UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO

SISTEMA DE DETECÇÃO DE SONOLÊNCIA, POR MEIO DE VISÃO COMPUTACIONAL

AUTOR: Rafael Barichello Ferrassini

ORIENTADOR: Evandro Luís Linhares Rodrigues

São Carlos
2014

RAFAEL BARICHELLO FERRASSINI

SISTEMA DE DETECÇÃO DE SONOLÊNCIA, POR MEIO DE VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em Eletrônica

ORIENTADOR: Evandro Luís Linhares Rodrigues

São Carlos

2014

Ferrassini, Rafael Barichello

F378s Sistema de detecção de sonolência, por meio de visão computacional / Rafael Barichello Ferrassini; orientador Evandro Luís Linhari Rodrigues. São Carlos, 2014.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2014.

1. Visão Computacional. 2. Odroid U2. 3. Open CV. 4. Biometria. 5. Reconhecimento de objetos. 6. Inteligência Artificial. I. Título.

FOLHA DE APROVAÇÃO

Nome: Rafael Barichello Ferrassini

Título: "Sistema de detecção de sonolência, por meio de visão computacional"

Trabalho de Conclusão de Curso defendido e aprovado em 28/11/2014,

com NOTA 10,0 (Dez, zero), pela Comissão Julgadora:

Prof. Associado Evandro Luís Linhari Rodrigues - (Orientador - SEL/EESC/USP)

Prof. Associado Adilson Gonzaga - (SEL/EESC/USP)

Prof. Dr. Marcelo Andrade da Costa Vieira - (SEL/EESC/USP)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

Agradecimentos

Primeiramente, a Deus, fonte da minha fé e determinação para superar os obstáculos encontrados. Ao meu avô Netto, meu maior ídolo e meu maior fã, pelo incentivo, amor, carinho e por ter sido, em toda sua vida, modelo de homem que busco me tornar. Aos meus pais, por me ensinarem que o conhecimento conquistado através dos estudos é a maior riqueza que qualquer homem pode adquirir. Obrigado pelo carinho e suporte para que eu atingisse todos os meus objetivos. À minha irmã Luísa, pela paciência, apoio e ajuda nos momentos mais inesperados.

Ao meu irmão Gustavo, pelos conselhos, aulas de programação, críticas e sugestões que foram essências para o desenvolvimento deste trabalho. À minha namorada Verena, pelo incentivo, sugestões, críticas, conselhos e carinho dedicados, sendo minha grande companheira. Aos funcionários e professores do Departamento de Engenharia Elétrica e Computação pela sua dedicação, proporcionando o ensino de qualidade aos jovens ingressantes nos cursos oferecidos pelo departamento todos os anos. Ao meu orientador, Professor Evandro, pelos ensinamentos, conselhos e “puxões de orelha” durante o desenvolvimento deste projeto, me lembrando sempre que a dedicação, a este projeto, seria de fundamental importância em minha vida pessoal, profissional e acadêmica. Aos meus colegas e amigos da Universidade de São Paulo, pelas alegrias, tristezas, sucessos e fracassos que compartilhamos nesses anos que estivemos unidos em São Carlos.

A todos vocês, “Obrigado, Obrigado e Obrigado!”.

Resumo

A sonolência é uma das principais causas de acidentes automobilísticos. Sendo assim, um sistema que identifique o estado de sonolência de um motorista em um veículo poderia prevenir inúmeros acidentes ao redor do mundo. Técnicas poderosas, como as existentes no ramo da visão computacional, podem ser empregadas para o desenvolvimento de tal sistema, tendo por base estudos da fisiologia do sono que provam que existe uma relação entre o tempo de uma piscada com a sonolência de um ser humano. Sendo assim, utilizando-se de um poderoso computador de dimensões reduzidas, a Odroid U2, foi desenvolvido na linguagem C++, um algoritmo, baseado no algoritmo de Viola e Jones, capaz de reconhecer se o olho do usuário está aberto ou fechado em um ambiente com iluminação controlada. Os testes realizados com 6 indivíduos indicaram uma taxa de acerto maior que 95% na determinação do estado dos olhos do usuário.

Palavras chave: Visão Computacional; Odroid U2; OpenCV; Biometria; Reconhecimento de objetos; Inteligência Artificial.

Abstract

Sleepiness is a major cause of traffic accidents. Thus, a system that identifies the state of drowsiness of a driver in a vehicle could prevent many accidents around the world. Powerful techniques, such as those existing in the computer vision field, can be used to develop such a system, based on sleep physiology studies demonstrating that there is a relationship between the time a blink with the drowsiness of a human being. Thus, using a powerful computer with reduced dimensions, the ODROID U2 was developed in C ++, an algorithm based on the Viola and Jones algorithm can recognize whether the user's eye is opened or closed in an environment with controlled lighting. Tests conducted with 6 subjects showed an accuracy rate above 95% in determining the user's eye state.

Key words: Computer Vision; Odroid U2; OpenCV; Biometrics; Object recognition; Artificial intelligence;

Lista de figuras

Lista de tabelas

Lista de Gráficos

Sumário

1. Introdução

Estudos realizados ao redor do mundo (TOBIAS, 2008) mostram que entre 26% e 32% dos acidentes de trânsito são provocados por motoristas que dormem na direção. Eles são responsáveis por índices entre 17% e 19% das mortes no local do acidente. Os dados foram obtidos em pesquisa da Universidade de Gênova, na Itália, já que no Brasil não há dados oficiais sobre o assunto. “Motoristas com distúrbios do sono correm duas a três vezes mais riscos de se envolver em acidentes.”, afirma Mello.

Dados da *National Highway Traffic Safety Administration* indicam que a cada ano ocorrem, nos Estados Unidos, cerca de 100.000 acidentes, sendo 1.550 fatais, em decorrência de motoristas que dormem ao volante. Pesquisas da Austrália, Inglaterra, Finlândia, além de outros países da Europa apontam que o sono ao volante como causa de 10% a 30% dos acidentes.

O risco de acidentes aumenta entre 12h30 e 14h e das 22h às 6 da manhã, quando temos uma diminuição na temperatura corporal, sendo que o período crítico fica entre 3h30 e 5h30. “Com a temperatura corporal baixa, começamos a liberar melatonina, que induz ao sono.”, diz Mello.

No Brasil, como dito anteriormente, faltam estatísticas oficiais, mas alguns dados dão dimensão do problema. Especialistas dizem que cerca de 35% dos condutores têm sonolência excessiva, e nas grandes cidades, 10% da população sofre de insônia crônica. Além disso, entre 5 e 8 milhões de pessoas têm apneia obstrutiva, (dificuldade de respirar enquanto se dorme), que prejudica o descanso, a qualidade de vida e causa de sonolência diurna.

A Clínica do Sono, com sede em Nova Lima – MG (FUNDASONO: FUNDAÇÃO DO SONO, 2014), realizou uma pesquisa com 639 motoristas de carros de passeio no ano de 1997. O objetivo era o de conhecer o grau de incidência de sonolência em motoristas não profissionais no estado. Os resultados, que ainda podem ser considerados atuais, constataram que:

- 55% dos motoristas entrevistados disseram que nunca se envolveram em acidentes de trânsito
- 38% já haviam se envolvido, mas alegaram não ter dormido ao volante.
- 7% admitiram ter dormido ao volante, provocando acidentes.

1.1. Relevância

Atentos a estes aspectos, pesquisadores, ao redor do mundo, tem projetado e lançado diversas soluções para minimizar este tipo de acidente, alertando o motorista do veículo que ele está em estado de sonolência antes de qualquer incidente.

A indústria já lançou produtos para atender essa demanda. A Volkswagem, por exemplo, tem em alguns de seus modelos comercializados, a função de detecção de fadiga, que alerta o condutor para uma perda de concentração. Ele analisa parâmetros como o comportamento de direção, uso do pedal e aceleração lateral. Se o sistema achar que a concentração do motorista está diminuindo, ele proporciona um aviso visual e sonoro. (VOLKSWAGEM, 2014)

Já a Ford desenvolveu o Lane Keeping System (CARTOLA, 2014), que monitora a trajetória do condutor na pista a fim de detectar sinais de fadiga ou sonolência, como mostrado na Figura 1. O dispositivo funciona com base em informações captadas por uma câmera instalada no para-brisa do veículo, atrás do espelho retrovisor. A câmera é capaz de detectar as linhas divisórias da pista e estabelecer onde o carro deveria estar posicionado.

Figura 1 -Ilustração explicativa sobre o sistema Lane Keeping Assist do fabricante Ford. Disponível em: < http://www.beachautomotive.com/blog/wp-content/uploads/2014/07/Blog-1-pic-1-Beach-Auto-Top-safety-checks.jpg> Acesso: 2 de nov. 2014.

Dentro das universidades existem pesquisas sobre o mesmo tema. Vários destes trabalhos(HORNG, 2004), (JI, YANG, 2002), (JAYANTHI, BOMMY, 2012), foram publicados inclusive pelo IEEE, e em sua maioria baseiam-se na utilização de métodos de visão computacional para sensoriamento de estado de fadiga de um motorista de um veículo automotivo.

Verifica-se que os trabalhos citados direcionam-se no sentido de reconhecer em imagens de faces humanas padrões que podem ser interpretados como sonolência do motorista.

Figura 2 - Exemplo de reconhecimento de olhos abertos e fechados com algoritmos de visão computacional. (CASSINELLI; PERRIN, 2014)

Destaca-se o uso da visão computacional para obtenção de resultados, visto que trata-se de uma técnica não intrusiva, funcionando a partir uma câmera posicionada de forma a obter a face do usuário.

1.2. Objetivo

Entendendo a problemática exposta, este trabalho tem por objetivo propor um sistema (hardware e software) baseado no processamento de imagens, capaz de detectar a região da face e dos olhos do usuário pelo algoritmo de Viola e Jones, e de determinar se o olho deste usuário está aberto ou fechado utilizando-se do Thresholding. Determinado o estado dos olhos, o sistema deve ser capaz de interpretar essa informação para reconhecer o estado de sonolência do usuário.
Entende-se simplificadamente que o sistema proposto seguiria o fluxo representado na Figura

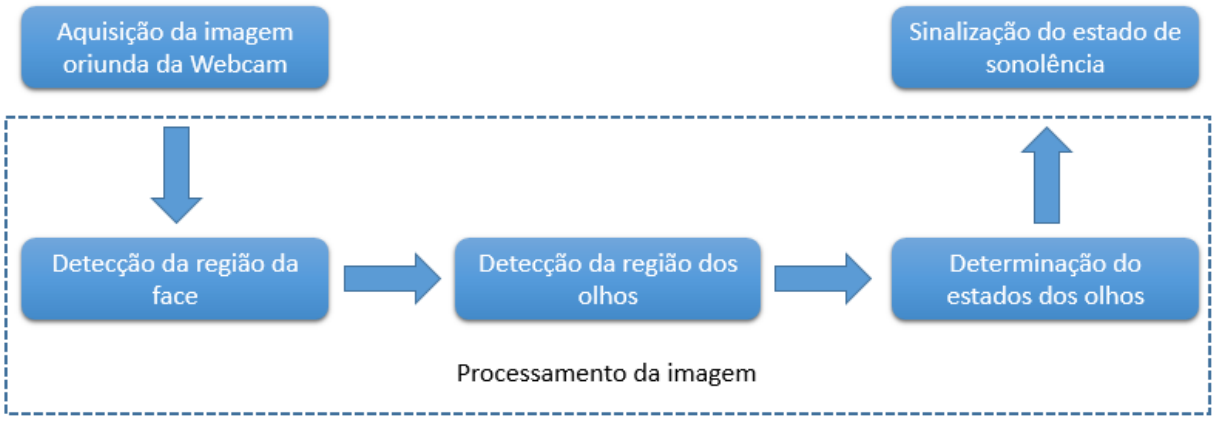

Figura 3: Fluxograma resumido do sistema proposto

Busca-se que o sistema proporcione um resultado confiável, utilizando-se de ferramentas de baixo-custo.

2. Embasamento teórico

Neste capítulo serão apresentados os conceitos que serão chave para o desenvolvimento de todo o projeto.

2.1. Visão Computacional

A área de Visão Computacional caracteriza-se primordialmente pela utilização de Imagens Digitais associadas a técnicas de reconhecimento de padrões. Além disso, o estudo de métodos cognitivos, processos biológicos, processos físicos e estatísticos têm gerado soluções importantes para a área de Visão Computacional.

Pode-se entender que o funcionamento padrão de um sistema baseado em visão computacional respeita o fluxo representado na Figura 4.

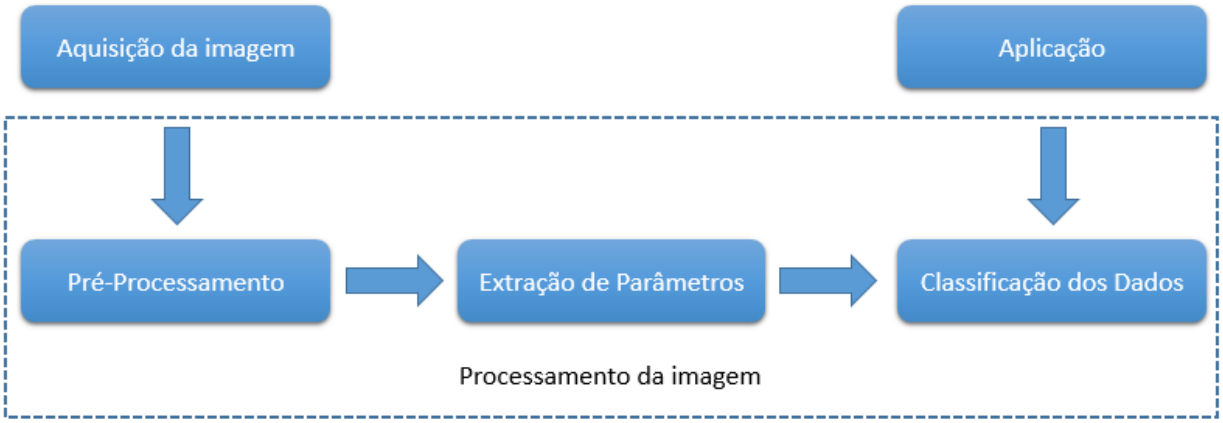

Figura 4: Fluxograma de um sistema genérico de visão computacional

Uma das mais difundidas aplicações da visão computacional é a Medicina, ou o processamento de imagens médicas. Tal área é caracterizada pela extração de informação de imagens para realizar diagnósticos sobre os pacientes. Fontes de imagens incluem imagens de microscopia, de radiografia, de angioplastia, de ultrassonografia, de tomografia e de ressonância magnética, como exemplificado na Figura 5.

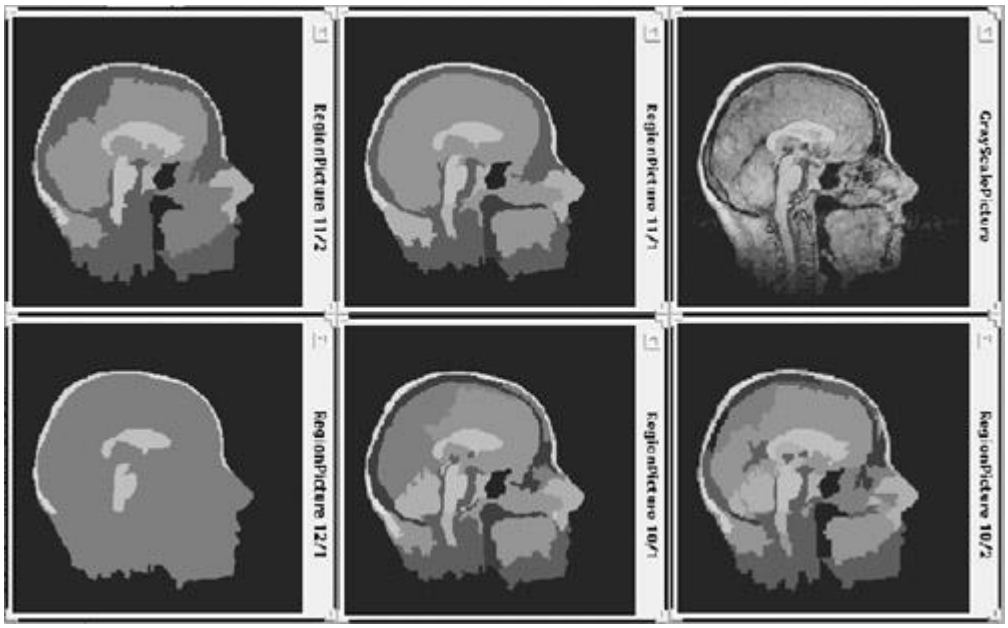

Figura 5 - Imagem de ressonância magnética tipo T1 em visão sagital de crânio humano (imagem superior à direita) e diversas segmentações realizadas utilizando-se o funcional de Mumford & Shah com diferentes parâmetros. (SEMINÁRIO INTRODUÇÃO À VISÃO COMPUTACIONAL, 2014).

Outra aplicação bastante difundida é na indústria, a Figura 6 ilustra a utilização da visão computacional na indústria madeireira. A informação obtida auxilia processos como a inspeção de controle de qualidade e cálculo de posição e orientação de detalhes para um braço robótico, por exemplo.

Figura 6 - Utilização de algoritmos de visão computacional para inspeção de qualidade na indústria madeireira. (SEMINÁRIO INTRODUÇÃO À VISÃO COMPUTACIONAL, 2014).

2.1.1. OpenCV

Devido a familiaridade do autor deste trabalho com a linguagem C/C++, entendeu-se por bem que essa linguagem seria adotada para desenvolvimento do sistema. Sendo assim, a melhor opção para o desenvolvimento de algoritmos de visão computacional, na linguagem C/C++, é o OpenCV.

O OpenCV, desenvolvida pela Intel, em 2000, é uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de Visão computacional, bastando seguir o modelo de licença da BSD Intel. O OpenCV possui módulos de Processamento de Imagens e Vídeo I/O, Estrutura de dados, Álgebra Linear, Interface Gráfica com sistema de janelas independentes, Controle de mouse e teclado, além de mais de 350 algoritmos de Visão computacional como: Filtros de imagem, calibração de câmera, reconhecimento de objetos, análise estrutural e outros. O seu processamento é em tempo real de imagens.

Esta biblioteca foi desenvolvida originalmente nas linguagens de programação C/C++. Porém também, dá suporte a programadores que utilizem Java, Python e Visual Basic que desejam incorporar a biblioteca a seus aplicativos.

Vale ressaltar que o OpenCV oferece uma série de métodos e possibilidades de detecção de objetos de imagens. Dentre eles, uma implementação do Algoritmo de Viola e Jones, algoritmo capaz de reconhecer padrões de imagens, como faces humanas (VIOLA; JONES, 2001), bem como um filtro em cascata treinado para detectar tais faces, como exemplificado na Figura 7.

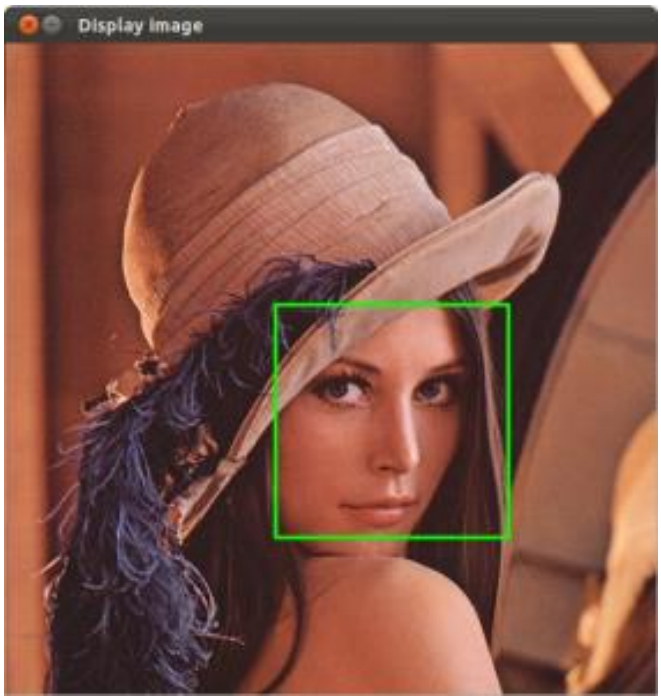

Figura 7 - Detecção de faces utilizando-se do algoritmo de Viola e Jones. Disponível: <https://pdincau.files.wordpress.com/2012/11/facedet.png>. Acesso em: 2 de nov. 14.

2.2. O Algoritmo de Viola e Jones

A utilização de algoritmos de visão computacional para extração de características de objetos, como faces humanas, tem demonstrado resultados robustos, eficientes e rápidos.

Entre os algoritmos mais conhecidos, destaca-se a proposta de Viola e Jones, por apresentar uma solução genérica quanto à detecção de qualquer tipo de objeto, além de possuir uma eficiência muito grande.

A descrição do Algoritmo foi embasada na Dissertação de Mestrado de Moreira, Capítulo 3 (2008).

2.2.1. Formato das Características de um Objeto

A detecção de objetos é baseada nas principais características de um objeto previamente extraídas a partir de um algoritmo de aprendizado de máquina. Tais características são responsáveis por diferenciar objetos uns dos outros, pois cada conjunto de características

encontradas em um dado objeto é altamente distintivo em relação aos conjuntos de características encontradas também em outros objetos.

A lógica para o uso de características de um objeto ao invés do uso de seus pixels, é que a velocidade da análise de uma imagem baseada no conjunto de suas principais características é maior do que a análise pontual baseada sobre seus pixels, devido ao fato do número de características ser substancialmente inferior em relação ao número de pixels de um objeto.

Sendo assim, o algoritmo de Viola e Jones é baseado em 3 formatos de características:

1. Característica de “dois retângulos”, onde o seu valor é a diferença entre a soma dos pixels dentro de duas regiões retangulares adjacentes de mesmo tamanho;
2. Característica de “três retângulos”, cujo valor é a soma em um retângulo central menos a soma de dois retângulos externos;
3. Característica de “quatro retângulos” que possui seu valor calculado através da diferença entre pares diagonais de retângulos.

Essas características são exemplificadas na Figura 8.

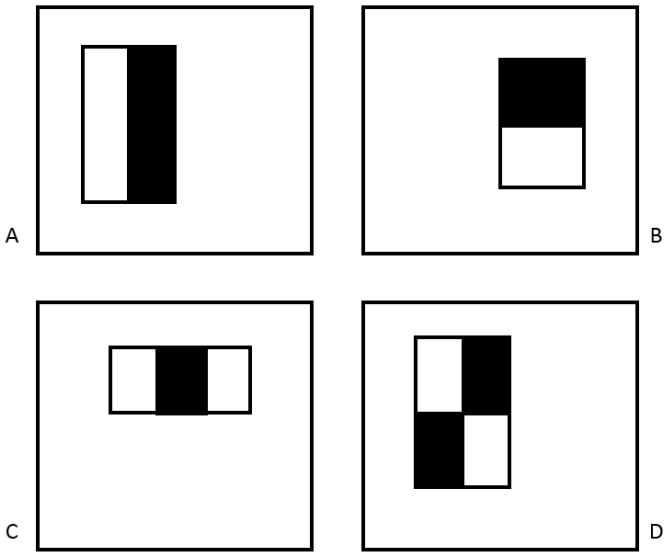

Figura 8 - Formatos de características possíveis. Os tipos de características representados em A e B são do tipo de “dois retângulos”, só que a primeira encontra-se no sentido vertical e a segunda

no sentido horizontal. Em C e D ilustram as características "três retângulos" e "quatro retângulos" respectivamente.(VIOLA; JONES, 2001, modificada).

Tais características no formato retangular são conhecidas como "características Haar". Baseado no estudo realizado previamente por Papageorgiou e Poggio (2000), no qual é proposto o uso de Wavelets de Haar para facilitar o treinamento de classificadores.

Esse estudo demonstrou que os usos destas características retangulares são suficientes para a extração de informações relevantes das formas de um objeto.

Com o objetivo de otimizar o cálculo de tais características retangulares, Viola e Jones introduzem, à visão computacional, a teoria da Imagem Integral. Nesta representação, como na Figura 9, cada ponto P(x,y) da imagem contém o somatório da intensidade luminosa de cada pixel da origem da imagem até a sua localização. Sendo assunto, a intensidade luminosa no ponto P(x,y), I(x,y) pode ser calculada com uma única iteração na imagem original, de acordo com a equação, onde i(x,y) é a intensidade em cada ponto.

$$I(x,y) = \sum_{\substack{0 \le x' \le x \\ 0 \le y' \le y}} i(x',y') \qquad (1)$$

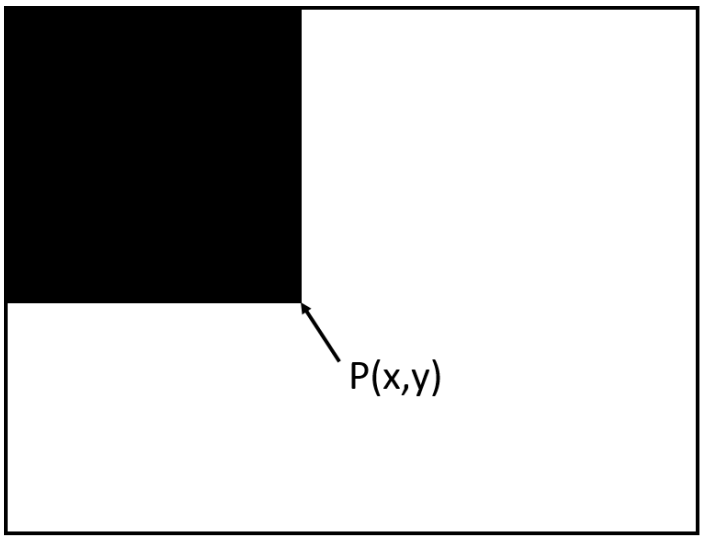

Figura 9 - Representação de área retangular da imagem integral

Tendo-se calculado a imagem integral, é possível encontrar o valor de intensidade de uma área retangular qualquer utilizando apenas os quatro pontos dos vértices da área desejada. Supondo que temos o valor total da área de origem da imagem até cada um dos quatro vértices, para se

encontrar a área desejada basta realizar uma subtração simples de retângulos. Caso deseja-se saber a área delimitada pelos pontos ABCD na Figura 10 seria necessário fazer o simples cálculo.

$$\sum_{\substack{x0 \le x \le x1 \\ y0 \le y \le y1}} i(x,y) = I(C) + I(A) - I(B) - I(C) \qquad (2)$$

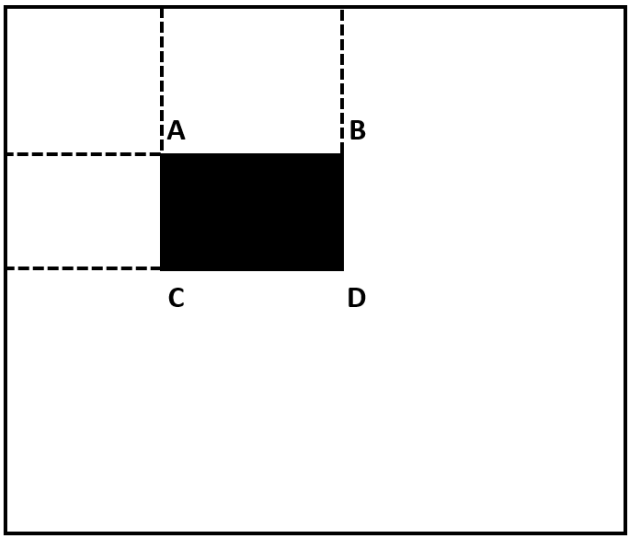

Figura 10 - Área delimitada pelos pontos ABCD. (VIOLA; JONES, 2001, modificada)

Desta maneira, o cálculo de intensidade luminosa em qualquer região de uma imagem integral pode ser realizado em tempo constante.

Por fim, com objetivo de melhorar a etapa de detecção de objetos, Lienhart e Maydt (2002) estenderam a quantidade de tipos de características retangulares, utilizando-se de rotação, como exemplificado na Figura 11.

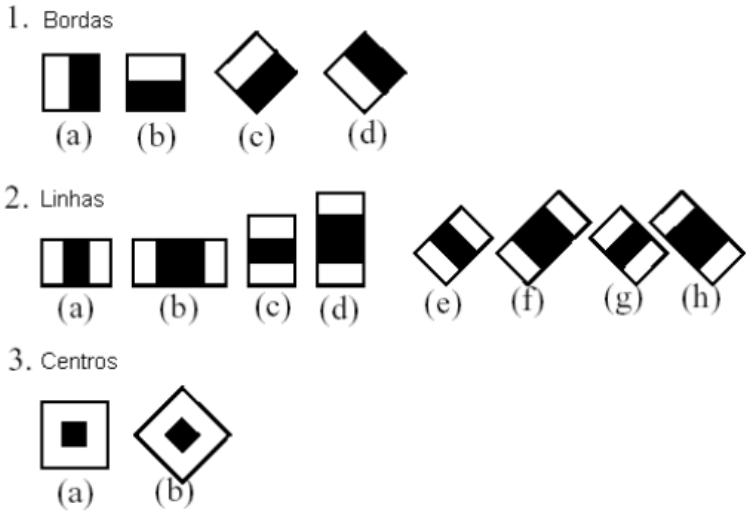

Figura 11 - Características retangulares estendidas por rotação. (VIOLA; JONES; 2001)

Com esta nova etapa, o processo de detecção se tornou 10% mais eficiente, diminuindo o número de falsos positivos.

2.2.2. Classificação

A função de classificação deve classificar corretamente um dado objeto a partir de um conjunto de suas principais características. Existem diversos algoritmos de aprendizado, como redes neurais e Support Vector Machine. O escolhido por Viola e Jones foi uma variante do algoritmo de AdaBoost(SCHAPIRE, 2013).

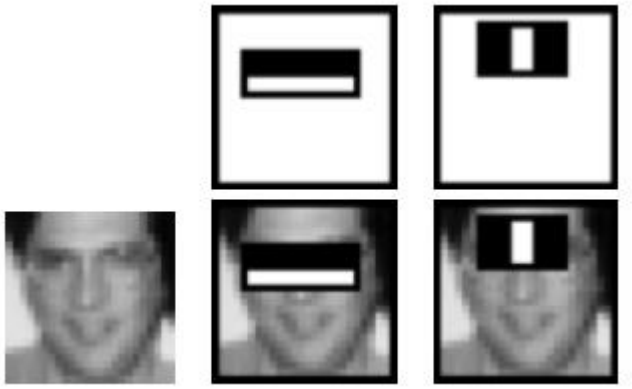

Figura 12 - Características obtidas através do treinamento em objetos do tipo "face humana". (VIOLA; JONES, 2001)

O classificador percorre a imagem procurando regiões com os mesmos padrões que as características do objeto desejado, na Figura 13 está ilustrado um exemplo em sua utilização na face humana. Na primeira linha estão exemplos da primeira e segunda características selecionadas pelo AdaBoost durante o treinamento do tipo de objeto “face humana”. Na segunda linha, as características estão sobrepostas sobre uma face, pode-se observar que a primeira característica mais marcante mostra que a região dos olhos é geralmente mais escura que a região das bochechas. Nota-se também que a segunda característica destaca a diferença de intensidade da região dos olhos e do nariz.

O algoritmo de aprendizado para ser executado necessita de um grupo de imagens de interesse, contendo imagens positivas, que contém o objeto de interesse, e imagens negativas, que não contém o objeto de interesse. O AdaBoost tem por objetivo tanto selecionar as características de interesse, quanto para treinar o classificador, destacando-se por encontrar um número reduzido de características relevantes no objeto, para determinação de positivos e negativos.

Com a utilização do algoritmo descrito, Viola e Jones conseguiram ótimos resultados. O classificador criado com 200 características e uma taxa de detecção desejada de 95%, obteve um único falso positivo num universo de 14.804 imagens de faces humanas testadas testadas.

2.2.3. Cascata de classificadores

A cascata de classificadores tem por objetivo otimizar o reconhecimento da imagem. São feitos estágios através da combinação de funções de classificação, criadas através do algoritmo de AdaBoost. A ideia é que classificadores fracos descartem grande número de regiões que não contém o objeto desejado, para diminuir a necessidade de processamento desnecessário, e que os estágios mais avançados sejam mais precisos, para evitar erros.

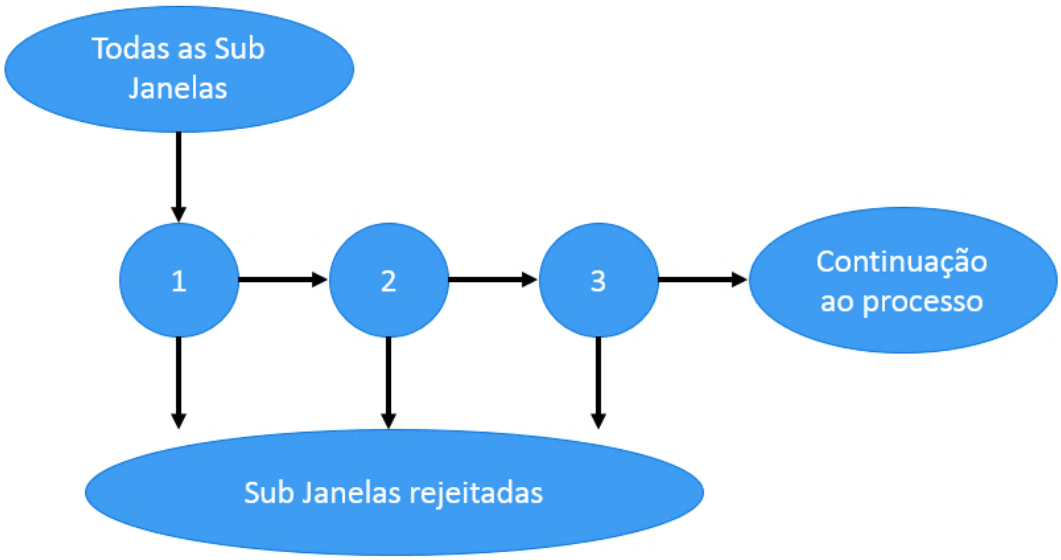

Figura 13 - Funcionamento de uma cascata de classificadores. (VIOLA; JONES, 2001, traduzida)

O processo de geração de cascatas é orientado por um conjunto de metas de detecção e desempenho. O número de estágios na cascata devem ser o suficiente para garantir uma taxa elevada de detecção de objetos, uma baixa ocorrência de falsos positivos e a minimização do tempo de processamento durante o teste de uma região da imagem.

2.2.4. A busca pelo objeto

Para que a detecção do objeto aconteça, são necessárias diversas varreduras na imagem, em diversos tamanhos, fazendo com que cada vez seja analisada uma região diferente, o que é chamado de janela de busca. O usuário define o tamanho mínimo que essa janela de busca pode ter de tal sorte que deve ser maior ou igual ao tamanho com o qual a cascata foi treinada, ou seja, se a cascata foi treinada com imagens positivas de 24 x 24 pixels, o tamanho mínimo da janela de busca deve seguir tal resolução.

A imagem deve ser percorrida por completo. Sendo assim, é necessário que se desloque a janela de busca pelo eixo x e pelo eixo y, com um determinado valor de deslocamento. Deve-se ter cuidado ao determinar o deslocamento, pois apesar de um valor alto fazer com que se tenha

menos janelas analisadas, pode-se diminuir o número de detecção, da mesma maneira, um valor muito baixo de deslocamento, pode gerar um aumento no tempo do processamento. Baseado em seus experimentos, Violas e Jones sugerem como deslocamento ideal o valor de 1,25 pixels.

2.2.5. O Algoritmo de Viola e Jones no OpenCV

A biblioteca OpenCV já possui uma implementação do algoritmo de Viola e Jones. Esse método já implementado, pode ser utilizado por meio da função:

detectMultiScale(const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, intminNeighbors=3, int flags=0, Size minSize=Size(), Size maxSize=Size())

Essa função possui 5 parâmetros de entrada e como resultado de saída retorna uma sequência de retângulos, e cada um desses retângulos representa um rosto. Os parâmetros de entrada e saída são:

• *image*: A imagem onde será feita a procura.
• *objects*: Sequência de retângulos com a localização das imagens procuradas.
• *cascade*: O classificador do tipo HaarCascade (na forma de arquivo XML) lido pelo OpenCV.
• *storage*: Espaço alocado na memória para armazenar a sequência dos resultados da detecção do objeto(resultados em forma de retângulos).
• *scale factor*: O fator pelo qual a janela de procura da função será aumentada para percorrer novamente a imagem.
• *min neighbors*: Faz o agrupamento de retângulos (da região da face) vizinhos, sendo esse parâmetro utilizado para mesclar retângulos similares.
• *flags*: Faz um processamento específico disponibilizado pela biblioteca OpenCV.
• *min size*: Valor em pixels da menor janela de procura, largura e altura da janela.

2.3. Thresholding

Outra técnica de ampla relevância na visão computacional que será necessária para verificar se os olhos do usuário do sistema estão abertos ou fechados é a binarização ou como é mais conhecida o Thresholding.

É uma técnica utilizada para binarização de uma imagem, transformando-a em uma imagem binária, ou seja, preto e branca.

Figura 14 - Exemplo de Threshold aplicado em uma imagem. Disponível em: < http://answers.opencv.org/upfiles/13506303131033303.png>. Acesso: 2 nov. 14.

Escolhendo-se um valor para limiar, a técnica do Thresholding converte todos os valores maiores que o limear para o 1 de intensidade de cor e os menores para o 0.

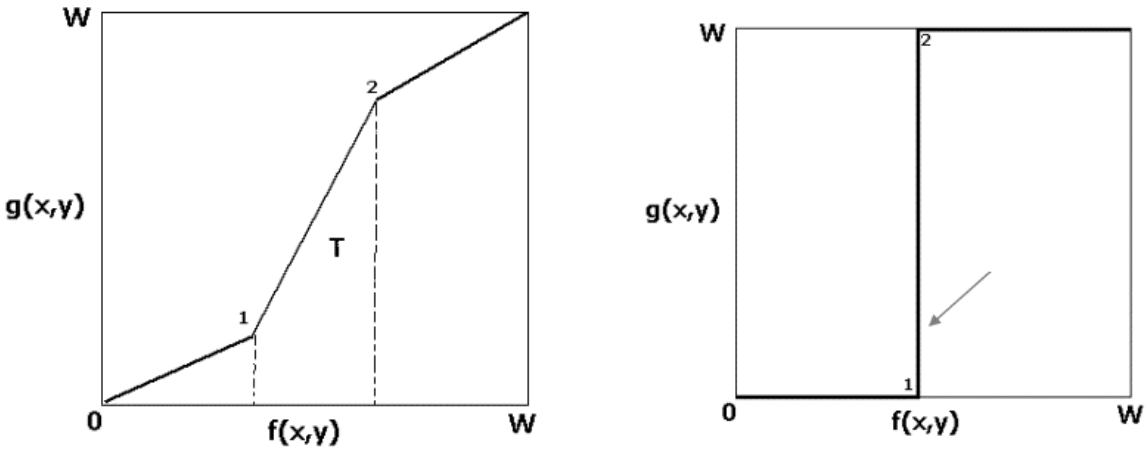

Gráfico 1 - Função intensidade de cor, antes e depois da utilização do Thresholding. (GONZAGA, AULA 2, modificado).

2.3.1. O Thresholding no OpenCV

A função Thresholding é disponível na biblioteca OpenCV.

double Threshold(InputArray src, OutputArray dst, double thresh, double maxval, int type)

Os parâmetros da função são:

- src – vetor de entrada (com único canal, 8-bit ou 32-bit de ponto flutuante).
- dst – vetor de saída com as mesmas características do vetor src.
- thresh – valor limiar.
- maxval – valor que será considerado o máximo.
- type – tipo de Thresholding(no caso utilizaremos o THRESH_BINARY, que tem dinâmica como explicado no item 3.4.).

2.4. Reconhecimento do estado de sonolência

2.4.1. Fisiologia do sono

Sono (do latim somnus) é um estado ordinário de consciência, complementar ao da vigília (ou estado desperto), em que há repouso normal e periódico, caracterizado, tanto no ser humano como nos outros vertebrados, pela suspensão temporária da atividade perceptivo-sensorial e motora voluntária.

2.4.2. O ciclo do sono

Um ciclo do sono dura cerca de noventa minutos, ocorrendo quatro a cinco ciclos num período de sono noturno. Oito horas é o ideal para dormir. Segundo LAVIE (1998, 45), o número de ciclos por noite depende do tempo do sono, acrescentando, ainda, que "o sono de uma pessoa jovem é, habitualmente, composto por quatro ou cinco desses ciclos, com tendência à redução com o avançar da idade". No entanto, o padrão comum varia entre quatro a cinco ciclos.

Durante o sono, o indivíduo passa, geralmente por ciclos repetitivos, começando pelo estágio 1 do sono NREM, progredindo até o estágio 4, regride para o estágio 2, e entra em sono REM. Volta de novo ao estágio 2 e assim se repete novamente todo o ciclo.

Nos primeiros ciclos do sono, os períodos de NREM (mais concretamente o estágio 3 e 4) têm uma duração maior que o REM. À medida que o sono vai progredindo, os estágios 3 e 4 começam a encurtar e o período REM começa a aumentar. Na primeira parte do sono predomina o NREM, sendo os períodos REM mais duradouros na segunda metade.

O sono divide-se em dois tipos fisiologicamente distintos:

- NREM (Non Rapid Eye Movement ou "Movimento Não Rápido dos Olhos"); e
- REM (Rapid Eye Movement ou "Movimento Rápido dos Olhos").

2.4.2.1. Sono NREM

O sono NREM (ou não-REM) ocupa cerca de 75% do tempo do sono e divide-se em quatro períodos distintos conhecidos como estágios 1, 2, 3 e 4.

Estágio 1:

Começa com uma sonolência. Dura aproximadamente cinco minutos. A pessoa adormece. É caracterizado por um EEG semelhante ao do estado de vigília. Esse estágio tem uma duração de um a dois minutos, estando o indivíduo facilmente despertável. Predominam sensações de vagueio, pensamentos incertos, mioclonias das mãos e dos pés, lenta contração e dilatação pupilar. Nessa fase, a atividade onírica está sempre relacionada com acontecimentos vividos recentemente.

Estágio 2:

Caracteriza-se por a pessoa já dormir, porém não profundamente. Dura cerca de cinco a quinze minutos. O eletroencefalograma mostra frequências de ondas mais lentas, aparecendo o complexo K. Nessa fase, os despertares por estimulação táctil, fala ou movimentos corporais são mais difíceis do que no anterior estágio. Aqui a atividade onírica já pode surgir sob a forma de sonho com uma história integrada.

Estágio 3:

Tem muitas semelhanças com o estágio 4, daí serem quase sempre associados em termos bibliográficos quando são caracterizados. Nessas fases, os estímulos necessários para acordar são maiores. Do estágio 3 para o estágio 4, há uma progressão da dificuldade de despertar. Esse estágio tem a duração de cerca de quinze a vinte minutos.

Estágio 4:

São quarenta minutos de sono profundo. É muito difícil acordar alguém nessa fase de sono. Depois, a pessoa retorna ao terceiro estágio (por cinco minutos) e ao segundo estágio (por mais quinze minutos). Entra, então, no sono REM.

Este estágio NREM do sono caracteriza-se pela secreção do hormônio do crescimento em grandes quantidades, promovendo a síntese proteica, o crescimento e reparação tecidular, inibindo, assim, o catabolismo. O sono NREM tem, pois, um papel anabólico, sendo essencialmente um período de conservação e recuperação de energia física.

2.4.3. Sono REM

O sono REM caracteriza-se por uma intensa atividade registrada no eletroencefalograma (EEG) seguida por flacidez e paralisia funcional dos músculos esqueléticos. Nesta fase, a atividade cerebral é semelhante à do estado de vigília. Deste modo, o sono REM é também denominado por vários autores como sono paradoxal, podendo mesmo falar-se em estado dissociativo.

Nesta fase do sono, a atividade onírica é intensa, sendo sobretudo sonhos envolvendo situações emocionalmente muito fortes.

É durante essa fase que é feita integração da atividade cotidiana, isto é, a separação do comum do importante. Estudos também demonstram que é durante o REM que sonhos ocorrem. A fase representa 20 a 25% do tempo total de sono e surge em intervalos de sessenta a noventa minutos. É essencial para o bem-estar físico e psicológico do indivíduo.

Pode-se resumir estas fases utilizando-se da Figura 15.

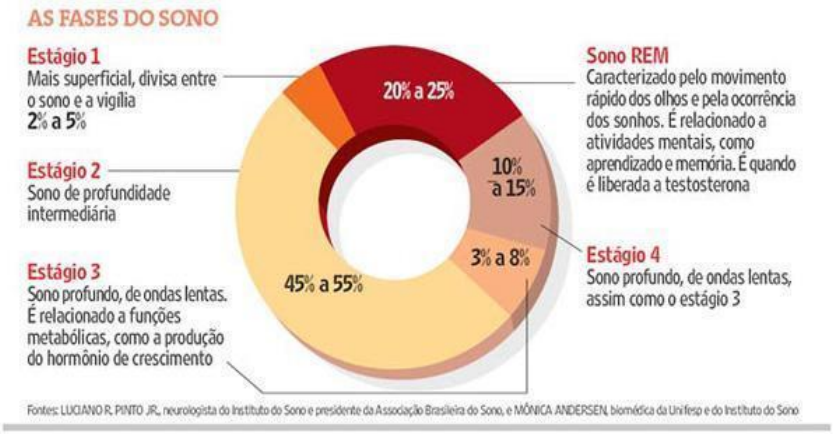

Figura 15 - Fases do sono. Disponível em: <http://3.bp.blogspot.com/_YpKuwYjJGwA/TQmdlHSlIhI/AAAAAAAAJlg/RSjknmOzsdE/s1600/fases_sono.jpg>. Acesso: 3 nov. 2014.

Para melhor caracterização dos estágios de sono, a Figura 16 demonstra a atividade cerebral de um ser humano durante estes estágios.

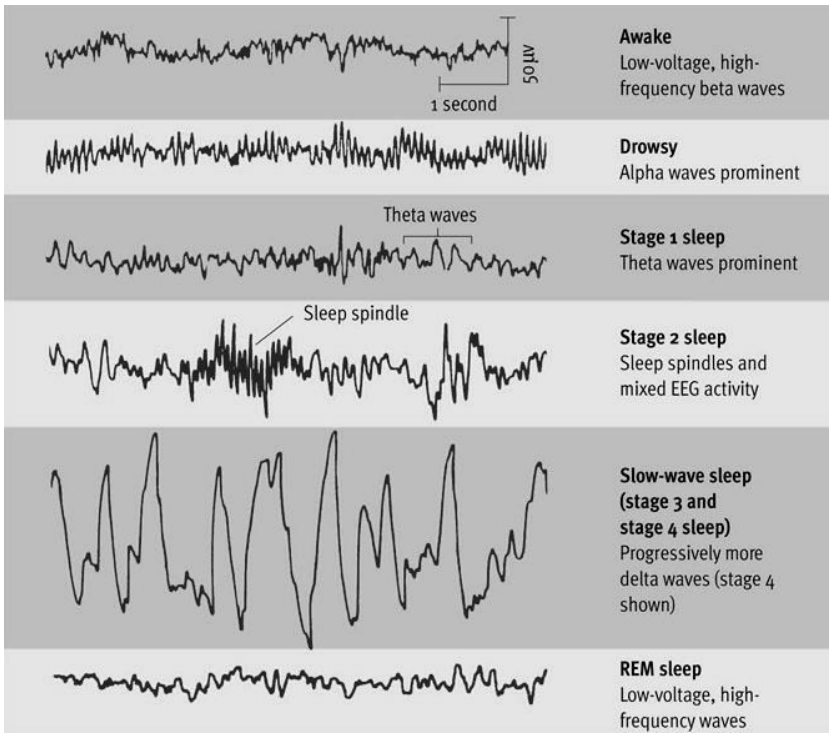

Figura 16 - Ondas cerebrais nos diversos estágios de sono. Disponível em: <http://3.bp.blogspot.com/-FvO8_SxhcF8/UehpqcZqTNI/AAAAAAAAAwU/8UoejBsZZSQ/s1600/101+sleep+stages.jpg>. Acesso: 3 nov. 2014.

Pode-se dizer popularmente que a fase NREM é a fase do sono sem sonhos e a fase REM é a fase com sonhos.

2.4.4. Sonolência

A sonolência pode ser definida como um estado fisiológico resultante da privação do sono, uma sensação subjetiva da necessidade de dormir.

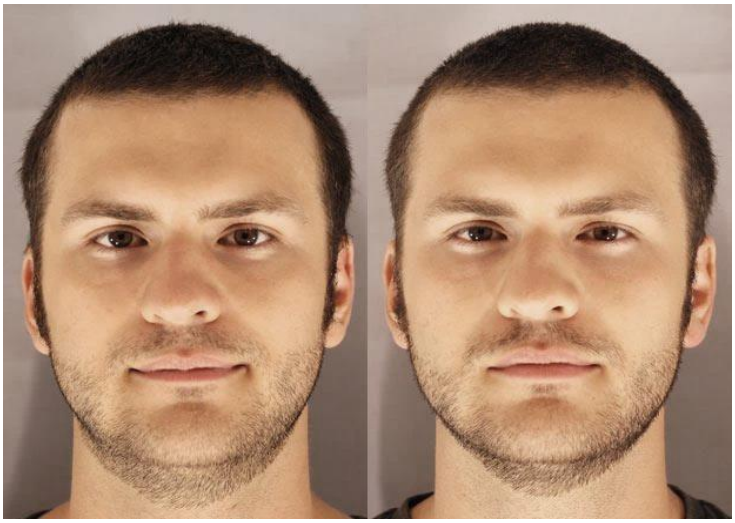

Figura 17 - Alteração da fisionomia de um indivíduo depois de 31 horas sem dormir. Disponível em: <http://s.glbimg.com/jo/g1/f/original/2010/12/15/sonobeleza3.jpg>. Acesso em: 2 nov. 14.

A Figura 17, exemplifica a mudança de fisionomia de um indivíduo depois de 31 horas privado do sono. Na foto à esquerda da Figura 17, verifica-se uma pessoa sem sono, já na foto à direta, verifica-se a mesma pessoa 31 horas após, sem que o indivíduo tenha dormido. Desta maneira, evidencia-se que a mudança mais significativa está na diminuição da área exposta da região ocular, ou seja, os olhos.

A diminuição da área exposta dos olhos tem uma explicação simples e objetiva. O cérebro, órgão responsável por todas as atividades do corpo humano, necessita de repouso quando estamos com sono, sendo assim devesse diminuir a luminosidade, visto que a mesma é um dos principais agentes de excitação da atividade cerebral. Por isso que quando estamos com sono, nossas piscadas se tornam mais longas e mais frequentes.

Horng (2004) define em seu algoritmo que uma vez indicado o estado do olho, aberto ou fechado, a sonolência do usuário poderá ser determinada, considerando que, se os olhos permanecerem fechados por mais de 5 quadros consecutivos, a uma taxa de 30 FPS.

A teoria é confirmada quando de posse dos trabalhos acerca da fisiologia da sonolência, por exemplo nos estudos de Bristow (2005) e Anund (2009). De acordo com experimentos em campo e em simuladores de direção, a duração da piscada de olho aumenta no estado de sonolência. A duração média de uma piscada em um indivíduo em estado de alerta é de 0.1 segundos, já para um indivíduo privado do sono 0.2 segundos. Outros parâmetros da piscada são alterados no estado de sonolência, como o tempo de fechamento dos olhos, tempo com os olhos fechados, tempo de abertura, amplitude da deflexão da pálpebra, além da frequência das piscadas (ANUND, 2009). Sendo assim, utilizando-se de uma taxa de 30 FPS, 1 quadro tem duração de 0,0333 s, portanto 6 quadros, 0,2 s.

3. Recursos

Os recursos utilizados foram escolhidos de tal forma a ter o melhor desempenho, utilizando-se do menor custo que fosse possível.

3.1. Recursos de Hardware

3.1.1. Plataforma

Considerando a necessidade de um grande processamento de informações e imagens, foi escolhida a Odroid U2, como plataforma para a aplicação. A Odroid-U2 é um computador com dimensões extremamente reduzidas (59 x 57 x 60 mm), como podemos verificar nas Figuras 18 e 19, fabricado na Coréia do Sul pela Hard Kernel.

Figura 18 - Odroid U2. (HARDKERNEL, 2014)

Figura 19 - Noção do dimensional da Odroid U2. (HARDKERNEL, 2014, traduzida)

É um sistema baseado no SoC Samsung Exynos4412 Prime Cortex-A9 Quad Core 1.7Ghz com 1MB L2 cache. Trata-se de um dos mais recentes e potentes CPUs disponíveis atualmente, que

já utiliza núcleos Cortex-A9, acompanhado por um GPU Mali-400 Quad Core a 440Mhz, e 2GB de RAM, e é vendida pelo preço de US$89,00 no site do fabricante. Além disso, vem equipado com portas USB, Ethernet, micro HDMI, microSD, e compatível com Android 4.x e Ubuntu 12.10.(HARDKERNEL)

Para melhor descrição da Odroid, suas caraterísticas estão descritas na Tabela 1.

Tabela 1 - Características relevantes da Odroid U2. (HARDKERNEL, 2014, traduzida)

Característica	**Descrição**
Processador	Samsung Exynos4412 Prime Cortex-A9 Quad Core 1.7Ghz com 1MB L2 cache
Memória	2048MB(2GB) LP-DDR2 880Mega
Acelerador 3D	Mali-400 Quad Core 440MHz
Vídeo	1080p via HDMI cable(H.264+AAC baseado MP4 formato)
Saída de Vídeo	micro HDMI
Áudio	Entrada headphone 3.5mm padrão
	HDMI Digital
LAN	10/100Mbps Ethernet with RJ-45 Jack (Auto-MDIX support)
USB2.0 Host	High speed standard A type connector x 2 ports
USB2.0 Device	ADB/Mass storage(Micro USB)
Armazenamento	MicroSD
	Modulo Emmc
Potência	10W - 5V 2A
Case	Corpo todo em alumínio com dissipador de calor. (59 x 57 x 60 mm)
Dimensões da PCB	48 x 52 mm

3.1.2. Webcam

A Webcam escolhida é o modelo Wc045 do fabricante MultiLaser, Figura 20. O fabricante (MULTILASER, 2014) cita como principais características da câmera:

- Microfone interno
- Lente de vidro de 2 camadas
- Botão Snapshot

- LED noturno
- Conexão: USB1.1 e 2.0
- Resolução: 1.3MP real ou até 16MP (interpolados)

Figura 20: Webcam Wc045. (MULTILASER, 2014)

A escolha desse componente deve-se exclusivamente ao seu baixo custo.

3.2. Recursos de Software

3.2.1. Sistema Operacional

O sistema operacional escolhido é uma compilação do Linux específica para a Odroid U2 denominada Linaro 12.11 Robotics Edition v2 (ROS+OpenCV+OpenNI+PCL) U2. Sua interface gráfica está representada na Figura 21.

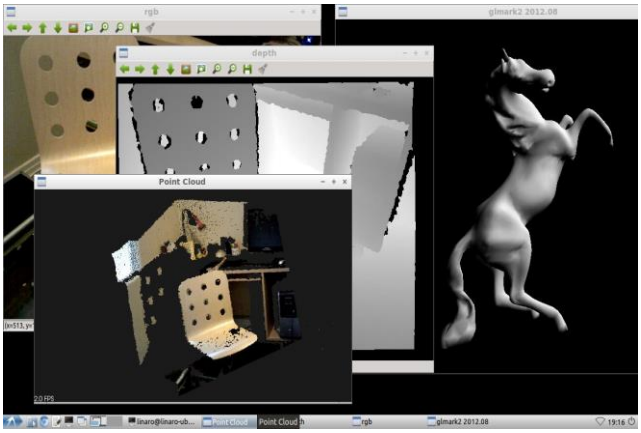

Figura 21 - Interface gráfica do Linaro 12.11 Robotics Edition v2 (ROS+OpenCV+OpenNI+PCL) U2. Disponível em: < http://forum.odroid.com/download/file.php?id=325&t=1&sid=9a9fc5773381eeff711ecfe5adbc99dd>. Acesso em: 29 de ago. 2014.

Suas características que tem destaque para o projeto são:

- Kernel 3.0.75;
- Baseada no sistema Debian;
- Mali habilitada;
- Lubuntu Desktop;
- OpenCV 2.4.6.1;
- Compilador C/C++ CMake;

Deve-se ressaltar a importância do OpenCV e do CMake nativos, visto que facilitam o desenvolvimento do projeto.

4. Método

4.1. Ligação dos componentes

No sistema proposto, os materiais foram conectados como ilustra a Figura 22.

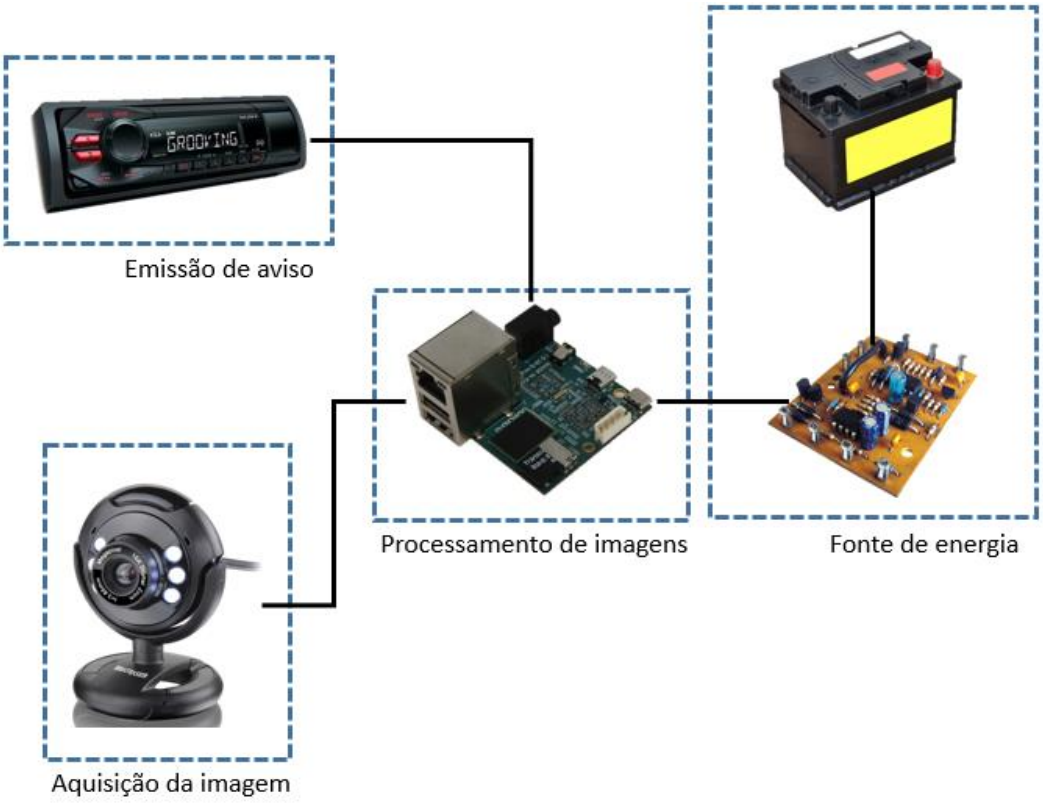

Figura 22 - Esquema elétrico de ligação dos materiais utilizados no sistema.

Para alimentar o sistema seria necessário um circuito que diminuísse a tensão da bateria do veículo, usualmente 12V ou 24V, para 5V, tensão de trabalho da Odroid U2, a câmera seria ligada em uma de suas portas USB disponíveis e a conexão com o rádio do veículo para emissão de alertas caso o usuário estivesse sonolento, pela saída P2.

4.2. Posição da câmera e da Odroid U2 dentro do veículo

A disposição proposta para os materiais dentro do veículo, segue a representação exposta na Figura 21.

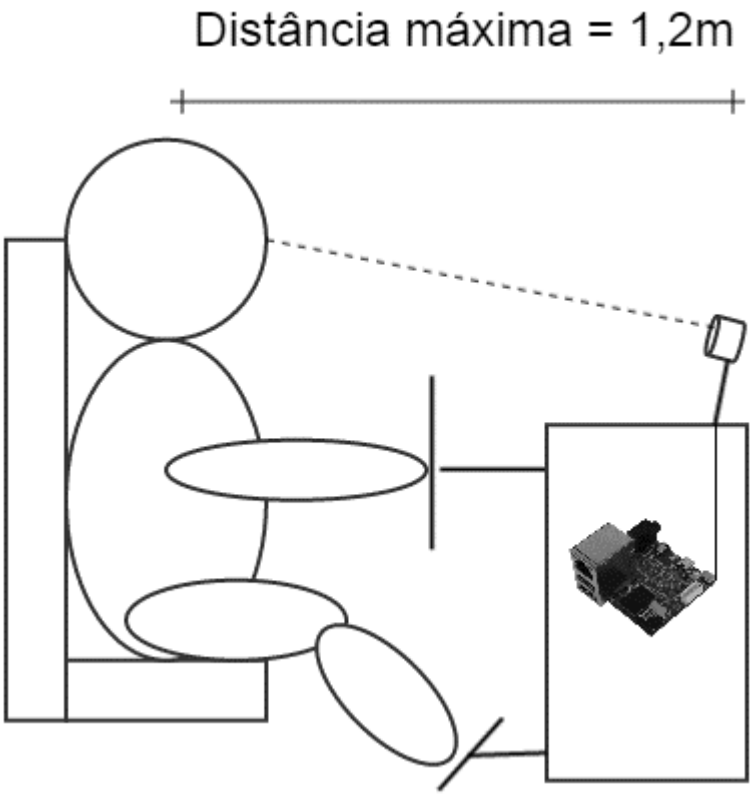

Figura 23 - Proposta de disposição dos materiais em um veículo automotivo

A distância observada na Figura 20 é de extrema importância, visto que foi dimensionada a atender os parâmetros dimensionais médios de um veículo automotivo.

4.3. O software

O desenvolvimento do software foi escrito no editor de texto gEdit na linguagem C++, como citado anteriormente, e compilado com o compilador cMake. Seu funcionamento é baseado no fluxograma representado na Figura 24.

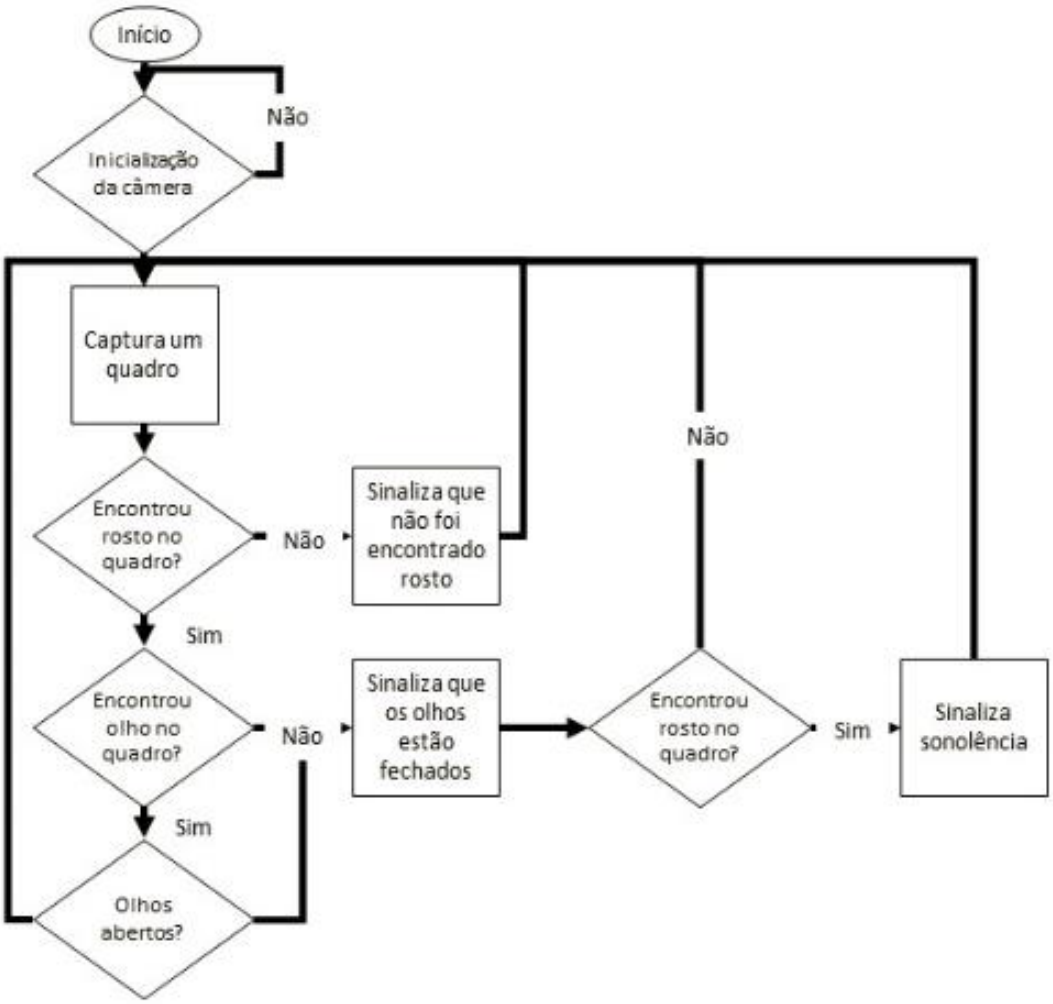

Figura 24: Fluxograma detalhado do sistema proposto.

Para melhor entendimento de seu funcionamento, cada uma das etapas chave de seu fluxograma será detalhada.

4.3.1. Inicialização da câmera

Nesta etapa, o programa busca a conexão com a câmera-Odroid, e determina todos os parâmetros de resolução, sistema de cor e FPS. Os parâmetros utilizados estão na Tabela 2.

Tabela 2 - Parâmetros de inicialização de câmera utilizados.

Parâmetro	Valor
Resolução	640x480 pixels
Sistema de cor	RGB
FPS	30 FPS

4.3.2. Captura de quadro

É capturado um quadro para análise. Esse quadro é convertido para escala de tons de cinza, visto que é um pré-requisito para funcionamento do algoritmo de Viola e Jones.

4.3.3. Detecção facial

A detecção facial é feita através da implementação do algoritmo de Viola Jones. Os parâmetros foram dimensionados para que o sistema tenha uma alta performance com a distância considerada para o projeto de 1,2 m (QUEIROZ, 2011).

- scale fator = 1.3
- min neighbors = 4
- min size = (40,40)
- flag = CV_HAAR_FIND_BIGGEST_OBJECT

O classificador, para detecção facial utilizado foi o arquivo haarcascade_frontalface_alt.xml, que já acompanha o OpenCV.

Encontrado um rosto, o programa procede para a etapa de detecção de olhos.

4.3.4. Detecção de olhos

A detecção é feita com a mesma função utilizada na detecção facial, com a alteração dos parâmetros(QUEIROZ, 2011).

- scale fator = 1.1
- min neighbors = 3

- min size = (30,20)
- flag = CV_HAAR_FIND_BIGGEST_OBJECT

O classificador, para detecção de olhos também já acompanha o OpenCV e é o arquivo “haarcascade_eye_tree_eyeglasses.xml”.

Caso a função não encontre qualquer padrão que represente um olho na imagem, o programa interpreta como olhos fechados, e escreve na tela a frase:” Olhos fechados”.
Reconhecendo um possível candidato a olho aberto, o programa procede para a etapa de determinação de olhos abertos/fechados.

4.3.5. Determinação do estado dos olhos

Nesta etapa do programa é necessário determinar o estado do olho de maneira precisa. Para realizar essa tarefa, foi desenvolvido um processo baseado na verificação da existência de área branca na região dos olhos.

O processo utiliza a técnica do Thresholding, e tem dinâmica como exemplificado nas Figuras 25 e 26.

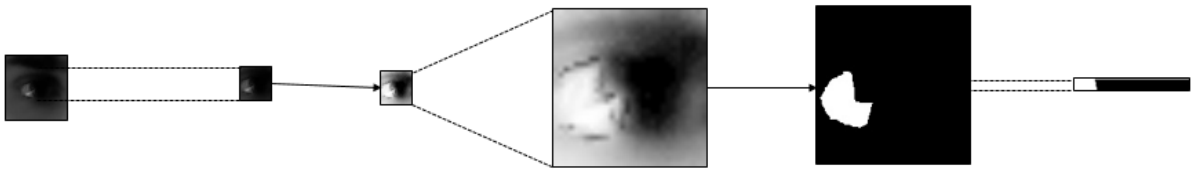

Figura 25 - Etapas de validação do estado dos olhos, utilizando-se de uma imagem de olhos abertos

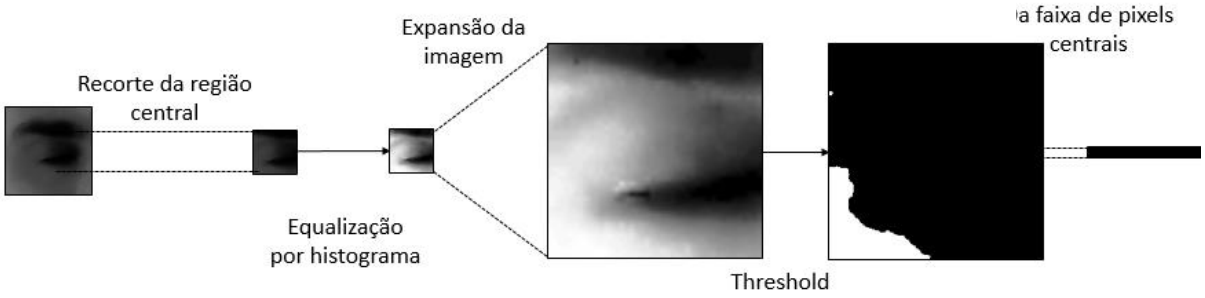

Figura 26 - Etapas de validação do estado dos olhos, utilizando-se de uma imagem de olhos fechados

O processo é iniciado com o recorte da região central da imagem reconhecida como “Olho” pelo algoritmo de Viola e Jones. A região recortada tem largura e altura igual à ¼ da região selecionada pelo reconhecimento de olhos. Esse recorte central é submetido a equalização por histograma, para que o contraste entre a região branca dos olhos e a pele do usuário fique mais nítido. Como a região recortada tem dimensões reduzidas, faz-se uma expansão nesta imagem em 5 vezes. A imagem enfim é submetida a técnica do Thresholding, para análise da região branca dentro dos olhos. O parâmetro de limiar de Thresholding foi determinado através de testes preliminares com valor igual a 235, de tal forma que extraísse as regiões brancas da imagem.

Por fim, separada a região de interesse, ou seja uma faixa de 4 pixels de altura do centro da imagem, como mostrado na Figura 24, caso haja qualquer pixel branco dentro desta região da imagem, o olho é considerado aberto, caso não haja, o olho é considerado fechado.

4.4. Testes

Os resultados que serão expostos nesse trabalho foram obtidos das imagens geradas pelo software de teste. O software de teste é uma variação do software proposto que salva, em formato .jpg, todos os quadros obtidos, com as sinalizações de ausência de rosto e olhos abertos ou fechados.

Para o desenvolvimento de testes, os componentes elétricos foram conectados como representado na Figura.

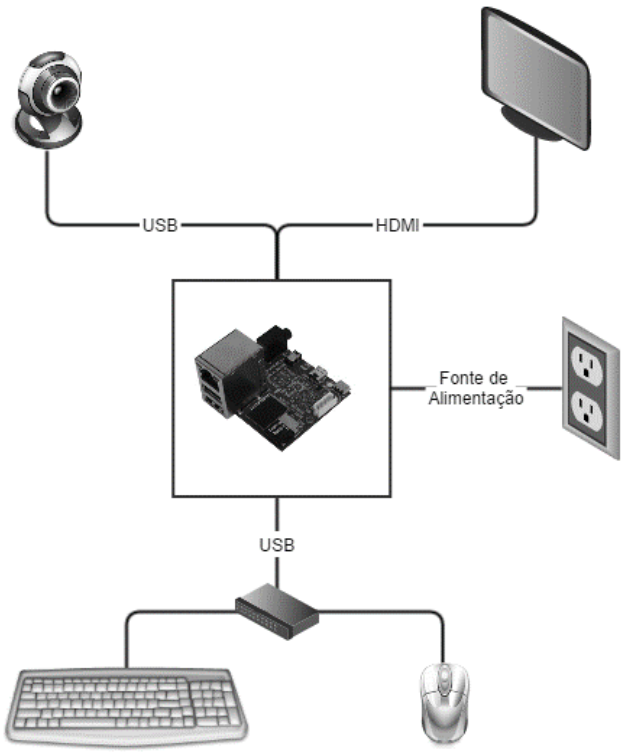

Figura 27: Esquema de ligação dos componentes na fase de testes

Além disso, um fator de extrema relevância nos testes é a luminosidade. Os testes foram realizados em ambiente fechado de aproximadamente 16 m², iluminado por duas lâmpadas de LED de 9W, 900 lúmens cada.

4.5. Limitações de projeto

Nos sistemas baseados em reconhecimento de padrões em imagem nota-se grande dependência da luminosidade ambiente. O sistema foi projetado para a luminosidade utilizada nos testes, portanto a alteração desse parâmetro resultaria em uma grande variação nos resultados. Além disso, caso o usuário estiver utilizando-se de óculos escuros, o algoritmo não reconhece o seu padrão de olhos, sendo assim, não consegue identificar a sonolência do condutor.

5. Resultados

Os resultados têm por base os dados obtidos através do software de testes. Os testes foram feitos em laboratório, devido a limitações de projeto, com 6 diferentes indivíduos. É importante ressaltar que os 6 usuários do sistema estavam em estado de vigília, e simularam sonolência através do fechamento de seus olhos.

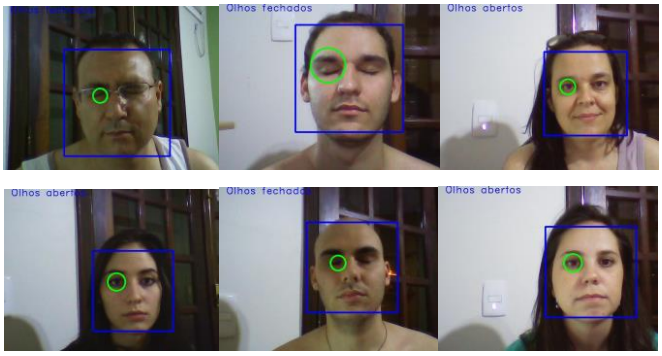

Figura 28 - Os usuários do sistema que passaram pelos testes

As imagens obtidas puderam ser classificadas em 6 tipos diferentes:

- Ausência de usuário. Não reconhecimento facial, sinalização de “Rosto não encontrado”.

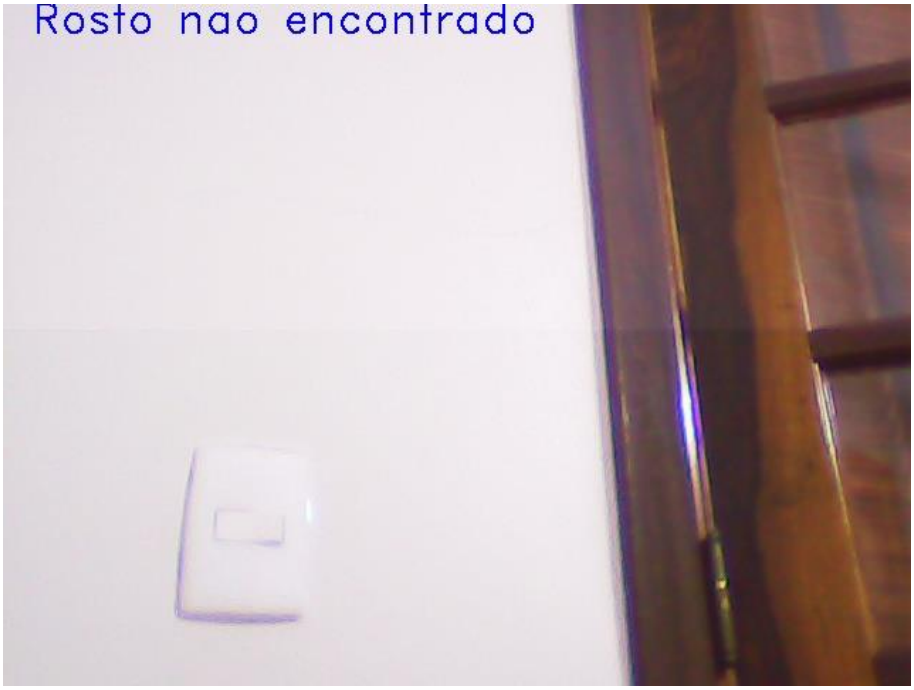

Figura 29: Ausência de usuário. Não reconhecimento facial, sinalização de “Rosto não encontrado”.

- Usuário presente com os olhos fechados. Reconhecimento da face, e o não reconhecimento dos olhos. Sinalização de “Olhos fechados”.

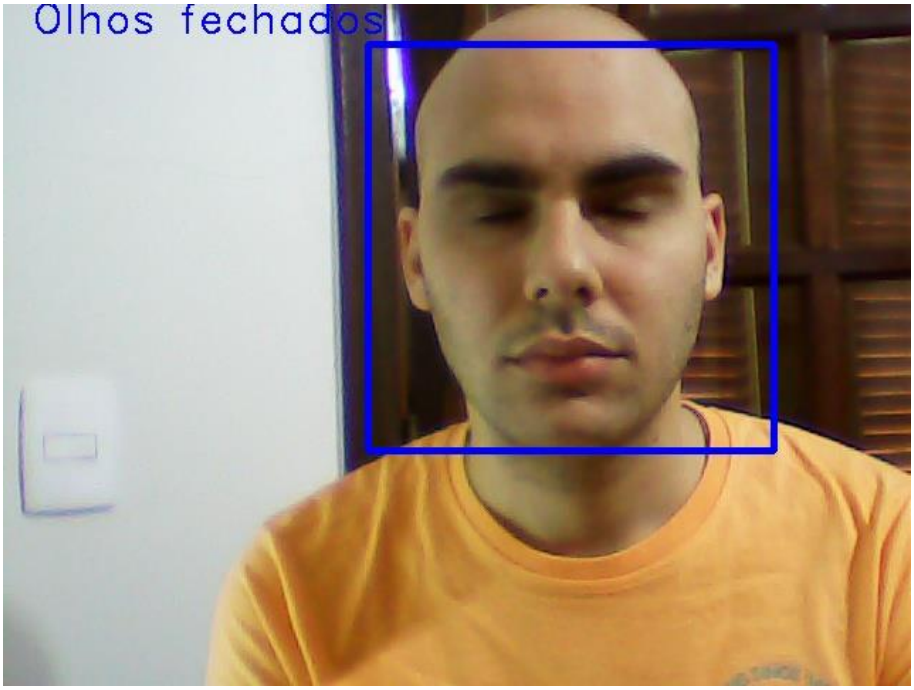

Figura 30: Usuário presente com os olhos fechados. Reconhecimento da face, e o não reconhecimento dos olhos. Sinalização de “Olhos fechados”.

- Usuário presente com olhos abertos. Reconhecimento da face, reconhecimento dos olhos, confirmação da hipótese de olhos abertos pelo Threshold. Sinalização de “Olhos abertos”.

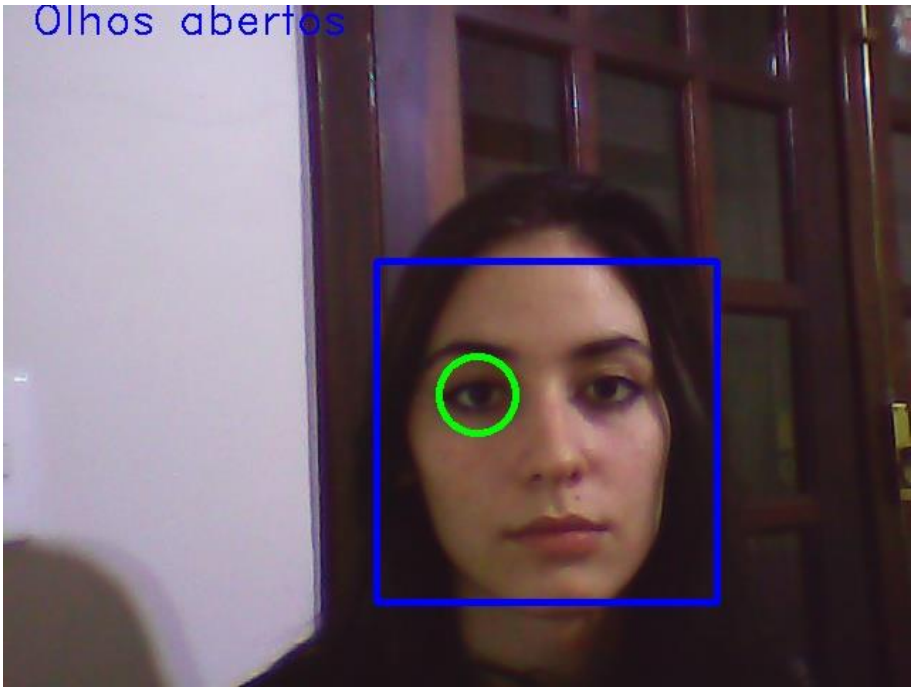

Figura 31: Usuário presente com olhos abertos. Reconhecimento da face, reconhecimento dos olhos, confirmação da hipótese de olhos abertos pelo Thresholding. Sinalização de "Olhos abertos".

- Usuário presente com olhos fechados. Reconhecimento da face, reconhecimento dos olhos, descarte da hipótese de olhos abertos pelo Thresholding. Sinalização de "Olhos fechados".

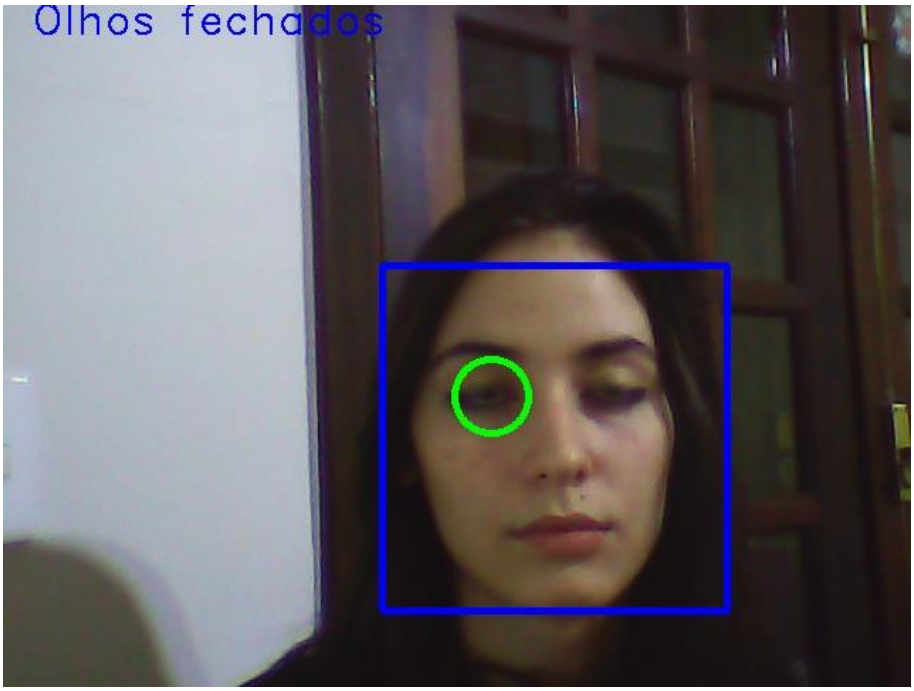

Figura 32: Usuário presente com olhos fechados. Reconhecimento da face, reconhecimento dos olhos, descarte da hipótese de olhos abertos pelo Thresholding. Sinalização de "Olhos fechados".

- Usuário presente com os olhos abertos. Reconhecimento da face, reconhecimento dos olhos, descarte indevido da hipótese de olhos abertos pelo Thresholding. Sinalização indevida de "Olhos fechados".

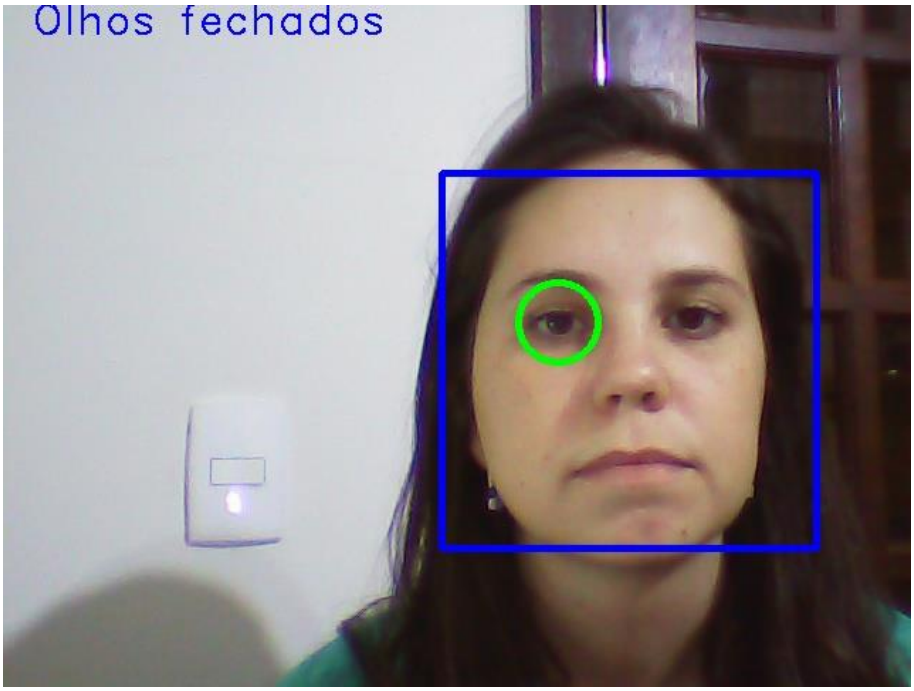

Figura 33: Usuário presente com os olhos abertos. Reconhecimento da face, reconhecimento dos olhos, descarte indevido da hipótese de olhos abertos pelo Thresholding. Sinalização indevida de "Olhos fechados".

- Usuário presente com os olhos fechados. Reconhecimento da face, reconhecimento dos olhos, validação indevida da hipótese de olhos fechados pelo Thresholding. Sinalização indevida de "Olhos abertos".

Figura 34: Usuário presente com os olhos fechados. Reconhecimento da face, reconhecimento dos olhos, validação indevida da hipótese de olhos fechados pelo Thresholding. Sinalização indevida de "Olhos abertos".

Através da observação dos resultados, foi possível determinar dois tipos de erros:

- Sinalização de olhos abertos quando os olhos estão fechados (Falsos Negativos(FN)).
- Sinalização de olhos fechados quando os olhos estão abertos (Falsos Positivos(FP)).

De posse das imagens salvas via software de testes, foram identificadas as imagens com determinação errônea de estado dos olhos e seus resultados estão dispostos na Tabela 3.

Tabela 3 - Resultados obtidos através do software de testes, detalhando erro total, erro tipo "Falso olhos abertos", erro tipo "Falso olhos fechados" e seus respectivos porcentuais.

Usuário	Total de amostras	Erro total	% Erro total	Falsos Positivos	% FP	Falsos Negativos	% FN
1	232	15	6,47%	8	3,45%	7	3,02%
2	362	17	4,70%	0	0,00%	17	4,70%
3	159	4	2,52%	0	0,00%	4	2,52%
4	268	2	0,75%	0	0,00%	2	0,75%
5	145	3	2,07%	1	0,69%	2	1,38%
6	310	7	2,26%	0	0,00%	7	2,26%

Baseado nos dados da tabela, podemos calcular a taxa de acerto dos testes preliminares do sistema, utilizando a equação:

$$Taxa\ de\ acerto = 1 - \frac{Erro\ total}{Total\ de\ amostras} \quad (3)$$

Calculando desta forma, chegamos a uma taxa de acerto de 96,75%. O desvio padrão entre o resultado entre os usuários é de 1,38%.

Para efeitos de discussão de resultados, é importante frisar que o erro encontrado está majoritariamente relacionado a falsos olhos fechados. Sendo assim, para que esse erro tivesse impacto na determinação de sonolência, ele deveria se perpetuar por mais de 6 quadros seguidos. Analisando as imagens obtidas foi possível verificar que esse tipo de erro é isolado, ou seja, ela acontece em um quadro específico, não se arrastando ao quadro seguinte. Desta forma, não causa danos ao resultado final.

Sendo assim, pode-se calcular a taxa de acerto no sistema baseando-se somente na quantidade de sinalizações de falsos olhos fechados:

$$Taxa\ de\ acerto = 1 - \frac{Falsos\ olhos\ abertos}{Total\ de\ amostras} \quad (4)$$

Desta forma, a taxa de acerto obtida é de 99,39%, com o mesmo desvio padrão 1,38%.

É importante destacar que 89% da recorrência deste tipo de erro foi encontrada nas amostras do usuário 1, que era o único utilizando óculos ao ser submetido aos testes.

6. Conclusão

De posse dos resultados encontrados, conclui-se que o sistema atende os objetivos deste projeto. Destaca-se duas caraterísticas, a alta taxa de confiança dentro dos testes preliminares (erro menor que 4%) e a velocidade na tomada de decisão. As características citadas garantem a eficiência do sistema para a aplicação proposta, pois sabe-se que fração de segundos podem ser essenciais quando falamos de sono ao volante.
Verifica-se o sucesso da escolha da plataforma Odroid-U2. Seu poder de processamento considerando suas dimensões é extraordinário. Além disso, a simplicidade de desenvolvimento, graças ao excelente sistema operacional distribuído gratuitamente em seu site, é outro fator relevante.

Por ter uma documentação completa, de fácil acesso e muito didática é importante citar o OpenCV. O desenvolvimento de um software simples para a tomada de decisão só foi possível graças a todas essas características citadas. Notou-se a diversidade de soluções possíveis de fácil aplicação disponíveis dentro das funções da biblioteca. Com poucas alterações no código desenvolvido seria possível a utilização do sensoriamento das piscadas do usuário do sistema para inúmeras soluções cotidianas.

O custo do projeto como um todo é outro ponto de sucesso. O que torna ainda mais viável a utilização da visão computacional como forma de tomada de decisão para eventos rotineiros.
Resumindo, a utilização do sistema é viável para diversas aplicações que utilizam-se da visão computacional.

7. Trabalhos futuros

É importante frisar que o sistema demanda luminosidade externa frontal, porém para a viabilidade do projeto como um produto comercial seria necessária uma alternativa a este ponto. Nota-se que a maioria dos acidentes de automóveis causados pela sonolência de seus motoristas acontece no período noturno, onde temos a carência de luminosidade.

Como alternativa, sugere-se a utilização de iluminação infravermelha. Esse tipo de iluminação contornaria a falta de luminosidade ambiente, além de melhorar a qualidade dos resultados, já que esse tipo de iluminação ressalta a região dos olhos nas imagens de rosto.

Além disso, considerando o alto poder de processamento da Odroid, é possível a utilização de um webcam com melhor resolução para melhora na qualidade dos resultados.

Referências

ANUND, A. Sleepiness at the wheel. Department of Public Health Sciences. National Prevetion of Suicide and Mental Health. Stockhoml. 2009. Disponível: <http://diss.kib.ki.se/2009/978-91-7409-431-2/thesis.pdf>. Acesso em: 30 set. 14.

BRADSKI, G; KAEHLER, A; Learning OpenCV computer vision with OpenCV library. O Relly. 2008.

BRISTOW, D ; FRITH, C. REES, G. Two distinct neural effects of blinking on human visual processing. NeuroImage. 2005. Disponível: <http://davinabristow.com/PhD_Research_files/Bristow%20D,%20Frith%20C,%20Rees%20G.%20Two%20distinct%20neural%20effects%20of%20blinking%20on%20human%20visual%20processing%20-%20Neuroimage%202005.pdf>. Acesso em: 30 out. 14.

CARTOLA, Agência de Conteúdo - Especial para o Terra; Dispositivos buscam manter motoristas atentos ao trânsito; Terra – Tecnologia; 27 de set. de 2014. Disponível: < http://tecnologia.terra.com.br/dispositivos-buscam-manter-motoristas-atentos-ao-transito,c3080fcc7696b310VgnCLD200000bbcceb0aRCRD.html > Acesso em: 2 de nov. 2014.

CASSINELLI, A.; PERRIN, S. To blink or not to blink. Disponível: <http://toblinkornot.wordpress.com/technical-statement/>. Acesso em: 20 out. 14

FUNDASONO: FUNDAÇÃO NACIONAL DO SONO. Sonolência e direção. Disponível: <http://www.fundasono.org.br/gera_conteudo.asp?materiaID=535>. Acesso em: 12 set. 14.

GONZAGA, A. Notas de aula: SEL 5886, Visão Computacional. Universidade de São Paulo. Escola de Engenharia de São Carlos. Departamento de Engenharia Elétrica e Computação. Disponível: <http://iris.sel.eesc.usp.br/sel886/>. Acesso em: 30 out. 14.

HARDKERNEL. Product Information, Odroid U2. Disponível: <http://www.hardkernel.com/main/products/prdt_info.php?g_code=G135341370451>. Acesso em: 30 de out. de 2014.

HORNG, W. et al. Driver fatigue detection based on eye tracking and dynamic template matching. Sensing and Control, p. 7-12, 2004.

JAYANTHI, D.; BOMMY, M. Vision-based Real-time Driver Fatigue Detection System for Efficient Vehicle Control. International Journal of Engineering and Advanced Technology (IJEAT), ISSN: 2249 – 8958, Volume-2, Issue-1, October 2012. Disponível: <http://www.ijeat.org/attachments/File/v2i1/A0808102112.pdf>. Acesso em: 25 jun. 14

JI, Q; YANG, XIAOJIE. Real-Time Eye, Gaze, and Face PoseTracking for Monitoring Driver Vigilance. Real-Time Imaging 8, 357–377 (2002). Elsevier Science Ltd. Disponível: <http://www.ecse.rpi.edu/~qji/rti.pdf>. Acesso em: 25 jun. 14.

LIENHART, R.; MAYDT, J. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE, 2002. Disponível: <http://nichol.as/papers/Lienhart/An%20Extended%20Set%20of%20Haar-like%20Features%20for%20Rapid%20Object.pdf>. Acesso em: 30 out. 14.

MOREIRA, G; FEIJÓ, B. Reconhecedor de objetos em vídeos digitais para aplicações interativas. Capítulo 3. Dissertação apresentada ao programa de Pós-Graduação em Informática da PUC-Rio como requisito parcial para obtenção do título de Mestre em Informática. Pontifícia Universidade Católica Do Rio De Janeiro - PUC-Rio. 2008. Disponível em: < http://www.maxwell.vrac.puc-rio.br/13069/13069_4.PDF>. Acesso: 3 nov. 14.

OPENCV. Documentation. Disponível: <http://docs.opencv.org/index.html>. Acesso em: 30 de out. de 2014.

MULTILASER. Informações sobre o Produto: Webcam Night Vision - Wc045. Disponível: <http://www.multilaser.com.br/produtos/detalhe/WC045/webcamnight-vision-wc045.html>. Acesso: 30 de out. 2014.

PAPAGEORGIOU, C; POGGIO, T. A Trainable System for Object Detection. International Journal of Computer Vision 38(1), 15–33, 2000. Kluwer Academic Publishers.

QUEIROZ, K. Sistema baseado em vídeo para detecção de sonolência do motorista. Dissertação de Mestrado. Universidade de Brasília. 2011. Disponível:

<http://repositorio.unb.br/bitstream/10482/10704/1/2011_KedsonLopesQueiroz.pdf>. Acesso em: 4 de set. 2014.

SEMINÁRIO INTRODUÇÃO À VISÃO COMPUTACIONAL. Técnicas de Análise de Imagens para o Controle de Qualidade e Inspeção de Produtos Industriais. Universidade Federal de Santa Catarina. Disponível: <http://www.inf.ufsc.br/~visao/qualidade.html>. Acesso em: 30 out. 14.

SCHAPIRE, R. Explain AdaBoost. Bernhard Schölkopf, Zhiyuan Luo, Vladimir Vovk, editors, Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik, Springer, 2013. Disponível: <https://www.cs.princeton.edu/~schapire/papers/explaining-adaboost.pdf>. Acesso em: 2 dez. 14.

TOBIAS, S. O sono ao volante. A sonolência e fadiga ao volante podem ser tão perigosas quanto o álcool. Revista Quatro Rodas On-line, setembro de 2008, Editora Abril, Disponível: <http://quatrorodas.abril.com.br/vai-viajar/reportagens/o-sono-ao-volante/>. Acesso em: 15 out. 14.

VIOLA, P; JONES, M. Rapid Object Detection using a Boosted Cascade of Simple Features. Accepted Conference On Computer Vision And Pattern Recognition. 2001. Disponível: <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>. Acesso em: 30 out. 14.

VOLKSWAGEM. Fale conosco. Perguntas e Respostas. Detector de fadiga. Disponível: <http://www.vw.com.br/pt/fale_conosco/perguntas_e_respostas/detector-de-fadiga.html>. Acesso em: 15 out. 14.

Apêndice A – Código fonte do Software de testes

```
//Bibliotecas necessarias
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/contrib/contrib.hpp"
#include "opencv2/core/core.hpp"
#include "opencv2/features2d/features2d.hpp"
#include "opencv2/nonfree/features2d.hpp"
#include <iostream>
#include <stdio.h>
#include <string.h>

using namespace cv;

//Função detecção de faces e olhos
void detectAndDisplay( Mat quadro );

//Declaração de variáveis
String face_cascade_name = "haarcascade_frontalface_alt.xml";
String eye_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
CascadeClassifier face_cascade;
CascadeClassifier eye_cascade;
string window_name = "TCC - Rafael Ferrassini";
RNG rng(12345);
int sono;
int cont = 0;
char buffer[1000];
int closed = 0;
char text[100];
int facial;

//Funcao main
int main( int argc, const char** argv )
```

```
  {

    CvCapture* capture;
    Mat quadro;

    //1. Carregando os classificadores em Cascata
    if( !face_cascade.load( face_cascade_name ) ){ printf("--(!)Erro - (Classificador de face)\n");
return -1; };
    if( !eye_cascade.load( eye_cascade_name ) ){ printf("--(!)Erro - (Classificador de olho)\n");
return -1; };
   //2. Obtendo as imagens da camera
    capture = cvCaptureFromCAM( -1 );

    if( capture )
    {
      while( true )
      {
      quadro = cvQueryQuadro( capture );
   //3. Chamando a funcao detectAndDisplay para reconhecimento de face e olho
        if( !quadro.empty() )
        { detectAndDisplay( quadro ); }
        else
        { printf(" --(!)Erro na captura de informacoes!"); break; }

        int c = waitKey(10);
        if( (char)c == 'c' ) { break; }
      }
    }
    return 0;
  }

 // Funcao detectAndDisplay
 void detectAndDisplay( Mat quadro )
 {
```

```
facial = 0;
std::vector<Rect> faces;
Mat quadro_gray;
//Os classificadores estão em escala cinza, portanto temos a necessidade de converte-la
para GRAY e equaliza-la para diminuição dos problemas de luminosidade
cvtColor( quadro, quadro_gray, CV_BGR2GRAY );
equalizeHist( quadro_gray, quadro_gray );
//Detectando faces
face_cascade.detectMultiScale( quadro_gray, faces, 1.3, 4,
CV_HAAR_FIND_BIGGEST_OBJECT, Size(40, 40) );
        for( int i = 0; i < faces.size(); i++ )
        { facial = 1;
          Point face1(faces[i].x, faces[i].y);
          Point face2(faces[i].x + faces[i].width, faces[i].y + faces[i].height);
          rectangle( quadro, face1, face2, Scalar( 255, 0, 0 ), 4, 8, 0 );
          Mat faceROI = quadro_gray( faces[i] );
          std::vector<Rect> eye;
          //Em cada face, detecta-se os olhos
          eye_cascade.detectMultiScale( faceROI, eye, 1.1, 3,
CV_HAAR_FIND_BIGGEST_OBJECT, Size(30, 20) );
          closed = 0;
             if (eye.empty())
          closed = 1;
          else
          {
             for( int j = 0; j < eye.size(); j++ )
                {
              //Validação do estado dos olhos
              Mat eyeROI = faceROI( eye[j] );
              eyeROI = eyeROI(Rect(eye[j].width/4, eye[j].height/4, eye[j].width/2,
eye[j].height/2));
              Size size(5*eye[j].width/2, 5*eye[j].height/2);
              equalizeHist( eyeROI, eyeROI );
              resize(eyeROI, eyeROI, size);
```

```
                Threshold(eyeROI, eyeROI, 235, 255, CV_THRESH_BINARY);
                eyeROI = eyeROI(Rect(0, (eyeROI.rows/2)-5, eyeROI.cols, 10));
                double minVal, maxVal;
                minMaxLoc(eyeROI, &minVal, &maxVal);
                if (maxVal == 0)
                closed = 1;
                else
                closed = 0 + closed;
                Point center( faces[i].x + eye[j].x + eye[j].width*0.5, faces[i].y + eye[j].y +
eye[j].height*0.5 );
                 int radius = cvRound( (eye[j].width + eye[j].height)*0.25 );
                 circle( quadro, center, radius, Scalar( 0, 255, 0 ), 4, 8, 0 );
                  }
            }
        }
  cont ++;
  //Exibindo e salvando a imagem final
  if (facial == 0)
  sprintf(text, "Rosto nao encontrado");
  else
  {
   if ( closed == 0 )
   sprintf(text, "Olhos abertos");
   else
   sprintf(text, "Olhos fechados");
  }
  putText( quadro, text, Point(20,20), CV_FONT_NORMAL, 1, Scalar(255, 0, 0), 1, 1);
  imshow( window_name, quadro);
  sprintf(buffer, "face%u.jpg", cont);
  imwrite(buffer,  quadro);
  }
```

Anexo A

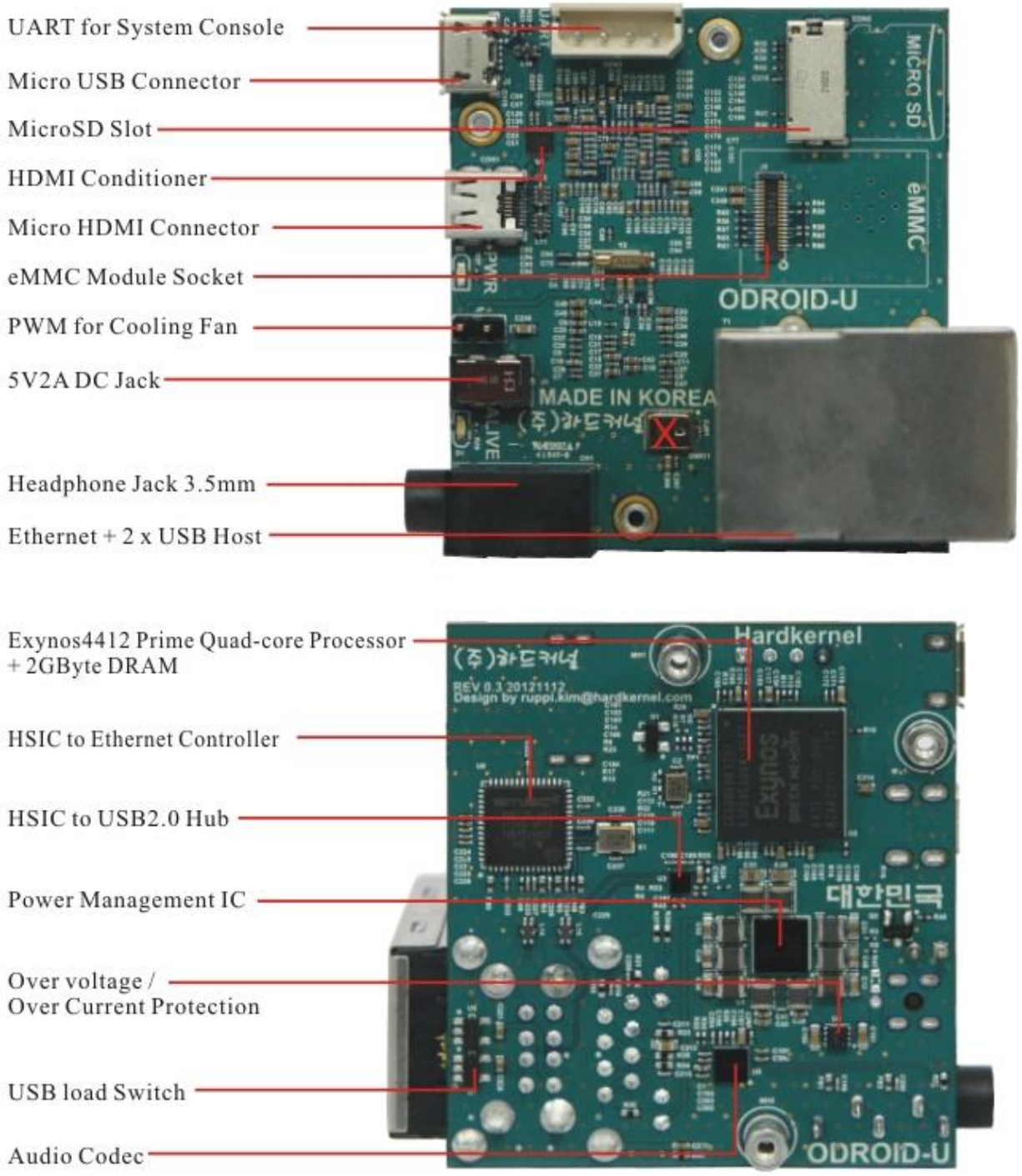

Figura 35 - Detalhamento dos componentes existente na Odroid U2. (HARDKERNEL)

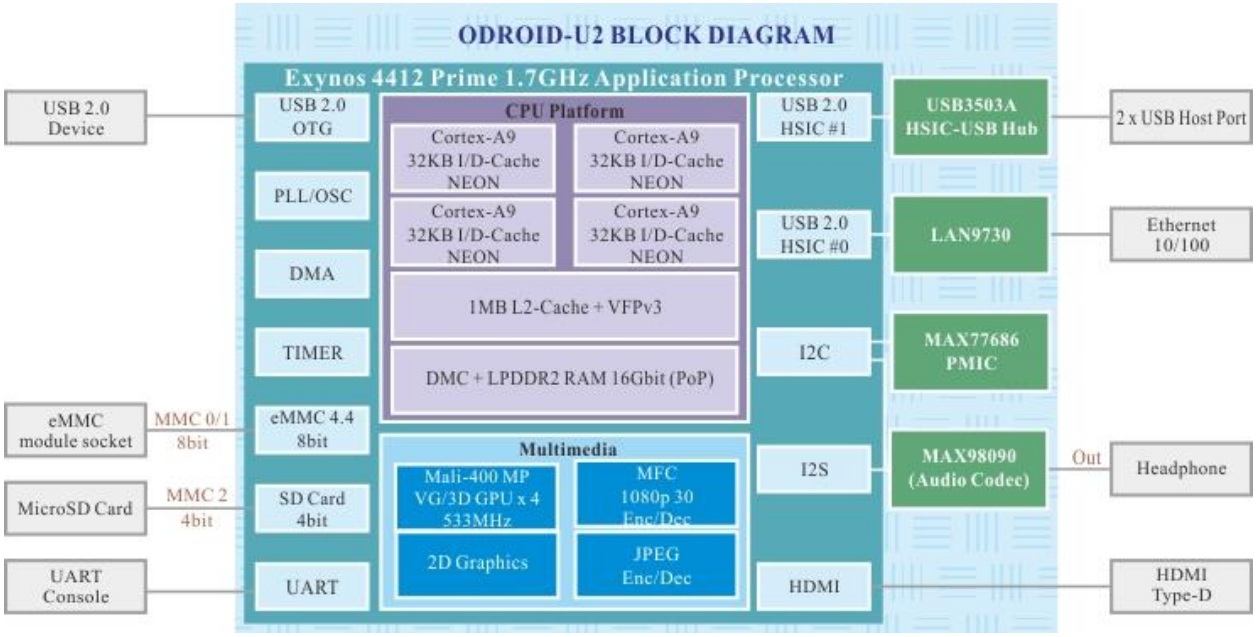

Figura 36 - Diagrama de blocos da Odroid U2. (HARDKERNEL).