que envolveram a adaptação e a integração das funcionalidades do *plugin* desenvolvido no produto mxfSPEEDRAIL F1000.

A principal característica dos produtos da MOG Technologies é o facto de estes apresentarem compatibilidade com a maior parte dos formatos de vídeo presentes nos *workflows* do mercado do audiovisual profissional. Associada a esta característica está uma grande variedade de formatos de *chroma subsampling* e *bit depth* presentes nas sequências de vídeo que são submetidas ao processo de *ingest*.

Na fase inicial deste capítulo, será efetuada uma breve descrição do funcionamento do módulo de controlo de qualidade `MOG QCM`, já que este serve de interface de ligação entre o *plugin* e a informação associada às sequências de vídeo que são submetidas ao processo de *ingest*.

Depois da apresentação do módulo `MOG QCM`, serão descritas as diferentes fases de adaptação da informação proveniente deste módulo ao *plugin* `MOG QC ECD`, e as modificações que foram efetuadas ao *plugin*, de modo a optimizar o seu consumo de recursos computacionais e o seu tempo de resposta.

## 5.1 Módulo `MOG QCM`

Para ser possível desenvolver um *plugin* compatível com o módulo de controlo de qualidade dos produtos da MOG Technologies, tornou-se necessário efetuar uma normalização ao protótipo desenvolvido anteriormente. Na versão mais atual do

módulo `MOG QCM`, apenas é garantido aos *plugins* o acesso a três tipos de dados: *(i)* Vídeo sem qualquer tipo de compressão; *(ii)* Vídeo comprimido; *(iii)* Áudio sem compressão;

A especificação deste módulo impõem que o *plugin* a ser integrado no produto deve conter pelo menos duas funções e quatro métodos específicos. A disponibilização do *plugin* deverá ser feita sobre a forma de uma Dynamic-link library (DLL) [60].

As duas funções associadas ao *plugin* são: a função `allocate` e `deallocate`. Estas efetuam a alocação de memória associada ao objeto e a libertação de memória no final do processo de análise da sequência de vídeo, respetivamente.

No fundo o *plugin* vai ser o objeto que vai incluir os métodos associados às normas impostas pelo módulo de controlo de qualidade. Os métodos associados ao *plugin* são: `getInformation`, `initialize`, `handleFrame` e `getResult`. Embora o *plugin* possa incluir mais métodos, apenas os referidos anteriormente são utilizados para efetuar a comunicação com o módulo `MOG QCM`. Na tabela 5.1, estão descritas as funcionalidades de cada um dos métodos.

**Tabela 5.1:** *Métodos associados ao desenvolvimento de plugin de acordo com as especificações do módulo `MOG QCM`*

| Método | Descrição |
|---|---|
| `getInformation` | É utilizado para o *plugin* especificar qual o tipo de informação que este necessita. Para tal, é necessário que este preencha corretamente a estrutura de dados `MOG_QC_DLL_Information` (apresentada no fim deste capítulo). |
| `initialize` | É utilizado para inicializar certos parâmetros associados ao *plugin*, e para verificar se o tipo de informação enviada pelo módulo corresponde ao tipo de informação requisitada no método `getInformation`. Este sinaliza o correto funcionamento do processo de inicialização. |
| `handleFrame` | É o único método que pode ser invocado mais do que uma vez durante o processo de comunicação com o módulo. No final da sua execução, este retorna o resultado da análise efetuada à informação recebida, sob a forma de uma variável do tipo `wstring` com o resultado codificado em XML. A este método está associado o objeto `element`. Este objeto contém dois tipos de dados: *(i)* dados que armazenam a informação de cada *frame*; *(ii)* dados de áudio. |
| `getResult` | É invocado quando não existem mais *frames* para serem analisadas. Este método retorna a informação de todas as *frames* analisadas em XML. |

A estrutura de dados `MOG_QC_DLL_Information` é composta por três elementos: `name`, `version` e `need`. Os dois primeiros elementos representam a informação associada ao *plugin* de controlo de qualidade. O elemento `need`, é preenchido de acordo com o tipo de informação que o *plugin* necessita.

```
typedef struct MOG_QC_DLL_Information{
   std::wstring name;
   std::wstring version;
   bool need[3];
} MOG_QC_DLL_Information;
```

Na figura 5.1, de uma forma genérica, está descrito o funcionamento do módulo MOG QCM e o seu enquadramento no fluxo de informação durante o processo de *ingest*.

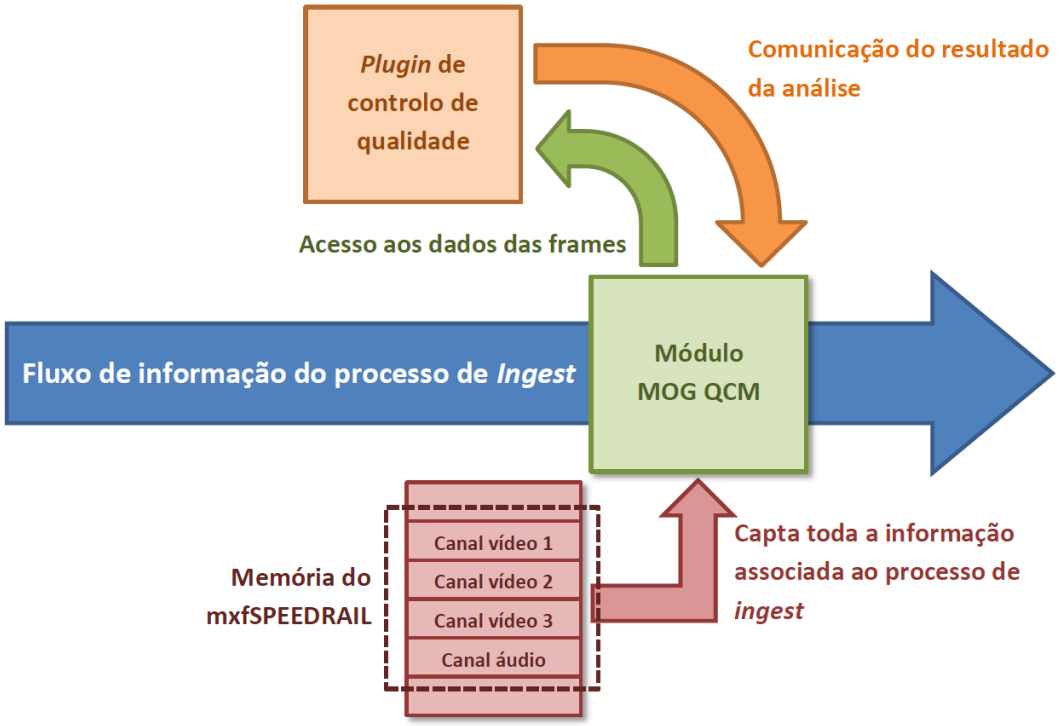


**Figura 5.1:** *Funcionamento do módulo de controlo de qualidade MOG QCM*

### 5.1.1 Processo de comunicação entre o módulo e o *plugin*

Na figura 5.2, está descrito o processo de comunicação entre o *plugin* de controlo de qualidade e o módulo `MOG QCM`, durante o processo de análise do conteúdo de cada uma das *frames* da sequência de vídeo submetida ao processo de *ingest*.

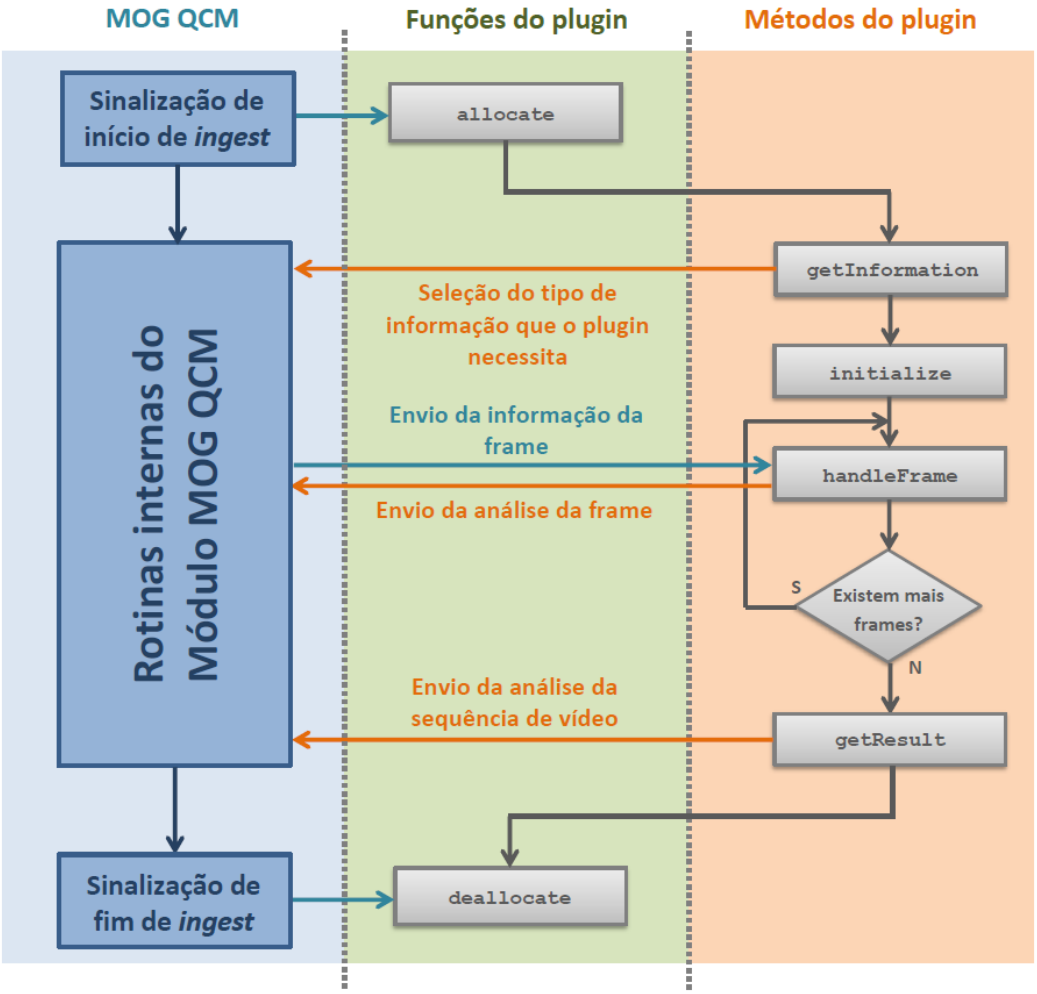


**Figura 5.2:** *Comunicação de informação entre o módulo `MOG QCM` e os plugins de controlo de qualidade*

O processo de comunicação é sempre iniciado pelo módulo `MOG QCM`. Logo que o processo de *ingest* é iniciado, o *plugin* executa executa a função `allocate`. Após a execução desta função, é utilizado o método `getInformation` de modo requisitar o tipo de informação que o *plugin* deve receber.

Durante o processo de *ingest* cada *frame* é processada individualmente e sempre que uma *frame* é processada, a comunicação entre o módulo e o *plugin* é efetuada através do método `handleFrame`.

Quando não existem mais *frames* para analisar é executado o método `getResult`,

de modo a comunicar ao módulo todos os resultados associados a cada frame analisada.

### 5.1.2 Adaptação do *plugin* MOG QC ECD ao módulo MOG QCM

O processo de adaptação do protótipo desenvolvido baseou-se nas especificações impostas pelo módulo MOG QCM tendo sido necessário adaptar o mesmo de modo a incluir as funções e os métodos descritos anteriormente. Na tabela 5.2, estão descritas as modificações efetuadas nos principais métodos do *plugin*.

**Tabela 5.2:** *Modificações efetuadas nos métodos do plugin MOG QC ECD*

| Método | Modificações efetuadas |
|---|---|
| getInformation | Este é utilizado para requisitar o tipo de informação, no contexto deste *plugin* é necessário preencher a estrutura de dados do tipo MOG_QC_DLL_Information de forma a receber a informação relativa aos canais Y, U e V que constituem a frame (sem qualquer tipo de compressão). |
| initialize | Embora seja utilizado para verificar a conformidade do tipo de informação recebida pelo *plugin*, a este método foram adicionadas funcionalidades capazes de efetuar o carregamento dos ficheiros XML associados aos classificadores utilizados no processo de deteção de objetos (faces e seios) nas *frames*. |
| handleFrame | Foram adicionadas as funcionalidades e os módulos que permitem efetuar a classificação do conteúdo da frame. |

Na figura 5.3 está descrita a forma de como o *plugin* deve preencher a estruturas MOG_QC_DLL_Information de modo a receber a informação de vídeo sem qualquer tipo de compressão, vulgarmente conhecido como RAW[1] em ambientes de pós-produção profissional.

| NAME | Version | CODED_VIDEO | UNCODED_VIDEO | UNCODED_AUDIO |
|---|---|---|---|---|
| MOG_QC_ECD | 1.0.0 | false | true | false |

**Figura 5.3:** *Preenchimento da estrutura MOG_QC_DLL_Information, de modo a receber apenas vídeo*

Na figura 5.4, está representada a implementação de toda a lógica do *plugin* associada ao processo de classificação de conteúdo da frame.

[1]RAW não é um acrónico, mas sim um adjetivo muito utilizado para descrever este tipo de formatos. A sua utilização deve-se ao significado da palavra *raw* em inglês. Neste contexto significa que o formato não possui qualquer tipo de compressão, representado a informação tal como foi capturada pelo sensor da câmara.

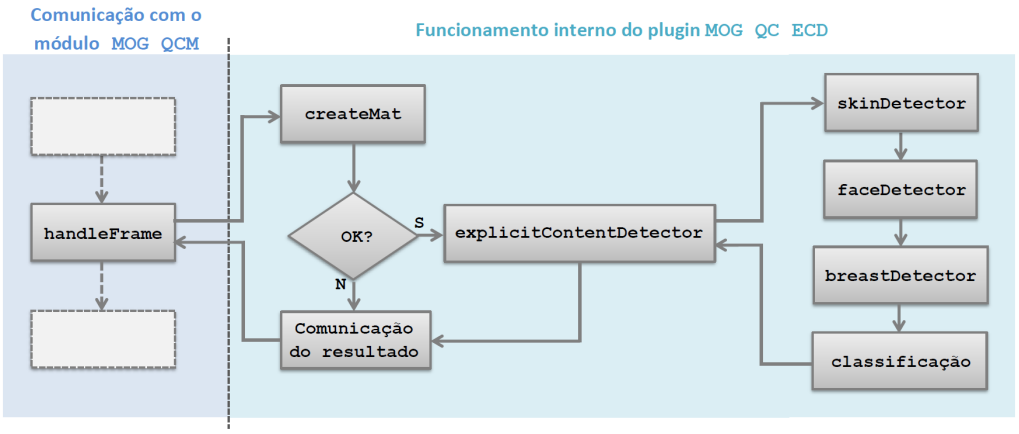


**Figura 5.4:** *Integração das funcionalidades do plugin utilizando as normalizações do módulo MOG QCM*

## 5.2 Adaptação da informação recebida

Toda a lógica do *plugin* desenvolvido é baseada nas funcionalidades de visão computacional presentes na *framework* OpenCV. Como consequência, toda a análise das *frames* é efetuada com base nos métodos e tipos de dados associados a esta *framework*. Desta forma. é necessário realizar a adaptação dos dados recebidos por parte do módulo `MOG QCM`.

Tal como no protótipo, a adaptação é efetuada através do método `createMat` associado ao *plugin* `MOG QC ECD`. Esta função realiza leitura do conjunto de três *arrays* de caracteres associados a cada um dos canais associado à *frame*, e armazena a informação das *frames* em objetos do tipo `cv::Mat`.

Certos parâmetros dos vídeos tornaram impossível a representação da *frame*, utilizando as funcionalidades do OpenCV. Esses parâmetros, estão associados aos diferentes tipos de formatos existentes no fluxo de informação de *ingest* no produto mxfSPEEDRAIL F1000.

Para representar a informação das sequências de vídeo, no produto existem dez tipos de formatos resultantes da combinação de diferentes tipos de sub-amostragem dos canais de crominância (*chroma subsampling*) e de diferentes valores de profundidade de cor (*bit depth*). Os diferentes formatos podem ser consultados na tabela 5.3. Face esta característica, foi necessário incluir um novo campo no método `explicitContentAnalyser` de modo a identificar os diferentes formatos de *chroma subsampling* e o valor da profundidade de cor, sendo este parâmetro fundamental para a correta adaptação da informação.

```
wstring MOG_QC_ECD::explicitContentAnalyser(char *Y, char
    *U, char *V, int Y_width, int Y_height, int U_width, int
    U_height, int V_width, int V_height, int format)
```

### 5.2.1 Formatos de *chroma subsampling*

Os formatos de chroma *subsampling* mais comuns no processo de *ingest* no produto mxfSPEEDRAIL F1000 são: 4:2:2, 4:2:0 e 4:1:1. Embora este produto esteja preparado para o formato 4:4:4, este tipo de formato é muito pouco utilizado durante a utilização regular do produto.

O processo de sub-amostragem apenas é realizado nos canais de crominância, sendo observáveis as diferenças apenas nesses canais. Durante a fase de adaptação foi necessário criar os objetos do tipo `cv::Mat` associados aos canais U e V, com as dimensões corretas, pois o processo de *chroma subsampling* altera apenas as resoluções das camadas de crominância. Como possível verificar na figura 5.5, nesta estão representadas as diferenças que existem na resolução dos diversos formatos:

- **Formato 4:4:4** - os canais U e V, mantêm-se inalterados a nível da resolução vertical e horizontal ;
- **Formato 4:2:2** - neste tipo de formato os canais U e V, mantêm a resolução vertical do canal Y e a resolução horizontal é reduzida para metade;
- **Formato 4:2:0** - tanto a resolução vertical como horizontal dos canais de crominância são reduzidos para metade;
- **Formato 4:1:1** - este formato mantêm a resolução vertical igual à original, e reduz a resolução horizontal com o fator 1/4.

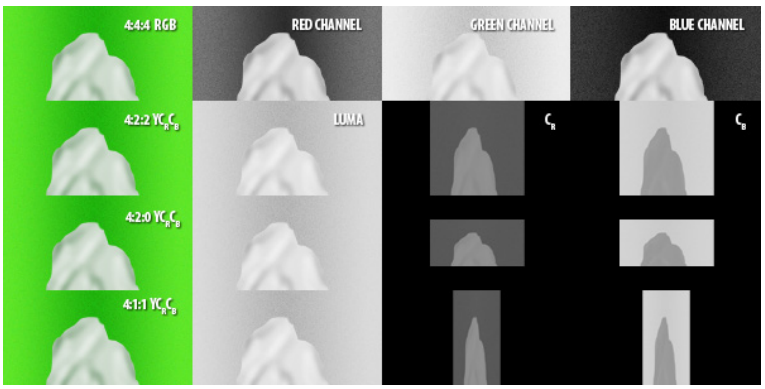


**Figura 5.5:** *Resoluções verticais e horizontais para diferentes formatos de chroma subsampling [61]*

### 5.2.2 *Bit depth*

O parâmetro *bit depth* (em português, profundidade de cor) está relacionado com o número de cores que cada pixel da imagem pode representar. Esta propriedade pode ser observada na figura 5.6, onde o parâmetro de *bit depth* utilizado para representar a mesma imagem é aumentado e verifica-se um aumento de qualidade da imagem representada.

**Figura 5.6:** *Qualidade de imagem associada a diferentes valores de bit depth (1 bit, 4 bits, 8 bits, 24 bits) [62]*

No caso do fluxo de dados existentes nos produtos da MOG Technologies, apenas são considerados dois tipos de valores de *bit depth* nos diversos formatos de vídeo: 8 bits ou 10 bits.

Embora o OpenCV apresente compatibilidades com imagens que apresentem valores de *bit depth* superiores a 8 bits, durante o processo de adaptação verificou-se que em *frames* em que o valor de *bit depth* fosse igual a 10 bits, o OpenCV era incapaz de representar corretamente a imagem (imagem do lado esquerdo da figura 5.7).

**Figura 5.7:** *Correção do problema associado ao bit depth*

Para tornar o *plugin* compatível com todos os formatos, foi criado o método `bitConvertion` de modo a ser possível adaptar sequências de vídeo com valores de *bit depth* iguais a 10 bits. Este método percorre todos os pixeis que compõem as *frames* e efetua a operação lógica de deslocamento de dois bits à esquerda de modo a converter a informação de 10 bits para 8 bits. Desta forma é possível tornar todos os formatos com valores de *bit depth* iguais a 10 bit compatíveis com as etapas de classificação do conteúdo presente nas *frames*.

Após o processo de adaptação do tipo de dados que o *plugin* recebe por parte do módulo `MOG QCM` ao tipo de dados compatíveis com as funções do OpenCV, este ficou preparado para interagir corretamente com a maioria de sequências que é submetida ao processo de *ingest* no produto. Na tabela 5.3 estão listados todos os tipos de formatos presentes no processo de *ingest* no mxfSPEEDRAIL F1000 e as suas características.

Embora seja possível adaptar o *plugin* ao formato YUV444P10, o funcionamento deste não foi verificado devido à inexistência de sequências de vídeo deste tipo de formato, no repositório de vídeos da MOG Technologies.

**Tabela 5.3:** *Formatos de chroma subsampling presentes no fluxo de dados do produto mxfSPEEDRAIl F1000 e as suas propriedades*

| **Formato** | ***Bit Depth*** | **Ordenação** | ***Chroma subsampling*** | **Grupo** | **Compatível com o *plugin*** |
|---|---|---|---|---|---|
| **UYVY422** | 8 bits | *Big Endian* | 4:2:2 | *Packed* | ✓ |
| **V210** | 10 bits | *Big Endian* | 4:2:2 | *Packed* | ✓ |
| **YUV420P** | 8 bits | *Big Endian* | 4:2:0 | *Planar* | ✓ |
| **YUV411P** | 8 bits | *Big Endian* | 4:1:1 | *Planar* | ✓ |
| **YUV422P** | 8 bits | *Big Endian* | 4:2:2 | *Planar* | ✓ |
| **YUV422P10** | 10 bits | *Big Endian* | 4:2:2 | *Planar* | ✓ |
| **YUV444P10** | 10 bits | *Big Endian* | 4:4:4 | *Planar* | ✗ |
| **YUV420P10LE** | 10 bits | *Liitle Endian* | 4:2:0 | *Planar* | ✓ |
| **YUVJ420P** | 8 bits | *Big Endian* | 4:2:0 | *Planar* | ✓ |
| **YUVJ422P** | 8 bits | *Big Endian* | 4:2:2 | *Planar* | ✓ |

## 5.3 Desempenho do *plugin* no mxfSPEEDRAIL F1000

Para efetuar a análise do desempenho utilizou-se um conjunto de sequências de vídeo com características distintas: diferentes formatos de *chroma subsampling*, vários valores de *bit depth* e resoluções distintas (720p e 1080p).

Os dados apresentados na figura 5.8 foram obtidos através da análise de sequências de vídeo distintas, totalizando 6458 *frames* analisadas durante o processo de *ingest*.

Como é possível verificar, a maior parte do tempo, 50 ms, é despendido na deteção de objetos representando assim cerca de 42% do tempo médio de análise de cada frame. O processo de deteção de seios tem uma duração média de 30 ms, representando 25% do tempo total. Por fim, os processos mais simples durante a análise do conteúdo da frame são: o processo de adaptação ao OpenCV e o processo de

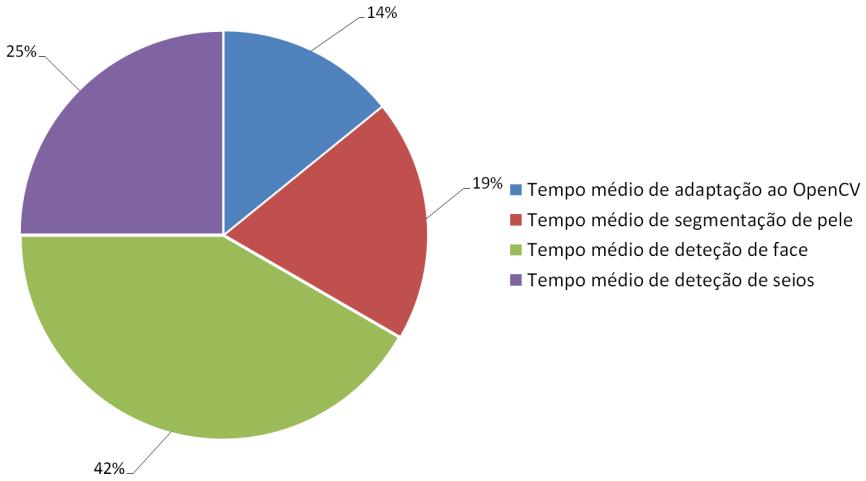


**Figura 5.8:** *Tempo despendido durante a execução das diferentes etapas de classificação de conteúdo explícito*

segmentação de pele, sendo despendido cerca de 17 ms e 23 ms em cada uma destas duas tarefas, respetivamente. Desta forma o tempo médio de classificação de cada frame é igual a 120 ms.

É importante frisar que o processo de deteção de faces é mais dispendioso que o processo de deteção de seios, pois durante o mesmo são efetuadas duas deteções distintas - deteção de faces na posição frontal e de perfil.

O consumo de recursos (processador e memória) em cada uma das etapas foi também avaliado. Utilizando as funcionalidades de *profiling* do ambiente de desenvolvimento do Microsoft Visual Studio, a figura 5.9 mostra que o consumo de recursos do processador se manteve relativamente constante e em cerca de 40%.

Nos resultados associados ao processo de *profiling*, foi possível verificar que maior parte do processamento foi gasto no processo de deteção de objetos associado aos métodos da *framework* de Viola-Jones e LBP.

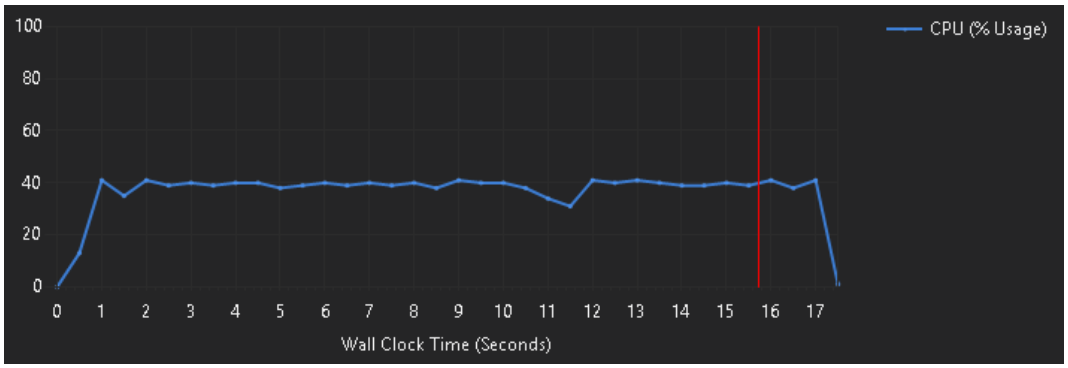


**Figura 5.9:** *Resultado do processo de profiling efetuado ao plugin*

Para conseguir obter os dados relativos à percentagem de memória utilizada pelo

*plugin*, foi necessário utilizar funções associadas à biblioteca `windows.h`. Depois do tratamento dos dados foi possível construir gráficos que representam o consumo de memória associado às sequências de vídeo com uma resolução de 720p (figura 5.10) e 1080p (figura 5.11). Neste gráficos estão representados os dados relativos ao consumo de memória por parte do processo de *ingest* (a azul) e o consumo associado à execução do *plugin* `MOG QC ECD`.

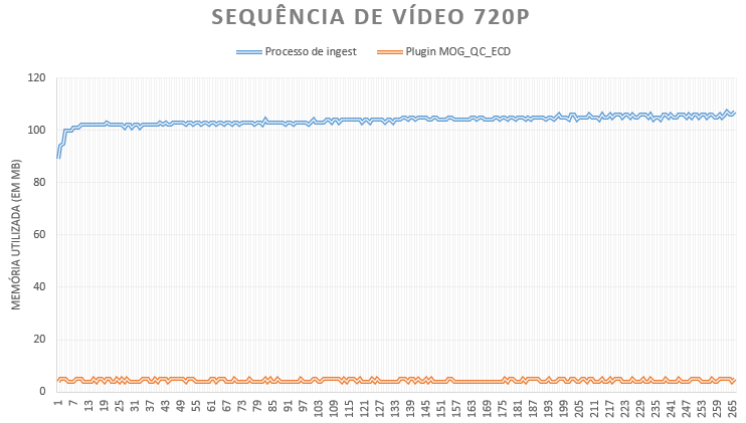


**Figura 5.10:** *Comparação do consumo de memória do processo de ingest e da execução do plugin numa sequência de vídeo com resolução de 720p*

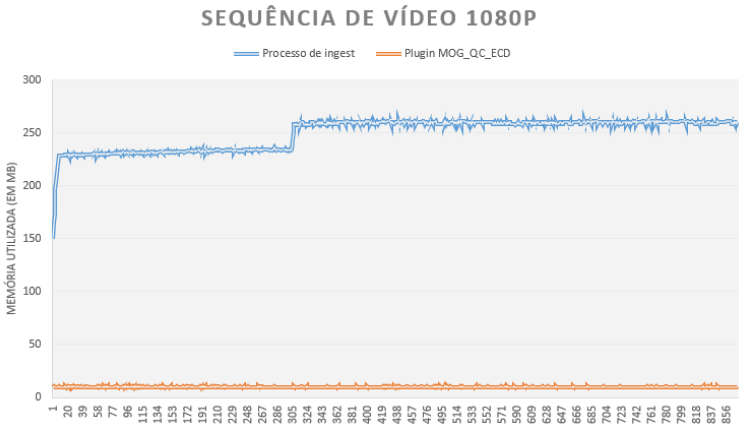


**Figura 5.11:** *Comparação do consumo de memória do processo de ingest e da execução do plugin numa sequência de vídeo com resolução de 1080p*

## 5.4 Optimizações efetuadas ao *plugin*

Como foi possível verificar no capítulo anterior, em determinadas partes do código eram utilizados mais recursos computacionais. Durante o processo de redimensionamento da frame (função `resize` do OpenCV) as dimensões originais da frame

são calculadas de forma desnecessária, pois o valor das mesmas é enviado durante o processo de comunicação entre o *plugin* e o módulo `MOG QCM`.

De modo a melhorar o funcionamento do *plugin*, este aspeto foi optimizado para ser possível reutilizar certas variáveis, não sendo assim despendido tempo e recursos no cálculo das mesmas. Depois destas modificações, realizou-se outra fase de testes para aferir se as modificações resultaram efetivamente numa diminuição do consumo de recursos e no tempo de resposta do *plugin*.

Outra abordagem utilizada com o efeito de diminuir o tempo médio, foi a integração de funcionalidades de multi-tarefa (*threads*). Desta forma, será possível executar em simultâneo, todas as etapas associadas à deteção de conteúdo explícito. Na figura 5.12 está apresentada a comparação entre os dois métodos que podem ser utilizados para efetuar a classificação do conteúdo da frame. No método A (método inicialmente utilizado) o processo de classificação é iterativo, demorando em média 120 ms a analisar e classificar cada frame da sequência de vídeo. O método B (método proposto como uma solução para a diminuição do tempo de classificação) tem como principal característica o facto da execução das diferentes etapas de classificação serem realizadas em simultâneo. Teoricamente o método B será capaz de reduzir o tempo médio despendido na classificação.

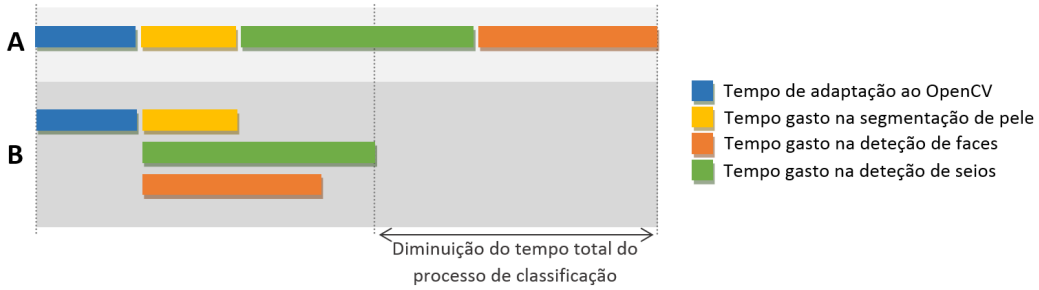


**Figura 5.12:** *Comparação do tempo despendido nos métodos A e B*

Após a adaptação do código do *plugin* para o método B, foram realizados novos testes utilizando as mesmas sequências de vídeo que foram utilizadas na primeira fase de testes, de modo a comprovar a melhoria efetuada ao *plugin*. Na tabela 5.4 estão representados dados resultantes da fase de testes efetuados ao *plugin* depois da adaptação do mesmo com as funcionalidades descritas no método B.

**Tabela 5.4:** *Resultados das melhorias efetuadas ao plugin `MOG QC ECD`*

| **Etapas** | **Adaptação ao OpenCV** | **Classificação de conteúdo** | **Total** |
|---|---|---|---|
| Tempo médio | 15 ms | 53 ms | 68 ms |

Embora o processo de adaptação da informação proveniente do módulo tenha sido

melhorado, foi a utilização método B que causou um maior impacto no funcionamento do *plugin*. A integração de funcionalidades de *multithreading* diminuiu o tempo despendido no processo de análise e classificação em cerca de 57% em relação ao tempo médio associado ao método A.

## 5.5 Integração do *plugin* no mxfSPEEDRAIL F1000

Uma das normalizações impostas pelo módulo `MOG QCM` é o facto dos métodos e funções associadas ao *plugin*, necessitarem de ser acessíveis através de uma DLL.

Após a criação da DLL referente ao *plugin* `MOG QC ECD`, foi necessário integrar a mesma no produto mxfSPEEDRAIL F1000.

Na Figura 5.13, estão apresentados os diretórios mais importantes para explicar o processo de integração da DLL associado ao *plugin* `MOG QC ECD`.

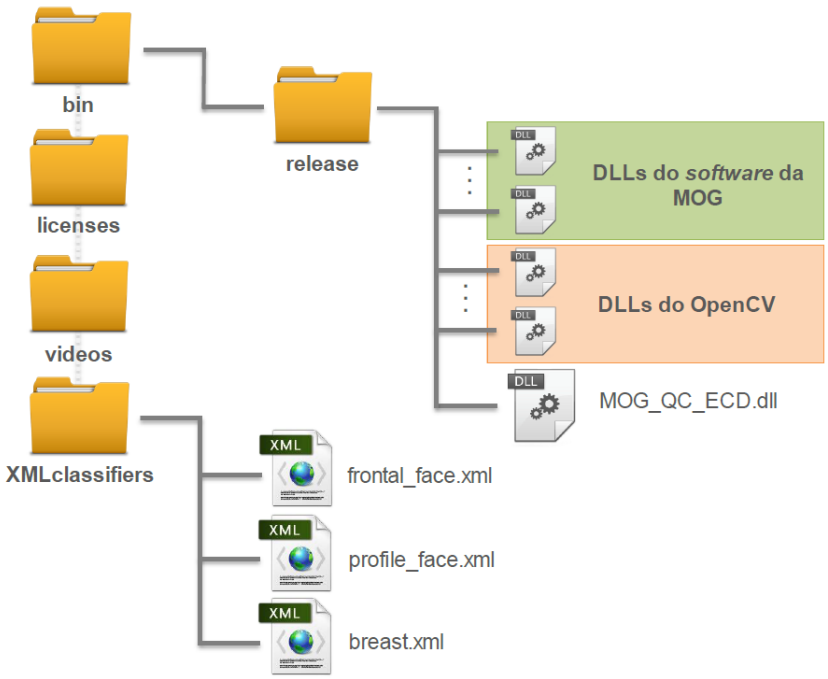


**Figura 5.13:** *Integração da DLL associada ao plugin `MOG QC ECD` no produto mxfSPEEDRAIL F1000*

No diretório `release` é necessário incluir todas as DLL associadas ao *software* da MOG e à *framework* OpenCV. Por fim, no mesmo diretório é incluída a DLL associada ao *plugin* que foi desenvolvido.

No diretório `licenses` estão incluídas as licenças que o módulo `MOG QCM` necessita para conseguir ser inserido no fluxo de dados do processo de *ingest*, sem as mesmas o módulo não conseguia disponibilizar a informação ao *plugin*.

Embora a implementação desta funcionalidade seja feita no produto, no contexto

desta dissertação é utilizada uma *custom build* do software do produto mxfSPEEDRAIL F1000 para efetuar todos os tipos de testes ao *plugin*.

Para que o processo de *ingest* seja iniciado é necessário forçar o mesmo através a execução de *scripts batch*. As sequências de vídeo a serem analisadas devem estar incluídas na pasta `videos`, de modo a que estas sejam submetidas ao processo de *ingest*.

Por fim, relacionado com o processo de deteção dos objetos são incluídos os ficheiros XML, sendo estes referentes aos classificadores necessários para a execução do processo de deteção faces e seios, utilizados pelas tecnologias LBP e *framework* de Viola-Jones.

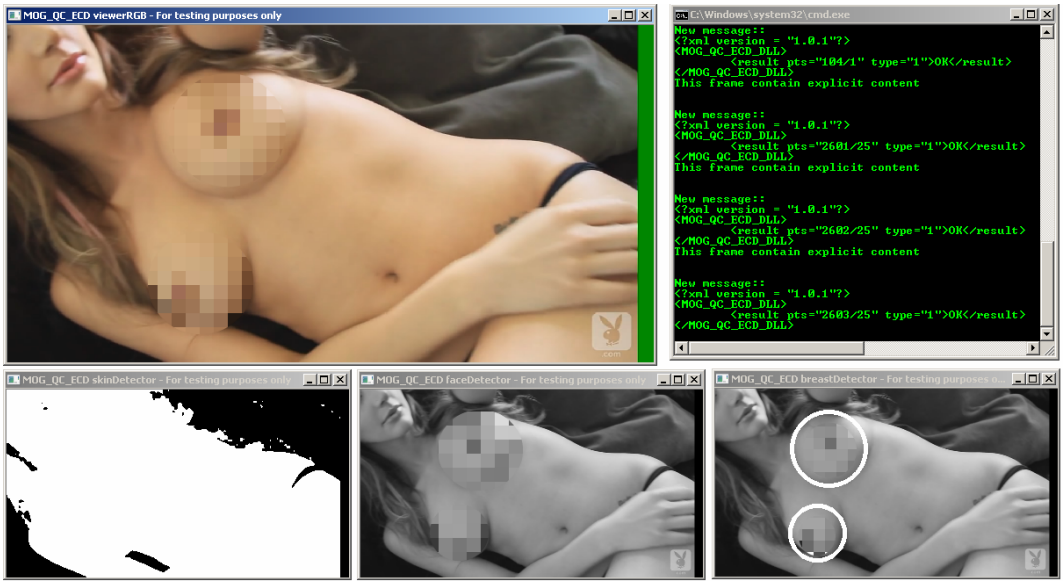


**Figura 5.14:** *Funcionamento do plugin* `MOG QC ECD` *e as respetivas fases de classificação de conteúdo explícito*

# 6

# Conclusão e Desenvolvimentos Futuros

O trabalho descrito nesta dissertação tem como objetivo o desenvolvimento de um módulo de deteção de nudez feminina a ser integrado num sistema de *ingest* profissional. De modo a aumentar a precisão do mesmo, o processo inclui três etapas: *(i)* segmentação de pele, *(ii)* deteção de faces (de modo a diminuir as falsas deteções associadas ao processo de segmentação); *(iii)* deteção de seios (para ser possível detetar conteúdo explícito com maior exatidão).

O facto do método de classificação desenvolvido não depender inteiramente da informação de cor, tornou esta solução mais robusta a falhas.

As principais desvantagens deste sistema são: o consumo de recursos computacionais (carga de processamento durante a deteção de faces e seios), e o facto de detetar conteúdo baseando-se apenas na deteção de seios na posição frontal. Uma das falhas detetadas nesta solução, foi sentida durante a análise de *frames* sucessivas, onde por vezes os objetos a serem detetados ficam desfocados durante o movimento, dificultando a deteção dos mesmos.

Este sistema será integrado no *workflow* do mxfSPEEDRAIL F1000, para que durante o processo de *ingest* seja possível gerar um relatório onde são sinalizadas as *frames* que possivelmente contêm nudez feminina. Desta forma o sistema apresenta-se como uma ferramenta útil para as equipas das estações televisivas, permitindo a simplificação do processo de classificação, não exigindo a visualização total das sequências de vídeo.

O primeiro aspeto que poderia ser melhorado seria o processo de segmentação de pele. Na solução descrita nesta dissertação são utilizados valores de *threshold*

fixos. Este aspeto pode ser melhorado através da utilização de um valor de *threshold* adaptável. Esta adaptação poderia ser efetuada através da análise do histograma da *frame*, de modo a diminuir os erros associados ao processo de segmentação de pele.

Apesar do processo de deteção de seios em imagens ter apresentado bons resultados durante a fase de testes ao *plugin*, o classificador utilizado poderá ser treinado de modo a tornar-se mais robusto. Para tal, como desenvolvimento futuro, seria interessante aumentar o *dataset* de imagens positivas utilizado durante o processo de treino do classificador de seios. Seria também necessário utilizar imagens negativas que contenham partes do corpo humano como olhos, faces, bocas, membros, etc., de modo a melhorar a qualidade de deteção e diminuir a percentagem de falsas deteções associadas a este processo.

Numa futura versão do *plugin*, poderia ser explorada a integração de novos classificadores associados a outras partes do corpo humano. Contudo, o aumento da diversidade de objetos a detetar nas *frames* tornaria o processo de classificação muito mais demorado e o *plugin* iria apresentar um maior consumo de recursos computacionais.

Um dos aspetos no qual esta dissertação se focou foi o estudo de como são consumidos os recursos computacionais durante o processo de deteção de material para adultos. Os testes realizados ao *plugin* `MOG QC ECD` utilizando a funcionalidade de *profiling*, associada ao ambiente de desenvolvimento Microsoft Visual Studio, demonstraram que a maior parte do tempo e processamento despendido, independentemente do tamanho e características do vídeo analisado, eram associados à etapa de deteção de faces e seios nas *frames*. Desta forma será importante, como um futuro desenvolvimento, encontrar uma solução que torne o processo de deteção de conteúdo explícito num processo mais fluído e menos exigente a nível de consumo de recursos do mxfSPEEDRAIL F1000.

A deteção de objetos na solução atual é efetuada em todas as *frames*. De maneira a diminuir a quantidade de processamento associado à sequência de vídeo, seria interessante dividir o processamento das *frames* em blocos de *frames*.

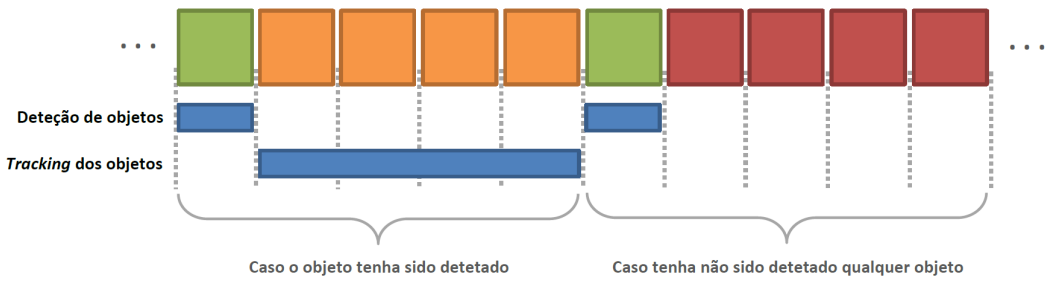


**Figura 6.1:** *Representação da solução baseada em tracking dos objetos durante o processo de classificação de conteúdo explícito*

Na figura 6.1, está apresentado um caso de uso desta solução. Neste caso os blocos

seriam resultantes da agregação de um conjunto de cinco *frames*. Desta forma, a sequência de vídeo seria dividida em blocos de *frames* e a maioria do processamento seria despendido na primeira frame de cada bloco.

Supondo que cada um dos quadrados da figura 6.1 representam uma *frame* da sequência de vídeo a ser analisada, apenas nas *frames* verdes seria efetuada a deteção de objetos. Caso no primeiro elemento do bloco de *frames* sejam detetados objetos, nas restantes quatro *frames* apenas seria efetuado o *tracking* dos mesmos.

O processo de *tracking* (seguimento de objetos em imagens) seria utilizado de modo a reduzir a área de procura dos objetos em cada bloco de *frames*, diminuindo assim tempo associado ao processo de procura do objeto, não sendo necessário procurar os mesmos na totalidade da *frame*. Caso os objetos não fossem encontrados na área associada ao processo de *tracking*, seria necessário verificar a existência dos mesmos ou de novos objetos em toda a *frame*. Este método seria repetido ao longo de todos os conjuntos de *frames*, melhorando assim o consumo de recursos associados ao processo de deteção de conteúdo explícito da sequência de vídeo a ser analisada.

De forma a tornar o sistema mais versátil, o intervalo de *frames* a agregar em blocos deverá poder ser especificado pelo utilizador do produto, sendo assim possível controlar a relação da qualidade de deteção *vs.* o consumo de recursos por parte dos processos de deteção e *tracking* dos objetos nas *frames* analisadas.

Esta página foi intencionalmente deixada em branco.

# Bibliografia


[1] Al Kovalick. A Quick Tour of Wrappers and MXF. *Advanced Media Workflow Association*, pages 1–2, 2012.

[2] Pedro Ferreira. MXF - a progress report. *EBU Technical Review*, pages 1–10, 2010.

[3] Sítio oficial da mog technologies, file based ingest. `http://www.mog-technologies.com/solutions/ingest-solutions/file-based-ingest/`, Acedido em 04-02-2014.

[4] Rashed Mustafa and Dingju Zhu. Objectionable Image Detection in Cloud Computing Paradigm-a Review. *Journal of Computer Science*, 9(12):1715–1721, December 2013.

[5] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6:21–44, 2006.

[6] Supervised learning workflow and algorithms. `http://www.mathworks.com/help/stats/supervised-learning-machine-learning-workflow-and-algorithms.html`, Acedido em 13-02-2014.

[7] Unsupervised learning. `http://www.mathworks.com/discovery/unsupervised-learning.html`, Acedido em 13-02-2014.

[8] R S Sutton and A G Barto. Reinforcement learning: an introduction. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 9:1054, 1998.

[9] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1—-27:27, 2011.

[10] Asa Ben-Hur and Jason Weston. A user's guide to support vector machines. *Methods in molecular biology (Clifton, N.J.)*, 609:223–239, 2010.

[11] Norm: No-reference image quality metric for realistic image synthesis. `http://www.cg.cs.tu-bs.de/teaching/seminars/ss13/CG/webpages/SoerenPetersen/`, Acedido em 23-06-2014.

[12] Raul Rojas. AdaBoost and the Super Bowl of Classifiers A Tutorial Introduction to Adaptive Boosting. *Freie Universitat Berlin*, pages 1–6, 2009.

[13] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory, Lecture Notes in Computer Science Volume 904*, volume 904, pages 23–37. 1995.

[14] Adaptative boosting. http://cmp.felk.cvut.cz/~sochmj1/adaboost_talk.pdf, Acedido em 23-06-2014.

[15] Julien Meynet. Fast face detection using adaboost. *PhD Thesis, École Polytechinque Fédérale de Lausanne*, 2003.

[16] J. Sochman and J. Malas. AdaBoost with totally corrective updates for fast face detection. *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, 2004.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1, 2001.

[18] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice-Hall, Inc., second edition, 2002.

[19] Image segmentation. https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic3.htm, Acedido em 08-03-2014.

[20] Sítio oficial da biblioteca opencv. http://www.opencv.org, Acedido em 05-03-2014.

[21] Luís Rebelo. Avaliação automática do resultado estético do tratamento conservador do cancro da mama. *MsC Thesis, Faculdade de Engenharia da Universidade do Porto*, 2008.

[22] João Manuel R. S. Carvalho, Fernando J. S. e Tavares. Detecção de Faces em Imagens baseada na Identificação da Pele e dos Olhos. *Polytechnical Studies Review*, 2008.

[23] Hardy Francke, Javier Ruiz-del solar, and Rodrigo Verschae. Real-Time Hand Gesture Detection and Recognition Using Boosted Classifiers and Active Learning. *Lecture Notes in Computer Science*, 4872:533–547, 2007.

[24] Margaret M. Fleck, David A. Forsyth, and Chris Bregler. Finding naked people. *Computer Vision ECCV 96*, 1065:593–602, 1996.

[25] P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recognition*, 40:1106–1122, 2007.

[26] Ahmed Elgammal, Crystal Muang, and Dunxu Hu. Skin detection - A short tutorial. *Encyclopedia of Biometrics*, pages 1–10, 2009.

[27] A comprehensive guide to parallel video decoding. http://emericdev.wordpress.com/2011/08/26/a-comprehensive-guide-to-parallel-video-decoding/, Acedido em 12-04-2014.

[28] Vladimir Vezhnevets. A Survey on Pixel-Based Skin Color Detection Techniques. *Cybernetics*, 85:85–92, 2003.

[29] A. Pinz K. Brkic and S. Segvic. Traffic sign detection as a component of an automated traffic infrastructure inventory system. *Austrian Association for Pattern Recognition (OAGM/AAPR)*, 2009.

[30] S. Nedevschi, R. Danescu, D. Frentiu, T. Marita, F. Oniga, C. Pocol, R. Schmidt, and T. Graf. High accuracy stereo vision system for far distance obstacle detection. *IEEE Intelligent Vehicles Symposium, 2004*, 2004.

[31] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6:271–288, 1998.

[32] Domingo Mery. Automated detection in complex objects using a tracking algorithm in multiple X-ray views. *CVPR 2011 WORKSHOPS*, pages 41–48, 2011.

[33] Diana Turcsany, Andre Mouton, and Toby P. Breckon. Improving feature-based object recognition for X-ray baggage security screening using primed visualwords. In *Proceedings of the IEEE International Conference on Industrial Technology*, pages 1140–1145, 2013.

[34] Dennis Mitzel and Bastian Leibe. Real-time multi-person tracking with detector assisted structure propagation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 974–981, 2011.

[35] E Bermejo, O Deniz, G Bueno, and R Sukthankar. Violence Detection in Video Using Computer Vision Techniques. *CAIP'11 Proceedings of the 14th international conference on Computer analysis of images and patterns - Volume Part II*, 2011.

[36] Staffan Ekvall, Danica Kragic, and Patric Jensfelt. Object detection and mapping for service robot tasks. *Robotica: International Journal of Information, Education and Research in Robotics and Artificial Intelligence*, 25(2):175–187, March/April 2007.

[37] P Viola and M J Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[38] Shengcai Liao, Xiangxin Zhu, Zhen Lei, Lun Zhang, and Stan Z. Li. Learning multi-scale block local binary patterns for face recognition. In *Proceedings of the 2007 International Conference on Advances in Biometrics*, ICB'07, pages 828–837, Berlin, Heidelberg, 2007. Springer-Verlag.

[39] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. *Proceedings. International Conference on Image Processing*, 1, 2002.

[40] Timo Ojala, Matti Pietikäinen, and Topi Mäenpää. A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification. *Advances in Pattern Recognition*, 2013:399–408, 2001.

[41] Di Huang, Caifeng Shan, Mohsen Ardabilian, Yunhong Wang, and Liming Chen. Local Binary Patterns and Its Application to Facial Image Analysis: A Survey. *IEEE Transactions on Systems Man and Cybernetic Part C-Applications and Reviews*, 41:765–781, 2011.

[42] Jo Chang-yeon and Jo Chang-yeon. Face Detection using LBP features. *CS 229 Final Project Report, Standford University*, pages 1–4, 2008.

[43] T. Ahonen, A. Hadid, and M. Pietikainen. Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28, 2006.

[44] C. Ries and R. Lienhart. A survey on visual adult image recognition. *Multimedia Tools and Applications*, 69:661–688, 2014.

[45] Jorge Alberto Marcial Basilio, Gualberto Aguilar Torres, Gabriel Sanchez Perez, Linda Karina Toscano Medina, Hector Manuel Perez Meana, and Enrique Escamilla Hernadez. Explicit Content Image Detection. *Signal and Image Processing : An International Journal(SIPIJ)*, 1:47–58, 2010.

[46] Jorge Alberto Marcial Basilio, Gualberto Aguilar Torres, Gabriel Sánchez Pérez, L. Karina Toscano Medina, and Héctor M. Pérez Meana. Explicit Image Detection using YCbCr Space Color Model as Skin Detection, 2011.

[47] Yue Wang, Jun Li, Heelin Wang, and Zujun Hou. Automatic nipple detection using shape and statistical skin color information. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5916 LNCS, pages 644–649, 2009.

[48] Sítio oficial do software paraben - porn detection stick. `http://www.paraben.com/porn-detection-stick.html`, Acedido em 06-03-2014.

[49] Sítio oficial do software ftk. `http://www.accessdata.com/products/digital-forensics/ftk`, Acedido em 07-03-2014.

[50] Sítio oficial do software pnwatch. `http://www.yangsky.com/products/porndetect/`, Acedido em 07-03-2014.

[51] Sítio oficial do script nude.js. `http://www.patrick-wied.at/blog/nude-js-update-1`, Acedido em 10-03-2014.

[52] Sítio oficial do software matlab. `http://www.mathworks.com/index.html`, Acedido em 08-03-2014.

[53] Computer vision with matlab. `http://web.mit.edu/8.13/matlab/MatlabTraining_IAP_2012/ComputerVision/CVmasterclass%20IAP%202012.pdf`, Acedido em 08-03-2014.

[54] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly Media, Inc., first edition, September 2008.

[55] Sítio oficial da framework libcvd. `http://www.edwardrosten.com/cvd/`, Acedido em 08-03-2014.

[56] Sítio oficial da framework fastcv. https://developer.qualcomm.com/mobile-development/add-advanced-features/computer-vision-fastcv, Acedido em 07-03-2014.

[57] Natham Oostendorp Kurt Demaagd, Anthony Oliver and Katherine Scott. *Practial Computer Vision with SimpleCV*. O'Reilly Media, Inc., first edition, 2012.

[58] Sítio oficial da framework simplecv. http://www.simplecv.org, Acedido em 05-03-2014.

[59] Vision blocks. http://hcordeiro.com, Acedido em 07-03-2014.

[60] Pedro Santos. MOG Quality Control Manager - Specification. 2014.

[61] Green screen primer - chroma subsampling. http://www.creativeimpatience.com/tag/chroma-subsampling/, Acedido em 20-05-2014.

[62] Color depht. http://en.wikipedia.org/wiki/Color_depth, Acedido em 21-05-2014.

Esta página foi intencionalmente deixada em branco.

# 7
# Anexos

Neste capítulo está incluído o anexo A de modo a ser possível entender o enquadramento do *plugin*, desenvolvido nesta dissertação (`MOG QC ECD`), no fluxo de informação do processo de *ingest*.

## Anexo A - Arquitetura de integração do *plugin* desenvolvido no produto mxfSPEEDRAIL F1000