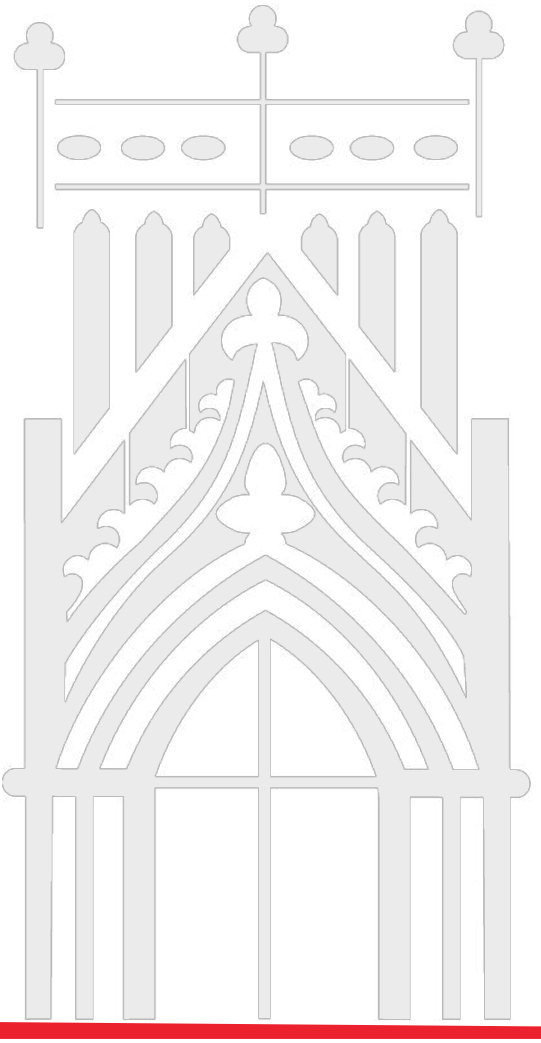
IPG Politécnico da Guarda
Polytechnic of Guarda

Mestrado em Computação Móvel

Sistema de Deteção de Cansaço para Condutores



Márcia Macedo Bernardino

janeiro | 2015

Escola Superior
de Tecnologia e Gestão

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

Sistema de Deteção de Cansaço para Condutores

Relatório de Projeto Aplicado submetido como requisite parcial para obtenção do grau de Mestre em Computação Móvel

Orientador: Professor Doutor Carlos Carreto

Márcia Macedo Bernardino

Janeiro | 2015

“Os conceitos e princípios fundamentais da ciência são invenções livres do espirito humano. ”

(Albert Einstein)

Agradecimentos

O presente trabalho resulta do meu esforço pessoal, em que a colaboração que me foi prestada assumiu um papel importante, por isso, gostaria de aqui expressar o meu reconhecimento e sinceros agradecimentos a todos os que me ajudaram a realizar este objetivo pessoal.

Assim, começaria por agradecer ao Professor Doutor Carlos Carreto, orientador do projeto aplicado, onde expresso o meu profundo agradecimento pela orientação e apoio incondicionais que muito elevaram os meus conhecimentos científicos e, sem dúvida, muito estimularam o meu desejo de querer, sempre, saber mais e a vontade constante de querer fazer melhor, bem como a partilha do saber e as valiosas contribuições para o trabalho. Acima de tudo, muito obrigada por me continuar a acompanhar nesta jornada e por estimular o meu interesse pelo conhecimento e pela vida académica.

A todos os meus amigos pelo incentivo e preocupação ao longo da realização deste trabalho.

Há minha mãe, pelo amor, incentivo e apoio incondicional.

Um agradecimento especial ao meu filho Hugo Alexandre pelas horas que não pude estar com ele.

À Cristiana o meu profundo obrigado e o mais sincero desejo de um dia poder retribuir toda a ajuda que me prestou.

O meu profundo e sentido agradecimento a todas as pessoas que contribuíram para a concretização desta dissertação, estimulando-me intelectual e emocionalmente.

Resumo

Ao longo dos últimos anos tem sido feito um grande esforço de investigação e desenvolvimento para criar os chamados Sistemas Avançados de Assistência ao Condutor capazes de monitorizar o seu estado ou a sua performance de condução de modo a alertá-lo aos primeiros sinais de perigo. Neste trabalho pretendeu-se estudar a viabilidade de usar *smartphones* para implementar este tipo de sistemas de assistência ao condutor.

O cansaço pode ser perigoso quando são realizadas tarefas que demandem concentração constante como é o caso da condução. As estatísticas têm mostrado que o cansaço dos condutores de veículos automóveis é uma das principais causas de acidentes de trânsito. O trabalho desenvolvido consistiu na implementação e validação de um sistema de deteção de cansaço para assistência a condutores, fazendo uso de recursos de Visão Computacional de modo que seja possível analisar a expressão facial do condutor e gerar um sinal de alarme se forem detetados sinais de cansaço através dos principais sintomas como olhos fechados e boca aberta. Além disso, procurou-se desenvolver um sistema que fosse robusto, confiável, de baixa complexidade computacional para o utilizador e que pudesse ser integrado num veículo de modo a não ser intrusivo ao condutor.

Os resultados obtidos através de testes experimentais realizados no trabalho mostram que é viável desenvolver este tipo de sistemas com base em tecnologias de *smartphone*.

Keywords Sistemas Avançados de Assistência ao Condutor, Deteção de Cansaço, Visão Computacional, Smartphone, Android

Abstract

Over the last few years a great effort has been made in the investigation and development of creating what is known as Advanced Driving Assistance Systems capable of monitoring the state or performance of the driver so as to alert him/her to the first signs of danger. In this assignment, we aim to study the feasibility of using smart phones to implement this type of assistance system to the driver.

Tiredness can be dangerous when tasks are carried out that require constant concentration, which is the case with driving. Statistics have shown that tiredness among automobile drivers is the main cause of road accidents. The work undertaken consists of the implementation and validation of a system that detects tiredness in order to help drivers. It resorts to computer vision resources to enable the analysis of the driver's facial expression and create an alarm if there are any signs of tiredness through the main symptoms such as eyes closing or an open mouth. Furthermore our aim was to develop a robust, trustworthy system that was easy for the user and which could be integrated into the vehicle in a non intrusive manner for the driver.

The results obtained from the experimental tests carried out at work showed that it is feasible to develop this type of system based on smart phone technologies.

Keywords Advanced Driving Assistance Systems for the Driver, Detection of tiredness, Computer Vision, Smartphone, Android

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

Este relatório descreve o trabalho realizado pela autora no âmbito do seu projeto aplicado do Mestrado em Computação Móvel do Instituto Politécnico da Guarda que consistiu no estudo e desenvolvimento de um sistema de deteção de cansaço para assistência a condutores, com base em tecnologias de *smartphones*.

1.1. Enquadramento e Motivação

O cansaço corresponde a um estado físico e emocional que condiciona o desempenho, e como consequência, uma diminuição das capacidades percetivas, cognitivas e motoras independentemente da atividade a que nos estejamos a referir. As capacidades necessárias à prática de uma condução segura de veículos ficam diminuídas logo que o estado de cansaço se desencadeia, muito antes de correr o risco de adormecer ao volante. Os principais sintomas são os bocejos frequentes, dificuldade de concentração, dificuldade em manter os olhos abertos e em os focar, sensação de picadas nos olhos ou de olhos pesados, entre outros.

Alguma vez adormeceu ao volante? Eis uma questão que pode ser respondida através de um estudo da Associação Portuguesa do Sono (2012) que confirma que doze por cento dos portugueses admitiu que fecharam os olhos enquanto conduziam. Esta é uma das causas de morte na estrada, portanto fica evidente a necessidade de encontrar um meio para reduzir a quantidade de acidentes que envolvem a sonolência ao volante.

Ao longo dos últimos anos tem sido feito um grande esforço de investigação e desenvolvimento para projetar sistemas capazes de monitorizar o condutor ou a sua performance de condução de modo a alerta-lo aos primeiros sinais de perigo o que orientou à inserção, no sistema rodoviário, de um conjunto de aplicações que funcionam através de computadores, sensores, sistemas de controlo, sistemas de comunicação e os mais diversos aparelhos eletrónicos. Todos estes elementos constituem os Sistemas Inteligentes de Transporte (também conhecidos por ITS – *Intelligent Transportation*

System). De acordo com o Instituto da Mobilidade e dos Transportes (2012), a segurança no sector rodoviário é o principal objetivo dos ITS e estes sistemas podem estar presentes na infraestrutura rodoviária ou nos próprios veículos. São exemplos destes sistemas os painéis de mensagens variáveis no ambiente rodoviário que podem informar acerca do estado da via, da existência de acidentes ou outras complicações no trânsito; os sistemas de navegação que disponibilizam e guiam o condutor através de um percurso, evitando que este se desvie da rota pretendida, consumindo menos tempo e gastando menos combustível; os sistemas de alerta em situação de emergência que permitem o envio automático de mensagens de auxílio em caso de acidente. Esta eficácia fornecida ao sistema pode também traduzir-se em benefícios individuais para o condutor e para o ambiente.

Os sistemas avançados de assistência ao condutor (ADAS - *Advanced Driver Assistance Systems*) fazem parte dos ITS e são sistemas que estão presentes nos veículos. Estes sistemas têm o desígnio de auxiliar, na tarefa de condução, na execução de manobras de diversas formas através de sinais ao condutor para alertar acerca de uma determinada situação, sendo o condutor informado da necessidade ou urgência de realização de uma ação sobre um comando. Podem também agir automaticamente antecipando ou até mesmo substituindo a ação do condutor. Os controladores de velocidade, os limitadores de velocidade, os sistemas de controlo de estabilidade dinâmica, o avisador acústico de marcha atrás, os sistemas de assistência à manutenção na faixa de rodagem são exemplos deste tipo de sistemas.

Os fabricantes de automóveis estão a apostar cada vez mais nos ADAS e já começam a surgir nos novos modelos de automóveis de algumas marcas. Enquanto esperamos pela adoção massiva desses sistemas por parte dos fabricantes de automóveis, uma opção interessante é desenvolver estes sistemas com base em plataforma de *smartphones* para o condutor utilizar como dispositivo de monitorização e alarme. A ideia surge no sentido que, estas plataformas, possuem muitos recursos tais como câmaras, sensores, etc., com capacidade de processamento cada vez maior, com interfaces com o utilizador muito ricas e prontas a usar e também porque são plataformas de uso massivo (os próprios condutores provavelmente já têm um *smartphone*).

Assim, neste trabalho pretendeu-se estudar a viabilidade de usar *smartphones* para implementar ADAS, através do desenvolvimento de um sistema de deteção de cansaço em condutores. A secção seguinte apresenta uma visão geral do sistema proposto.

1.2. Sistema Proposto

Visão Computacional, segundo os autores *Milano* e *Honorato* (2010), é considerada uma ciência muito recente e pode ser definida como o estudo e desenvolvimento de métodos através dos quais os sistemas computacionais são capazes de interpretar imagens, extraindo informações significativas a partir de imagens capturadas por câmaras de vídeo, sensores, *scanners*, entre outros dispositivos. Estas informações permitem reconhecer, manipular e pensar sobre os objetos que compõem uma imagem. Dessa forma, a Visão Computacional fornece ao computador uma infinidade de informações precisas a partir de imagens e vídeos, de forma que o computador consiga executar tarefas inteligentes, simulando e aproximando-se da inteligência humana.

Integrar os recursos de Visão Computacional é o objetivo principal deste projeto de modo que os seus meios sejam confiáveis e de baixa complexidade computacional, com a finalidade de desenvolver uma aplicação para *smartphones* que seja capaz de analisar a expressão facial e gerar um sinal de alarme se forem detetados sinais de cansaço, como por exemplo, os olhos fechados e boca aberta. A arquitetura do sistema tem como base os seguintes módulos:

- Aquisição de Imagem: É o primeiro passo para um sistema de Visão Computacional. Trata-se do processo de aquisição de uma imagem ou de um conjunto de imagens a partir de sensores de câmaras.
- Deteção de Faces: Processo realizado para destacar regiões relevantes da imagem, segmentando-as para processamento posterior, nomeadamente a deteção de face do condutor. O algoritmo inicialmente proposto foi o método de Viola-Jones pelo facto de ser provavelmente o método de deteção de faces humanas mais citado na literatura especializada. Contudo, após a implementação deste método verificou-

se que apesar de ser muito eficiente possuía pontos negativos e houve necessidade de recorrer as outras soluções.

- Extração de Atributos: Como os sinais de cansaço refletem-se através dos olhos e da boca, é imprescindível que haja uma extração da região dos mesmos.
- Classificação dos Atributos: Processo que inclui validação da satisfação dos dados obtidos e classificação dos objetos obtidos em diferentes categorias, como olhos / boca aberto ou fechado.
- Tomada de Decisão: Em função da classificação do estado dos olhos e da boca é aplicado um conjunto de regras para classificar o cansaço do condutor.

1.3. Organização do Relatório

O resto do relatório está organizado como se descreve a seguir.

O Capítulo 2 apresenta o estado da técnica relacionada com o projeto desenvolvido. O capítulo começa por descrever alguns exemplos de ADAS desenvolvidos por diversos fabricantes de automóveis e que podem ser encontrados atualmente em alguns dos modelos automóveis que comercializam. A seguir, o capítulo apresenta uma revisão de alguns dos principais trabalhos publicados sobre monitorização e deteção de cansaço em condutores. O capítulo termina com um estudo dos principais algoritmos de Visão Computacional para deteção de faces, o qual serviu para selecionar os algoritmos a usar no projeto.

O Capítulo 3 apresenta uma revisão das características das principais plataformas de *smartphone* com o intuito de se escolher a plataforma mais adequada para o projeto.

O Capítulo 4 descreve o sistema de assistência ao condutor desenvolvido, na forma de uma aplicação para *smartphone*, apresentando a abordagem seguida para a deteção de cansaço em condutores e os algoritmos implementados.

Os testes realizados para testar a aplicação desenvolvida e os resultados obtidos, assim como a discussão dos mesmos, são apresentados no Capítulo 5.

Finalmente, o Capítulo 6 contém as conclusões e as principais contribuições e limitações deste trabalho e a lista de referências bibliográficas encerra o relatório.

2. Estado da Técnica

O Capítulo 2 apresenta o estado da técnica relacionada com o projeto desenvolvido. Começa por descrever alguns exemplos de ADAS desenvolvidos por diversos fabricantes de automóveis e que podem ser encontrados atualmente em alguns dos modelos automóveis que comercializam. A seguir, apresenta uma revisão de alguns dos principais trabalhos publicados sobre monitorização e deteção de cansaço em condutores. O capítulo termina com um estudo dos principais algoritmos de Visão Computacional para deteção de faces, o qual serviu para selecionar os algoritmos a usar no projeto.

2.1. Exemplo de ADAS propostos por fabricantes de automóveis

Um exemplo de ADAS foi proposto pelos fabricantes da marca Volkswagen que criaram um sistema de deteção de cansaço, *Driver Alert System* que inicialmente foi aplicado em alguns modelos, nomeadamente, *Passat Variant* e é uma inovação na sua área. Segundo informações da Volkswagen (2013), a função do detetor de cansaço é de alertar o condutor para uma perda de concentração a partir dos 65km/h. Este sistema consiste em avaliar as alternâncias do volante e identifica um possível cansaço do condutor. O sistema emprega um tipo de tecnologia baseada em sensores do carro, uma vez que analisa as características de direção do condutor, considerando o uso dos pedais e os ângulos de inclinação e correção do volante.

A figura 1 retrata um comportamento do condutor ao volante divergir dos movimentos leves concentrados de uma condução atenta.

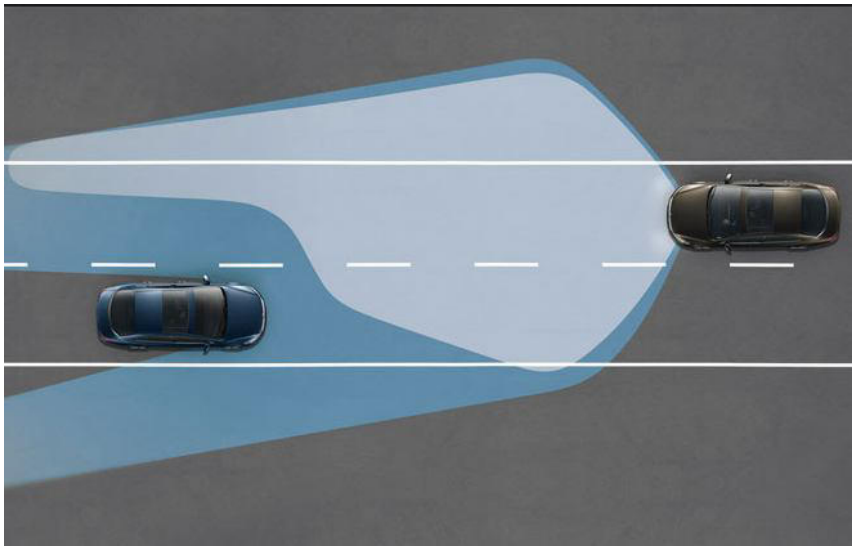


Figura 1. Parâmetros a indicar um desvio em relação ao comportamento da direção (adaptada de Volkswagen (2013))

Quando os parâmetros saem do padrão, um aviso sonoro e visual no painel do computador de bordo alerta o condutor. Após a emissão do primeiro sinal, caso o condutor não faça a pausa durante os próximos 15 minutos, o carro emite outro aviso sonoro, até que o condutor pare e descanse.

Outro exemplo de ADAS é proposto pelos fabricantes da marca Ford (2013) que desenvolveram um sistema de deteção de cansaço intitulado *Lane Keeping System* que monitora a trajetória do condutor na pista da estrada a fim de detetar sinais de cansaço ou sonolência, adaptado aos modelos *Fusion* e *Explorer* do ano de 2013. O dispositivo funciona com base em informações captadas por uma câmara instalada no para-brisa do veículo, atrás do espelho retrovisor, tal como podemos observar na figura seguinte.

Figura 2. Sistema de deteção de cansaço através de uma câmara instalada no para-brisa do veículo (adaptada de Ford (2013))

Este tipo de sistema utiliza uma tecnologia baseada em Visão Computacional, extraindo informações a partir de imagens capturadas pela câmara instalada no para-brisa do veículo. A aplicação também utiliza a mesma tecnologia usando imagens capturadas pela câmara do dispositivo. O sistema acompanha os movimentos do condutor e transmite um aviso caso ele saia da pista ou faça mudanças bruscas na direção do veículo. Nesse caso, o *Driver Alert System*, que integra a ferramenta da Ford, produz um som com o objetivo de despertar o motorista. No painel também aparece um alerta luminoso no formato de uma xícara de café, uma recomendação para que o condutor pare para descansar. Porém, caso isso não seja feito e o sensor continue a detetar instabilidade na direção, outro som é emitido. O sistema só termina os alertas quando o condutor para o carro e abre a porta, ou quando o motor é desligado.

O *Attention Assist* é outro exemplo de ADAS desenvolvido pela marca da Mercedes-Benz, em 2009, nomeadamente em modelos de carro da Classe V. É um sistema de apoio ao condutor que reconhece atempadamente o cansaço do condutor. O sistema *Attention Assist,* segundo a marca da Mercedes–Benz (2013), monitoriza o comportamento de condução do condutor, formando um perfil individual do condutor que é constantemente comparado com os dados transmitidos pelo sensor do carro de grande sensibilidade que tem como finalidade monitorizar com precisão os movimentos do volante e a sua velocidade. Esta constante monitorização é importante para o sistema poder reconhecer a transição do condutor do estado de alerta para o de sonolência e o avisar atempadamente. A base para essa avaliação em curso é fornecido por mais de 70 variáveis de medidas, tais como velocidade, aceleração e indicador de uso pedal.

A figura que se segue ilustra todas as variáveis que o veículo registra para reconhecer o cansaço do condutor.

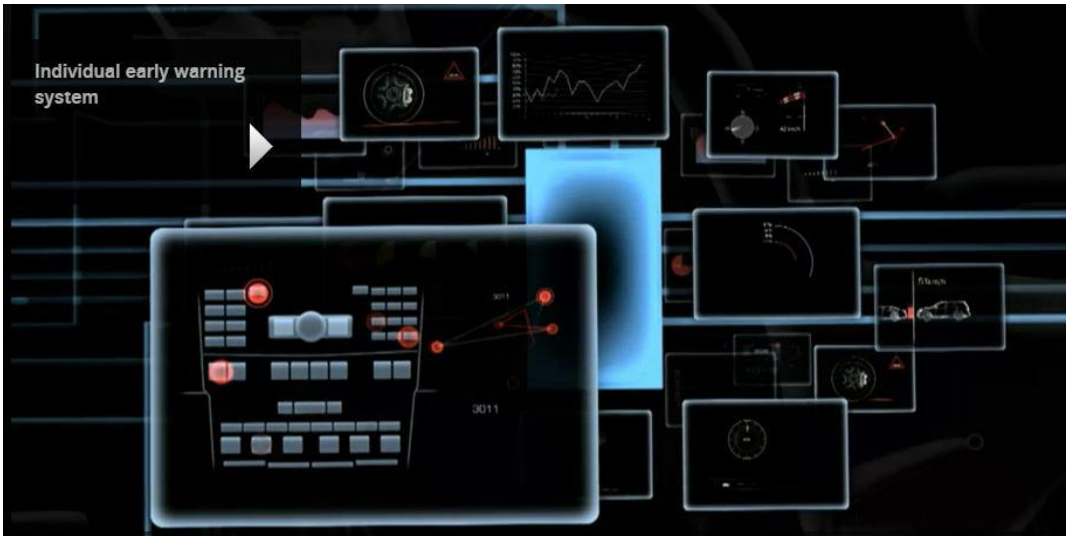


Figura 3. Constante monitorização dos dados para avaliar comportamento de condução do condutor (adaptada de Mercedes-Benz (2013))

Nos primeiros momentos de cada condução, o sistema utiliza estes dados para criar um padrão individual do comportamento do condutor e constantemente compará-lo com o atual comportamento de condução e a situação de condução através da unidade eletrónica de comando a bordo do veículo. Isto permite ao sistema reconhecer os sinais típicos de excesso de cansaço e alertar o condutor. O método utilizado no projeto, no que se refere ao estado de classificação dos olhos e da boca, é semelhante, uma vez que usa um método que utiliza modelos de imagens para estar constantemente a comparar com a imagem atual. Durante as investigações, os especialistas da marca Mercedes-Benz, utilizaram uma equipa de engenheiros, matemáticos e psicólogos e testaram uma variedade de métodos na deteção de cansaço no condutor. Uma destas técnicas é o método de observação *eye-blink*, e segundo a marca, refere que consiste numa câmara de infravermelhos direcionada para a cabeça do condutor que monitoriza em permanência a frequência do piscar de olhos, permitindo detetar o micro sono num momento em que os olhos se encontram fechados durante mais do que um certo período de tempo. Para obter indicadores objetivos de cansaço, são também usados outras leituras psicológicas como o eletroencefalograma e a análise de dados dinâmicos de condução. Um dos sistemas, aciona um alarme se o condutor não mexer o volante durante um período prolongado de tempo.

Um outro tipo de exemplo de ADAS é proposto pelos fabricantes da Opel (2013) que criaram um sistema de assistência ao condutor presente em modelos da *Opel Insígnia Country Tourer* do ano de 2012. Consiste numa câmara dianteira designada por *Opel Eye* que integra uma câmara especial de alta resolução que o ajuda a conduzir com mais segurança. Através da figura 4 é possível ver que está posicionada na base do espelho retrovisor interior e disponibiliza três componentes inovadores: sistema de reconhecimento dos sinais de trânsito, aviso de desvio de trajetória, aviso de perigo de colisão frontal.

Figura 4. Sistema de deteção de cansaço através de uma câmara posicionada na base do espelho retrovisor (adaptada de Opel (2013))

É um sistema que utiliza uma tecnologia baseada em Visão Computacional devido à câmara que possui, o mesmo acontece na aplicação que usa a mesma tecnologia fazendo uso da câmara do dispositivo, e faz com que reconheça a linha delimitadora da faixa de rodagem através da função de aviso de desvio de trajetória, que é ativada instantaneamente quando o veículo transpõe inadvertidamente uma linha delimitadora da faixa de rodagem sem indicação do condutor. O aviso de desvio de trajetória emite um sinal sonoro e uma luz de emergência acende-se no painel de instrumentos, alertando imediatamente o condutor.

2.2. Sistemas de Deteção de Cansaço em Condutores

Enquanto esperamos pela adoção massiva do tipo de sistemas descritos na secção anterior, por parte dos fabricantes de automóveis, tem sido realizado um esforço de investigação e desenvolvimento muito grande para desenvolver ADAS baseados em diferentes tipos de plataformas, tais como: Computadores, Laptops, Consolas de Jogos, Plataformas Móveis e Sistemas Embebidos. Nesta secção descrevem-se alguns desses sistemas.

Os autores Ren, et al. (2012) apresentam um sistema de deteção de linhas da faixa de rodagem formado por uma aplicação para *smartphones iPhone*. A deteção da linha da faixa de rodagem é um método muito usado em aplicações ADAS, que permite determinar a geometria da estrada, bem como a posição lateral do veículo, ou seja, é usado para localizar os limites da pista em imagens de estradas dadas, e tem sido amplamente estudado em vários cenários. Um exemplo real de deteção de linha, utilizando um método que a seguir será comentado é mostrado na figura seguinte.

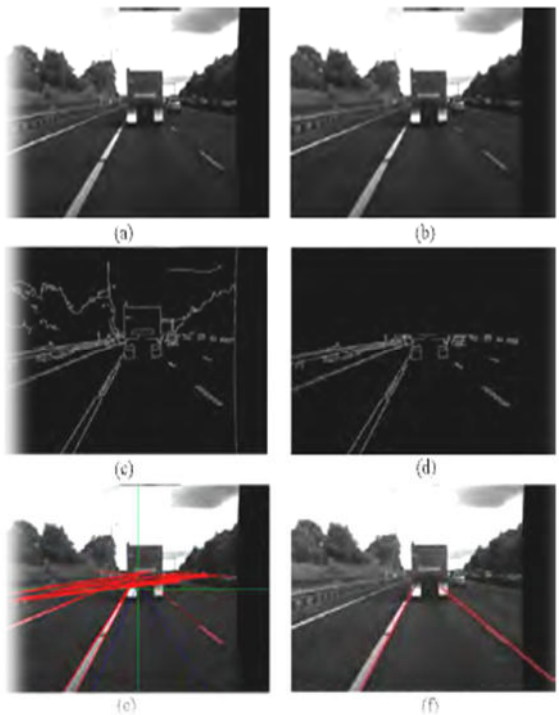


Figura 5. Um exemplo de deteção de linha: (a) Uma imagem de entrada; (b) Alisamento; (c) Deteção de Bordas; (d) ROI; (e) Hough transformar; (f) Resultado da deteção da linha (adaptada de Ren et al. (2012))

A figura 6 mostra a localização do *smartphone iPhone* dentro de um veículo.

Figura 6. Localização do smartphone iPhone dentro do veículo (adaptada de Ren et al. (2012))

O método de deteção baseia-se na aquisição de uma imagem a partir da câmara de um *iPhone*, à qual é aplicado um algoritmo de suavização de forma a eliminar o ruído. Em seguida, é realizada a deteção de arestas e é aplicado o algoritmo da Transformada de *Hough* para poder detetar as linhas da estrada. A Transformada de *Hough,* segundo Jamundá (2013), é um método padrão para a deteção de formas que são facilmente parametrizadas (linhas, círculos, eclipses, etc.) em imagens computacionais. Em geral, a transformada é aplicada após a imagem sofrer um pré-processamento, comumente a deteção de arestas. Como existem muitas linhas, além dos limites da estrada, foi aplicado um ROI (*Region of Interest*, em português região de interesse), a fim de selecionar a zona da imagem correspondente à estrada onde aparecem as linhas e ignorar a restante da imagem. Esta técnica ROI também serviu para a aplicação no módulo da extração da região dos olhos e da boca, ou seja, a deteção dos olhos e da boca não foram procurados em toda a imagem, optou-se por extrair uma região de interesse dos olhos e da boca através do retângulo da face previamente detetado.

Os autores Devi e Bajaj (2013) desenvolveram um sistema que utiliza uma câmara de vídeo que aponta diretamente para o condutor de modo a detetar sinais de cansaço. Se

for detetado cansaço será emitido um sinal de alarme para alertar o condutor. Consideraram então um sistema baseado no movimento visual da pálpebra para detetar o estado de cansaço do condutor. Através da monitorização dos olhos, os autores referem que os sintomas de cansaço do condutor podem ser detetados suficientemente cedo para evitar um acidente de carro. Referem ainda no artigo, que a taxa normal de frequência do piscar de olhos aumenta de acordo com o estado de cansaço. Nesse sentido um piscar de olhos com duração de 3 a 4 segundos é considerado um bom indicador desse estado. O método implementado pelos autores é descrito no fluxograma da figura seguinte.

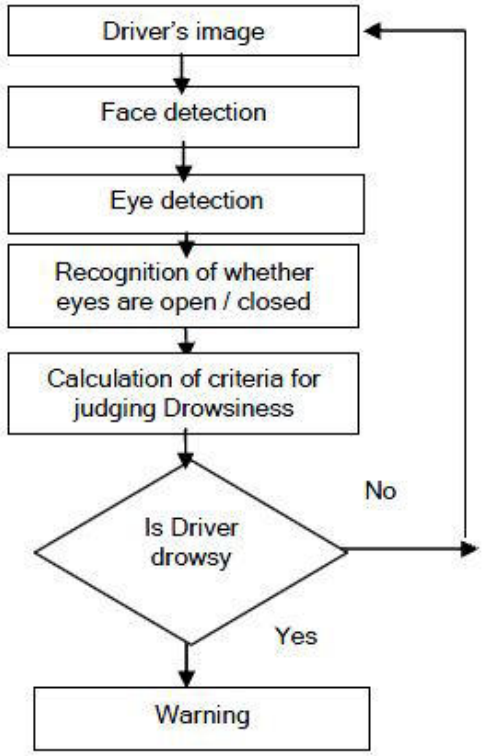


Figura 7. Fluxograma do Driver Fatigue Detection Based on Eye Tracking (adaptada de Devi e Bajaj (2013))

O sistema começa por adquirir a imagem do condutor através de uma câmara de vídeo e posteriormente deteta a face com base no método das características através da cor da pele. Este método de deteção de pele humana em imagens tem por finalidade detetar qualquer padrão de cor de pele existente, tais como branco, negro, indígena, asiático e qualquer outra cor de pele existente.

A deteção do olho é feita apenas em parte da imagem, uma vez que os olhos encontram-se na metade superior da face e a metade inferior da face é removida para

diminuir a área de pesquisa em relação aos olhos. Em seguida é utilizada a deteção de arestas e é encontrado o centro da face que vai ser utilizado como referência para comparar os olhos. Para localizar a região dos olhos é feito através das significativas mudanças de intensidade na região da face, para isso é necessário converter a imagem em tons de cinzento. De seguida, são calculadas as médias da projeção horizontal (média do valor da intensidade para cada coordenada x) da área da face a partir do topo para baixo, onde se pode visualizar na figura 8.

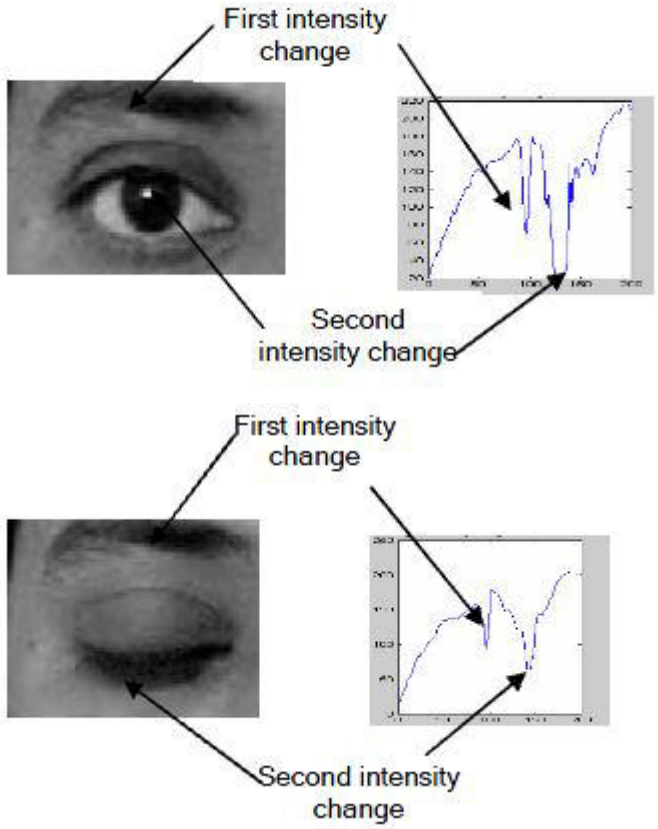


Figura 8. Média de intensidade na região da face quando os olhos estão abertos e fechados (adaptada de Devi e Bajaj (2013))

Na figura anterior, observa-se que existem duas significativas mudanças de intensidade. A primeira variação de intensidade é a sobrancelha e a próxima mudança é o limite superior do olho, assim com o conhecimento destes dois valores é encontrada a posição dos olhos na face. Usando os valores da média da projeção horizontal, consegue-se verificar se o estado dos olhos detetados estão abertos ou fechados, ou seja, quando os olhos estão fechados, a distância da coordenada x das alterações de intensidade é maior

comparada quando os olhos estão abertos. Se os olhos se encontrarem fechados por cinco *frames* consecutivos, o sistema chega à conclusão de que o condutor encontra-se em estado de cansaço e emite um sinal de alerta.

O autor Queiroz (2011) apresenta um sistema de deteção de cansaço baseado em vídeo utilizando a plataforma computacional FITPC2, que é um microcomputador pequeno e compacto (Dimensões 104x96 mm, e altura 22.9 mm), inserido no painel do carro, como é ilustrado na figura 9.

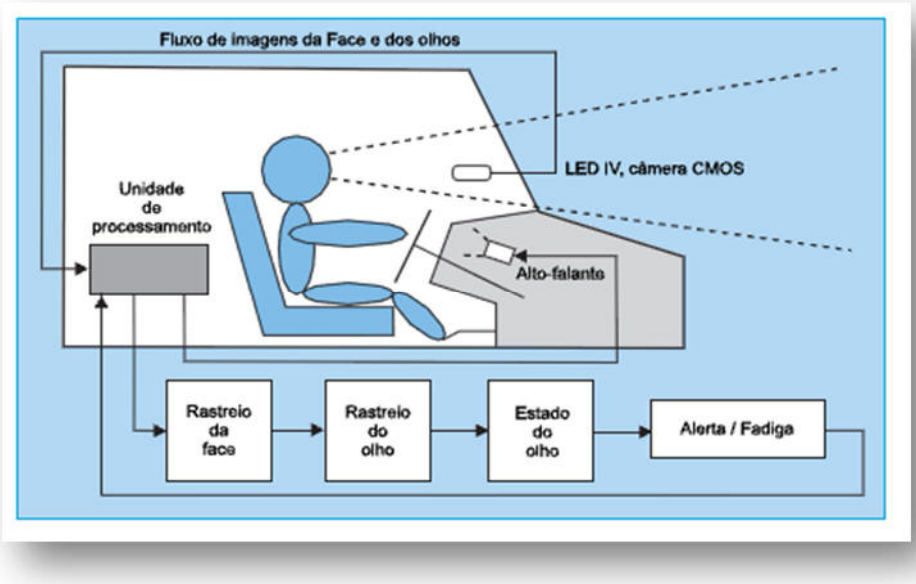


Figura 9. Esquema ilustrando a integração dos sistemas de hardware e software e a localização destes dentro de um veículo (adaptada de Queiroz (2011))

Os autores consideram que a unidade de processamento analisa o fluxo de imagens fornecidas pela câmara para extrair informações sobre a pálpebra do condutor, padrões de piscar e características da face. O ponto de conexão é o lugar onde o condutor se posiciona à frente do volante a uma distância pré-determinada (máximo 1,2 m) da câmara. O indicador informa o condutor acerca do estado do sistema através de um aviso sonoro. A biblioteca de Visão Computacional utilizada neste sistema foi *OpenCV – Open Source Computational Vision* devido ao facto de ser baseado unicamente na análise visual das expressões faciais do condutor, focando principalmente os olhos. Inicialmente esta biblioteca foi utilizada na aplicação para a deteção da face.

O fluxograma que descreve o funcionamento do sistema é apresentado na figura 10.

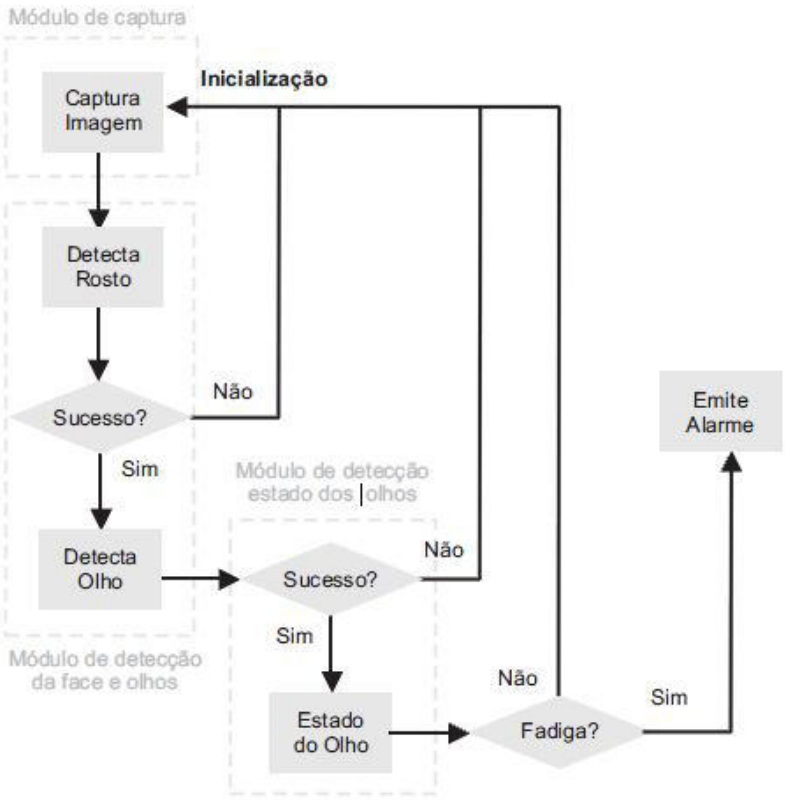


Figura 10. Sistema proposto para um sistema baseado em vídeo para a deteção de cansaço (adaptada de Queiroz (2011))

O sistema começa por adquirir a imagem do condutor através de uma câmara de vídeo e posteriormente deteta a face utilizando a biblioteca *OpenCV*, como foi referido anteriormente. Diferentemente da face, na primeira fase do procedimento de deteção, os olhos não foram procurados em toda a imagem, optou-se por extrair uma ROI dos olhos, direito e esquerdo, de dentro do retângulo da face previamente detetado. Portanto, este autor considera que se o retângulo do rosto é [0,0, largura, altura], define-se o retângulo de ROI dos olhos como [0,altura/4,largura/2, altura/4]. Este processo para o módulo da extração de atributos foi muito semelhante ao que se aplicou na aplicação. Para encontrar o melhor método para identificar o estado do olho (aberto/fechado), o autor implementou e testou três algoritmos baseados em três métodos distintos: limiarização automática baseada em histograma, template matching e limiarização elevada. Os três métodos são descritos a seguir.

Método da Limiarização Automática baseada em Histograma

Para determinar um limiar automático de forma a delinear o contorno dos olhos, da iris e das pálpebras, é calculado o valor do histograma da imagem do olho, onde um valor *H* é o índice com valor máximo no histograma (o eixo x é o nível de cinzento de 0 a 255, e o eixo y é o numero de pixéis com cada nível de cinzento, o valor máximo será a coordenada x com maior valor de y), sugerido por Ma et al. (2008). O valor de $H \times 2/3$ foi determinado como bom valor de limiar para delinear o segmento de contorno dos olhos, figura 11.

Figura 11. (A) Olho, (B) Histograma do olho, (C) Imagem Binária do Globo Ocular e Pálpebras (adaptada de Queiroz (2011))

A distância da pálpebra é definida como sendo a distância entre a pálpebra superior e a inferior. Essa distância é relativamente grande quando o condutor está em estado de alerta, figura 12 (A), e tornando-se pequena à medida que ele vai ficando cansado figura 12 (B e C).

Figura 12. Exemplo dos três estados que o olho pode assumir: (A) Aberto, (B) Meio aberto, (C) Fechado (adaptada de Queiroz (2011))

Através da distância da pálpebra, pode-se decidir se o condutor está em alerta ou sonolento. A média dos valores da distância da pálpebra é calculada para 100 *frames* contínuos. Quando a média cai abaixo de um limiar de 60% do valor normal o condutor

é considerado que está a entrar em estado de cansaço, permanecendo abaixo do limiar de 60%, o condutor é considerado sonolento, figura 13.

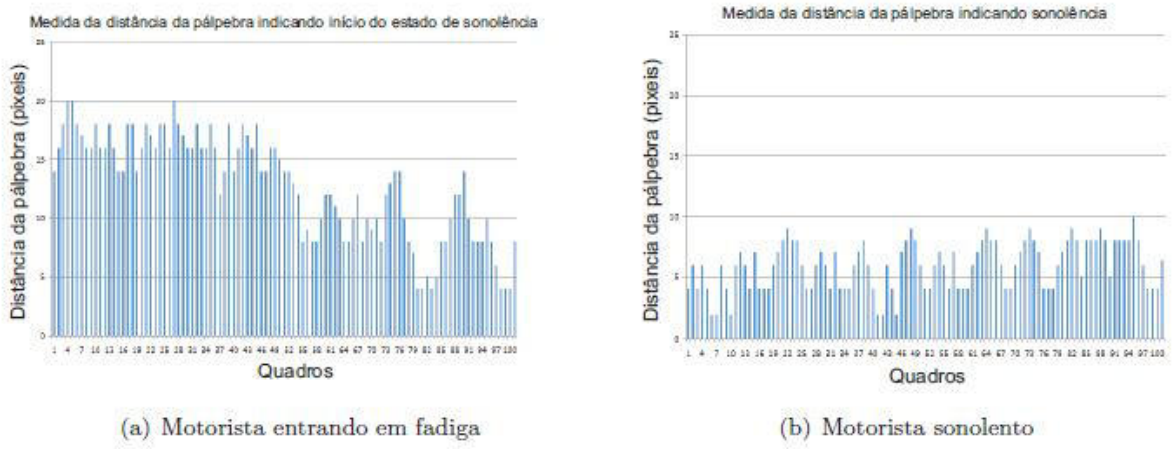


Figura 13. Gráficos que descrevem o motorista a entrar em estado de fadiga (a) e Sonolento (b) (adaptada de Queiroz (2011))

Este método foi implementado na aplicação para o módulo da classificação do estado dos olhos mas não obteve resultados diferenciados de se poder classificar olho aberto e olho fechado.

Método Template Matching

Para este caso, a classificação do estado dos olhos e a análise de duração desse evento são baseados na observação e comparação dos coeficientes de correlação obtidos durante rastreamento da ROI do olho. Para tal, são usados *templates* do olho, ou seja, imagens guardadas como modelos de olhos, e para que esse modelo seja guardado, o autor considerou que o condutor deverá olhar para a câmara durante 10 segundos. Este intervalo de tempo é suficiente para que o algoritmo guarde uma imagem do olho que será usada posteriormente como um modelo de olho aberto. Na aplicação também foi aplicado este método, mas no nosso caso o utilizador terá de fazer a captura do *template* manualmente. Na figura que se segue pode-se observar vários modelos dos olhos.

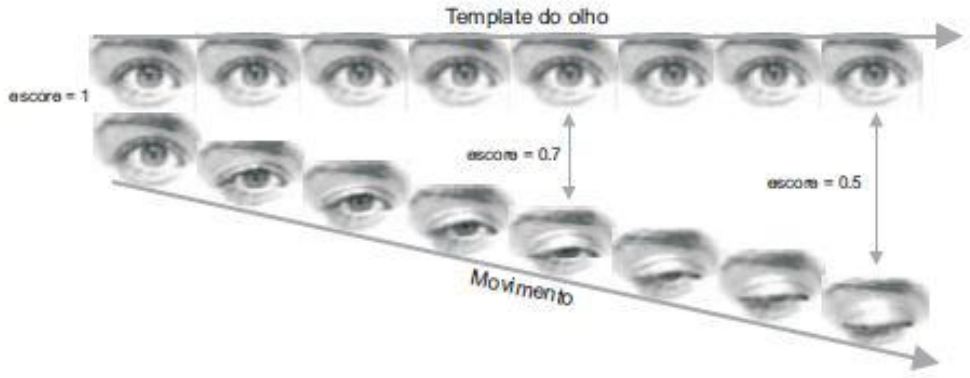


Figura 14. Modelos dos olhos (Templates) (adaptada de Queiroz (2011))

Através da figura, o autor considerou que o aumento e diminuição na similaridade corresponde diretamente aos coeficientes de correlação entre o modelo e a imagem da região do olho que está sendo processada no *frame* atual durante a captura de vídeo. Considerou também que para detetar o cansaço, este algoritmo avalia através dos *frames* onde os valores do coeficiente de correlação estão entre 0.8 e 0.55, ou seja, o olho é considerado fechado. Nos *frames* com valores maiores que 0.8, considera-se olho aberto. Os *frames* com valores inferiores a 0.55 são perdidos e sinalizam para reinicialização do rastreamento da face e dos olhos. Este processo também foi implementado na aplicação.

Método da Limiarização Elevada

O terceiro método é resultante da propriedade de reflexão dos raios Infravermelhos pela córnea. Essa propriedade faz com que o olho, quando iluminado por uma luz infravermelha, apareça na imagem de uma câmara com um círculo de alta intensidade na região da córnea. Uma característica tão marcante na imagem pode ser usada para diminuir a complexidade computacional em algoritmos para determinar o estado dos olhos. A proposta que o autor apresenta acerca da decisão do algoritmo baseia-se em aplicar um limiar alto à imagem. Assim, quando a imagem do olho é transformada em imagem binaria e é aplicado a ela um valor de limiar muito alto, torna-se uma imagem preta, com apenas a região de córnea em branco. Para olho fechado, a região estaria inteiramente preta, posteriormente calcula-se a intensidade média da imagem. Se a média do valor da intensidade for zero, o olho está fechado. Caso contrário, quando a intensidade média é diferente de zero, a córnea é detetada e o olho está aberto. Para determinar o

estado de cansaço é o mesmo método utilizado no algoritmo baseado em Template Matching, ou seja, monitorizando o estado dos olhos, quando permanecerem fechados por 8 ou mais *frames* consecutivos é emitido o alerta de cansaço.

2.3. Métodos de Visão Computacional para Deteção de Faces

Este capítulo apresenta um estudo dos principais algoritmos de Visão Computacional para deteção de faces proposto por Zhao et al. (2003), o qual serviu para selecionar os algoritmos a usar na aplicação.

2.3.1. Métodos Baseados em Conhecimento (Métodos Top – Down / Feature-Based)

Os métodos baseados em conhecimento representam as técnicas de deteção de faces que utilizam alguma base de regras estabelecida a partir do conhecimento prévio sobre o problema, ou seja, métodos que possuem regras que definem o que é uma face, de acordo com o conhecimento do investigador. Por exemplo, uma face humana possui características válidas tais como dois olhos, um nariz e uma boca, que por sua vez se encontram distribuídos de maneira específica sobre a face, o que permite estabelecer regras que identificam uma face humana, como se pode ver na figura 15.

Figura 15. Uma face típica usada em métodos baseados em conhecimento (adaptada de Zhao et al. (2003))

As desvantagens que este método possui têm a ver com a dificuldade em traduzir o conhecimento humano em regras bem definidas. Se as regras forem muito específicas elas podem ser ineficazes na deteção de faces se não satisfazerem todas as regras. Por

outro lado, se as regras são muito gerais, corre-se o risco de apresentar uma alta taxa de falsos positivos, ou seja, elementos incorretamente identificados como face.

O autor Heisele et al. (2001) descreve um exemplo utilizando o método baseado no conhecimento.

2.3.2. Métodos Baseados em Características Invariantes (Métodos Bottom-Up)

Ao contrário dos métodos baseados em conhecimento, os métodos em questão utilizam técnicas que tem por objetivo encontrar características invariantes da face. Estes métodos são baseados na capacidade que os seres humanos possuem de identificar facilmente faces e objetos em diferentes posições e condições de iluminação. Existem vários métodos propostos para detetar componentes faciais e então deduzir a presença de uma face. Componentes faciais como sobrancelhas, olhos, nariz, boca, cabelo e o contorno da face são extraídos geralmente usando detetores de arestas e também utilizando a projeção horizontal e vertical, baseando-se nos componentes faciais extraídos, um modelo estatístico é construído para verificar a existência de face. A principal desvantagem desta abordagem é que as características da imagem podem ser severamente danificadas devido às condições de iluminação, ruído e obstrução, comprometendo assim, a eficiência da abordagem. A figura 16 retrata uma imagem onde as sombras podem causar numerosas arestas fortes o que no seu conjunto torna uma imagem danificada.

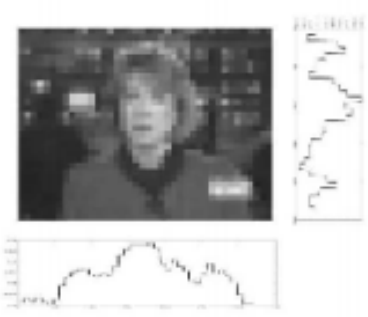


Figura 16. Exemplo de uma face numa determinada resolução utilizadas para detetar faces através de regras que utilizam o conhecimento sobre a distribuição de luminosidade da imagem (adaptada em Zhao et al. (2003))

As principais características que podem ser utilizadas para separar a face de outros objetos que estão presentes na imagem são a textura, cor da pele ou ambas.

Textura

Os rostos humanos têm uma textura diferente que pode ser usada para separá-los de diferentes objetos. Três tipos de características são considerados: pele, cabelo e outros.

Cor da pele

A cor da pele humana é uma característica bastante utilizada para separar a face de outros objetos presentes numa imagem com fundo complexo. As informações sobre as cores da pele constituem uma importante ferramenta para identificar áreas da face e os componentes faciais específicos. Existe uma grande variedade de cores de pele (branca, negra, amarela, etc.), e existe um grande número de pesquisas que utilizam a cor da pele. Contudo, modelos da cor de pele, não são eficazes, onde é significativa a variação do espectro da fonte de luz, isto porque a aparência da cor é muitas vezes instável devido a mudanças tanto no fundo como na iluminação de primeiro plano. A cor da pele só, geralmente não é suficiente para a deteção de faces.

Múltiplas características

Recentemente, vários métodos que combinam várias características faciais têm sido propostos para localizar ou detetar faces. A maioria deles utiliza características globais, tais como cor da pele, tamanho e forma. Depois são verificados esses candidatos usando as características locais, tais como sobrancelhas, nariz e cabelo. Uma abordagem típica começa com a deteção de regiões de pele. Em seguida os *pixels* da pele são agrupados em conjunto, utilizando a análise de componentes conectados ou algoritmos de agrupamento. Se a forma de uma região ligada tiver uma forma elíptica ou oval, torna-se uma face candidata. Finalmente, as características locais são utilizadas para a verificação.

Os autores Devi e Bajaj (2010) descrevem um exemplo utilizando o método baseado em características invariantes.

2.3.3. Métodos Baseados em Templates (Template-Matching)

Uma técnica clássica de detetar objetos é procurar pelo mesmo objeto dentro da imagem e testar se ele corresponde a um modelo prévio da sua forma. Uma das formas mais comuns de modelar a forma de um objeto é descrevê-lo através de seus componentes geométricos básicos, como círculos, quadrados ou triângulos, esta técnica é denominada *Templates*. Dada uma imagem de entrada, os valores de correlação com os modelos de padrão são computados para o contorno da face, olhos, nariz e boca de forma independente. A deteção do objeto, portanto, consistirá em achar a melhor correspondência, definida através de uma função de energia, entre o objeto presente na imagem e o seu modelo (*Templates*). No caso de deteção de faces o *template* mais utilizado é aquele que trata a face como uma elipse. Esta abordagem tem a vantagem de ser simples de implementar. No entanto, tem revelado ser insuficiente para a deteção da face uma vez que não pode lidar efetivamente com variação de escala, postura e forma. Multi-resolução, multi-escala e modelos deformáveis foram posteriormente propostos para alcançar a invariância de escala e forma, figura 17.

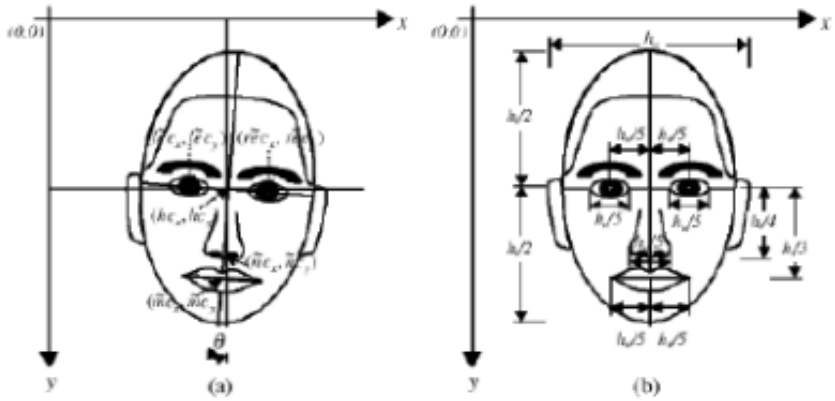


Figura 17. (a) Modelo utilizado para detetar a cabeça (b) Modelo para detetar os componentes faciais (adaptada de Zhao et al. (2003))

Os autores Horng e Chen (2008) descrevem um exemplo utilizando o método baseado em *templates*.

2.3.4. Métodos Baseados na Aparência (Appearence-based)

Nos métodos baseados na aparência, os modelos são definidos utilizando técnicas de análise estatística e máquinas de aprendizagem para encontrar as características relevantes de "imagens face" e "imagens não face", não sendo utilizados nenhum conhecimento ou características sobre o objeto previamente informado, como os vistos nos métodos descritos na sessão anterior. Eles irão extrair de uma base de dados contendo inúmeras imagens do objeto de interesse informações sobre o mesmo com base em regras de reconhecimento de padrões pré-estabelecidas. Exemplos desta abordagem são *Eigenfaces* e Redes Neurais.

2.3.4.1. Eigenfaces

O método de *Eigenfaces* foi o primeiro algoritmo de deteção facial utilizado com sucesso para reconhecer faces na Visão Computacional. A funcionalidade do método é gerar, como resposta, um conjunto de vetores que serão usados para contornar o problema da deteção dos padrões das imagens. O algoritmo utiliza vetores de distribuição probabilística para gerar a informação matemática do rosto de um ser humano. Neste método é necessário realizar o treino de uma base de dados de imagens, através da técnica PCA (*Principal Component Analysis*). Esta técnica é responsável por transformar as informações visuais em vetores, denominados *eigenvectors* com o intuito de representar os pontos da imagem que mais se destacam. A representação de cada *eigenvectors* é analisada para que seja possível a criação de um padrão que determine se a imagem representada em questão é o objeto, forma ou face humana que se deseja encontrar.

2.3.4.2. Redes Neuronais

Outra técnica que utiliza os conceitos do método baseado na aparência é a técnica de deteção de faces por meio de Redes Neuronais. Nesta técnica, a deteção de face é tratada como um problema de reconhecimento de padrões, assim como o reconhecimento ótico de caracteres, reconhecimento de objetos e condução de robôs autónomos, cuja aplicação da técnica obtém bastante sucesso. A vantagem de se utilizar Redes Neuronais para deteção de faces é que o algoritmo é treinado para aprender os padrões que

representam as faces e assim distinguir face de não-face, este é o ponto principal para solucionar o problema de classificação. Esta aprendizagem tem por finalidade extrair as características da face e codificá-las a fim de se obter um padrão de face. No entanto, uma desvantagem é que a arquitetura de rede tem de ser amplamente configurada (número de camadas, número de nós, taxas de aprendizagem, etc.) para obter um desempenho excecional.

Os autores Devi e Bajaj (2008) e Ma et al. (2008) descrevem exemplos utilizando o método baseado na aparência.

2.3.5. Comparação dos Métodos

Uma das desvantagens que os métodos baseados em conhecimento possuem tem a ver com a construção do conjunto de regras, ou seja, se as regras são muito gerais corre-se o risco de que o sistema que as utiliza apresentar uma taxa de falsos positivos e o inverso também é verdadeiro. Um conjunto de regras muito específico pode ser ineficaz ao tentar detetar faces se estas não satisfizerem todas as regras, caindo muito a precisão da deteção. A aplicação corria um grande risco ao usar este tipo de método, uma vez, que teríamos de criar muitas regras pelo facto de existir muitos formatos diferentes de faces. Além disso, os fundos do cenário onde o projeto pode ser aplicado podem ser complexos o que implica que a deteção da face e as caraterísticas faciais ficam comprometidas devido ao método em questão.

Apesar dos métodos baseados em características invariantes serem mais eficientes e simples de implementar que o anterior, implica alguma capacidade de processamento e uma elevada resolução de câmara, uma vez que faz uso dos objetos presentes numa imagem, tais como a textura e cor da pele. Como os *smartphones* ainda não possuem uma boa capacidade de processamento são limitações que podemos encontrar para a aplicação. Além disso como a aplicação é para ser utilizada em diversas fases do dia, irá proporcionar diferentes tipos de iluminação que é uma outra desvantagem desta abordagem, as características da imagem podem ser severamente danificadas devido às condições de iluminação.

Os métodos baseados na aparência como não utilizam nenhum conhecimento à partida sobre o objeto ou caraterísticas a ser detetada, utilizam técnicas de análise estatística e máquinas de aprendizagem para encontrar as características relevantes de "imagens face" e "imagens não face", o que implica que a sua implementação seja mais complexa o que envolve um custo computacional mais elevado. Acrescenta ainda, o facto de a computação ser intensa tanto em termos temporais quanto no uso de memória.

Um método que parece ser adequado aos objetivos propostos para a aplicação é o método *Template Matching* baseados em *Templates* porque além de ter a vantagem de ser simples de implementar, a técnica de Templates é extremamente flexível e o objeto pode ser procurado numa imagem utilizando um modelo padrão que pode ser definido manualmente.

3. As principais Plataformas de Smartphone

O Capítulo 3 apresenta uma revisão das características das principais plataformas de *smartphone* com o intuito de se escolher a plataforma mais adequada para o projeto.

3.1. Plataforma Android

O *Android* é um sistema operativo *Open Source* (código aberto) baseado em Linux. Inicialmente foi desenvolvido pelo Google e posteriormente pela *Open Handset Alliance*, uma associação comercial constituída por mais de trinta empresas, incluindo Google, HTC, Dell, Intel, Motorola e Samsung, entre outras. Estas empresas são responsáveis por criar padrões abertos para os dispositivos móveis de todas as marcas participantes e um conjunto de *software* focado em dispositivos móveis.

A maior vantagem do *Android* sobre outras plataformas é o desenvolvimento em Java. Hoje a linguagem Java é uma das linguagens de programação mais utilizada no mundo. As principais características deste sistema são: experiência de navegação completa, rápida e intuitiva; acesso a um conjunto de serviços Google - *Gmail, Google Talk, Google Maps, YouTube*, etc.; acesso a um mundo de aplicações gratuitas ou disponíveis através de pagamento e personalização dos conteúdos. *Android* é uma plataforma recente, a primeira versão foi criada em 2008. Desde então foram lançadas outras versões. A Google adotou uma tradição, a partir do lançamento da terceira versão do sistema operativo, que é a de batizar o *software* com o nome de um doce. A ideia surgiu em 2009, com o lançamento do *Android* 1.5. O *software* passou-se a chamar de *cupcake*.

A figura 18 fornece dados, que são divulgados pelo *Android Developers* (2015), sobre o número relativo de dispositivos que executam uma determinada versão da plataforma *Android*.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	7.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	6.7%
4.1.x	Jelly Bean	16	19.2%
4.2.x		17	20.3%
4.3		18	6.5%
4.4	KitKat	19	39.1%

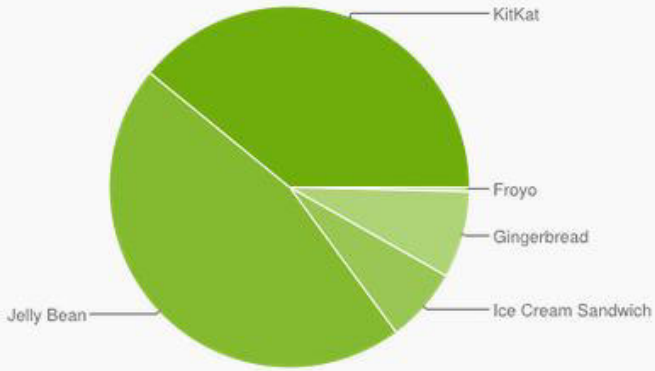


Figura 18. Versões e Distribuições da Plataforma Android. Dados de 5 de janeiro de 2015 (adaptada de Android Developers (2015))

No quadro da figura 18 as versões mais antigas não estão incluídas, no entanto, em agosto de 2013, as versões inferiores ao *Android* 2.2 possuíam uma distribuição inferior a 1%. A coluna "*Version*" diz respeito ao número comercial das versões; Na coluna "*Codename*" é o nome dado a cada versão; API (*Application Programming Interface*) refere o nível utilizado; *Distribution* exibe o percentual das distribuições atuais das versões da plataforma. É importante conhecer as informações acerca da versão e da API para configurar o ambiente de desenvolvimento. Quando surgem novas versões da plataforma é mantida a compatibilidades em relação às versões anteriores e são fornecidas normalmente pelo fabricante do aparelho, é gratuito e o processo de atualização do sistema operativo é fácil.

É de referir que até ao momento da realização do trabalho tinha saído uma nova versão, *Android* 5.0 *Lollipop* (API *level* 21), mas esta versão não estava referenciada para os dados estatísticos.

A arquitetura do sistema operativo *Android* é dividida em camadas, onde cada parte é responsável por gerir os seus respetivos processos. Tal como podemos observar na figura 19, o sistema operativo *Android* é construído por 4 níveis, *Linux Kernel*; *Libraries* e *Android Runtime*; *Application Framework*; e *Applications*. Cada nível agrupa vários programas que suportam funções específicas do sistema operativo.

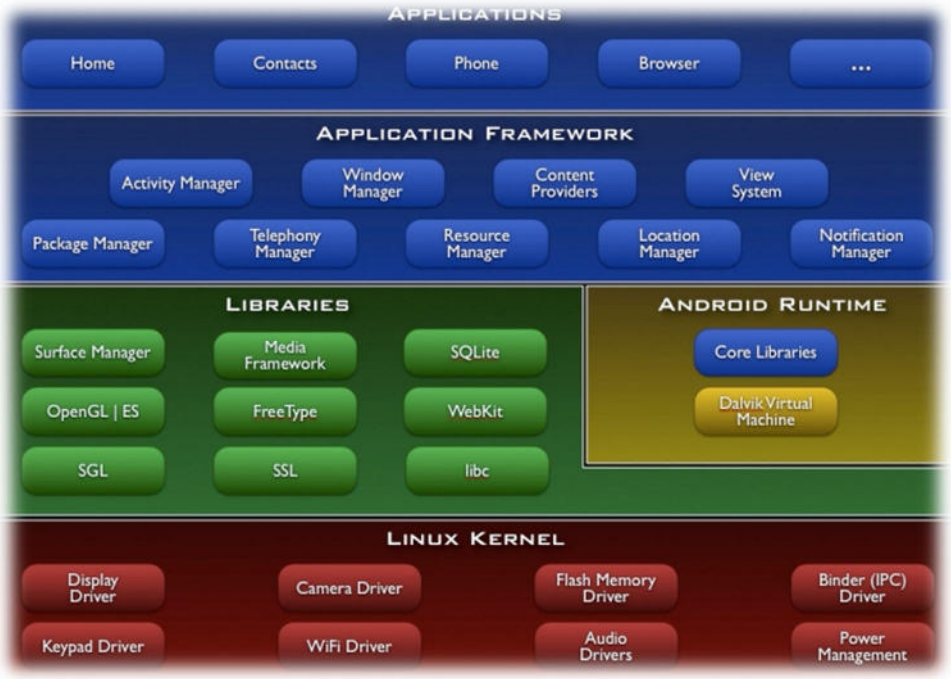


Figura 19. Arquitetura da Plataforma Android (adaptada de Android Developers (2013))

No primeiro nível, temos a base, ou seja, Linux Kernel, o núcleo do sistema operativo *Android* é derivado do kernel 2.6 do Linux, herdando diversas características dessa plataforma. Parte importante que se utiliza do Linux na conceção do Google *Android* é o controlo de processos, gestão de memória, protocolos de rede, configurações de segurança, o *software* de gestão de energia e vários drivers de hardware, ou seja, programas que controlam dispositivos de hardware.

O próximo nível de *software* inclui as *Libraries*, bibliotecas do *Android*. São um conjunto de classes e métodos que dizem ao dispositivo como lidar com diferentes tipos de dados, incluindo um conjunto de biblioteca C / C + + usadas por diversos componentes do sistema e são expostas a desenvolvedores através da estrutura de aplicativo *Android*. No mesmo nível da camada de bibliotecas, existe a camada *Runtime Android* que inclui um conjunto de bibliotecas do núcleo Java, bem como DVM (*Dalvik Virtual Machine)*, a máquina virtual. A utilização desta máquina virtual vai permitir que nenhuma aplicação seja dependente de outra e se uma aplicação parar, ela não afeta as outras aplicações que estejam a correr no dispositivo. Este processo simplifica a gestão de memória, uma vez que a máquina virtual está desenvolvida de forma otimizada para requerer pouca memória

e permitir que múltiplas instâncias executem ao mesmo tempo. No nível três aparece a camada *Application Framework*, que inclui programas que fazem a gestão das funções básicas do dispositivo. Os desenvolvedores de aplicações têm acesso total ao *framework* de aplicações do *Android*. Isso possibilita que eles tirem vantagem das capacidades de processamento do *Android* e suportem recursos quando estão construindo uma aplicação *Android*. Por último, temos a camada de *Applications*, aqui podemos encontrar as funções básicas do dispositivo, tais como fazer chamadas, aceder ao navegador web, lista de contatos, entre outros. De uma forma sucinta é possível referir que o *Android* possui as seguintes características, Tabela 1:

Tabela 1. Características do *Android*

Ferramentas do Android	
SDK é suportado pelos SO's:	Windows XP; Vista; 7; Mac OS X 10.5.8 ou posterior; Linux
JDK:	5 ou 6 (apenas a JRE não é o suficiente)
IDE:	Recomenda-se o eclipse
Plugin ADT (Android Development Toll):	DDMS (Dalvik Debug Monitor Service); QEMU (emulador)

Na Tabela 2, podemos ver as ferramentas de apoio ao *Android*.

Tabela 2. Ferramentas do *Android*

Características do Android
Código fonte é Open Source
Foi baseado no Kernel do Linux
Possui um Kit de desenvolvimento Java chamado Android SDK
Existe um SGBD nativo, SQLite
Tem suporte a gráficos 3D baseado na especificação 1.0 da OpenGLES

3.2. Plataforma Windows Phone

A Microsoft é responsável pela criação do sistema operativo *Windows Mobile*, especialmente desenvolvido para dispositivos móveis. No ano 2000 surge a primeira versão do *Windows Mobile* (Queirós 2008), com a chegada dos *Pocket PC* que criou assim as primeiras oportunidades legítimas para que os programadores pudessem criar as suas

primeiras aplicações para dispositivos móveis. Com o aparecimento do sistema operativo do *Windows Mobile*, a *Microsoft* disponibilizou um conjunto de ferramentas para desenvolver aplicações nativas para os *Pocket PC*. No entanto, é de referir que inicialmente existiam alguns problemas, os telemóveis apresentavam um ecrã de pequena dimensão o que não era compatível com a maior parte das aplicações, para além de serem bastante caros, a subscrição de um serviço de dados, não se justificava derivado a não ter a rapidez necessária para o utilizador final e era também necessário a utilização de uma caneta.

Como a Microsoft não queria ficar atrás do sucesso do *IPhone* e do aparecimento de um novo sistema operativo *Android*, decidiu procurar um novo sistema para garantir as necessidades dos utilizadores. E, de facto, a nova solução da Microsoft, seria uma versão completamente diferente da anterior, recorrendo a tecnologia de *multi-touch* e uma interface. Para além disso a *Microsoft* em vez de colocar no mercado uma gama de dispositivos móveis, com o seu sistema operativo, como já tinha feito anteriormente, padronizou o seu *hardware*, e colocou à disposição dos programadores a plataforma de desenvolvimento de aplicações móveis para o seu sistema operativo.

A plataforma tem como base a tecnologia. NET e *Silverlight*. "*Windows Phone* 7" foi o nome escolhido para o novo sistema que veio substituir o sistema operativo móvel *Windows Mobile*. A arquitetura do *Windows Phone* é dividida em quatro componentes: *Runtime – On "Screen"; Tools; Cloud Services* e *Portal Services*. A figura seguinte ilustra esses quatro componentes.

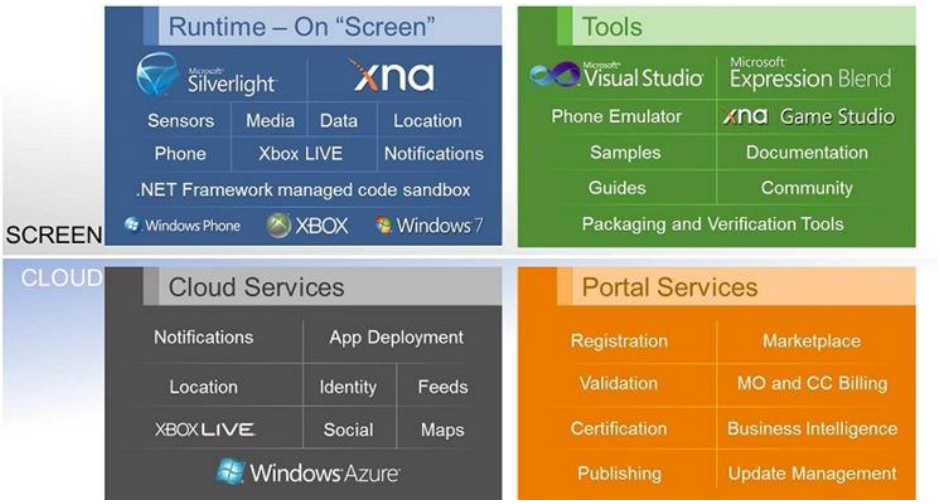


Figura 20. Arquitetura da plataforma Windows Phone (adaptada de Queirós (2008))

Rutime – On " Screen"

Os dois principais componentes de desenvolvimento *Runtime* consistem na utilização da tecnologia do *Silverlight* e XNA Framework. *Silverlight* é a mais recente tecnologia da Microsoft para auxiliar no desenvolvimento de RIA - *Rich Internet Applications*. A plataforma XNA, que tinha como objetivo inicial ajudar os programadores a desenvolver jogos em DirectX para computador, agora tem a possibilidade de fornecer um conjunto de biblioteca e ferramentas necessárias para desenvolver jogos personalizados para Xbox 360. Além disso, os componentes estão associados ao dispositivo Zune da Microsoft com o objetivo de gerir ficheiros do tipo multimédia e reproduzi-los.

Os componentes do *Runtime* são também constituídos por poderosos sensores, que podem ser usados nas aplicações. Todo o desenvolvimento de aplicações é assim baseado nas duas principais plataformas *Silverlight* e XNA Framework, em que todos os pedidos realizados à API (*Application Programming Interface)*, é sempre feito de forma protegida e controlada. O programador apenas tem de realizar esses pedidos como cliente e construir as suas aplicações. Assim, estas duas *frameworks* juntamente com os componentes *Windows Phone frameworks*, e todas as classes disponíveis na biblioteca, disponibilizam todos os componentes necessários para construir as aplicações, tal como se pode ver em mais pormenor na figura 21.

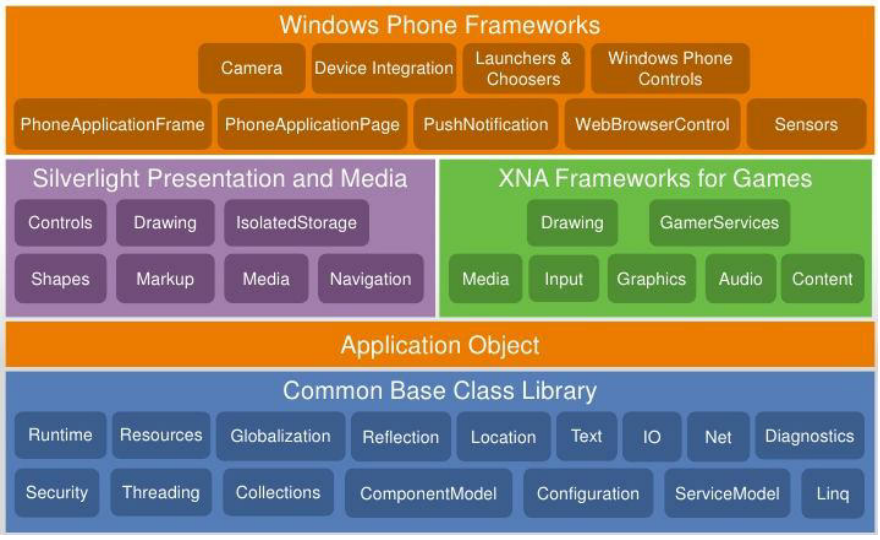


Figura 21. Arquitetura Windows Phone (adaptada de Queirós (2008))

Cloud Services

A Microsoft fornece um conjunto de serviços alojados em nuvem (*cloud services*), tornando assim as aplicações e os serviços mais eficazes e funcionais. A tecnologia *Windows Azure*, Serviços Xbox LIVE, serviços de alerta, serviços de localização, e outros serviços de web, permitem que os programadores partilhem dados na nuvem. O componente *Notification Service* fornece ao utilizador alertas e o *Bing Maps Control* que permite a consulta de mapas.

Portal Services

O Microsoft Windows proporciona um local para as aplicações puderem ser certificadas e comercializadas para que os consumidores finais possam comprar ou atualizar. Para tal, existe um processo que é efetuado por etapas, primeiro é o registo e a validação da aplicação, o responsável pelo desenvolvimento da aplicação, terá que inscrever-se na aplicação *Hub Suite* com o seu Live ID, ao concluir o seu registo poderá obter todas as ferramentas necessárias.

A certificação, publicação e a gestão das atualizações da aplicação, obriga a apresentação de uma candidatura no formato .XAP, que contém todos os ficheiros da aplicação compactados. Posteriormente podem definir-se os preços e selecionar os mercador onde deseja publicar a aplicação. Caso a aplicação sofra alterações, novas funcionalidades ou uma versão completamente diferente da original, deve realizar os mesmos passos.

Tools

Atualmente existe um conjunto de ferramentas de ambiente e desenvolvimento das aplicações, sendo o *Visual Basic* o principal. Para além desta ferramenta, o Microsoft disponibiliza o IDE, *Expression Blend* que permite aos *designers* trabalhar de forma mais fluida. Uma das ferramentas de integração com o *Visual Studio* e com o *Express Blend* de modo a realizar testes e *debug* das aplicações de forma mais eficiente é o emulador do *Windows Phone*. Outro IDE utilizada é XNA *Game Studio*, que integra todas as ferramentas necessárias para o desenvolvimento de jogos, bem como a documentação que poderá auxiliar o utilizador na sua aplicação.

3.3. Plataforma iPhone

Segundo Reis et al. (2012), o sistema operativo iOS (iPhone OS) é um sistema da *Apple* desenvolvido inicialmente para o *Smartphone iPhone* e posteriormente adaptado ao *iPod Touch*, *iPad* e *Apple* TV. Este *software* só pode funcionar em equipamentos da *Apple*, uma vez que a *Apple* não permite que o iOS seja executado em *hardware* de terceiros. É um dos sistemas operativos mais antigos em desenvolvimento até hoje e foi o primeiro a ter um interface com ecrã sensíveis a multi-toque.

Inicialmente, o *iPhone* possuía uma plataforma totalmente fechada, uma vez que não estavam disponíveis as ferramentas que permitiam desenvolver aplicativos para a plataforma, o chamado SDK, com exceção de alguns parceiros escolhidos pela *Apple*, a única forma de desenvolver aplicativos para ele era através de *widgets*, que são páginas *web* destinadas a exibirem informações específicas. Com o aparecimento do *iPhone* 3G, isso mudou embora ainda com restrições através de uma rigorosa cláusula que impede que os desenvolvedores troquem informações entre si. Assim, a *Apple* passou a disponibilizar o SDK do *iPhone* OS, possibilitando o desenvolvimento para a plataforma, no entanto continuou sendo fechado, uma vez que a *Apple* continua a controlar a distribuição das aplicações. A única forma autorizada de instalar aplicações adicionais no *iPhone* é através da *AppStore*, que é diretamente controlada pela *Apple*. Todas as aplicações passam por uma revisão e apenas as que forem aprovadas são disponibilizadas através da loja.

A nível de arquitetura o iOS consiste em quatro camadas de abstração, figura 22.

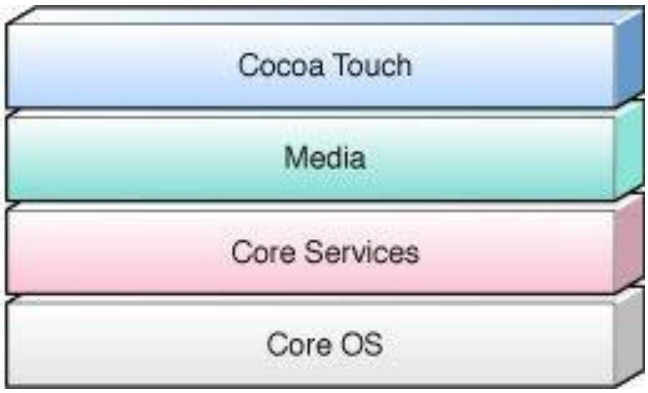


Figura 22. Estrutura do Sistema Operacional iOS (adaptada de Reis et al. (2012))

As camadas da arquitetura do iOS sao: *Core OS, Core Services, Media e Cocoa Touch*, sendo *a Cocoa Touch* a de maior nível e a Core OS a de mais baixo nível. A seguir uma explicação detalhada sobre a função de cada camada.

Camada Core OS (núcleo do sistema operacional)

Esta camada é a mais próxima do *kernel* e inclui acesso direto a funcionalidades de baixo nível tais como: OS X *Kernel*, *Sockets*, Segurança, Gerenciamento de Energia, Certificados, Sistema de Arquivos. Sua função é gerir o funcionamento do sistema operativo, cuidando da gestão de memória, de bateria, da luminosidade da tela, da segurança, entre outras funções.

Camada Core Services (serviços oferecidos pelo sistema)

Fornece os serviços fundamentais usados por todas as aplicações. Serviços como comunicação, acesso à agenda de contatos, acesso ao calendário, visualização e acesso de ficheiros com formatos suportados pelo sistema sqlite e rede.

Camada Média (serviços de mídia ios)

Inclui *frameworks* de reprodução e gravação de áudio e vídeo e bibliotecas gráficas aceleradas por *hardware* para animação e renderização 2D e 3D. A renderização de gráficos 3D é feita através da API *OpenGL* e leitor pdf.

3.4. Comparação das Plataformas

Após um estudo detalhado acerca das três principais plataformas de *smartphones*, chegou a altura de decidir qual a melhor plataforma que se adequa à aplicação. Para isso foi feita uma comparação das vantagens e desvantagens de cada um deles.

Uma das vantagens que o sistema *Android* tem, é que pode ser utilizada em diversos modelos de *smartphones* que são fabricados por diversos fabricantes, ao contrário do *iPhone* que é utilizado em poucos aparelhos diferentes e todos são da *Apple*.

Quando falamos de aplicativos, o *iPhone* predomina em relação ao *Android*, o *Windows Phone* possui poucos aplicativos e somente certificados pela *Microsoft*.

No que diz respeito a preços, o *iPhone* continua a ser dos *smartphones* mais caro. O sistema operativo *Android* como é encontrado em diversos aparelhos é possível encontrar *smartphones* com configurações mais básicas com ótimos preços. Já o *Windows Phone* ainda é um pouco caro, pois tem poucas opções, e fica na mesma faixa de preços dos *smartphones Android* mais poderosos.

A nível mundial, quando se fala em *smartphones*, os sistemas operativos mais utilizados são *iPhone* e o *Android*, embora o *Windows Phone* já tenha evoluído muito mas ainda continua atrás destes dois sistemas. No que diz respeito à flexibilidade, eficiência, ajuda e documentação o *Android* é o que traz mais benefícios.

No que se refere à programação, o *Android* foi desenvolvido para funcionar em *Windows*, *Linux* e até mesmo no *Mac*, o que dá mais liberdade de escolha de *hardware.* Como se trata de um sistema operativo livre, de código aberto, a documentação é vasta por abranger utilizadores de diversos sistemas operativos, o que não acontece com os outros dois sistemas, que são restritos a eles próprios e tem de se pagar para poder publicar as aplicações. Além disso, todas as aplicações, tanto no *Windows Phone* como no *iPhone*, têm de passar por uma burocracia "apertada" e apenas os que forem aprovados é que são disponibilizados. Isso traz uma desvantagem para o *Android* pela falta de segurança, mas o utilizador consome o que quiser e assume os riscos por isso.

Do ponto de vista de recursos das plataformas computacionais, qualquer das plataformas estudadas parece ter recursos adequados para a aplicação, contudo a plataforma *Android* foi a que mais se adaptou devido a vários fatores, tais como, é de código aberto, ou seja, o *Android* é um sistema baseado no código *Linux* que possui licença aberta, existem uma variedade de dispositivos em diferentes marcas, possui uma infinidade de aplicações grátis na *Android Market* e a nível de preços é possível encontrar *smartphones* a um valor acessível. O ambiente de desenvolvimento que se vai usar para desenvolver a plataforma é o Eclipse IDE (Android Developers, 2013).

4. Trabalho Desenvolvido

O Capítulo 4 descreve o sistema de assistência ao condutor desenvolvido, na forma de uma aplicação para *smartphone*, apresentando a abordagem seguida para a deteção de cansaço em condutores e os algoritmos implementados.

4.1. Arquitetura do Sistema

A abordagem seguida para implementar a deteção de cansaço em condutores foi usar os métodos de Visão Computacional capazes de analisar as imagens capturadas pela câmara do *smartphone*, de modo a detetar sinais de cansaço em condutores.

O diagrama da figura 23 apresenta os módulos que foram criados para o sistema desenvolvido.

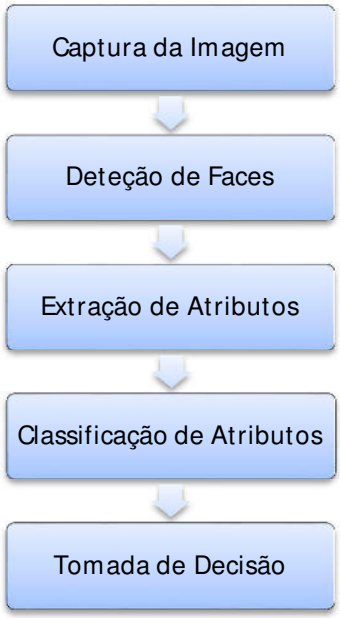


Figura 23. Módulos referentes ao sistema desenvolvido

A captura da imagem é o primeiro passo para um sistema de Visão Computacional. Trata-se do processo de aquisição de uma imagem a partir de sensores de câmaras, onde os *pixels* de cada imagem obtida indicam coordenadas de luz e propriedades físicas.

Deteção e segmentação é um processo realizado para destacar regiões relevantes da imagem, segmentando-as para processamento posterior, nomeadamente a deteção de face do condutor.

Extração de atributos tem como o objetivo localizar regiões da imagem que contenham caraterísticas significativas. Como os principais sintomas de cansaço refletem-se através dos olhos e da boca, é imprescindível que haja uma extração dessas mesmas regiões.

A classificação de atributos é o processo que inclui validação da satisfação dos dados obtidos, estimativa de parâmetros sobre a imagem e classificação dos objetos obtidos em diferentes categorias. Com a referência dos olhos e da boca classifica-se o estado dos olhos e da boca como aberto ou fechado.

A tomada de decisão é feita em função da classificação do estado dos olhos e da boca através de um conjunto de regras implementadas para identificar o cansaço do condutor.

As secções a seguir descrevem a implementação de cada módulo e as diferentes alternativas que foram testadas para alguns deles.

4.2. Smartphone Utilizado

Para escolher o *smartphone* foi necessário ter em conta alguns fatores relevantes do sistema, nomeadamente, o método escolhido para a deteção de faces usa um sistema operativo Android superior a 4.0, as imagens têm de ter alguma qualidade para se poder extrair caraterísticas significativas e como estamos a trabalhar com imagens em tempo real é necessário possuir um processador aceitável. De acordo com estas exigências o *smartphone* utilizado na aplicação é o Samsung Galaxy S3 que vem equipado com o sistema operativo *Android* 4.3, câmara de 8 *megapixels*, a câmara frontal é de 1.9 *megapixels* e o ecrã tem uma resolução de 1280x720 *pixel*. Possui memória RAM de 16 GB e um processador 1.4 GHz *Quad Core*.

4.3. Módulo de Captura da Imagem

A captura da imagem através da câmara de um dispositivo pode ser feita de duas formas. A primeira, e mais simples, utiliza a aplicação nativa da câmara que já existe no *Android*. E a segunda permite que o programador tenha controlo total do *hardware* da câmara. Para a aplicação optou-se pela programação do *hardware* da câmara pelo facto de haver necessidade de criar interfaces gráficas no *Android*.

A resolução escolhida para a captura de imagem na aplicação, foi de 640 x 480 *pixels*, pelo facto de ser uma resolução o mais pequena possível para se ter um processamento rápido das imagens, mas também por ser uma resolução suficientemente grande para que não se perca detalhe das imagens e os algoritmos possam funcionar bem. Além disso, foram realizados testes com diferentes resoluções, nomeadamente no âmbito dos algoritmos de deteção de faces, e a resolução escolhida foi com a qual se obteve o melhor compromisso entre rapidez de processamento e qualidade de resultados.

O dispositivo móvel utilizado possui duas câmaras, a frontal e a traseira. Optou-se por configurar a câmara frontal por ser mais prático para o desenvolvimento do projeto na medida em que é permitido ao utilizador a visualização dos testes ocorridos. É de salientar, segundo a documentação do *hardware* da câmara, que a tela de visualização da câmara frontal é invertida horizontalmente antes da rotação, ou seja, a imagem é refletida ao longo do eixo vertical central do sensor da câmara, é como se estivéssemos a olhar para um espelho. Este detalhe é intrínseco do funcionamento da câmara e que se deve ter em conta na sua aplicação. A orientação de visualização da câmara escolhida foi em modo paisagem porque na maioria dos dispositivos é a orientação padrão.

Ainda referente à captura da imagem é de referir que em dispositivos *Android*, o formato padrão da imagem da câmara de visualização encontra-se no formato Y'UV420sp (NV21). É necessário converter a imagem em formato RGB para poder desenhar no ecrã do dispositivo de modo a permitir ser percetível à visão humana. A conversão entre o Formato YUV e RGB foi realizado com base no algoritmo descrito em *Java interface to OpenCV* (2013).

4.4. Módulo de Deteção de Faces

A deteção de faces é um método de Visão Computacional que determina os locais e tamanhos de rostos humanos em imagens digitais. Deteta a face e ignora objetos presentes na imagem. Uma das principais áreas de desenvolvimento do domínio da Visão Computacional é a da deteção de faces em imagens.

No caso da deteção de faces com base nas APIs 1.5 e 14, (que irão ser descritas a seguir), teve-se e, conta o facto de o espaço de coordenadas usado pelos algoritmos de deteção de faces, não ser igual ao espaço de coordenadas do ecrã onde depois a imagem capturada pela câmara é apresentada. Esta situação é ilustrada na figura 24.

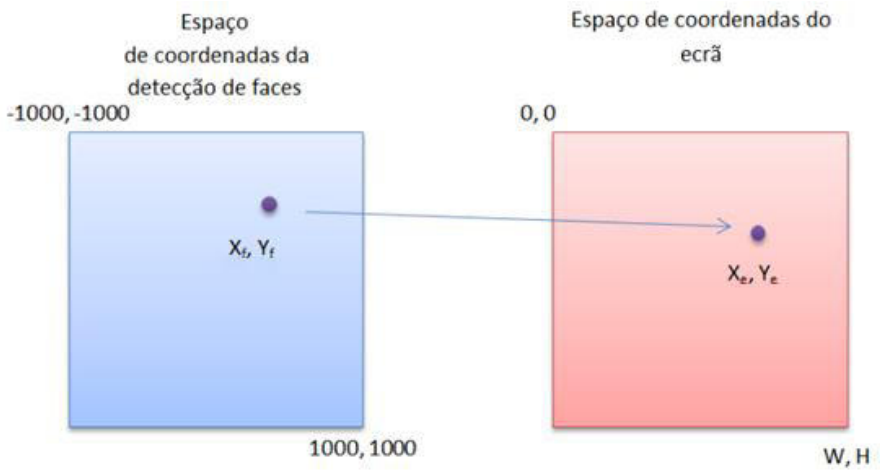


Figura 24. Espaço de coordenadas da deteção de faces e Espaço de coordenadas do ecrã

Isso significa que os algoritmos de deteção de faces devolvem o resultado da deteção (um retângulo que identifica a posição da face), em coordenadas do espaço de algoritmo. Para se poder representar visualmente (desenhar) esse retângulo sobre a imagem que aparece no ecrã, é necessário realizar uma conversão de coordenadas entre os dois espaços de coordenadas. Isto é feito multiplicando um ponto P_f no espaço de coordenadas de deteção de faces por uma transformação geométrica composta M de modo a obter o correspondente ponto P_e do espaço de coordenadas de ecrã (equação 4-1).

$$\boldsymbol{P_e = M.P_f} \qquad \textbf{(4-1)}$$

A matriz *M* é obtida pela equação 4-2.

$$\boldsymbol{M} = \boldsymbol{S}_{(sx,sy)} \cdot \boldsymbol{E}_{(ex,ey)} \cdot \boldsymbol{T}_{(tx,ty)} \quad \textbf{(4-2)}$$

Onde

T é uma translação com $\boldsymbol{tx} = \frac{\boldsymbol{W}}{\boldsymbol{2}}$ e $\boldsymbol{ty} = \frac{\boldsymbol{H}}{\boldsymbol{2}}$ que alinha os centros dos dois espaços de coordenadas;

E é uma variação de escala com $\boldsymbol{ex} = \frac{\boldsymbol{W}}{\boldsymbol{2000}}$ e $\boldsymbol{ey} = \frac{\boldsymbol{H}}{\boldsymbol{2000}}$ que converte as coordenadas entre espaços de coordenadas;

S é outra variação de escala com *sx*= -1"e " *sy*= 1 para provocar uma simetria horizontal e desfazer o efeito espelho característico da imagem capturada pela câmara frontal de um dispositivo *Android*.

De seguida será feita uma descrição das alternativas testadas.

4.4.1. Deteção de Faces com Base no Método HarCascade

O algoritmo inicialmente utilizado para a deteção da face, foi o método criado por Viola-Jones (2008) baseado em filtros de *HarCascade*. Segundo Viola-Jones, o seu método possui três contribuições principais que o distingue de outros algoritmos e o torna computacionalmente mais eficiente, elas são: a imagem integral, o algoritmo de aprendizado baseado em *AdaBoost* e um método para combinar classificadores de complexidade crescente baseados em *Haar-like features* para criar um filtro em cascata eficiente.

Com o intuito de otimizar o cálculo de tais características retangulares, Viola-Jones propõe a utilização de uma representação intermediária da imagem chamada de Imagem Integral. Esta técnica foi utilizada para que fosse possível computar de forma eficiente a presença ou não de uma característica em cada uma das diversas posições e escalas que uma janela de observação pode ser colocada na imagem em que se quer detetar um objeto.

O objetivo principal a ser alcançado por uma função de classificação é o de ser capaz de classificar corretamente um dado objeto a partir do conjunto das suas principais características. O algoritmo escolhido por Viola-Jones é uma variante do algoritmo de aprendizado *AdaBoost* (Freund e Schapire 1995). Este algoritmo permitirá selecionar as melhores características aleatórias, treinando assim o classificador.

O conceito de Cascata usado na implementação do algoritmo de Viola-Jones deriva do facto de que as subjanelas devem ser rapidamente descartadas em caso de não haver uma face na mesma. O funcionamento básico de uma estrutura como essa ocorre pela passagem de todas as subjanelas de uma imagem por diversos classificadores, sendo os primeiros mais simples que os seguintes. Se é obtido um resultado positivo pela análise do primeiro classificador, a janela segue para o seguinte, que se também for aprovada, contínua pelos próximos classificadores até ser rejeitada ou então chegar ao final da cascata. Em qualquer uma das etapas, uma janela pode ser rejeitada, isso quer dizer que o detetor entende que ali não se encontra uma face.

Para usar o método *HaarCascade* na aplicação recorreu-se à API *OpenCV* para *Android*. *OpenCV* gera um arquivo XML a partir de amostras de treinamento, que podem ser utilizados para a deteção de objetos rápido. Tal arquivo XML é fornecido com API *OpenCV* para a deteção de face. Para utilizar a biblioteca *OpenCv* houve necessidade de alguns procedimentos para a sua configuração que podem ser encontrados no *Java interface to OpenCV* (2013).

Após a implementação do algoritmo proposto por Viola-Jones, verificou-se que apesar de ser muito eficiente, possui alguns pontos negativos. Ao se utilizar o algoritmo de deteção em vídeos, a taxa de deteção deixa de ocorrer em tempo real e a taxa de *frames* por segundo cai de forma considerável, valores entre 1 a 3. As outras desvantagens estão relacionadas com a capacidade do tamanho do projeto, pois apenas para a deteção da face tem-se um tamanho de valor igual a 34 MB, a deteção da face era muito lenta e ainda restava desenvolver os restantes módulos. Tendo em conta estes resultados não seria o método mais eficiente para aplicar na aplicação.

4.4.2. Deteção de Faces com Android API 1.5

A partir da API 1.5, a plataforma passou a contar com duas importantes APIs, *FaceDetector.Face* e *FaceDetector*. A API *FaceDetector.Face* contém todas as informações que identifica a localização da face num bitmap e a API *FaceDetector* identifica as faces das pessoas num objeto gráfico Bitmap.

Após uma pesquisa profunda constatou-se que, através de *Underlying technique of Android's FaceDetector* (2010), é criada através de um processo de deteção de face que faz parte de uma técnica de aprendizagem de uma máquina de reconhecimento de objetos usando um conjunto de características. Essa técnica é baseada na Transformada de Fourier onde o reconhecimento das faces é feito por encontrar a correspondência mais próxima entre vetores de características contendo os coeficientes de Fourier em frequências selecionadas.

Como a plataforma possui estas duas APIs de deteção de faces não há necessidade de instalar as aplicações, uma vez que elas vêm com a base de APIs do *Android*, não a partir de pacotes de opcionais. Um tutorial que se concentra em utilizar essas APIs para realizar a tarefa de deteção de face pode ser encontrado em *Android Developers* (2013).

O desempenho deste método em relação ao tamanho da aplicação é bastante satisfatório, pois é apenas de 444 KB. Contudo, no que diz respeito à taxa de *frames* por segundo é de 4 valores e a deteção em condições de movimento era muito lenta. Tendo em conta estes resultados não seria o método mais eficiente para aplicar no projeto.

4.4.3. Deteção de Faces com Android API 14

A partir do nível 14, o *Android* faz o reconhecimento da face através de duas classes, *Camera.Face* e *Camera.FaceDetectionListener*. A classe *Camera.Face* possui informações sobre uma face identificada através de deteção de face da câmara. Quando a deteção de face é usada com a câmara, o *Camera.FaceDetectionListener* retorna uma lista de objetos de face para uso em foco e medição. *Camera.FaceDetectionListener* é o interface de retorno de chamada para a face detetada na área.

Apesar de muita procura sobre qual o método usado por estas classes para realizar a deteção de faces, não foi possível encontrar nada em concreto, contudo deduz-se que

seja uma versão melhorada do método usada na versão do nível 1.5 e que empregue as mesmas técnicas referidas anteriormente.

Neste caso, também não há necessidade de uma instalação da deteção de faces com o *Android*, uma vez que aplicações vêm com a base de APIs do *Android*. Um tutorial que se concentra em utilizar essas APIs para realizar a tarefa de deteção de face pode ser encontrado em *Android Developers* (2013).

O desempenho deste método em relação ao tamanho da aplicação é bastante satisfatório, pois apenas é de 464 KB, a taxa de *frames* por segundo é compreendida entre 16 a 20 valores (considerado bastante razoável) e a deteção em condições de movimento era muito eficaz. Tendo em conta todos estes fatores satisfatórios este método foi o escolhido.

4.5. Módulo de Extração de Atributos

Este módulo tem como o objetivo localizar regiões da imagem que contenham atributos significativos. Como os principais sintomas de cansaço refletem-se por exemplo através dos olhos e da boca, é imprescindível que haja uma extração dessas mesmas regiões.

Diferentemente da face, na primeira fase do procedimento de deteção, os olhos e a boca não foram procurados em toda a imagem, optou-se por extrair uma ROI dos olhos e da boca através do retângulo do rosto previamente detetado. A localização dos olhos e da boca é encontrada com base em determinadas dependências geométricas conhecidas para o rosto humano. As tradicionais regras de proporção, segundo os autores Krolak e Strumillo (2010), mostram a face dividida em seis quadrados iguais, ou seja, dois por três. De acordo com essas regras podemos definir que os olhos estão situados cerca de 0.4 em relação à parte superior da cabeça, figura 25.

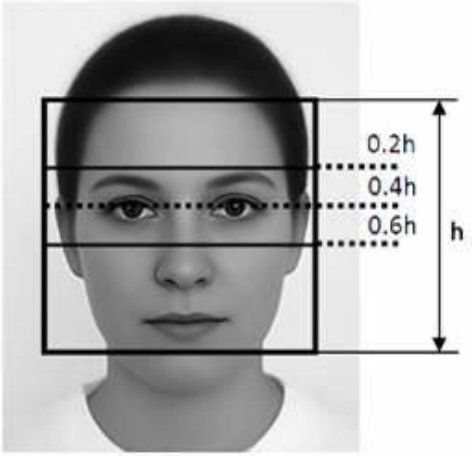


Figura 25. Regras de proporções da face humana (adaptada de Krolak, e Strumillo (2010))

Com base nestas regras de proporções da face humana e após vários testes com diferentes faces chegou-se à conclusão que o tamanho ideal para os retângulos dos olhos é que apenas foque essencialmente os olhos e que altura máxima do retângulo esteja abaixo da sobrancelha. Para o retângulo da boca será de modo que capture toda a região da mesma com altura máxima abaixo do nariz. Considerando as coordenadas do retângulo *(l, t, r, b)* temos as seguintes proporções:

Retângulo do olho esquerdo:

$$R = (0.5w,\ 0.3h,\ 0.8w,\ 0.4h)$$

Retângulo do olho direito:

$$R = (0.2w,\ 0.3h,\ 0.4w,\ 0.4h)$$

Retângulo da boca:

$$R = (0.2w,\ 0.6h,\ 0.7w,\ 0.8h)$$

Onde:

w corresponde à largura do retângulo a partir das coordenadas do retângulo da face;

h corresponde à altura do retângulo a partir das coordenadas do retângulo da face;

Na figura 26 podemos ver o resultado final da extração de atributos, ou seja, a identificação das regiões de interesse correspondentes aos olhos e à boca.

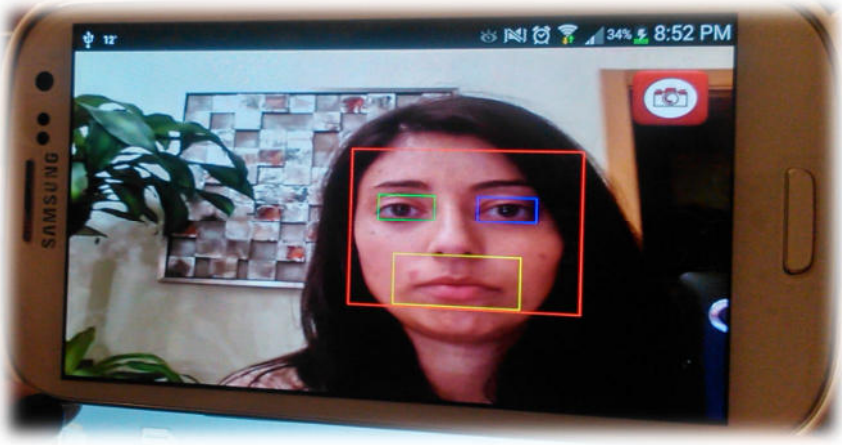


Figura 26. Identificação das regiões de interesse correspondentes aos olhos e à boca

4.6. Módulo de Classificação de Atributos

A classificação de atributos é o processo que inclui validação da satisfação dos dados obtidos, estimativa de parâmetros sobre a imagem e classificação dos objetos obtidos em diferentes categorias. Após a referência dos olhos e da boca na respetiva imagem, verifica-se o estado de classificação referente aos olhos e à boca (aberto/fechado).

Para a classificação de atributos o método utlizado aplica-se em imagens em tons de cinzento. Para o efeito é capturado um bitmap a partir do retângulo de deteção da face, cujos *pixels* resultam da conversão do formato YUV da imagem original para o formato RGB em tons de cinzento. Essa conversão foi realizada com base no algoritmo referido em *YUV420SP Grayscale* (2014).

De seguida é feita uma descrição das alternativas testadas.

4.6.1. Método Baseado no Histograma de Intensidades

Um possível método poderá ter como base a análise do histograma do pedaço da imagem correspondente ao olho ou à boca e extrair algumas características do histograma que permitam ao programa decidir se o olho ou a boca estão abertos ou fechados.

O histograma de uma imagem é um conjunto de valores que representa a quantidade de *pixels* que a imagem tem para cada tom possível de cinzento. Cada *pixel* pode variar de 0 (preto absoluto) até 255 (branco absoluto). Para a análise dos tons, é recomendado que se usasse o histograma de luminosidade e não separado em canais, uma vez que é mais difícil de ser analisado. A ideia é implementar o método referenciado na secção 2.2 Sistemas de Deteção de Cansaço em Condutores apresentado na figura 8. Através do histograma calculamos valores aritméticos, tais como, a média e o desvio padrão e consegue-se verificar se o estado dos olhos detetados estão abertos ou fechados, ou seja, quando os olhos estão fechados, a distância da coordenada x das alterações de intensidade é maior comparada quando os olhos estão abertos.

Após a implementação deste método, podemos ver os resultados nas figuras que se seguem.

Figura 27. Histograma das intensidades referente ao olho aberto

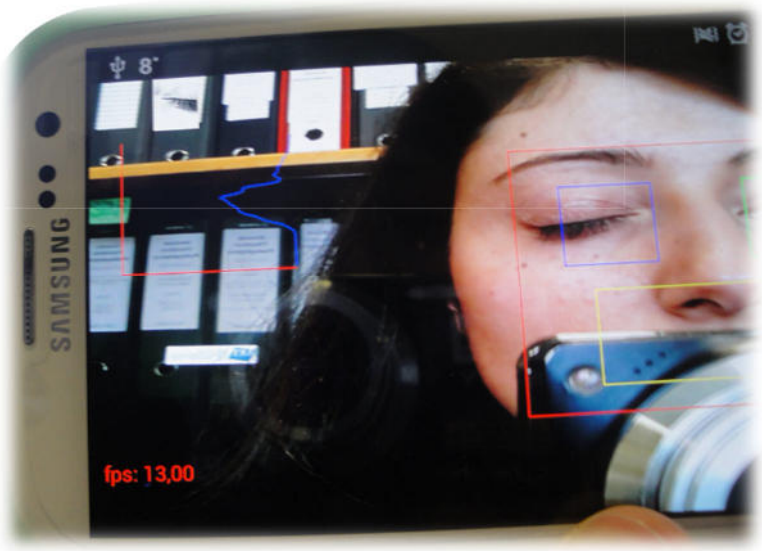


Figura 28. Histograma das intensidades referente ao olho fechado

A partir das imagens é possível observar que existem diferenças no gráfico do olho aberto e do olho fechado, o nível de cristas do histograma difere nas duas situações, porém em termos de valores não são diferenças significativas. Optou-se por melhorar a qualidade da imagem aplicando um contrate e uma binarização, mas continuava a não resultar. Posteriormente, foram calculados valores aritméticos, tais como, a média e o desvio padrão durante uma situação de 30 segundos onde os primeiros 10 segundos o olho está aberto, nos próximos 10 segundos o olho está fechado e depois o olho fica aberto novamente.

A figura que se segue é uma representação gráfica (feito no Excel), da média e do desvio padrão da situação que foi descrita anteriormente.

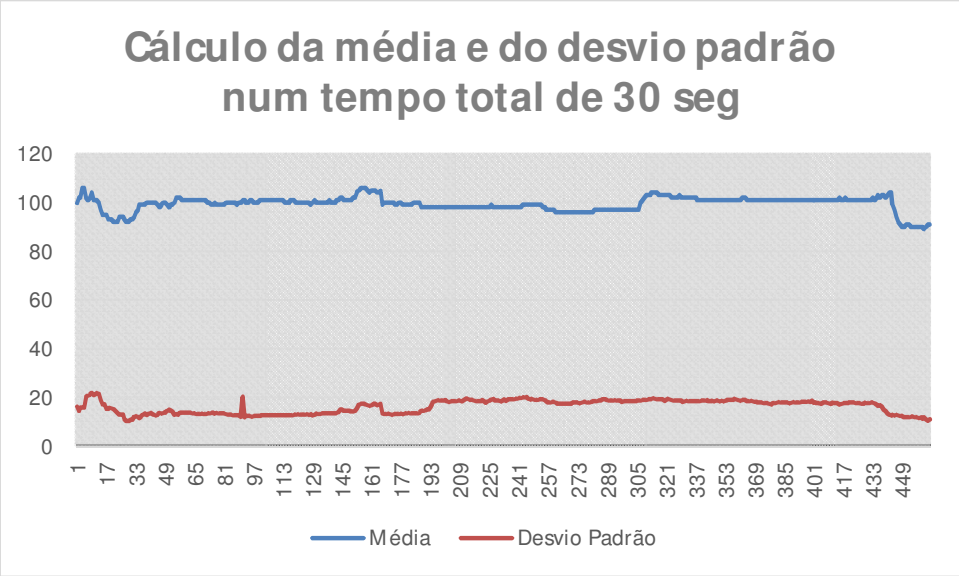


Figura 29. Gráfico com medidas aritméticas tiradas do histograma

Através do gráfico é possível verificar que existem algumas variações de quando o olho está aberto e fechado, no entanto essas variações continuam a ser mínimas o que não nos é permitido assegurar um intervalo para definir o estado do olho.

Visto não estar a obter os resultados desejados com base no tipo de metodologia usada até ao momento, optou-se que fosse implementado um tipo de método mais sofisticado para a classificação dos estados dos olhos e da boca.

4.6.2. Método de Template Matching

Template Matching consiste em comparar a informação visual extraída de uma imagem com a informação extraída de um *template*. Um *template* é normalmente uma informação visual que contém uma imagem de referência do objeto, ou o conjunto de objetos, a serem detetados na imagem em estudo. Existem dois tipos de métodos de *Template Matching*, o método de correspondência deformável e de correspondência por padrão (Rodrigues 2013).

O método de correspondência deformável é um modelo geométrico parametrizado do objeto a ser reconhecido, em conjunto com uma medida de quão bem ajusta os dados de imagem. As variações nos parâmetros correspondem às deformações admissíveis do

objeto e pode ser especificado por um modelo probabilístico. Após a etapa de extração dos parâmetros do modelo deformável pode ser utilizado para a descrição do objeto e reconhecimento.

O método de correspondência por padrão, que foi o utilizado para a aplicação, consiste em comparar a distribuição de níveis de cinzento de uma matriz amostra (padrão) com matrizes candidatas pertencentes a uma matriz de pesquisa usando uma função de correlação adequada. Neste caso foi feita a comparação de imagens de *template* com as imagens capturadas em tempo real, ou seja, primeiro são capturados *templates* de cada olho e da boca e posteriormente são feitas comparações das imagens em tempo real dos olhos e da boca com as imagens capturadas no *template*.

Segundo o autor Medeiros (2009), várias são as funções para este tipo de correspondência: erro, erro quadrático, correlação, covariância cruzada e covariância cruzada modificada. A função que foi escolhida para ser usada na aplicação foi a covariância cruzada modificada ou coeficiente de correlação por ser a mais citada na literatura e que proporciona melhores resultados. Para a determinação da correspondência a matriz amostra (ou janela de referência) é comparada com cada matriz candidata na janela de referência, comparação é feita com o cálculo de uma das funções especificadas anteriormente.

A função de coeficiente de correlação resulta da normalização da função covariância cruzada (Equação 4-3). O coeficiente de correlação, neste caso, varia de –1 a 1. O valor 1 para o coeficiente de correlação corresponde à medida de similaridade máxima, o valor 0 indica que não há correlação e o valor -1 indica correlação inversa.

$$CC = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{(\sum(x_i - \overline{x})^2)(\sum(y_i - \overline{y})^2)}} \quad \textbf{(4-3)}$$

De seguida irá ser feita uma apresentação da transformação da fórmula que foi feita para a implementar de forma mais eficiente no programa.

Denotamos as variáveis:

y = imagem de template;

x = imagem capturada em tempo real;

n = número de amostras;

$\bar{x}$ = média da imagem capturada em tempo real;

$\bar{y}$ = média da imagem do template;

$$s_{xx} = \sum x_i^2 - n(\bar{x})^2 \quad \textbf{(4-4)}$$

$$s_{yy} = \sum y_i^2 - n(\bar{y})^2 \quad \textbf{(4-5)}$$

$$CC = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{s_{xx}s_{yy}}} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sqrt{s_{xx}s_{yy}}} \quad \textbf{(4-6)}$$

Uma outra situação a ter em conta é como é que o sistema se comporta quando o utilizador move a cabeça em relação à imagem que está a ser processada. Se o movimento for rápido, a imagem fica momentaneamente diferente do *template*, não por causa dos estados dos olhos ou da boca, mas por causa do movimento. Se a imagem fica instável devido a movimentos bruscos, durante alguns *frames* podem ser capturados retângulos que não correspondem exatamente à área pretendida, pelo que é normal que ao serem comparados com o *template*, resultem num valor de semelhança próximo de zero, mesmo se o utilizador tem os olhos ou a boca no mesmo estado que está no *template*. Para resolver essa questão, foram comparadas as coordenadas do canto superior esquerdo do retângulo

da face no *frame* atual com as do *frame* anterior. Se o valor absoluto da diferença for maior do que o limite que foi definido, então é porque houve um movimento brusco da imagem e o Coeficiente de Correlação não é calculado nesta situação, fica com o valor que tinha anteriormente.

Para o exemplo da aplicação, após várias tentativas de simulação, considerou-se, para a classificação do estado dos olhos, que acima do valor 0.7 para a média dos últimos cinco valores do coeficiente de correlação corresponde à medida de similaridade da captura do *template* e abaixo de 0.7 indica que não há correlação. Isto quer dizer, se a captura for feita no estado de olho aberto, um limiar abaixo de 0.7 corresponde ao estado de olho fechado. Para a classificação do estado da boca é o mesmo processo para a classificação do estado dos olhos mas de valor igual a 0.8. Ponderou-se a média dos últimos cinco valores de coeficiente de correlação para não estar a considerar todos os seus valores, pelo facto de em vez de armazenar cada valor e calcular a tomada de decisão referente aos olhos e à boca se estão abertos ou fechados com cada valor, faz-se antes com a média de cinco em cinco valores para o programa ser mais eficiente no tempo da sua execução.

4.7. Módulo de Tomada de Decisão

O módulo de Tomada de Decisão é o responsável pela classificação de cansaço em função da classificação do estado dos olhos e da boca realizada no módulo anterior. A classificação do cansaço é realizada com base num conjunto de regras usadas também para criar três níveis de alarme para o condutor, (verde, amarelo e vermelho), para permitir informar o utilizador acerca da classificação de cansaço. O nível de alarme de cor verde transmite que não que não existe qualquer perigo para o condutor, ou seja o cansaço é Negativo; Amarelo corresponde a um alerta, o cansaço é Negativo mas já começa a ter indícios de cansaço e o vermelho diz que o cansaço é realmente Positivo. Após vários testes considerou-se que quando os olhos permanecerem fechados e a boca manter-se aberta por mais de 2 segundos é emitido o alerta de cansaço. A tabela que se segue descreve as regras usadas para classificar o cansaço do condutor.

Tabela 3. Regras usadas para classificar o cansaço do condutor

Estado do Olho Direito	Estado do Olho Esquerdo	Estado da Boca	Cansaço	Níveis de alarme
Fechado	Fechado	Aberto	Positivo	Vermelho
Fechado	Fechado	Fechado	Negativo	Amarelo
Fechado	Aberto	Aberto	Negativo	Amarelo
Aberto	Fechado	Aberto	Negativo	Amarelo
Aberto	Aberto	Aberto	Negativo	Amarelo

Caso não seja nenhuma destas situações, significa que não existe qualquer perigo para o condutor, ou seja o cansaço é Negativo e o nível de alarme é de cor verde.

4.8. Funcionamento da Aplicação

Quando o utilizador inicia a aplicação terá de ser feita a captura do *template* referente aos olhos e à boca. Para que o *template* tenha uma captura mais controlada, ou seja, para que o *template* corresponda à região da imagem pretendida, é necessário capturá-lo só quando não houver movimento da imagem (da face do utilizador). Para uma melhor referência ao utilizador, foi desenhado um retângulo de cor branca que permite informá-lo qual a melhor posição que deve ter em relação à sua face durante a captura do *template*. Essa posição foi encontrada, após vários testes, que devido à simetria da câmara seria a localização ideal para permitir com que não houvesse grandes oscilações na captura do *template*. A partir do ecrã do dispositivo o utilizador irá clicar no botão e só nesse momento é que é feita a captura do *template*. O utilizador clica no botão só quando achar que a imagem está estabilizada e os olhos bem como a boca estarem bem visíveis dentro do correspondente retângulo, tendo em conta, como já foi referido na secção 4.5 Módulo da Extração de Atributos, que o tamanho ideal para os retângulos dos olhos é que apenas foque essencialmente os olhos e que altura máxima do retângulo esteja abaixo da sobrancelha. Para o retângulo da boca será de modo que capture toda a região da mesma com altura máxima abaixo do nariz. Esta captura deve ser feita nas mesmas condições (mesma distância) em que a aplicação vai depois funcionar. A figura 30 mostra como irá aparecer o interface da captura do *template* ao utilizador.

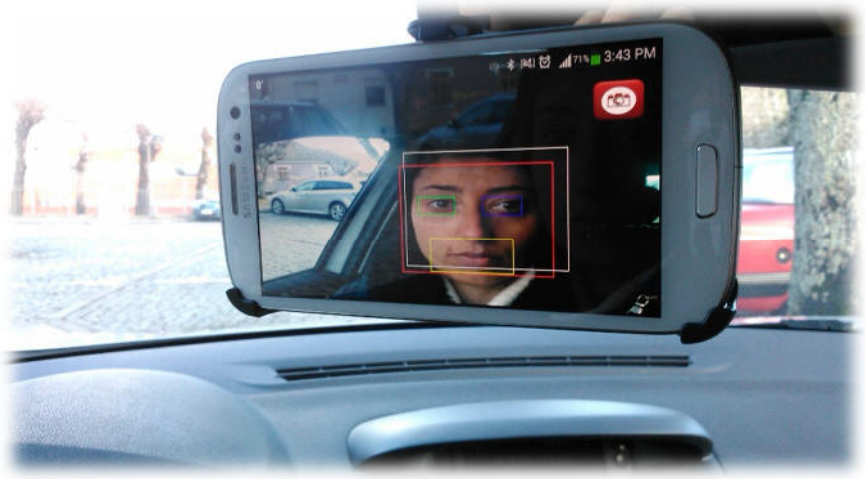


Figura 30. Interface da captura do template referente aos olhos e boca

A partir daqui, o utilizador apenas terá de tentar manter-se na mesma posição do momento da captura e a classificação de cansaço será visualizada através de um círculo de cor corresponde ao nível de alarme que foi definido na secção 4.7 Módulo de Tomada de Decisão (verde, amarelo e vermelho). É de referir ainda que, caso o utilizador esteja perante uma situação de estado de cansaço, além de visualizar na tela do dispositivo o círculo de cor vermelha também será alertado através de um aviso sonoro, que indica que o condutor deve parar o veículo. Esse aviso sonoro irá parar quando o condutor passar para o estado de cansaço negativo. A figura 31 apresenta o interface com uma situação em que ambos os olhos estão abertos e a boca fechada, logo corresponde a um cansaço negativo com um nível de alarme de cor verde, que é representado pela circunferência da mesma cor.

Figura 31. Interface que retrata uma situação de estado negativo de cansaço

5. Testes e Resultados

O Capítulo 5 apresenta os testes realizados para testar a aplicação desenvolvida e os resultados obtidos, assim como a discussão dos mesmos.

5.1. Metodologia

De modo a avaliar o desempenho do sistema, no que se refere à deteção de sinais de cansaço nos condutores, foi tido em conta que a simples quantificação de acertos num grupo de teste não reflete necessariamente o quão eficiente é o sistema na deteção de cansaço, pois essa quantificação dependerá fundamentalmente da qualidade e distribuição dos dados nesse grupo de teste. Por exemplo, considere-se um caso em que o sistema é aplicado num grupo de teste onde os indivíduos vão simular situações de cansaço e de não cansaço, as quais podem ser observáveis e comparáveis com o resultado dado pelo sistema. Considere-se que o sistema tem uma taxa de deteção de 90% das situações de cansaço. Parece ser um resultado bom. Mas pode não ser, pois não levámos em conta a distribuição da condição de cansaço no grupo de teste. Pode acontecer que 90% do grupo de teste simulou a condição de cansaço e o sistema limitou-se a detetar sempre cansaço conseguindo mesmo assim 90% de acertos. Nesta situação já não se pode ter a certeza que o sistema é realmente bom.

Assim, para avaliar o sistema desenvolvido optou-se por aplicar o método da Matriz de Confusão (ou Tabela de Contingência) que segundo o autor Souza (2009), permite calcular um conjunto de diferentes medidas e resolver eventuais desequilíbrios de balanceamento nos grupos de teste. A estrutura da matriz é apresentada na tabela 4.

Tabela 4. Matriz de Confusão

		Valor da Deteção	
		Negativo	**Positivo**
Valor Real	**Negativo**	VN Verdadeiro Negativo	FP Falso Positivo
	Positivo	FN Falsos Negativos	VP Verdadeiro Positivo

O significado dos valores da matriz é o seguinte:

VP - Valores positivos que o sistema julgou positivos (são Verdadeiros Positivos e correspondem a acertos dos sistema);

FN - Valores positivos que o sistema julgou negativos (são Falsos Negativos e correspondem a erros do sistema);

VN - Valores negativos que o sistema julgou negativos (são Verdadeiros Negativos e correspondem a acertos do sistema);

FP - Valores negativos que o sistema julgou positivos (são Falsos Positivos e correspondem a erros do sistema).

Com base nos valores da Matriz de Confusão, é possível calcular diversas medidas sobre o desempenho do sistema, sendo as principais as que se descrevem a seguir.

Acurácia - É a proporção de predições corretas, sem levar em consideração o que é positivo e o que é negativo. Esta medida é altamente suscetível a desbalanceamentos do conjunto de dados e pode facilmente induzir a uma conclusão errada sobre o desempenho do sistema.

$$ACC = \frac{Total\ de\ Acertos}{Total\ de\ Dados\ do\ Conjunto} = \frac{(VP + VN)}{(VP + VN + FP + FN)} \quad (5\text{-}1)$$

Sensibilidade - É a proporção de verdadeiros positivos e corresponde à capacidade do sistema em predizer corretamente a condição para casos que realmente a têm.

$$SENS = \frac{Acertos\ Positivos}{Total\ de\ Positivos} = \frac{VP}{(VP + FN)} \quad (5\text{-}2)$$

Especificidade - É a proporção de verdadeiros negativos e corresponde à capacidade do sistema em predizer corretamente a ausência da condição para casos que realmente não a têm.

$$SPEC = \frac{Acertos\ Negativos}{Total\ de\ Negativos} = \frac{VN}{(VN + FP)} \quad (5\text{-}3)$$

Eficiência - É a média aritmética da Sensibilidade e Especificidade. Na prática, a sensibilidade e a especificidade variam em direções opostas. Isto é, geralmente, quando um método é muito sensível a positivos, tende a gerar muitos falso-positivos, e vice-versa. Assim, um método de decisão perfeito (100 % de sensibilidade e 100% especificidade) raramente é alcançado, e um balanço entre ambos deve ser atingido.

$$EFF = \frac{(SENS + ESPEC)}{2} \quad (5\text{-}4)$$

Preditividade Positiva - É a proporção de verdadeiros positivos em relação a todas as predições positivas. Esta medida é altamente suscetível a desbalanceamentos do conjunto de dados e pode facilmente induzir a uma conclusão errada sobre o desempenho do sistema.

$$PPV = \frac{Acertos\ Positivos}{Total\ de\ Predições\ Positivas} = \frac{VP}{(VP + FP)} \quad (5\text{-}5)$$

Preditividade Negativa - É a proporção de verdadeiros negativos em relação a todas as predições negativas. Esta medida é altamente suscetível a desbalanceamentos do conjunto de dados e pode facilmente induzir a uma conclusão errada sobre o desempenho do sistema.

$$NVP = \frac{Acertos\ Negativos}{Total\ de\ Predições\ Negativos} = \frac{VN}{(VN + FN)} \quad (5\text{-}6)$$

5.2. Aplicação da Metodologia

A avaliação do desempenho do sistema de deteção de cansaço foi realizada através de testes em que se pediu a diferentes utilizadores que durante um dado período de tempo, simulassem sinais de cansaço reconhecíveis pelo sistema, de acordo com as regras definidas na tabela 3 da secção 4.7 Módulo de Tomada de Decisão, baseadas no estado dos olhos e da boca.

Considerou-se que os valores reais positivos e negativos da Matriz de Confusão correspondem, respetivamente, a momentos no tempo em que se observou que o individuo exibia ou não sinais de cansaço de acordo com a referida tabela. Da mesma maneira, os valores de deteção positivos e negativos da Matriz de Confusão, correspondem ao resultado do sistema, nos mesmos momentos no tempo, de acordo com as mesmas regras da tabela referida.

Embora a saída do sistema corresponda a um alarme de cansaço de três níveis (verde, amarelo e vermelho), para efeitos de quantificação dos valores da matriz de confusão, considerou-se apenas dois valores possíveis. O valor positivo correspondente à existência de sinais de cansaço (alarme de nível vermelho) e o valor negativo correspondente à não existência de sinais de cansaço (correspondente ao alarme de nível verde ou amarelo).

Foram realizados dois testes para testar o desempenho do sistema desenvolvido. O Teste 1 em que o sistema foi testado num cenário de condução real, sem controlo das condições ambientais, com o *smartphone* montado num automóvel, e o Teste 2 onde o

sistema foi testado em cenário de laboratório, com controlo das condições ambientais. A descrição de como foram realizados os testes e os respetivos resultados são apresentados nas secções a seguir.

5.3. Teste 1 – Deteção de Cansaço em Cenário de Condução

Para a deteção de cansaço em cenário de condução real, o primeiro passo foi colocar o *smartphone* no interior do veículo. A melhor localização, após vários testes, para posicionar o *smartphone* será colocar o suporte do aparelho na parte inferior ao espelho retrovisor e direcionar o *smartphone* para a face do condutor a uma distância máxima permitida para a deteção de 1,12m. A figura 32 mostra a localização do *smartphone* dentro de um veículo.

Figura 32. *Localização de um smartphone em cenário de condução*

Para testar o desempenho do sistema do Teste 1 em cenário de condução foram escolhidos três percursos diferentes. A estrada nacional que se encontra próxima à saída da cidade da Guarda na localidade do Alvendre é descrita como uma estrada com algum trânsito, tem um piso irregular e é favorecida com algumas curvas fechadas e contracurvas. O IP5 - Itinerário Principal das Beiras Litoral e Alta, possui um piso irregular e em mau estado de conservação, com alguma movimentação de veículos e

curvas pouco acentuadas. O IP 2 - Itinerário Principal do Interior, é um percurso que faz a ligação entre Bragança e Guarda. É uma estrada recente, praticamente não possui curvas, tem pouco trânsito e o piso do lado direito já se apresenta com remendos por causa dos buracos.

Para estudar melhor o seu comportamento foi feito uma simulação no percurso das 3 estradas, onde o condutor era do sexo feminino, estatura normal e um tom de pele moreno. O utilizador tinha um guião para saber em que momento no tempo devia simular os sinais de cansaço que é descrito na tabela que se segue. Após várias tentativas de simulação, considerou-se que o parâmetro do intervalo de tempo para considerar o estado dos olhos como sendo estado Fechado ou da boca como sendo estado Aberto (como referido na secção 4.7 Módulo de Tomada de Decisão) era de 3 segundos.

Tabela 5. Guião da simulação do Teste 1

Tempo (segundos)	Estado do Olhos	Estado da Boca	Cansaço
80	Piscar normal	Fechada	Negativo
160	Fechados por 3 segundos	Fechada	Negativo
240	Piscar normal	Aberta por 3 segundos	Negativo
320	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
400	Fechados por 3 segundos	Fechada	Negativo
450	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
530	Piscar normal	Fechada	Negativo
600	Piscar normal	Aberta por 3 segundos	Negativo
680	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
760	Fechados por 3 segundos	Fechada	Negativo
800	Piscar normal	Fechada	Negativo
900	Piscar normal	Aberta por 3 segundos	Negativo
980	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
1060	Piscar normal	Fechada	Negativo
1200	Fechados por 3 segundos	Aberta por 3 segundos	Positivo

De seguida irá ser apresentada a Matriz de Confusão para esta simulação, como é possível ver na tabela 6.

Tabela 6. Matriz de Confusão da simulação do Teste 1

		Valor da Deteção	
		Negativo	**Positivo**
Valor Real	**Negativo**	9	1
	Positivo	4	1

5.3.1. Resultados

Na tabela 7 podemos encontrar diversas medidas calculadas referentes ao desempenho da simulação do Teste 1 com base nos valores da Matriz de Confusão da tabela 6. As medidas escolhidas foram: Acurácia, Sensibilidade, Especificidade e a Eficiência. Consideraram-se estas medidas pelo facto de serem as das mais usadas na literatura.

Tabela 7. Medidas calculadas para a simulação do Teste 1

Medidas	Simulação
Acurácia	67%
Sensibilidade	20%
Especificidade	90%
Eficiência	55%

De uma forma geral, podemos ver que os valores diferem um pouco uns dos outros. Em relação à Sensibilidade (20%) denota-se uma grande discrepância em relação à Especificidade (90%), ou seja, o sistema tem dificuldades em conseguir identificar corretamente o cansaço do condutor. Para o sistema é mais fácil reconhecer quando não existe cansaço porque, tal como foi referido anteriormente, para efeitos de quantificação dos valores da matriz de confusão, o valor negativo correspondente à não existência de sinais de cansaço (correspondente ao alarme de nível verde ou amarelo). Existem mais hipóteses para quantificar a proporção de verdadeiros negativos.

As diferenças dos valores calculados das medidas refletem-se dos vários obstáculos que foram encontrados durante a simulação do Teste 1, nomeadamente, problemas de velocidade do sistema para conseguir manter a deteção da face com a trepidação do carro devido ao piso ser irregular. Isso deve-se às capacidades de processamento do *smartphone* que são limitadas para este tipo de aplicação, e também aos próprios métodos de Visão Computacional implementados no sistema que eventualmente poderão ser melhorados e otimizados. Outro fator a ter em conta tem a ver com problemas de luminosidade inconstante que afeta tanto a deteção da face como a deteção dos sinais de cansaço. Os testes que foram realizados ao início da tarde, onde o sol incidia sobre a face do condutor originavam imagens muito claras o que tornava difícil a deteção da face bem como a deteção dos sinais de cansaço. Uma solução para esse problema seria o sistema estar preparado automaticamente para quando esse tipo de problema acontecer, ou seja, o sistema devia ser capaz de escurecer a imagem nas zonas relevantes. Ao anoitecer deparamo-nos que se durante o percurso houvesse iluminação o sistema conseguia detetar a face mas não era totalmente fiável, caso não houvesse iluminação o sistema era incapaz de detetar. Para resolver este problema seria o *smartphone* possuir uma câmara com infravermelho. De acordo com os problemas encontrados verificou-se que não havia necessidade de realizar mais testes em estrada nem ser testado com diferentes utilizadores.

5.4. Teste 2 – Deteção de Cansaço em Cenário de Laboratório

A necessidade de haver um Teste 2 para a deteção de cansaço em cenário de laboratório, deve-se ao facto de que o sistema ainda tem algumas limitações para puder ser testado em cenários reais de condução como concluído na seção anterior.

Para a deteção de cansaço em cenário de laboratório o primeiro passo foi encontrar a melhor localização para o *smartphone*. Para manter as mesmas condições que foram realizadas no Teste 1 e após vários testes verificou-se que o ideal seria o *smartphone* estar cerca de 30 cm de altura sobre uma secretária, a uma distância de 40 cm em relação à face do utilizador.

O Teste 2 foi testado em cinco utilizadores, dois do sexo masculino e os restantes do sexo feminino. O utilizador 1 é uma jovem de 18 anos, cor de pele clara, com olhos bem salientes e com uma estatura alta. O utilizador 2 é um jovem de 20 anos, cor de pele normal, olhos rasgados e pouco salientes e com uma estatura média. O utilizador 3 é uma senhora de 35 anos, cor de pele com uma tonalidade mais escura que os restantes utilizadores, olhos bem salientes e com uma estatura normal. O utilizador 4 é um senhor de 38 anos, cor de pele normal, com olhos pouco salientes, com uma estatura baixa e alguma penugem na face. Por último, o utilizador 5 é uma senhora de 60 anos, cor de pele clara, olhos salientes e com uma estatura baixa. Os 5 utilizadores descritos podem ser vistos na figura seguinte.

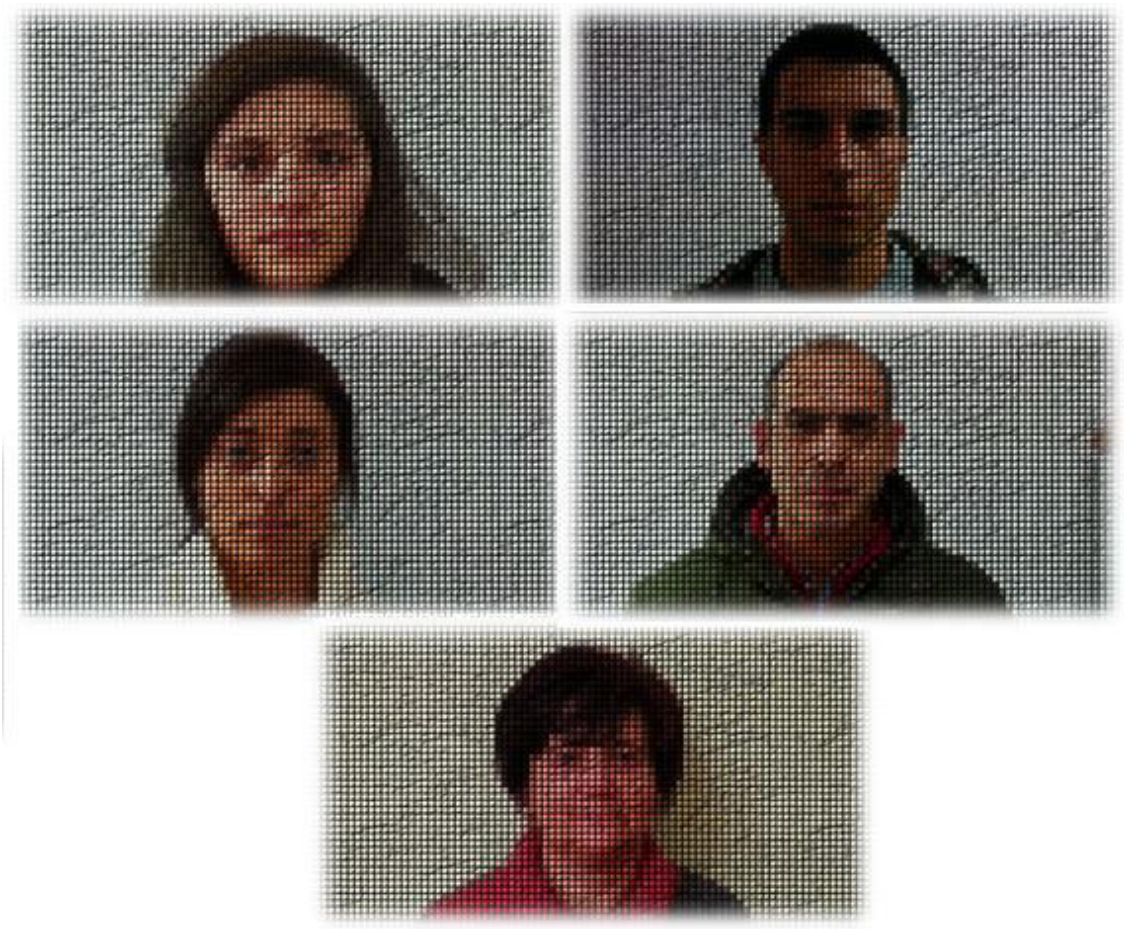


Figura 33. Utilizadores usados no Teste 2

É de referir ainda, que nas simulações que foram feitas no Teste 2, as condições ambientais foram mantidas constantes, foi utilizada a mesma iluminação e a mesma distância entre utilizador e telemóvel.

Cada utilizador tinha um guião, tal como no Teste 1, para saber em que momento no tempo devia simular os sinais de cansaço. Para não se tornar tão exaustivo para o utilizador, em vez de uma simulação, como no teste anterior, foram realizadas três simulações para cada um. Tal como no Teste 1, após várias tentativas de simulação, considerou-se que o melhor tempo para os olhos estarem fechados ou a boca estar aberta era de 3 segundos. Descrevem-se a seguir as três simulações realizadas e os respetivos resultados.

5.4.1. Simulação 1

O guião da Simulação 1 é apresentado na tabela que se segue.

Tabela 8. Guião da Simulação 1

Tempo (segundos)	Estado do Olhos	Estado da Boca	Cansaço
15	Piscar normal	Fechada	Negativo
40	Fechados por 3 segundos	Fechada	Negativo
60	Abertos por 3 segundos	Aberta por 3 segundos	Negativo
75	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
90	Fechados por 3 segundos	Fechada	Negativo
105	Fechados por 3 segundos	Aberta por 3 segundos	Positivo

De seguida irá ser apresentada a Matriz de Confusão para a Simulação 1 com os valores de todos os utilizadores, como é possível ver na tabela 9.

Tabela 9. Matriz de Confusão da Simulação 1 do Teste 2

		Valor da Deteção	
		Negativo	**Positivo**
Valor Real	**Negativo**	19	1
	Positivo	3	8

5.4.2. Simulação 2

O guião da Simulação 2 é apresentado na tabela que se segue.

Tabela 10. Guião da Simulação 2

Tempo (segundos)	Estado do Olhos	Estado da Boca	Cansaço
15	Piscar normal	Aberta por 3 segundos	Negativo
30	Piscar normal	Fechada	Negativo
40	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
60	Fechados por 3 segundos	Fechada	Negativo
75	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
90	Piscar normal	Aberta por 3 segundos	Negativo
105	Fechados por 3 segundos	Aberta por 3 segundos	Positivo

De seguida irá ser apresentada a Matriz de Confusão para a Simulação 2 com os valores de todos os utilizadores, como é possível ver na tabela 11.

Tabela 11. Matriz de Confusão da Simulação 2 do Teste 2

		Valor da Deteção	
		Negativo	**Positivo**
Valor Real	**Negativo**	19	1
	Positivo	1	14

5.4.3. Simulação 3

O guião da Simulação 3 é apresentado na tabela que se segue.

Tabela 12. Guião da Simulação 3

Tempo (segundos)	Estado do Olhos	Estado da Boca	Cansaço
15	Fechados por 3 segundos	Fechada	Negativo
30	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
40	Piscar normal	Aberta por 3 segundos	Negativo
60	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
70	Piscar normal	Fechada	Negativo
85	Fechados por 3 segundos	Aberta por 3 segundos	Positivo
95	Piscar normal	Aberta por 3 segundos	Negativo
110	Fechados por 3 segundos	Fechada	Negativo
130	Fechados por 3 segundos	Aberta por 3 segundos	Positivo

De seguida irá ser apresentada a Matriz de Confusão para a Simulação 3 com os valores de todos os utilizadores, como é possível ver na tabela 13.

Tabela 13. Matriz de Confusão da Simulação 3 do Teste 2

		Valor da Deteção	
		Negativo	**Positivo**
Valor Real	**Negativo**	22	3
	Positivo	10	10

5.4.4. Resultados das Simulações

Na tabela 14 podemos encontrar diversas medidas calculadas referentes ao desempenho das três simulações do Teste 2 com base nos valores da Matriz de Confusão. Tal como no Teste 1, as medidas escolhidas foram: Acurácia, Sensibilidade, Especificidade e a Eficiência.

Tabela 14. Medidas calculadas para as três simulações do Teste 2

Medidas	Simulação 1	Simulação 2	Simulação 3	Média
Acurácia	87%	94%	68%	83%
Sensibilidade	72%	93%	50%	71%
Especificidade	95%	95%	88%	92%
Eficiência	83%	94%	69%	82%

A avaliação global do sistema em função dos valores totais das médias é bastante satisfatório, pois o valor mais baixo é de 71% na medida da Sensibilidade e o mais elevado é de 83% na medida da Acurácia. Em termos de valores totais das médias temos uma avaliação de 82%. Através da tabela podemos ver que o sistema nas três simulações teve um comportamento semelhante, apesar dos valores da Simulação 3 serem mais baixos em relação às outras duas, mas isso explica-se pelo facto de a simulação possuir mais situações. Verifica-se que a especificidade e a eficiência das duas primeiras simulações são muito semelhantes, contudo a Simulação 2 foi a que teve melhores resultados. Na

Simulação 3 podemos reparar que o valor da Eficiência destacou-se um pouco do valor da Especificidade, isso resulta que nesta simulação foi mais difícil detetar o cansaço positivo, isso pode resultar por haver mais situações de deteção de cansaço positivo em relação às outras simulações. De uma forma geral, podemos concluir que nas três simulações os valores mais baixos foram na medida da Sensibilidade, o que significa que o sistema tem mais dificuldade em detetar o cansaço positivo. Isso pode resultar pelo facto que na deteção de cansaço positivo, quando o utilizador boceja provoca alguma movimentação na deteção da face e o sistema não tem velocidade suficiente para manter a deteção da face e afeta a deteção dos sinais de cansaço. Isso deve-se às capacidades de processamento do *smartphone* que são limitadas para este tipo de aplicação.

6. Conclusão

Este trabalho teve como finalidade o desenvolvimento de um sistema de deteção de cansaço para assistência a condutores, com base em Visão Computacional e tecnologias de *smartphones*. O cansaço corresponde a um estado físico e emocional que condiciona o desempenho e como consequência uma diminuição das capacidades percetivas, cognitivas e motoras independentemente da atividade a que nos estejamos a referir. Integrar os recursos de Visão Computacional foi o objetivo principal deste projeto, de modo que os seus métodos fossem confiáveis e de baixa complexidade computacional, com a finalidade de desenvolver uma aplicação para *smartphones* que fosse capaz de analisar a expressão facial e gerar um sinal de alarme quando fossem detetados sinais de cansaço através dos principais sintomas como por exemplo, os olhos fechados e boca aberta.

A deteção de faces é uma tecnologia do computador que determina os locais e tamanhos de rostos humanos em imagens digitais. Para isso, foram testados vários métodos e constatou-se que o sistema que foi mais eficaz para a deteção da face foi o método que o *Android* possui a partir da versão 4.0 (API Nível 14) de APIs para a identificação de rostos e cálculo de ajustes de imagem usando tecnologia de reconhecimento de face. A escolha deste método deveu-se ao facto, que em relação aos outros métodos testados, o tamanho da aplicação era apenas de 464 KB, a taxa de *frames* por segundo é compreendida entre 16 a 20 valores e a deteção em condições de movimento era eficaz.

Como os sinais de cansaço são detetados através dos olhos e da boca, houve a necessidade de fazer a extração desses atributos. Estes não foram procurados em toda a imagem, optou-se por extrair uma ROI dos olhos e da boca através do retângulo da face previamente detetado. A localização dos olhos e da boca foi encontrada com base nas tradicionais regras de proporção de face humana.

A classificação do estado dos olhos e da boca foi baseada no método *Template Matching*, pelo facto de ter sido o sistema que obteve melhores resultados. A deteção do estado dos olhos, da boca e a análise de duração desse evento são baseados na observação

e comparação dos valores de coeficiente de correlação obtidos durante o rastreamento da ROI dos olhos e da boca, onde o valor 1 para o coeficiente de correlação corresponde à medida de similaridade máxima e o valor 0 indica que não há.

Em função da classificação do estado dos olhos e da boca foi descrito um conjunto de regras para classificar o cansaço do condutor e foram criados três níveis de alarme, (verde, amarelo e vermelho), que permite informar o utilizador acerca da classificação de cansaço. O nível de alarme de cor verde transmite que não que não existe qualquer perigo para o condutor, ou seja o cansaço é Negativo; Amarelo corresponde a um alerta, o cansaço é Negativo mas já começa a ter indícios de cansaço e o vermelho diz que o cansaço é realmente Positivo. Após vários testes, o sistema considerou que quando os olhos permanecerem fechados e a boca manter-se aberta por mais de 2 segundos é emitido o alerta de cansaço com um sinal sonoro.

Para avaliar o comportamento do sistema, foi feita uma simulação em cenário de condução e verificaram-se algumas limitações, nomeadamente, problemas de velocidade do sistema para conseguir manter a deteção da face com a trepidação do carro devido ao piso ser irregular e problemas de luminosidade inconstante que afeta tanto a deteção da face como a deteção dos sinais de cansaço. Devido às limitações que foram encontradas em cenários reais de condução houve necessidade de realizar outro tipo de testes em cenários de laboratório, em que as condições ambientais foram mantidas constantes, foi utilizada a mesma iluminação e a mesma distância entre utilizador e telemóvel. Foram efetuadas três simulações para cada utilizador e verificou-se que os resultados obtidos neste teste foram mais satisfatórios em relação ao teste anterior. A nível de valores totais das médias o valor mais baixo é de 71% na medida da Sensibilidade e o mais elevado é de 83% na medida da Acurácia. Podemos ver que o sistema nas três simulações teve um comportamento semelhante, contudo a simulação 2 foi a que teve melhores resultados. Na Simulação 3 o valor da Eficiência destacou-se um pouco do valor da Especificidade, isso resulta que nesta simulação foi mais difícil detetar o cansaço positivo. De uma forma geral, podemos concluir que nas três simulações os valores mais baixos foram na medida da Sensibilidade, isso resulta porque o sistema tem mais dificuldade em detetar o cansaço positivo. Isso pode ser justificado pelo facto de que na deteção de cansaço positivo, quando o utilizador boceja provoca alguma movimentação na deteção da face e o sistema

não tem velocidade suficiente para manter a deteção da face e afeta a deteção dos sinais de cansaço.

Notoriamente, ainda são grandes as possibilidades de amadurecimento desta solução, a primeira seria a utilização de uma câmara com infravermelho, onde problemas relacionados à mudança de luminosidade ambiente seriam sanados. Outro detalhe importante a considerar é a melhoria no desempenho dos processadores utilizados em *smartphones*, pelo facto de ainda possuírem pouca capacidade de processamento o que traduz o ajuste da deteção da face bem como a deteção dos sinais de cansaço lenta.

Embora o sistema implementado tenha as suas limitações, o estudo realizado e os resultados obtidos, mostram que é possível desenvolver sistemas avançados de apoio à condução com base em tecnologias de *smartphone*.

É de salientar que foi deveras importante a realização do projeto para a compreensão e aprofundamento do tema, visto que permitiu desenvolver e aperfeiçoar diversas técnicas de investigação, seleção, organização e comunicação de informação veiculada. Gostaria também de reforçar a ideia do quanto foi fundamental para a minha formação e enriquecimento pessoal e profissional, uma vez que permitiu desenvolver capacidades, aptidões e adquirir novos conhecimentos e competências.

Referências Bibliográficas

Android Developers, (2013), "Developers". Acedido em 10 de março de 2013, em: http://developer.android.com

Android Developers, (2015), "Number of devices running a given version of the Android platform". Acedido em 15 de janeiro de 2015, em: http://developer.android.com/about/dashboards/index.html

Associação Portuguesa do Sono (2012), "Índice de condutores que adormeceram ao volante". Acedido em 15 de setembro de 2012, em: www.apsono.com

Devi, M.S., Bajaj, P.R. (2008), "Driver Fatigue Detection Using Mouth and Yawning Analysis". IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.6.

Devi, M.S., Bajaj, P.R. (2010), "Fuzzy Based Driver Fatigue Detection". Research Associate B.D.College of Engineering, Sevagram, Wardha, India, Electronics Dept G.H.Raisoni College of Engineering Nagpur, India.

Devi, M.S., Bajaj, P.R. (2013), "Driver Fatigue Detection Based on Eye Tracking", First International Conference on Emerging Trends in Engineering and Technology

Ford (2013), "Lane Keeping System". Acedido em 22 de novembro de 2013, em: http://www.ford.pt/Automoveis-Veiculos-Comerciais-ligeiros/Transit-2013/SegurancaeProteccao

Heisele, B., Ho, P., Poggio, T. (2001), "Face Recognition with Support Vector Machines: Global versus Component-based Approach". Acedido em 10 de março de 2013, em: http://cbcl.mit.edu/cbcl/publications/ps/iccv2001.pdf

Horng, W., B., e Chen, C., Y. (2008), "A Real-Time Driver Fatigue Detection System Based on Eye Tracking and Dynamic Template Matching". Tamkang Journal of Science and Engineering, Vol. 11, No. 1, pp. 65-72.

Instituto da Mobilidade e dos Transportes, I.P (2012), "Manual do Ensino da Condução". Acedido em 17 de outubro de 2012, em: http://www.imtt.pt/sites/IMTT/Portugues/EnsinoConducao/ManuaisEnsinoCon

ducao/Documents/Fichas/FT_SistemasApoioConducao.pdf

Jamundá, T. (2013), "Reconhecimento de Formas: A Transformada de Hough". Acedido em 28 de fevereiro de 2013, em: http://www.inf.ufsc.br/~visao/2000/Hough/index.html

Java interface to OpenCV and more, (2013), "Convert YUV to RGB". Acedido em 14 de março de 2013, em: https://code.google.com/p/javacv/issues/detail?id=327

Krolak, A., Strumillo, P. (2010), "Eye-Blink Detection System for Human-Computer Interaction". Unitech2010 The International Conference on Universal Technologies Oslo University College, Oslo, Norway, May 19-20.

Ma, H., Yang, Z., Song, y., Jia, P. (2008), "A Fast Method for Monitoring Driver Fatigue Using Monocular Camera". Proceedings of the 11th Joint Conference on Information Sciences.

Medeiros, A. (2009), "Correlação Digital De Imagens". Tese de Licenciatura em Computação. Universidade Do Estado De Mato Grosso Campus Universitário Vale Do Teles Pires Departamento De Licenciatura Em Computação.

Mercedes-Benz (2013), "Attention Assist ". Acedido em 25 de novembro de 2013, em: http://www.mercedes-benz-intelligent-drive.com/com/en/1_driver-assistance-and-safety/16_attention-assist

Milano, D. e Honorato, L.B. (2010), "Visão Computacional". Acedido em 20 de outubro 2003, em: http://www.ft.unicamp.br/liag/wp/monografias/monografias/2010_IA_FT_UNICAMP_visaoComputacional.pdf

Opel (2013), "Opel Eye Sistema de Deteção". Acedido em 20 de dezembro de 2013, em: http://media.opel.com/media/intl/en/opel/vehicles/driver_assistance_systems/2011.html

Queirós, Ricardo (2008), "Programação para Dispositivos Móveis em Windows Mobile *6*" 2008º edn (FCA - Editors Informática).

Queiroz, K. (2011), "Sistema Baseado em Vídeo para Deteção de Sonolência em Motoristas". Tese Mestrado em Engenharia Elétrica. Universidade de Brasília Faculdade de Tecnologia, Brasília.

Reis, Catarina, Catarina Silva, Nuno Fonseca, Vitor Carreira, e Luis Marcelino (2012), *Desenvolvimento em iOS iPhone, iPad e iPod Touch*, 2012º edn (FCA - Editora Informática).

Ren, F., Huang, J., Terauchi, M., Jiang, R., Klette, R. (2012), "Lane Detection on the iPhone". Acedido em em 20 de dezembro de 2012, em: http://www.mi.auckland.ac.nz/tech-reports/MItech-TR-43.pdf

Rodrigues, J. (2013), "Segmentação, seguimento e avaliação automática de bactérias em imagens de microscópio". Tese de Mestrado em Engenharia Biomédica – Faculdade de Ciências e Tecnologia Universidade Nova de Lisboa.

Souza, C. (2009), "Análise de Poder Discriminativo Através de Curvas ROC". Acedido em 20 de julho de 2014, em http://crsouza.blogspot.pt/2009/07/analise-de-poder-discriminativo-atraves.html

Underlying technique of Android's FaceDetector (2010), "Technique of androids face detector". Acedido em 06 de junho de 2013, em http://stackoverflow.com/questions/3353696/underlying-technique-of-androids-facedetector

Viola, P., e Jones, M., (2001), "Rapid object detection using a boosted cascade of simple features". In CVPR 2001 – IEEE Conference on Computer Vision and Pattern Recognition.

Volkswagen (2013), "Driver Alert System". Acedido em 2 de novembro de 2013, em: http://www.volkswagen.pt/pt/models/golf-variant/

YUV420SP Grayscale, (2014). "RGB to Grayscale". Acedido em 20 de julho de 2014, em https://github.com/dps/nnrccar/blob/master/android/src/org/davidsingleton/nnrccar/NNRCCarActivity.java

Zhao, W.; Chellappa, R.; Phillips, P.J. e Rosenfeld, A. (2003), "Face recognition: A literature survey". ACM Computing Surveys, 35(4):433-440.