UNIVERSIDADE FEDERAL DE SANTA CATARINA CENTRO TECNOLÓGICO

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA CURSO DE CIÊNCIAS DA COMPUTAÇÃO

MONITORAÇÃO DE SONOLÊNCIA COM IPHONE VIA PROCESSAMENTO DE IMAGENS EM TEMPO REAL

Cezar Augustus Signori

José João Junior

Florianópolis - SC

2011/1

Cezar Augustus Signori

José João Junior

Monitoração de Sonolência com iPhone via Processamento de Imagens em Tempo Real

Trabalho de Conclusão de Curso Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciências da Computação

Orientador: Professor Aldo Wangenheim

BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CENTRO TECNOLÓGICO

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Florianópolis

Junho 2011/1

“O Conhecimento custa tanto quanto a ignorância e é no mínimo tão satisfatório quanto”

Barrin, O Mago Mestre

AGRADECIMENTOS

Agradeço ao meus pais por terem me ajudado a ser quem sou. Agradeço também à professora Maria da quarta série do primário, pode ter se negado a me explicar o que é um átomo. Agradeço à Day por ter me ajudado com o Word, pois sem ela este trabalho não teria um sumário. Agradeço ao JJ por ter contribuído significativamente com este trabalho. Mas hoje não é Dia de Ação de Graças, então eu termino agradecendo pela comida. ;)

- Cezar Signori

Agradeço ao meus pais pois sem eles eu não estaria aqui. Agradeço a minha esposa Priscilla por ter aturado o meu mau humor após noites em claro, não somente após noites em claro. Agradeço minhas sobrinhas que fazem a vida parecer tão mais simples e bonita. Agradeço as gigantescas filas do RU que sempre possibilitaram um ambiente perfeito para discussões sem sentido. Também agradeço ao meu melhor e mais esquisito amigo Cezar pois sem ele estes anos de faculdade não teriam sido tão divertidos.

- José João Junior

RESUMO

Sonolência ao volante representa um importante risco nas rodovias, dado que esta é uma das principais causas de acidentes de trânsito.

Neste trabalho é desenvolvido um sistema de monitoração de sonolência em tempo real que executa em um dispositivo móvel que possa ser aclopado a qualquer veículo de 4 ou mais rodas, avisando-o progressivamente sobre sua condição de forma que este possa corrigir seu comportamento ou parar de dirigir. Para tanto, técnicas de processamento de imagens são utilizadas para localizar os olhos do motorista e contabilizar estatísticas relacionadas à alterações da frequência de piscadas das pálpebras.

Palavras-chave: Sonolência do Motorista, Análise da Frequencia de Piscadas, Processamento de Imagens, iOS

ABSTRACT

Driver sleepiness is one of the most important and maybe common causes of car accidents during the daytime today.

In this work a mobile system for sleepiness monitoring in real time is developed aiming to be used inside vehicles with 4 or more tires. The system advises the driver about his state of vigilance so that he can fix his behavior or stop driving.

In order to accomplish these goals, image processing techniques are used to recognize the driver's eye's region so that we are able to get blink frequency related statistics to infer the driver's vigilant state.

Keywords: Driver Sleepiness, Blink Frequency Analysis, Image Processing, iOS

Sumário

Lista de Figuras

1.Introdução

Sonolência ao volante representa um importante risco nas rodovias, dado que esta é uma das principais causas de acidentes de trânsito. Este fato tem sido reportado por vários estudos que estabeleceram fortes relações entre a sonolência do motorista e acidentes de trânsito. Aldrich por exemplo, reportou que mais de 13% dos acidentes de trânsito se devem à sonolência do motorista [1]. Ele também aponta um estudo de Lafont, que afirma que 34% dos acidentes são causados por sonolência.

Os resultados acima mencionados demonstram a importância de uma pesquisa com o objetivo de reduzir o risco de acidentes com esta causa.

Até então vários estudos tem tentado modelar o comportamento de um motorista sonolento através do estabelecimento de relações entre a sonolência e certos parâmetros relacionados ao veículo e ao próprio motorista (e.g., velocidade do veículo, posição das rodas, etc) [2, 3, 4, 5, 6]. Estes estudos foram eventualmente a base para desenvolvimentos futuros, visto eles são limitados a determinar quais parâmetros realmente estão relacionados à condição de sonolência.

O objetivo deste trabalho é desenvolver um sistema de monitoração de sonolência em tempo real que execute em um dispositivo móvel que possa ser aclopado a qualquer veículo de 4 ou mais rodas, avisando-o progressivamente sobre sua condição de forma que este possa corrigir seu comportamento ou parar de dirigir.

Este trabalho organiza-se da seguinte maneira: Primeiro são apresentados e justificados os parâmetros utilizados para a detecção da sonolência. Segue-se com a descrição superficial da plataforma, e detalhada da solução proposta, incluindo os testes de validação. Então os resultados são apresentados e analisados e

finalmente são apresentadas as conclusões e trabalhos futuros.

1.1.Objetivos

Os objetivos do presente projeto foram subdivididos em objetivo geral e objetivos específicos, como apresentados a seguir.

1.1.1.Objetivo Geral

O objetivo geral do presente projeto é o desenvolvimento de um sistema de monitoração de sonolência em tempo real via processamento de imagens em um dispositivo móvel aclopado ao veículo.

1.1.2.Objetivos Específicos

Para alcançar o objetivo geral é necessário que se atinja os seguintes objetivos específicos:

· Realizar 2 revisões sistemáticas de literatura, procurando validar os parâmetros relacionados ao piscar das pálpebras como indicadores confiáveis de sonolência, bem como avaliar a viabilidade de realizar o processamento de imagens necessário em uma plataforma móvel e avaliar o estado da arte

· Construir um referencial teórico sobre o tema;

· Levantar soluções existentes para problema da sonolência no trânsito;

· Definir um conjunto de requisitos funcionais e não funcionais que o sistema proposto deve atender, com especial atenção aos requisitos ligados à identificação do estado de sonolência em tempo real;

· Modelar os processos e o sistema;

- Desenvolver o projeto Piloto;
- Testar e validar o conjunto.

1.2.Justificativa

De acordo com a Highway Traffic Safety Administration dos Estados Unidos, aproximadamente 4700 desastres envolvendo veículos no ano de 2000 tem como causa a falta de atenção do motorista quanto ao trânsito [7]. Destes, cerca de 1700 estão relacionados à presença de sonolência do motorista.

Detectar a falta de atenção visual cedo o suficiente para alertar o motorista sobre a sua falta de atenção devido à sonolência pode diminuir consideravelmente o índice de mortes e/ou gastos da população. Se faz, portanto, necessário explorar novas tecnologias para resolver o problema.

1.3.Metodologia

O estudo iniciará com a construção do referencial teórico, levantamento de requisitos e modelagem da aplicação. Para a construção do referencial teórico serão desenvolvidas três Revisões Sistemáticas de Literatura seguindo o modelo de Kitchenham [8] a fim de responder a três pontos determinantes. O primeiro diz respeito à abordagem biológica da solução proposta para detectar a presença do estado de sonolência do indivíduo sendo avaliado. E deve indicar se a observação do ritmo de piscadas e tempo que os olhos permanecem fechados é um indicador confiável para determinação do estado de sonolência. O segundo ponto a ser levantado é a possibilidade de identificar e processar em tempo real os olhos em um rosto para a determinação dos fatores de indicação de sonolência. O último e não

menos importante fator, é a possibilidade de realizar este tipo de processamento de imagens em tempo real num dispositivo móvel de baixo poder de processamento.

O referencial teórico servirá como auxilio no levantamento dos requisitos funcionais da solução, bem como na modelagem onde serão utilizadas tabelas comparativas de soluções já existentes levantadas pelas Revisões Sistemáticas de Literatura realizadas na fase anterior do projeto.

Uma vez realizada a modelagem da solução inicia-se a implementação de um sistema que seja executado em um dispositivo móvel capaz de ser utilizado em um veículo e que possua câmera para monitoração do estado de sonolência via processamento de imagem.

1.4.Limitações do Trabalho

No que tange ao trabalho como um todo deve-se ressaltar que a proposta está restrita à uma implementação por software embarcado para a plataforma iOS de uma solução de detecção e alarme de sonolência de um indivíduo durante o período diurno, onde o ambiente é bem iluminado e levando em consideração apenas alguns fatores determinantes do estado de sonolência.

2.Referencial Teórico

Para a concretização da primeira etapa do presente projeto realizou-se um estudo teórico a cerca dos assuntos mais relevantes dentro do atual contexto.

O estudo teórico começa na determinação de um conjunto de indicadores de sonolência restrito à região facial que sejam fatores confiáveis, como as alterações no ritmo de piscadas e o tempo em que os olhos do indivíduo em observação permanecem fechados. Segue-se o estudo com os algoritmos e técnicas utilizadas em trabalhos similares, restringindo o foco às soluções baseadas em processamento de imagens e destinadas à aplicativos móveis para dispositivos embarcados de alta tecnologia. Por último, será explanado a cerca da plataforma, das ferramentas, metodologias e padrões uitlizados na implementação do projeto. Cada um destes assuntos encontra-se nas seguintes seções.

2.1. Estado da Arte

Para fins de revisão do estado da arte, vários trabalhos correlatos foram avaliados. Quando se diz respeito ao diagnóstico de sonolência ou de estados fisiológicamente similares como a fadiga ou efeitos relacionados à monotonia no trânsito, a maioria das soluções buscam realizar análises livres de contato, através do processamento de imagens capturadas enquanto o motorista está ao volante. Cada solução busca informações diferentes quando da análise das imagens, mas em geral todos os trabalhos fazem verificações dos movimentos da cabeça ou das expressões faciais e dos olhos.

Estas verificações são realizadas pois constatou-se que determinados movimentos da cabeça, e dos olhos começam a ocorrer quando o indivíduo está a beira de

perder ou já perdendo alguns importantes aspectos de observação e reação, necessários para tarefas específicas como dirigir um veículo. Alguns trabalhos ainda fazem uso de outros fatores não precursores da sonolência ou fadiga, mas que deixam de acontecer quando um destes estados fisiológicos é atingido pelo motorista. Um exemplo seria o motorista não mais verificando com frequência os retrovisores do veículo ou não olhando para a frente mas sim para a paisagem ao redor ou para baixo, indicando a falta de atenção do mesmo quando ao trânsito.

Em relação aos precursores fisiológicos da sonolência ou fadiga, os fatores unipresentes na determinação do diagnóstico são parâmetros referentes à frequencia do piscar das pálpebras, como por exemplo o tempo decorrente entre uma piscada e outra ou ainda o tempo que as pálpebras permanecem fechadas.

Do ponto de vista computacional, a verificação dos parâmetros precursores da sonolência é realizada em etapas. Geralmente inicia-se com a detecção da região da face à fim de reduzir o escopo de procura pelos olhos - algumas soluções ainda fazem uso da região facial para identificar a posição da cabeça, tornando possível avaliar o nível de atenção do motorista para com o trânsito. Esta redução de escopo é realizada gradativamente, ou seja, reduz-se a área da imagem a ser avaliada à região da face. Para tanto, os algoritmos mais utilizados são a Diferenciação Temporal e o AdaBoost com caracterítiscas Haar-like. O primeiro, proposto por Turk em 1991 [21] explora o fato de que seres humanos não conseguem manter a cabeça completamente parada por longos períodos de tempo. O segundo algoritmo de destaque, AdaBoost, é um algoritmo de aprendizado de máquina adaptativo que faz uso de características fracas para descrever características mais fortes, formulado por Yoav Freund e Robert E. Schapire em 1997 [22], este algoritmo tem sido

amplamente utilizado para reconhecimento de padrões em imagens em aplicações de tempo real. Dentre as características utilizadas juntamente ao AdaBoost, destacam-se as características Haar-like conforme proposto por P. Viola e M. Jones em 2001 [23].

Após a redução do escopo para a região facial, aplica-se algum algoritmo para a detecção da região dos olhos. A partir daí, vários parâmetros podem ser contabilizados, desde a frequência de piscadas, a duração e o intervalos entre as mesmas, até o mensuramento do diâmetro das pupilas.

Uma vez contabilizadas as estatísticas mensionadas, cada trabalho utiliza-se de métricas estabelecidas em estudos correlatos ou em experimentos realizados durante os estudos para diagnosticar a presença de sonolência, fadiga ou falta de consciência do motorista. Uma medida presente em praticamente todas as soluções em tempo real é o PERCLOS, que diz respeito ao percentual de fechamento palpebral sobre a pupila ao longo do tempo e reflete o lento fechamento das pálpebras ("droops") ao invés do piscar dos olhos. Referente a esta medida, a métrica mais comum entre as soluções existentes foi estabelecida em 1994 e definida como a proporção de tempo em um minuto em que os olhos estão pelo menos 80% fechados [20].

2.2.Fundamentos Fisiológicos

Sonolência no trânsito é uma preocupação emergente entre pesquisadores na área de segurança no trânsito. Do ponto de vista psicológico e médico, a sonolência reduz a atenção do motorista, aumentando o risco de ocorrência de acidentes no trânsito. Esta sessão se dedica a descrever e identificar o estado de sonolência bem

como descorrer brevemente sobre suas causas.

2.2.1.Sonolência

A sonolência caracteriza-se pelo decréscimo da memória, tempo de reação e processamento de informação, bem como inlfuencia negativamente na tomada de decisões e no estado de vigilância do indivíduo.

Alguns pesquisadores classificam a influencia da sonolência em estados fisiológicos endógenos e exógenos [15]. Neste contexto, fatores endógenos afetam a base da preparação de um indívido quando realizando uma tarefa específica como por exemplo, dirigir um veículo. Estes estão associados com alterações flutuantes do nível de consciência ou vigilância. Os principais fatores endógenos incluem variações circadianas associadas ao horário do dia, o cansaço gerado pela duração de uma atividade ou ainda problemas de falta de sono.

Bons exemplos de precursores endógenos de sonolência que deteriorizam as capacidades do motorista, são longos períodos de trânsito ou ainda dirigir um veículo durante o meio do período vespertino ou tarde da noite [16].

Quando o nível de consciência cai, várias mudanças fisiológicas aparecem, por exemplo a frequência de piscadas e o tempo que os olhos permanecem fechados aumentam.

2.2.1.1.O Piscar das Pálberas

Uma pessoa adulta pisca cerca de 20 vezes por minuto sendo que cada piscada dura entre 0.2 e 0.3 segundos. Geralmente, piscadas são classificadas em três classes, por exemplo, aqueles que ocorrem periodicamente, aqueles que tem sua

ocorrência forçada por estimulação externa como a variação da intensidade de luz incidida sobre os olhos e aqueles que ocorrem intencionalmente. Além disto, acredita-se que as pessoas tem seu ritmo de piscadas reduzido quando se concentram em algo, bem como aumentam quando ficam nervosas. No que tange a este artigo, cabe-se citar que a frequência de piscadas e o tempo que os olhos permanecem fechados aumentam conforme o nível de consciência cai [17, 18 e 19].

2.3 Fundamentos Tecnológicos

Neste tópico são apresentados conceitos fundamentais para o entendimento do ambiente onde o sistema de detecção e monitoração de sonolência estará atuando, bem como a respeito das ferramentas utilizadas em seu desenvolvimento e das metodologias e padrões de projeto utilizados na concepção da solução.

2.3.1. A Plataforma iOS

Entre os dispositivos móveis com interface gráfica adequada, e recursos essenciais como câmera frontal e capacidades de processamento de imagens em tempo real, destacam-se o iPhone 4 e alguns modelos de dispositivo rodando sobre o Android.

De acordo com um estudo realizado pela Nielsen em setembro de 2010 [13], embora o sistema operacional Android esteja emergindo no mercado, este ainda não superou o Market Share do iOS da Apple que lidera com 28.6% do mercado americano de sistemas operacionais para smartphones.

Desta forma entende-se que o iOS tem uma maior base de usuários, aumentando portanto o potencial da aplicação proposta neste trabalho de diminuir o número de acidentes de trânsito decorrentes à sonolência no volante em periodo diurno.

O sistema operacional iOS também conhecido como iPhone OS até junho de 2010, é o sistema operacional para dispositivos móveis da Apple. Foi originalmente desenvolvido para o iPhone e tem sido extendido para outros dispositivos do fabricante, tais quais o iPod Touch, iPad e Apple TV.

A interface com o usuário disponível no iOS é baseada em toques e gestos. Esta interface é composta por componentes como sliders, switches, entradas de texto e botões. Entre os gestos suportados por UIs - User Interfaces - mais elaboradas, são o swipe, tap, pinch e pinch reverso. Todos estes gestos tendo significados específicos no contexto do sistema operacional e sua interface baseada em toques.

O iOS é dividido em basicamente quatro camadas de abstração: Core OS, Core Services, Media e Cocoa Touch. No que tange à este trabalho, serão abordadas apenas as características do framework AV Foundation, uma das partes integrantes da camada de Media.

2.3.2. Ferramentas

Para desenvolvedor aplicações iOS, a Apple fornece um ambiente de desenvolvimento integrado chamado XCode que provê as ferramentas necessárias para a criação da interface da aplicação e do código de backend que a traz a vida.

Nesta sessão são descritas em alto nível, as ferramentas a serem utilizadas neste trabalho.

2.3.2.1. iPhone Simulator

O sistema de simulação iOS permite ao desenvolvedor compilar e executar aplicações para iPhone ou iPad em um computador convencional. Este ambiente é

utilizado para encontrar e solucionar problemas na aplicação durante os estágios de design e testes durante o desenvolvimento. Obviamente o ambiente de simulação iOS é também utilizado para distribuir os componentes visuais da aplicação em tela, bem como tem seu uso indicado para avaliar o uso de recursos como memória e processador de forma detalhada antes de executar a aplicação em um dispositivo real.

A maior parte do ambiente de simulação iOS é o iOS Simulator. Esta aplicação apresenta a interface com o usuário do iPhone ou iPad em uma janela no computador e provê várias formas de interação com o usuário, dentre as quais pode-se citar o uso do teclado e do mouse para a simulação de toques da tela, gestos e rotação do dispositivo.

2.4. AV Foundation Framework

AV Foundation é um dos vários frameworks que podem ser utilizados para reprodudizr e criar conteúdos multimídia baseados em tempo. O framework provê uma interface em Objective-c que pode ser usada para trabalhar em alto grau de detalhe com arquivos de dados audio-visuais. Além de examinar, criar, editar e re-codificar arquivos de mídia, o AV Foundation framework é capaz de obter fluxo de dados de entrada dos dispositivos de captura multimídia e manipular o video durante a captura ou reprodução do mesmo em tempo real.

2.4.1.Captura de dados multimídia

A gravação da entrada de câmeras e microfones é gerenciada por uma sessão de captura. Esta sessão coordena o fluxo de dados da entrada dos devices para saídas

como por exemplo um arquivo de mídia. Multiplas entradas e saídas podem ser configuradas para uma única sessão, a qual é controlada por mensagens de inicio e pausa do fluxo de dados.

Adicionalmente o framework provê uma layer de pré-visualização para mostrar ao usuário o que está sendo gravado da respectiva entrada de dados multimídia.

Para gerenciar a captura de um dispositivo como a câmera, alguns objetos são compilados para representar as entradas e saídas, e uma instância de AVCaptureSession é utilizada para coordenar o fluxo de dados entre elas.

Basicamente, o necessário para capturar informações multimídia do dispositivo se resume a uma instância de AVCaptureDevice para representar o dispositivo de entrada - por exemplo a câmera, uma subclasse concreta de AVCaptureInput para configurar as portas do dispositivo de entrada, e outra subclasse concreta de AVCapturaOutput para gerenciar a saída para um arquivo de vídeo ou objeto de imagem em memória. E é claro, uma instância de AVCaptureSession para coordenar o fluxo de dados entre as entradas e saídas.

Para muitas aplicações este é o nível de detalhe necessário, mas para algumas operações tais quais o monitoramento de quadros do vídeo gravados em tempo real pela câmera do dispositivo móvel, se faz necessário considerar como são representadas as várias portas de um dispositivo de entrada e como estas portas estão conectadas aos dispositivos de saída.

Uma conexão entre a entrada da captura e sua respectiva saída em uma sessão de captura é representada por um objeto da classe AVCaptureConnection. Entradas de captura (instâncias de AVCaptureInput) possuem uma ou mais portas de entrada (AVCaptureInputPort), assim como instâncias de AVCaptureOutput podem aceitar

dados de uma ou mais fontes (por exemplo, AVCaptureMovieFileOutput aceita dados de aúdio e vídeo).

Quando uma entrada ou saída é adicionada a uma sessão, esta sessão forma conexões entre todas as portas de entrada e portas de saída compatíveis. Essas conexões são representadas por objetos do tipo AVCaptureConnection.

2.4.2.Captura de Video em Tempo Real

Para realizar a captura de conteúdo multimídia em tempo real, uma sessão de captura deve ser criada através da instanciação de um objeto do tipo AVCaptureSession. Este objeto é utilizado para coordenar o fluxo de dados do dispositivo entrada, no caso a câmera para as saídas.

A seguir, cria-se um provedor de dados de entrada para fornecer dados de vídeo para a sessão de captura atráves da criação de um objeto AVCaptureDeviceInput. Este dispositivo de entrada deve ser adicionado à sessão através do método addInput.

Um destino de saída deve ser atribuído à sessão de captura de forma a trabalhar sobre os dados de vídeo capturados do dispositivo de entrada. Esta saída é criada através da instância de AVCaptureVideoDataOutput e sua adição à sessão de captura representada pelo objeto AVCaptureSession através do método addOutput.

AVCaptureVideoDataOutput é a subclasse concreta de AVCaptureOutput selecionada pois ela realiza o processamento de descompressão dos frames do vídeo sendo capturado. Uma instância deste objeto produz frames de vídeo que podem ser processados por outra APIs multimídia. Os frames produzidos por este objeto podem ser acessados via o método delegado captureOutput:

didOutputSamleBuffer: fromConnection: percentence ao protocolo AVCaptureVideoDataOuputSampleBuffer -Delegate que deve ser adotado pela implementação para a interceptação dos callbacks fornecidos pela API.

A qualidade da saída pode ser especificada pelo atributo sessionPreset e o inicio e parada do fluxo de dados da entrada para a saída é controlado pela invocação dos métodos startRunning e stopRunning pertencentes ao objeto que representa a sessão de captura.

Cada exemplar de imagem obtido dessa forma é representado pela estrutura CMSampleBufferRef e pode então ser convertido para a representação Cocoa Touch de imagens, ou seja, UIImage de forma facilitar o tratamento da imagem via algoritmos desenvolvidos neste trabalho.

2.5.Metodologias e Padrões de Projeto

De acordo com o iOS Application Programming Guide [14], toda aplicação iOS é construida utilizando o UIKit framework e possui essencialmente a mesma arquitetura. UIKit provê os objetos essenciais para executar a aplicação, coordenar o tratamento da entrada de dados do usuário e exibir conteúdo em tela. A forma como uma aplicação se diferencia de outra em termos de arquitetura é como ela organiza estes objetos e onde eles são incorporados em objetos customizados para amplificar a experiência do usuário em termos de interface e comportamento da aplicação.

Existem muitas interações ocorrendo entre o sistema e a aplicação em execução, e muitas destas interações são automaticamente tratadas pela infraestrutura do UIKit. Porém, existem momentos em que a aplicação precisa estar ciente dos eventos vindos do sistema operacional. Por exemplo, quando o usuário termina a aplicação,

esta precisa ser notificada para que seja possível armazenar qualquer informação relevante que a aplicação esteja fazendo uso. Para situações como esta, o UIKit faz uso de vários mecanismos que requerem que a aplicação siga determinados design patterns, dentre os quais se destacam o MVC e o Delegation.

2.5.1.Model View Controller

Model View Controller – MVC é um padrão de projeto de software, cuja objetivo é separar a interface gráfica da camada de dados, ou seja, a lógica de negócio, a lógica de apresentação e métodos de processamento são definidas e declaradas em locais distintos. Segundo Leff e Rayfield (2001), o padrão de projetos MVC defendem que a aplicação do MVC facilita o desenvolvimento e a manutenção de software desde que:

- A “aparência” da aplicação possa ser modificada sem alterar suas estruturas de dados e a sua lógica.
- A aplicação consiga manter facilmente diferentes interfaces.

A camada Model representa uma abstração de um domínio específico real em uma aplicação, ou seja, os dados propriamente ditos. Esta camada não tem nenhum vínculo ou conhecimento sobre a interface com o usuário. A representação dos elementos da camada Model é definida, ou seja, disposta na tela, na camada View. Para cada View existe um Controller associado responsável por todas as ações possíveis que o usuário pode executar. Cada Model pode possuir mais de uma View associada, e essas Views podem ser diferentes entre si, pois o mesmo conjunto de

dados pode ser representado de muitas formas distintas. (VEIT e HERRMANN, 2003)

2.5.2.Delegate

O padrão de projeto Delegation define uma maneira de modificar o comportamento de objetos complexos sem a necessidade de extendê-los via criação de subclasses. Ao invés disto, o objeto é utilizado tal como foi implementado e qualquer código customizado para a modificação do comportamento daquele objeto é descrito em um objeto separado, o qual é referenciado como o objeto delegado pelo objeto a ter seu comportamento modificado. Ou seja, em momentos pré-definidos, o objeto complexo realiza chamadas a métodos do objeto delegado dando a este a chance de executar código customizado.

3. Sistema de Monitoração Visual

Este trabalho se propõe a desenvolver um sistema de monitoramento do nível de sonolência do motorista em tempo real através do processamento de imagens capturadas pela câmera de um dispositivo móvel. Estas imagens são utilizadas como entrada para o algoritmo de avaliação do AdaBoost já treinando para o reconhecimento de um dos olhos. Em seguida faz-se uso de mesmo algorimo com um classificador em cascata desenvolvido para decidir se o olho se encontra total ou parcialmente fechado e aberto. Os resultados obtidos dessas classificações são contabilizados a fim de calcular o PERCLOS [20],

indice utilizado por este trabalho para determinar o estado de sonolência do indivíduo em análise. O sistema de monitoramento alerta o usuário visual e

sonoramente, uma vez classificado como sonolento ou à beira da sonolência, a fim de atrair a atenção do motorista.

3.1. Detecção do Olho

A detecção do olho é realizada através de um método baseado em características Haar-like, proposto por Viola e Jones [23] em 2001, onde uma janela é movida pela imagem em análise e para cada subseção da imagem suas correspodentes características Haar-like são calculadas. Esta diferença é comparada a um limite de aprendizado, separando olhos de não-olhos.

A principal vantagem de utilização destas características é a velocidade de cálculo. Devido à representação integral da imagem, uma característica Haar-like de qualquer tamanho pode ser calculada em tempo constante.

No entanto em um número qualquer de sub-janelas de uma imagem, o número total de características Haar-like é muito grande, algumas vezes superior do que a quantidade de pixels. Para assegurar uma rápida classificação, o processo de aprendizagem deve excluir uma larga quantidade de características e focar em um conjunto menor de características dadas como críticas para a detecção do olho. Esta seleção é feita através de uma modificação do AdaBoost [24], onde o aprendiz fraco (weak learner) é restringido de tal forma que cada classificador fraco (weak classifier) retornado pode depender de apenas uma característica. Como resultado, cada estágio do AdaBoost que seleciona um novo classificador fraco, pode ser visto como um processo de seleção de uma característica.

Outra contribuição importante do trabalho de Viola e Jones [23], é o treinamento e avaliação destes classificadores em uma cascada, aumentando drasticamente a

velocidade do detector pois mantém o foco de atenção em regiões promissoras da imagem. A noção por trás de abordagens baseadas em foco de atenção é que geralmente é possível determinar mais rapidamente onde um objeto pode ocorrer em uma imagem [25]. Desta forma, computações mais complexas podem ser realizadas apenas nas regiões selecionadas.

Este tópico descreve um processo para treinamento de um classificador simples e eficiente que pode ser utilizado como um operador supervisionado de foco de atenção - treinado para detectar exemplos de uma classe em particular.

Viola e Jones [23] relatam em seu artigo de detecção de faces que é possivel obter menos de 1% de falsos negativos e 40% de falsos positivos utilizando um classificador construído com duas características Haar-like. Portanto o efeito deste filtro é reduzir pela metade o número de locais onde o detector final deve ser avaliado.

Aquelas janelas que não forem rejeitadas pelo classificador inicial são processadas por uma sequência de classificadores, cada um um pouco mais complexo que o anterior. Se algum classificador rejeitar a janela, nenhum processamento adicional é realizado.

3.1.1. Características Haar-like

Existem muitas motivações para o uso de características ao invés do uso direto de pixels. Além de concentrar conhecimento de domínio de difícil aprendizado quando utilizando uma quantidade finita de dados para o treinamento, elas operam em uma velocidade maior do que sistemas baseados em pixels.

Estas caracterísitcas são remanescentes da funções Haar utilizadas por Papageorgiou [26]. Mais especificamente, este trabalho faz uso de seis diferentes tipos de características, apresentadas Figura 1. O valor de uma *característica de dois retângulos* é a diferença entre a soma dos pixels das duas regiões retangulares. Esta característica pode comportar dois retângulos com as mesmas medidas na vertical ou horizontal. A *característica de três retângulos* calcula a soma de pixels dos retângulos mais externos e deduz do somatório do retângulo mais interno. Novamente esta característica é encontrada na forma vertical e horizonta, com a mesma forma e tamanho. Uma *característica de quatro retângulos* que computa a diferença entre os pares diagonais de retângulos. E finalmente uma característica de retângulos concêntricos que calcula a diferença de área entre o retângulo mais externo e o mais interno.

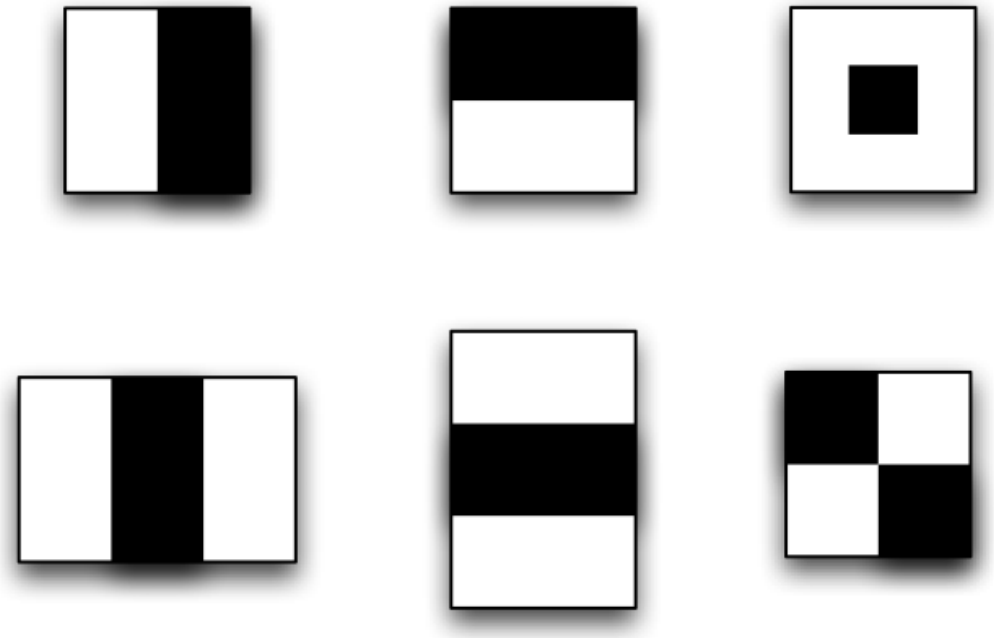

Figura 1 – Conjunto de características Haar-like utilizadas no projeto.

3.1.2. Imagem Integral

Conforme descrito anteriormente, a fim de otimizar o cálculo das características retangulares este trabalho faz uso de uma representação intermediária da imagem chamada Imagem Integral (Figura 2). A imagem integral na posição (x,y) contem a soma dos pixels acima e a esquerda de (x,y) na imagem original, ou seja:

$$I(x,y) = \sum_{x\prime \leq x \, \wedge y\prime \leq y} i(x',y')$$

onde I(x,y) é a imagem integral e i(x,y) é a imagem original. Esta representação pode ser computada em apenas uma passada na imagem original, através das seguintes recorrências:

$$s(x,y) = s\,(x,y-1) + i(x,y)$$

$$I(x,y) = I(x-1,y) + s(x,y)$$

onde s(x,y) é a soma cumulativa da linha, s(x, -1) = 0 e I(-1, y) = 0.

Uma vez computada a imagem integral, qualquer soma retangular pode ser computada com quatro referências, ou seja, a diferença entre duas somas retangulares pode ser calculada com o dobro de referências. Maiores otimizações podem ser obtidas caso as características envolvam somas retangulares adjacentes, ocasionando em uma baixa de duas referências ou mais, dependendo de quantas características retangulares são utilizadas.

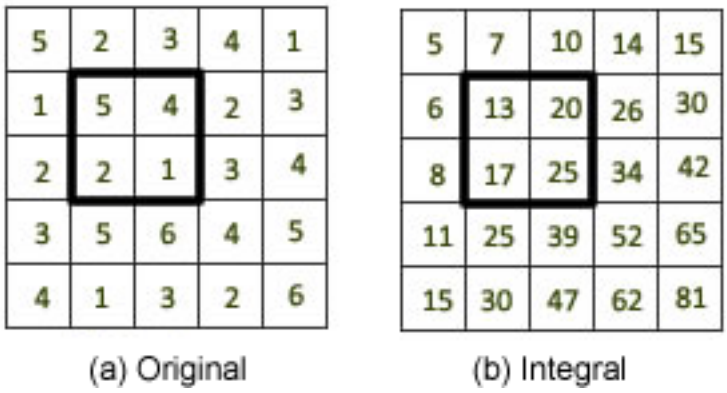

Figura 2 – Representações de Imagens: (a) Imagem Original; (b) Imagem Integral

3.1.3. Aprendizado de funções classificadoras

Dado um conjunto de características e um conjunto de trainamento composto de imagens positivas e negativas, qualquer abordagem baseada em aprendizado de máquina poderia ser utilizada para aprender a função de classificação. Neste trabalho uma variante do AdaBoost é utilizada para selecionar uma pequeno conjunto de características e treinar o classificador [24]. Na sua forma original o algoritmo de aprendizado AdaBoost é utilizado para impulsionar a performance da classificação de um algoritmo de aprendizado fraco. Existem várias garantias formais fornecidas pelo AdaBoost. Schapire provou que o erro de treinamento de um classificador forte se aproxima de zero exponencialmente ao número de rodadas. Mais do que isso, resultados foram posteriormente apresentados sobre performance de generalização. O aspecto chave é que performance de generalização é relacionado à margem de exemplos, e o AdaBoost atinge longas margens rapidamente.

Visto que existem mais características retangulares do que pixels na imagem,

mesmo que cada característica seja eficientemente computada, computar o conjunto completo é proibitivo. Viola e Jones [23] concluiram através de experimentação que um número muito pequeno destas características pode ser combinado para formar um classificar efetivo.

Para encontrar este conjunto, o algoritmo de aprendizado fraco é criado para selecionar a característica retangular que melhor separa os exemplos positivos dos negativos. Para cada característica, o aprendiz fraco determina o limite ótimo da função de classificação, de forma que o menor número de exemplos são classificados erroneamente. Um classificar fraco hj(x) consiste portanto de uma característica fj, um limite oj e uma paridade indicando a direção do sinal de não-igualdade:

$$h_j(x) = \begin{cases} 1, & se\ p_j\ f_j(x) = p_j\ \Theta_j \\ 0, & caso\ contrário \end{cases}$$

Onde x é uma sub-janela da imagem.

As características selecionadas nas primeiras rodadas do AdaBoost possuem taxas de erro entre 0.1 e 0.3, subindo para até 0.5 em rodadas posteriores quando os classificadores se tornam mais complexos.

Vide a Figura 3 para referência ao algoritmo AdaBoost proposto por Viola e Jones [23] em 2001 e utilizado neste trabalho.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:

  1. Normalize the weights,

     $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

     so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:

     $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

     where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

Figura 3 – AdaBoost conforme proposto por Viola e Jones

3.1.4. Organização em Cascada

Visando uma melhoria incremental de performance, além da diminuicão do tempo de computação, esta seção descreve um algoritmo para a contrução de uma cascada de classificadores. Basicamente constrói-se classificadores menores e portanto mais eficientes destinados a rejeitar várias das sub-janelas negativas enquanto detecta quase todas as instâncias positivas, ou seja, o limite de um classificador pode ser

ajustado de tal forma que a taxa de falsos negativos seja próxima de zero. Então faz-se uso dos classificadores mais simples para rejeitar a maioria das sub-janelas antes que os classificadores mais complexos sejam invocados, diminuindo a taxa de falsos positivos.

Este processo é chamado de cascata pois tem a forma de uma árvore de decisão. Um resultado positivo do primeiro classificador dispara a avaliação de um segundo classificador, o qual também foi ajustado para obter altas taxas de detecção. Um resultado positivo do segundo classificador dispara um terceiro classificador e assim por diante. Um resultado negativo em qualquer ponto resulta na imediata rejeição da sub-janela sob avaliação.

Estágios em cascata são construídos através do treinamento de classificadores utilizando AdaBoost, tendo seus limites ajustados de forma a minimizar a ocorrência de falsos negativos. Note que o limite padrão do AdaBoost é definido de forma a obter uma baixa taxa de erros durante o treinamento. No geral, um limite baixo resulta em uma taxa de detecção maior, bem como em maiores taxas de falsos positivos.

Um classificador forte de duas características, por exemplo, pode resultar num excelente classificador para o primeiro estágio, através da redução do limite a fim de minimizer falsos negativos. Quando medido contra um conjunto de validação do treinamento, o limite pode ser ajustado para detectar 100% dos olhos com uma taxa de falsos positivos próxima a 40%.

Segundo Viola e Jones [23], a computação realizada por um classificador de duas características soma cerca de 60 instruções de microprocessador. Por comparação,

uma rede neural de uma camada requer pelo menos 20 vezes mais operações por sub-janela.

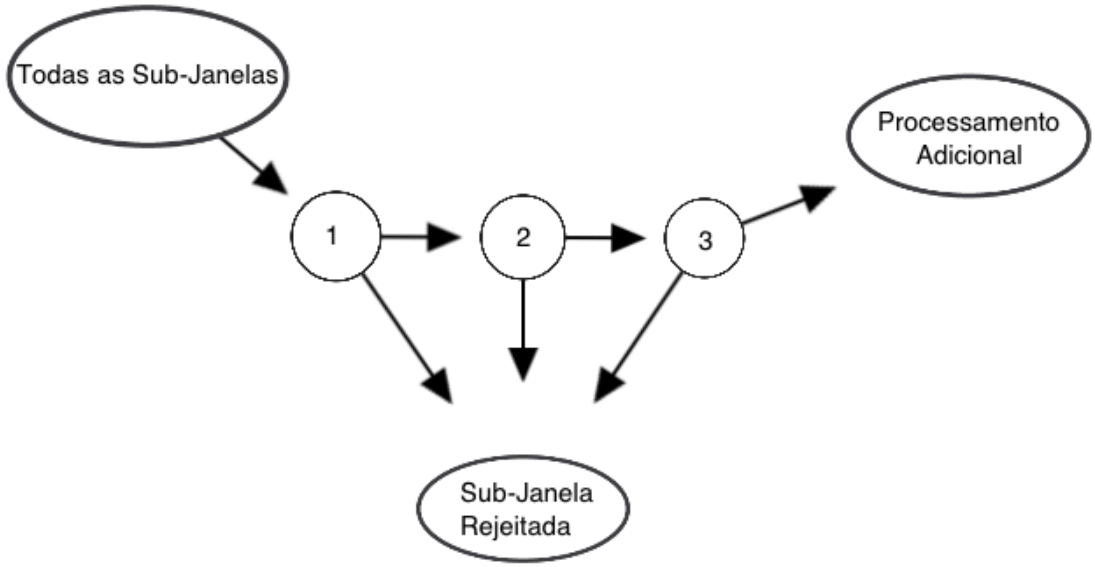

Figura 4 – Representação da Cascada de Classificadores

A estrutura de cascata reflete o fato de que dentro de uma única imagem, existe uma imensa quantidade de sub-janelas negativas. E portanto, procura-se rejeitar o máximo possível de janelas o mais cedo possível, enquanto uma ocorrência positiva ocasiona na execução de cada classificador na cascata (Figura 4).

Exatamente como uma árvore de decisão, classificadores subsequentes são treinados utilizando aqueles exemplares que passaram por todos os estágios anteriores. Como um resultado, o segundo classificador enfrenta uma tarefa mais árdua do que o primeiro. Isso resulta em taxas de falso positivos mais altas conforme os classificadores descem a cadeia.

3.1.4.1 Treinando uma cascata de classificadores

Na maioria dos casos, classificadores com mais características atingem maiores taxas de detecção e taxas mais baixas de falsos positivos. Ao mesmo tempo classificadores com mais características requerem mais tempo de processamento. Por este motivo se faz necessário achar uma combinação ótima do número de estágios, do número de características em cada estágio e o limite de cada estágio.

A fim de produzir um classificador efetivo e altamente eficiente, cada estágio na cascada reduz a taxa de falsos positivos e de detecção. Cada estágio é treinado pela adição de características até que o objetivo de detecção e taxas de falso positivos são atingidas (testando o detector com um conjunto de validação). Finalmente, estágios são adicionados até que o objetivo de detecção e falsos positivos sejam atingidos.

3.2. Detecção de Sonolência

Esta seção introduz a métrica utilizada na inferência sobre o estado de sonolência do motorista durante o trânsito, bem como descreve minunciosamente como esta métrica é contabilizada e utilizada pela solução proposta.

3.2.1 Perclos

Dentre os fatores precursores da sonolência avaliados neste estudo, a medida referida como PERCLOS é a mais confiável e válida para a determinação do nível de consciência do motorista. PERCLOS é o percentual de fechamento da palpebra sobre a pupila ao longo do tempo e reflete fechadas lentas das pálpebras ao invés

de piscadas.

Esta medida foi publicada pela primeira vez em um estudo de simulação de direção de 1994 como a proporção de tempo em um minuto em que os olhos permaneceram pelo menos 80% fechados. De acordo com uma pesquisa de Wierwille [5], FWHA e NHTSA consideram o PERCLOS a medida conhecida mais promissora para uso em tempo real por sistemas de deteção de sonolência no trânsito [20].

3.2.2. Inferência sobre a sonolência

A inferência sobre a sonolência do motorista se dá após a detecção do olho no streaming da camera, em três etapas: avaliação do estado do olho, contabilização e decisão.

A primeira etapa verifica a imagem do olho a fim de determinar se suas pálpebras se encontram abertas ou fechadas. É importante ressaltar que devido ao uso de PERCLOS, as pálpebras são dadas como fechadas caso elas estejam pelo menos 80% fechadas. Para tanto, fizemos uso da mesma implementação do AdaBoost, mas com uma segunda base de dados de olhos abertos e fechados para treinamento.

Para o cálculo do PERCLOS, foram definidas três variáveis que tem seus valores atualizados ao longo do tempo de execução do aplicativo: tempo de observação, tempo em que os olhos permaneceram fechados e tempo a excluir. O tempo de duração da observação é configurada previamente ao uso do aplicativo. Toda vez que o olho for detectado como fechado (80% a 100% fechados), o tempo em que os olhos permaneceram fechados tem sua contagem iniciada. Esta contagem se encerra assim que o olho é dado como aberto. Toda vez que o aplicativo não é

capaz de determinar o estado do olho, o tempo a excluir tem sua contagem iniciada até a próxima detecção bem sucedida ou término do tempo de observação. Uma vez encerrado o tempo de observação, o PERCLOS é calculado pela divisão do tempo contabilizado com o olho fechado divido pelo tempo total decorrido, sendo este ultimo o tempo de observação deduzido o tempo a excluir.

Trabalhos correlatos [27, 28, 29, 30] apresentam variadas medidas para a avaliação do PERCLOS a fim de determinar o estado de sonolência do individuo em análise. A maioria dos estudos apontam como tempo de observação ideal valores na faixa de um a três minutos, observando sonolência quando o índice de sonolência se mostra superior a 25% ou 40%. Especificamente WANG, ZHOU e YING [30] utilizam-se de uma métrica diferente em sua solução de tempo real, diminuindo o tempo de observação para dez segundos e utilizando valores iguais ou superiores à 30% para PERCLOS na determinação de sonolência.

Embora a implementação proposta neste trabalho possibilite a configuração destes parâmetros, fixaram-se os mesmos valores utilizandos por WANG, ZHOU e YING [30] pois estes demonstram maiores taxas de acerto quando utilizando-se apenas do índice PERCLOS como fator preocursor determinante de sonolência.

Uma vez atingido o valor limite do PERCLOS, a aplicação notifica o motorista com indicações visuais e sonoras, a fim de captar a atenção do mesmo.

4. Conclusões

Este trabalho apresentou uma solução de detecção e monitoração de sonolência em tempo real para dispositivos móveis, baseada no uso de características Haar-like da região ocular e do recentemente publicado algoritmo AdaBoost. Algumas

otimizações a nível de processamento e memória foram feitas no algoritmo a fim de viabilizar seu uso em plataformas de baixo poder de processamento. A abordagem de implementação utilizada nas etapas de detecção do olho do motorista e determinação de seu estado (pálpebras abertas, semi-fechadas ou fechadas), minimiza o tempo computacional enquanto alcança altas taxas de acerto.

A contabilização dos dados estatísticos referentes à variação do tempo em que as pálpebras permanecem fechadas ao longo do tempo é realizada através de equações simples, ocupando ao mínimo o tempo de CPU. Por fazer uso de uma métrica estabelecida conhecida como PERCLOS, este trabalho conta com o apoio de pesquisas correlatas envolvendo a validade do uso deste fator como um precursor confiável da sonolência.

Vale ressaltar que uma ampla pesquisa – presente no **Apêndice A** - foi realizada para averiguar o estado da arte tanto do ponto de vista fisiológico quanto tecnológico da solução proposta, levantando quais os fatores precursores da sonolência - e até mesmo da fadiga - mais frequentes na literatura, além de aspectos técnicos como tamanhos de imagem, precisões obtidas por outras soluções, desempenho em termos de tempo de execução, capacidade de processamento do hardware utilizado em experimentos, vantagens e desvantagens dos diversos algortimos que podem ser utilizados para resolver o problema motivador deste trabalho.

Concluiu-se que apesar do processamento a ser realizado para a completude da tarefa de determinação do nível de consciência do motorista em tempo real seja demasiadamente complexa, esta pode ser otimizada através de certas técnicas apresentadas neste trabalho, tornando possível detectar a presença ou ausência de sonolência com alta precisão em dispositivos móveis. E que a solução aqui proposta

tem potêncial de auxilar com a diminuição do número de acidentes no trânsito no periodo diurno, atingindo o objetivo humano do trabalho.

5. Trabalhos Futuros

Neste trabalho foi desenvolvida uma solução que embasa sua decisão sobre o nível de sonolência de um indivíduo na proporção de tempo em que as pálpebras permanecem entre 80% e 100% fechadas. Outros fatores precursores da sonolência poderiam ser utilizados em conjunto ao PERCLOS para diminuir a taxa de erro da solução, como por exemplo considerar a frequência de piscadas ou a ocorrência de bocejos. Mais do que isso o escopo da solução poderia ser aumentado, a fim de detectar fadiga e possivelmente falta de atenção no trânsito, ocasionando na necessidade de implementação de localização e tracking da cabeça, a fim de determinar a direção na qual o motorista está olhando. Dentro deste aspecto, a direção do olhar também poderia ser averiguado, tornando possível classificar pontos cegos do motorista, verificar se ele está olhando no retrovisor central ou lateral, por exemplo. Isto ainda pode ser integrado às demais funcionalidades oferecidas pela plataforma alvo, como por exemplo o acelerômetro, permitindo decidir sobre a presença do carro na estrada.

6.Referências

1. Aldrich, M.S., 1989 "Automobile Accidents in Patients with Sleep Disorders", **Association of Professional Sleep Societie**s, 12 (6), 487-494.
2. Boisvert, É., 1993 "**Acquisition et traitement de données dans une étude de la vigilance du conducteur d'automobile**", Mémoire de maîtrise, École Polytechnique de Montréal.
3. Gabrielsen, K. and Sherman, P., 1994 "Steering Wheel Data and Random Processes", **Proceedings of the 27th ISATA Conference**.
4. Seko, Y., Kataoka, S., and Senoo, T., 1986 "Analysis of Driving Behavior Under a State of Reduced Alertness", Int. J. of **Vehicle Design**, Special Issue on Vehicle Safety, 318-330.
5. Skipper, J.H. and Wierwille, W.W., 1986 "Drowsy Driver Detection Using Discriminant Analysis", **Human Factors**, 28 (5), 527-540.
6. Transport Canada, 1993 "Étude inédite sur la fatigue des conducteurs de poids lourds", Actualités R&D, **Centre de développement des transports**, 3 (3).
7. “Traffic Safety Facts 2000: A Compilation of Motor Vehicle Crash Data From the Fatality Analysis Reporting System and the General Estimates System,” **U.S. Dept. Transportation Administration**, National Highway Traffic Safety Report DOT HS 809 337, 2001.
8. **Procedures For Performing Systematic Reviews**, Barbara Kitchenham. ISSN:1353-7776.
9. VEIT, Matthias; HERRMANN, Stephan. Model-View-Controller an Object Teams: A Perfect Match os Paradigms. **Association For Computing Machinery - Acm,** New York, p. 140-149. 2003. Disponível em:

<http://doi.acm.org/10.1145/643603.643618>. em: 16 dez. 2010.

10. LEFF, Avraham; RAYFIELD, James T. Web-Application Development Using the Model/View/Controller Design Pattern. **Ieee: International Enterprise Distributed Object Computing Conference**, , p. 118-127. 2001. Disponível em: <http://doi.ieeecomputersociety.org/10.1109/EDOC.2001.950428>. Acesso em: 16 dez. 2010.
11. APPLE INC. (Org.). **The AV Foundation Framework**. Disponível em: <http://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/03_MediaCapture.html%23//apple_ref/doc/uid/TP40010188-CH5-SW2>. Acesso em: 17 jun. 2011.
12. APPLE INC. (Org.). **IOS Development With XCode**. Disponível em: <http://developer.apple.com/library/ios/#DOCUMENTATION/Xcode/Conceptual/iphone_development/000-Introduction/introduction.html%23//apple_ref/doc/uid/TP40007959-CH1-SW1>. Acesso em: 13 jun. 2011.
13. NIELSEN CORP (Org.). **Apple Leads Smartphone Race While Android Attracts Most Recent Customers**. Disponível em: <http://blog.nielsen.com/nielsenwire/online_mobile/apple-leads-smartphone-race-while-android-attracts-most-recent-customers/>. Acesso em: 10 jun. 2011.
14. APPLE INC (Org.). **IOS Programming Guide**. Disponível em: <http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/CoreApplication/CoreApplication.html>. Acesso em: 15 jun. 2011.
15. P. Cabon, R. Mollard, and A. Coblentz. Prevention of decreases of vigilance

of aircrew during long haul flights: Practical recommendations. In **Eighth International Symposium** 011 Aviation Psychology, pages 9 1 6 920, Columbus (Ohio), 1995.

16. Pierre Thiffault and Jacques Bergeron. **Monotony of road environment and driver fatigue: A simulator study. Accident Analysis and Prevention**, 35(3):381- 39I, 2003.
17. H. Kitajima, N. Numata, K. Yamamoto and Y .Goi; “Prediction of automobile driver sleepiness”, **Transactions of the Japan Society of Mechanical Engineers**, 63-613, pp.3059-3066,1997.
18. N. Numata, H. Kitajima, Y . Goi and K. Y& mamoto; “Analysis of driver’s behavior before and after crashes in simulated expressway to predict sleepiness for doze alarm”, **Transactions of Society of Automotive Engineers of Japan**, vol.29, No.2, pp.127-132, 1998.
19. K. Sugiyama, M. Mizuno, T. Nakano and S. Ya- mamoto; “Drowsiness level detection by measuring blinks utilizing image processing”, **The R&D Review of Toyota CRDL**, vo1.31, No.2, pp.51-60, 1996.
20. Federal Highway Administration. **PERCLOS: A Valid Psychophysiological Measure of Alertness As Assessed by Psychomotor Vigilance**. Disponível em: <http://www.fmcsa.dot.gov/documents/tb98-006.pdf>. Acesso em: 01 jul. 2011.
21. TURK, Matthew Alan. **Interactive-Time Vision: Face Recognition as a Visual Behavior. Disponível** em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.3926&rep=rep1&type=pdf&rct=j&q=Interactive-Time Vision: Face Recognition as a Visual Behaviour&ei=mEoRTv3KNMXw0gGHstyGAQ&usg=AFQjCNE2wOVNOtznRPC-

_JC6IZDsIhjQ8Q>. Acesso em: 27 jun. 2011.

22. FREUND, Yoav; SCHAPIRE, Robert E.. **A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting**. Disponível em: <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.8918>. Acesso em: 20 jun. 2011.

23. VIOLA, P; JONES, M. **Rapid object detection using a boosted cascade of simple features.** Disponível em: <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=990517>. Acesso em: 25 jun. 2011.

24. YOAY, F; SCHAPIRE, R. “A decision-theoretic generalization of on-line learning and an application to boosting” **Computational Learning Theory: Eurocolt**, paginas 23-37.

25. AMIT, Y; GEMAN, D; WILDER, K. **Joint induction of shape features and tree classifier**, 1997x

26. PAPAGEORGIOU; OREN; POGGIO, "A general framework for object detection", **International Conference on Computer Vision**, 1998.

27. BERGASA, L; NUEVO, J.;SOLETO, M. **Real-Time System for Monitoring Driver Vigilance.** Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1603553>. Acesso em: 20 ago. 2011.

28. FRIEDRICHS, F; YANG, B. **Camera-based Drowsiness Reference for Driver State Classification under Real Driving Conditions**. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5548039>. Acesso em 10 ago. 2011.

29. PAPADELIS, C;CHEN, Z; KOURTIDOU, C; **Monitoring sleepiness with on-board electrophysiological recordings for preventing sleep-deprived traffic accidents.** Disponível em: <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VNP-4P899C7-1-1&_cdi=6184&_user=687353&_pii=S1388245707002945&_origin=search&_zone=rslt_list_item&_coverDate=09%2F30%2F2007&_sk=998819990&wchp=dGLzVlz-zSkWW&md5=9e3cf31aa01c96cf6c840953c078f990&ie=/sdarticle.pdf>. Acesso em: 10 set. 2011.

30. WANG, H; ZHOU, L; YING, Y. **A Novel Approach for Real Time Eye State Detection in Fatigue Awareness System.** Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5513139> Acesso em: 16 set. 2011.

APÊNDICES

Apêndice A – Relatório Técnico: Detecção de Sonolência via Processamento de Imagens em Tempo Real

INCoD
Instituto Nacional para Convergência Digital

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

INCoD - Nº 001/2011 - P - GQS

Relatórios Técnicos do INCoD

Detecção de Sonolência via Processamento de Imagens em Tempo Real

Cezar Augustus Signori
José João Junior

Setembro – 2011

Relatórios Técnicos do INCoD

Relatórios Técnicos do INCoD

Detecção de Sonolência via Processamento de Imagens em Tempo Real

Autores:
Cezar Augustus Signori
José João Junior

Versão 1.0
Status: Final
Distribuição: Interna
SETEMBRO - 2011

INCoD – Instituto Nacional para Convergência Digital
Universidade Federal de Santa Catarina - UFSC
Campus Universitário João David Ferreira Lima - Trindade
Departamento de Informática e Estatística - Sala 320
Florianópolis-SC - CEP 88040-970

Fone / FAX: +55 48 3721-9516 R.17
www.incod.ufsc.br

ISSN 2236-5281

Detecção de Sonolência via Processamento de Imagens em Tempo Real / Cezar Augustus Signori, José João Junior – Florianópolis: INCoD, 2011.

1. Detecção Sonolência - 2. Revisão Sistemática - 3. Processamento de Imagens - 4. Tempo Real

Resumo

Este trabalho tem o intuito de analisar o estado atual das pesquisas relacionadas com a detecção de sonolência baseado em processamento de imagens em tempo real. Para alcançar este objetivo são realizadas duas revisões sistemáticas em bibliotecas digitais na área de tecnologia. A primeira visa identificar quais os precursores fisiológicos da sonolência mais comumente utilizados, bem como qual a validade e grau de confiança destes fatores. A segunda revisão sistemática já toma um papel mais computacional, visando levantar quais algoritmos e técnicas têm sido utilizadas em conjunto com estes fatores. Após as revisões sistemáticas são apresentados resumos da analise de cada artigo classificado como relevante nas etapas de inclusão e exclusão. Neste trabalho ainda é apresentado um capitulo de conclusões elaboradas a partir das revisões sistemáticas e dos resumos. Com estes trabalhos apresentados, este relatório técnico serve como uma visão do estado da arte em pesquisas na área de detecção de sonolência através do uso de processamento de imagens.

Sumário

Lista de Figuras

1. Inferências sobre Indicadores de Sonolência

1.1. Revisão Sistemática

Este trabalho tem o intuito de analisar o estado atual das pesquisas relacionadas à determinação do estado de sonolência em seres humanos, através da monitoração do ritmo de piscadas e do tempo que os olhos do indivíduo permanecem fechados. Neste contexto, procura-se verificar a confiabilidade destas atividades oculares enquanto indicadores de sonolência.

1.1.1. Fontes de dados e estratégias de busca

Esta revisão utiliza como fonte de dados o IEEEXplore, ACM Digital Library, Compendex EI, ISI (Institute for Scientific Information) Web of Science, ScienceDirect e WILEY Interscience Database.
Listam-se abaixo todas as fontes de dados e suas respectivas estratégias de busca.

IEEEXplore
(("Abstract":"Eye") AND (("Abstract":"Sleepiness") OR ("Abstract":"Drowziness") OR ("Abstract":"Awakening Level") OR ("Abstract":"Dullness") OR ("Abstract":"Doze state") OR ("Abstract":"Consciousness level") OR ("Abstract":"Blinking") OR ("Abstract":"Blink") OR ("Abstract":"Pupillary response") OR ("Abstract":"vigilance state")))

ACM Digital Library
(Abstract:eye) and (Abstract:sleepiness or Abstract:drowsiness or Abstract:awakening or Abstract:dullness or Abstract:doze or Abstract:consciousness or Abstract:pupillary or Abstract:vigilance or Abstract:alertness) and (FtFlag:yes)

ISI (Institute for Scientific Information) Web of Science
Abstract=(eye) AND (Abstract=(sleepiness) OR Abstract=(drowsiness) OR Abstract=(awakening) OR Abstract=(dullness) OR Abstract=(doze) OR Abstract=(consciousness) OR Abstract=(pupillary) OR Abstract=(vigilance) OR Abstract=(alertness))

ScienceDirect
“eye” AND (“sleepiness” OR “drowsiness” OR “awakening” OR “dullness” OR “doze” OR “consciousness” OR “pupillary” OR “vigilance” OR “alertness”)

1.1.2. Critérios de Inclusão e Exclusão

Inclui-se todo conteúdo publicado em inglês abordando indicadores de sonolência em seres humanos, com foco biológico e/ou computacional, que estão disponíveis na internet via bibliotecas digitais e bancos de dados, publicados entre Janeiro de 1990 e Junho de 2011. Os artigos estão limitados a peer reviewed work, incluindo apenas artigos publicados em revistas ou anais de congressos.

Exclui-se desta revisão todo e qualquer projeto ou exercício realizado por estudantes (comumente referenciados por "simulação"), bem como pesquisas no âmbito computacional que não forneçam embasamento biológico ou estatístico a respeito do uso do ritmo de piscadas como indicador de sonolência.

1.1.3. Identificação e Seleção de Artigos

A busca inicial retornou 174 artigos para o IEEE. Num primeiro momento, títulos e resumos foram rapidamente revisados levando em consideração os critérios de inclusão. Trabalhos irrelevantes ou duplicados foram removidos, reduzindo a lista para 22 artigos. O mesmo processo foi aplicado para os artigos retornados pela ACM, reduzindo a lista de 29 à 1 artigo. No ISI, foram retornados 8 artigos, mas apenas 4 se encaixaram nos critérios de inclusão, enquanto a busca realizada na Science Direct foram retornados 12 artigos e após a aplicação dos mesmos critérios foram reduzidos à 3 artigos. No total, 31 artigos foram selecionados e resumidos.
Abaixo é fornecida a listagem de artigos inclusos e exclusos de cada fonte de dados utilizada nesta revisão.

1.1.3.1. Artigos Incluídos

1.1.3.1.1. IEEE

1.Software for an expert system for human fatigue analysis
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=95192

2.On an Image Processing of Eye Blinking to Monitor Awakening Levels of Human Beings
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=652663

3.Monitoring Head/Eye Motion for Driver Alertness with One Camera
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=902999

4.Detection of Consciousness Degradation and Concentration of a Driver for Friendly Information Service
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=961722

5.Determining Driver Visual Attention With One Camera
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1260587

6. Affordable Visual Driver Monitoring System for Fatigue and Monotony
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1401415

7. Development of Non-contact Real-time Blink Detection System for Doze Alarm
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1491689

8. Hypovigilence Analysis: Open or Closed Eye or Mouth? Blinking or Yawning
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1577268

9. Real-Time System for Monitoring Driver Vigilance
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1603553

10. Efficient Measurement of Eye Blinking under Various Illumination Conditions for Drowsiness Detection Systems
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1698913

11 A Pupil Diameter Measurement System for Accident Prevention
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4274098

12. Measurement of Driver's Consciousness by Image Processing
-A Method for Presuming Driver's Drowsiness by Eye-Blinks coping
with Individual Differences -
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4274320

13. A Drowsiness and Point of Attention Monitoring System for Driver Vigilance
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04357702

14. A study of classification of the level of sleepiness for the drowsy driving prevention
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4421521

15.Using the Active Appearance Model to Detect Driver Fatigue
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4544792

16.Estimation of Sleepiness using Frequency Components of Pupillary Response
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4696948

17.Estimation of Drowsiness Level Based on Eyelid Closure and Heart Rate Variability
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5334766

18.Drowsiness detection based on visual signs: blinking analysis based on high frame rate video
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5488257

19.Camera-based Drowsiness Reference for Driver State Classification under Real Driving Conditions
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5548039

20.Estimation of Drivers' Drowsiness Level using a Neural Network Based 'Error Correcting Output Coding' Method
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5624964

21.Drowsy Driver Detection System Using Eye Blink Patterns
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5648121

22.Eye detection and recognition in the fatigue warning system
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5693673

1.1.3.1.2. ACM

1. Driver Drowsiness Warning System Using Visual Information for Both Diurnal and Nocturnal Illumination Conditions
Link: http://delivery.acm.org/10.1145/1930000/1928491/p3-flores.pdf?ip=150.162.246.33&CFID=31652632&CFTOKEN=90441450&__acm__=1309388743_9fb27f3a7b3feb298ac884b62b212420

1.1.3.1.3. ISI

1. Experimental evaluation of eye-blink parameters as a drowsiness measure
Link: http://www.springerlink.com/content/e6x53ulv1yrm9w3t/fulltext.pdf

2.The spontaneous eye-blink as sleepiness indicator in patients with obstructive sleep apnoea syndrome-a pilot study
Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6W6N-4FDNBX4-1-5&_cdi=6603&_user=687353&_pii=S1389945704002230&_origin=&_coverDate=03%2F31%2F2005&_sk=999939997&view=c&wchp=dGLbVzW-zSkWB&md5=04a960a2029cdd4148dae6e6faccf869&ie=/sdarticle.pdf

3.Predicting drowsiness accidents from personal attributes, eye blinks and ongoing driving behaviour
Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V9F-3XH3HFM-D-1&_cdi=5897&_user=687353&_pii=S0191886999000896&_origin=&_coverDate=01%2F01%2F2000&_sk=999719998&view=c&wchp=dGLzVlz-zSkWW&md5=59b37ea4119f75ddc3bb02b81f6e9d0b&ie=/sdarticle.pdf

4.Diurnal variation of spontaneous eye blink rate in the elderly and its relationships with sleepiness and arousal
Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6T0G-4WS2J1D-1-7&_cdi=4862&_user=687353&_pii=S0304394009008568&_origin=&_coverDate=09%2F29%2F2009&_sk=995369998&view=c&wchp=dGLzVzz-zSkWB&md5=4beb7408c4f67da983362409856a374c&ie=/sdarticle.pdf

1.1.3.1.4. Science Direct

1. Monitoring sleepiness with on-board electrophysiological recordings for preventing sleep-deprived traffic accidents
Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VNP-4P899C7-1-1&_cdi=6184&_user=687353&_pii=S1388245707002945&_origin=search&_zone=rslt_list_item&_coverDate=09%2F30%2F2007&_sk=998819990&wchp=dGLzVlz-zSkWW&md5=9e3cf31aa01c96cf6c840953c078f990&ie=/sdarticle.pdf

2.Increased spontaneous eye blink rate following prolonged wakefulness
Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6T0P-4M6SBM8-2-1&_cdi=4868&_user=687353&_pii=S0031938406004185&_origin=search&_zone=rslt_list_item&_cover

Date=01%2F30%2F2007&_sk=999099998&wchp=dGLzVlz-zSkWA&md5=a441d72f311834bc25802462129f8267&ie=/sdarticle.pdf

3.Analysis of drivers' behavior before and after crashes in simulated expressway driving to predict sleepiness levels for doze alarm activation
Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V3Y-3V57C42-9-P&_cdi=5743&_user=687353&_pii=S0389430498000113&_origin=search&_zone=rslt_list_item&_coverDate=07%2F01%2F1998&_sk=999809996&wchp=dGLzVlb-zSkWW&md5=9072492c59f24eac8a9180af91a87916&ie=/sdarticle.pdf

1.2. Tabela Comparativa

IDs	Condições De Iluminação	Técnica	Fatores Considerados	Medida	Precisão	Experimentos
1	Diurna e Noturna	Processamento de Sinais e Imagens	PERCLOS	5 frames seguidos ou .25 segundos	98.9%(estado do olho)	Não
2	Diurna e Noturna	Processamento de Sinais	EEG, EOG, EMG, EGG	Histórico / Comparativo	Estudo	Sim
3	Diurna e Noturna	Drogas (Dopamina)	EEG, EOG, EMG, EGG, ELS	Histórico / Comparativo	Estudo	Sim
4	Diurno	Processamento de Sinais e Imagens	EEG, EOG, ECG, Respiração, Velocidade	Mental Work Strain	Estudo	Sim, Simulado
5	Diurno	Processamento de Sinais	PERCLOS e frequência de piscadas	Questionário	Estudo	Sim
6	Diurno	Subjetivo	Frequência de Piscadas, Tempo de Reabertura das Pálpebras	Questionário	Estudo	Sim
7	Diurna	Subjetivo	Personalidade e Ritmo de Piscadas	Questionário	Estudo	Sim, Simulado
8	Diurna	Drogas e Análise de Dados	EOG, EGG	Questionário	Estudo	Sim
9	Diurna	Processamento de Imagens	Ritmo de piscadas, cardiaco e movimento da cabeça	Sistema Especialista	Não fornecido	Não
10	Diurna	Processamento de Imagens	Ritmo de Piscadas	Comparação com EGG	Estudo	Não
11	Diurna	Processam	Ritmo de	Olhos	Não	Não

		ento de Imagens	piscadas, boca e posição da cabeça, PERCLOS	fechados por 40/60 frames indicam sonolência		
12	Diurna	Processamento de Imagens	PERCLOS	Comparativo	95%	Sim
13	Diurna	Processamento de Imagens	Ritmo de piscadas e oclusão da boca	Olhos fechados por 40/60 frames indicam sonolência	Não	Não
14	Diurna	Processamento de Imagens	Movimentação da Cabeça e Ritmo de Piscadas	Comparativo com EGG, EOG, EKG e GSR	Não	Sim
15	Diurna	Processamento de Imagens	Ritmo de Piscadas	Comparação com Eletrocefalograma	Estudo	Não
16	Diurna	Processamento de Imagens	Ritmo de Piscadas e Sonolência	Comparação	Não	Não
17	Diurna	Processamento de Imagens	PERCLOS e ritmo de piscadas	Experimentação	84% e 93%	Sim
18	Diurna e Noturna	Processamento de Imagens	PERCLOS	PERCLOS > 40%	99%	Não
19	Diurna	Processamento de Imagens e Sinais	Diametro da Pupila, Ritmo de piscadas e direção do olhar	Diametro <= 2mm	Estudo	Sim
20	Diurna	Processamento de Imagens e Ondas	Ritmo de Piscadas	Enquadramento de Escalas	96%	Sim
21	Diurna	Processamento de Imagens	PERCLOS, atributos da boca e sobrancelhas	PERCLOS > 40%	Não	Sim, Simulado
22	Diurna	Processam	Ritmo de	Comparativo	Estudo	Não

		ento de Imagens	Piscadas, PERCLOS, movimentação da cabeça e expresses faciais	com Eletrocefalograma		
23	Diurna	Processamento de Imagens	PERCLOS	PERCLOS > 40%	Não	Sim
24	Diurna	Processamento de Imagens	Diametro da Pupila e Ritmo de Piscadas	VAS, SSS, ESS	71% com VAS e SSS, e 94% com ESS	Sim
25	Diurna	Processamento de Imagens e Sinais	Ritmo de Piscadas e Batimento Cardiaco	Classificação por ECOC	89%	Sim

1.3. Resumo dos Artigos

A seguir são fornecidos resumos dos artigos classificados de acordo com a aplicação dos critérios de inclusão após a execução da string de busca em cada uma das fontes de dados utilizadas nesta revisão sistemática de literatura.

[1] Driver Drowsiness Warning System Using Visual Information for Both Diurnal and Nocturnal Illumination Conditions

Link:http://delivery.acm.org/10.1145/1930000/1928491/p3-flores.pdf?ip=150.162.246.33&CFID=31652632&CFTOKEN=90441450&__acm__=1309388743_9fb27f3a7b3feb298ac884b62b212420

Ano de Publicação: 2010

Autores: Marco Javier Flores, José Maria Armingol e Arturo de la Escalera

Resumo:

Este trabalho apresenta uma solução para localizar e analisar o rosto e os olhos de motoristas para computar o nível de fadiga ou sono do mesmo durante o processo de direção do veículo. O sistema apresentado trabalha em tempo real e sob diversas condições de iluminação, tanto diurna quanto noturna. Para tanto foram desenvolvidos dois subsistemas, um para uso diurno via processamento de imagens e outro para uso noturno via processamento de sinais infra-vermelhos.
No que tange ao caso diurno, foram utilizados fatores como frequência de bocejo e piscadas, bem como movimentos da cabeça para determinar o estado de sonolência do motorista. Sendo o ritmo de piscadas e o tempo que os olhos permanecem fechados o indicador mais confiável utilizado no trabalho. O sistema utiliza este fator para alertar o motorista no caso de seus olhos terem sido identificados como fechados por cinco frames consecutivos ou durante 0.25 segundos.

Figura 5 – Viola Jones Method

[2] Monitoring sleepiness with on-board electrophysiological recordings for preventing sleep-deprived traffic accidents

Link:http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VNP-4P899C7-1-1&_cdi=6184&_user=687353&_pii=S1388245707002945&_origin=search&_zone=rslt_list_item&_coverDate=09%2F30%2F2007&_sk=998819990&wchp=dGLzVlz-zSkWW&md5=9e3cf31aa01c96cf6c840953c078f990&ie=/sdarticle.pdf

Ano de Publicação: 2007

Autores: Christos Papadelis, Zhe Chen, Chrysoula Kourtidou-Papadeli, Panagiotis D. Bamidis, Ioanna Chouvarda, Evangelos Bekiaris, Nikos Maglaveras

Resumo:

Este estudo desenvolveu e avaliou o uso estatístico de sinais neurofisiológicos na determinação do estado de sonolência de motoristas, visando a construção de um sistema automatizado para tomada de providências à respeito da falta de atenção do motorista em trânsito. Neste estudo, métodos como EEG, EOG, EMG e ECG foram utilizados em motoristas com privação de sono dirigindo veículos reais. Como resultado vários fatores de indicação de sonolência foram observados através da comparação dos índices de sua ocorrência antes e após o inicio da ocorrência de erros humanos. Dentre os quais se pode citar o ritmo de piscadas que aumentou linearmente em relação ao tempo em que os motoristas permaneceram em trânsito. Mais do que isso, não apenas a quantidade de piscadas por minuto mas também a duração das piscadas por minuto aumentaram significativamente no minuto prévio às primeiras manifestações de erro humano na direção do veículo. Os autores concluem que o ritmo e a duração das piscadas são fatores decisivos na determinação do estado de sonolência e devem ser utilizados na elaboração de um sistema de tomada de providência em tempo real à fim de reduzir o percentual de erro humano em trânsito.

Figura 6 - Um dos indivíduos em experimento vestindo uma capa EGG

[3] Increased spontaneous eye blink rate following prolonged wakefulness

Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6T0P-4M6SBM8-2-1&_cdi=4868&_user=687353&_pii=S0031938406004185&_origin=search&_zone=rslt_list_item&_coverDate=01%2F30%2F2007&_sk=999099998&wchp=dGLzVlz-zSkWA&md5=a441d72f311834bc25802462129f8267&ie=/sdarticle.pdf

Ano de Publicação: 2006

Autores: Giuseppe Barbato, Vittoria De Padova, Antonella Raffaella Paolillo, Laura Arpaia, Eleonora Russo, Gianluca Ficca

Resumo:

Este artigo analisa o papel da dopamina na privação de sono, uma técnica de manipulação da sonolência que tem sido utilizada no tratamento de indivíduos depressivos. Neste estudo são observadas mudanças no estado de sonolência através de uso de hormônios que agem sobre neurotransmissores do sistema nervoso simpático ao invés de drogas anti-depressivas. No que tange ao escopo desta revisão de literatura, como resultados obtidos têm-se que a quantidade de piscadas por unidade de tempo aumentou positivamente em relação ao tempo em que os indivíduos foram mantidos acordados pela aplicação da dopamina, até o momento que em caíram no sono. Os autores concluem que o aumento do ritmo de piscadas pode refletir o nível da ativação da dopamina que pode ser utilizada para neutralizar o impulso de sono dos objetos de estudo.

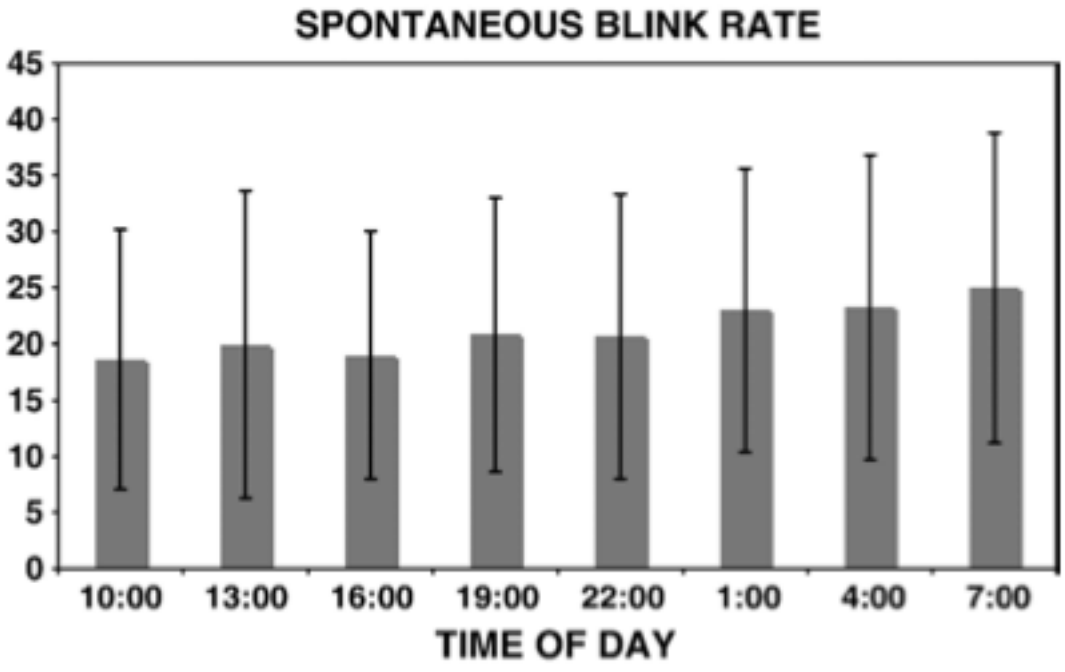

Figura 7 - Médias de piscadas durante período prolongado sem dormir

[4] Analysis of drivers' behavior before and after crashes in simulated expressway driving to predict sleepiness levels for doze alarm activation

Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V3Y-3V57C42-9-P&_cdi=5743&_user=687353&_pii=S0389430498000113&_origin=search&_zone=rslt_list_item&_coverDate=07%2F01%2F1998&_sk=999809996&wchp=dGLzVlb-zSkWW&md5=9072492c59f24eac8a9180af91a87916&ie=/sdarticle.pdf

Ano de Publicação: 1997

Autores: Nakaho Numata, Hiroki Kitajima, Yoshihiro Goi, Keiichi Yamamoto

Resumo:

Este trabalho objetiva-se a definir um método de determinação do nível de sonolência de indivíduos sem levar em consideração características pessoais, ou seja, únicas ao indivíduo em observação. O trabalho afirma ter realizado vários experimentos com pessoas em um ambiente simulado, no caso uma via expressa com características de trânsito monótono a fim de induzir a presença de sono. Constatou-se que vários fatores foram observados em comum em quase todos os participantes do experimento, entre estes fatores destacam-se o ritmo de piscadas e a diminuição da frequência de execução de algumas atividades comuns ao trânsito, como por exemplo, verificar os retrovisores e o velocímetro do veículo. O trabalho também define matematicamente os níveis de sonolência como valores entre 1 e 5. De acordo com esta definição, o nível 5 seria onde os acidentes ocorrem, sendo que na maioria dos acidentes simulados pelo experimento, os participantes tiveram seu nível de sonolência acima de 3 no último minuto prévio ao acidente, sendo portanto necessário avisar o motorista sobre seu estado de sonolência - através de um alerta sonoro - quando este atingir o nível 3.

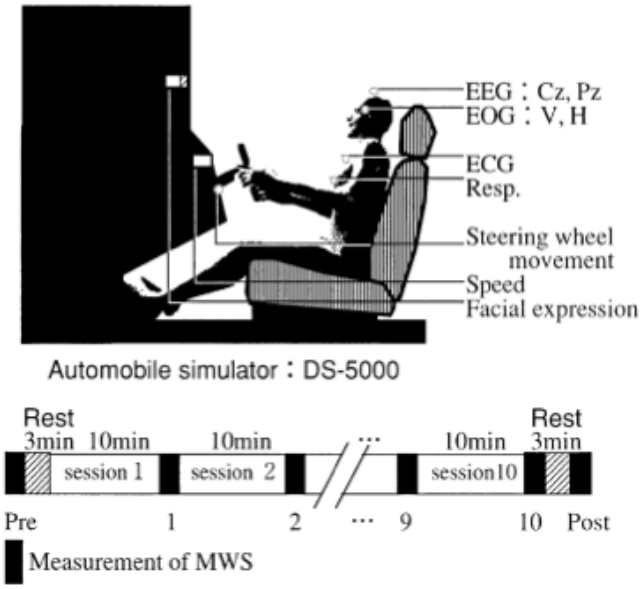

Figura 8 - Design do experimento realizado no artigo

[5] Experimental evaluation of eye-blink parameters as a drowsiness measure

Link: http://www.springerlink.com/content/e6x53ulv1yrm9w3t/fulltext.pdf

Ano de Publicação: 2003

Autores: Philipp P. Caffier, Udo Erdmann, Peter Ullsperger

Resumo:

O presente trabalho realizou um experimento com 60 adultos saudáveis a fim de avaliar diversos parâmetros relacionados ao piscar dos olhos como uma medida do nível de sonolência de um indivíduo. O experimento foi realizado com um sensor infra-vermelho anexado às lentes dos óculos dos participantes, à fim de medir continuamente a movimentação dos olhos sem a necessidade de estabelecimento de contato. Segundo os autores, como o estado de sonolência é subjetivo, foram utilizados questionários imediatamente antes da gravação e interpretação dos movimentos oculares. Os resultados demonstram que vários parâmetros relacionados ao piscar dos olhos podem ser utilizados como fatores confiáveis de sonolência, sendo a duração da piscada e o tempo de re-abertura das pálpebras particularmente definitivas. Além disto, outro parâmetro informativo é o tempo que as pálpebras permanecem fechadas durante uma piscada. O estudo finaliza concluindo que estes parâmetros podem ser usados de forma confiável para a monitoração contínua da tendência do indivíduo a cair no sono.

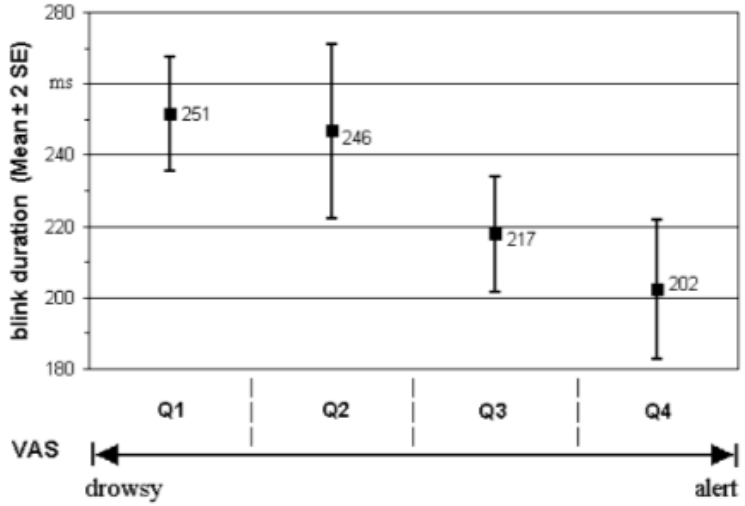

Figura 9 - Medidas e correspondentes erros padrão da duração das piscadas em dependência a subjetiva sonolência.

[6] The spontaneous eye-blink as sleepiness indicator in patients with obstructive sleep apnoea syndrome-a pilot study

Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6W6N-4FDNBX4-1-5&_cdi=6603&_user=687353&_pii=S1389945704002230&_origin=&_coverDate=03%2F31%2F2005&_sk=999939997&view=c&wchp=dGLbVzW-zSkWB&md5=04a960a2029cdd4148dae6e6faccf869&ie=/sdarticle.pdf

Ano de Publicação: 2004

Autores: Philipp P. Caffier, Udo Erdmann, Peter Ullsperger

Resumo:

Este trabalho avaliou as piscadas espontâneas de olhos como um indicador de sonolência em indivíduos com síndrome da apnéia obstrutiva de sono através um sensor livre de contato para a captura continua das piscadas de 21 pacientes. Antes do estudo todos os pacientes foram submetidos a uma noite de polissonografia. O piscar dos olhos foi estudado na manhã seguinte antes da terapia e novamente após a primeira noite de terapia com pressão positiva dos canais nasais (nCPAP). A avaliação do estado de sonolência foi feita através de questionários imediatamente antes da captura do piscar dos olhos. Estes estudos foram conduzidos em dois hospitais distintos. A redução da duração da piscada e do tempo de reabertura dos olhos assim como o aumento do ritmo de piscadas foi significante em pacientes apresentando sono diurno excessivo. O trabalho encerra concluindo que o uso do ritmo de piscadas é um fator confiável para o diagnóstico da sonolência, porém seria necessária uma pesquisa mais profunda para avaliar as vantagens do uso deste parâmetro em relação à outros comumente utilizados em estudos clínicos.

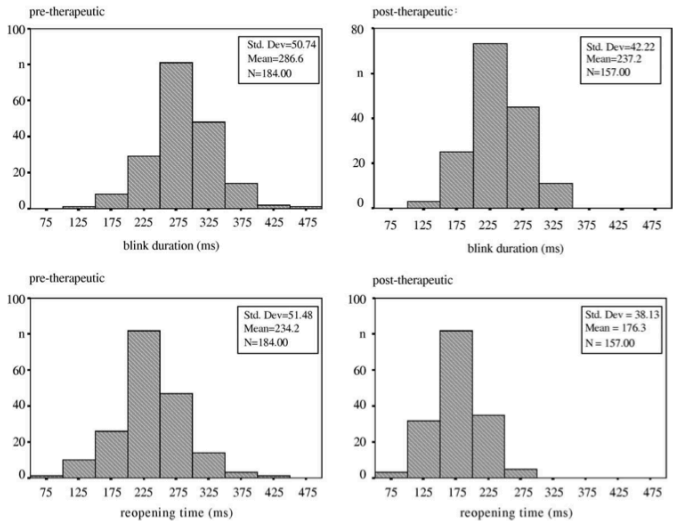

Figura 10 - Exemplo de histogramas individuais de duração de piscadas e tempo de reabertura das pálpebras de medições pre e pós-terapeuticas a respeito de um paciente com EDS

[7] Predicting drowsiness accidents from personal attributes, eye blinks and ongoing driving behavior

Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V9F-3XH3HFM-D-1&_cdi=5897&_user=687353&_pii=S0191886999000896&_origin=&_coverDate=01%2F01%2F2000&_sk=999719998&view=c&wchp=dGLzVlz-zSkWW&md5=59b37ea4119f75ddc3bb02b81f6e9d0b&ie=/sdarticle.pdf

Ano de Publicação: 1999

Autores: Willem B. Verwey, David M. Zaidel

Resumo:

Um experimento foi realizado por 135 minutos em uma pista rural simulada com 26 participantes à fim de avaliar correlações entre o estado de sonolênca e alguns atributos de personalidade, comportamento apresentado durante o experimento e o ritmo de piscada dos olhos. Dos 26 participantes, 17 afirmaram ter caído no sono durante o experimento. Destes 9 caíram no sono na segunda metade do experimento, sendo que 10 motoristas sofreram algum acidente durante a simulação. A maioria dos parâmetros foram indicados como medidas informativas, tendo significativa importância na determinação do diganóstico do estado de sonolência o ritmo de piscadas e a frequencia de ocorrência de piscadas mais longas.

[8] Diurnal variation of spontaneous eye blink rate in the elderly and its relationships with sleepiness and arousal

Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6T0G-4WS2J1D-1-7&_cdi=4862&_user=687353&_pii=S0304394009008568&_origin=&_coverDate=09%2F29%2F2009&_sk=995369998&view=c&wchp=dGLzVzz-zSkWB&md5=4beb7408c4f67da983362409856a374c&ie=/sdarticle.pdf

Ano de Publicação: 2009

Autores: Vittoria De Padova, Giuseppe Barbato, Francesca Conte, Gianluca Ficca

Resumo:

Este estudo realizou um experimento com 12 idosos voluntariados. Quatro vezes ao dia foram medidas o nível de sonolência segundo a escala Karolinska e o ritmo de piscadas também foi capturado via EOG. Como medida objetiva do nível de sonolência foi utilizada a potência alpha EEG. Como resultado obteve-se que os idosos tendem a sentir mais sono no final da tarde e que o ritmo de piscadas não sofre grandes mudanças durante o dia. O trabalho aponta que a não alteração da frequência de piscadas dos olhos atenta à modificações do nível de dopamina relativas a idade do orgamismo. Este mesmo estudo também cita que o ritmo de piscadas espontâneas é um fator confiável para a determinação do estado de sonolência em indivíduos jovens, mas que para o caso de idosos entre 64 e 79 anos, esta medida não é de grande valia.

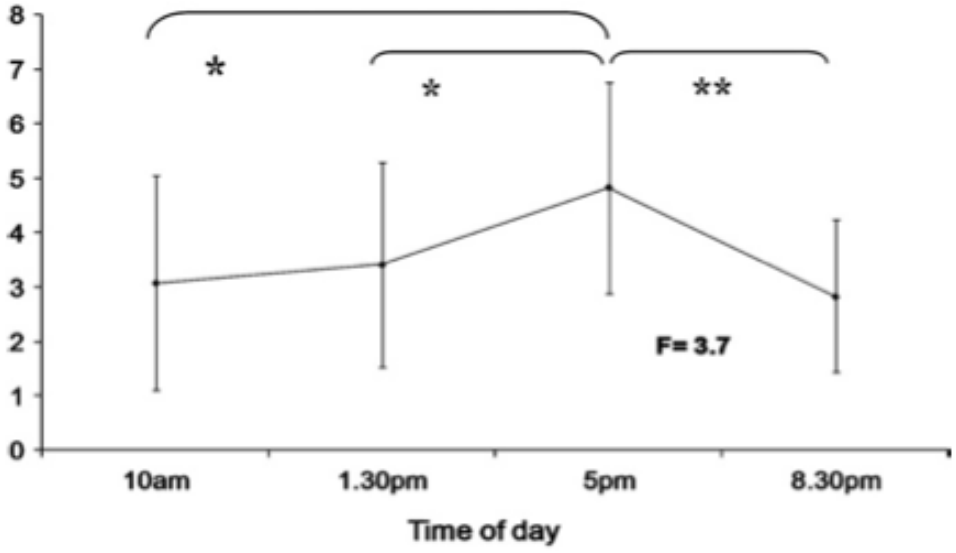

Figura 11 - Variação de sonolência Karolinska (*p<=0.05 e **p<=0.01)

[9] Software for an expert system for human fatigue analysis

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=95192

Ano de Publicação: 1988

Autores: Victor J. Vincent e Wunnava Subbarao

Resumo:

Este artigo apresenta uma proposta de software para analise de fadiga humana baseada em uma técnica não invasiva para medição de precursores da fadiga. Um sistema especialista foi desenvolvido através da aplicação de uma série de algoritmos sobre dados coletados a respeito do ritmo de piscadas, ritmo cardíaco e movimento da cabeça. O trabalho provê informações a respeito do sistema especialista, mas não comenta sobre os fundamentos que levam os autores a confiar no ritmo de piscadas como um fator precursor de sonolência, assim como não foi realizado nenhum experimento para embasar suas afirmações. Os autores, no entanto afirmam que o sistema fornece de fato bons resultados no diganóstico da sonolência tomando como base o ritmo de piscadas e as regras do sistema especialista especificado pelos autores.

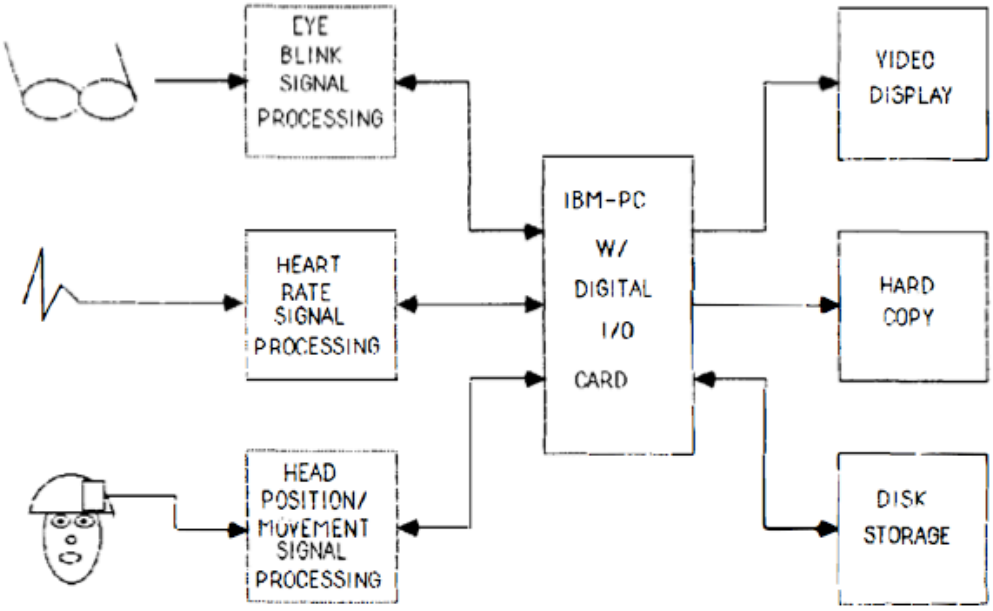

Figura 12 - Representação do Sistema

[10] On an Image Processing of Eye Blinking to Monitor Awakening Levels of Human Beings

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=652663

Ano de Publicação: 1996

Autores: Mariko Fujikake Funada, Satoki P. Ninomija, Satoshi Suzuki, Kyoko Idogawa, Yusuke Yam, Hideto Ide

Resumo:

Este trabalho desenvolveu uma solução baseada em processamento de imagens para monitorar o nível de sonolência de seres humanos através de uma técnica de reconhecimento de piscadas dos olhos. Para determinar a confiabilidade da solução baseada no ritmo de piscadas dos olhos, uma comparação entre os resultados da mesma foi realizada contra um sensor utilizando eletrocelafogramas (EEGs). Para a efetivação desta comparação os autores fizeram uso de uma unidade de tempo nomeada "tempo de atividade" que consiste no período de tempo entre um fechamento das pálpebras e o próximo. Para o mesmo grupo de indivíduos, ondas alpha indicando a atividade cerebral foi coletada por eletrocefalogramas. Como resultado da comparação, obteve-se que existe uma forte relação entre os dois resultados, concluindo assim que o uso do ritmo de piscadas é um parâmetro adequado e confiável para o diagnóstico de sonolência em um ser humano.

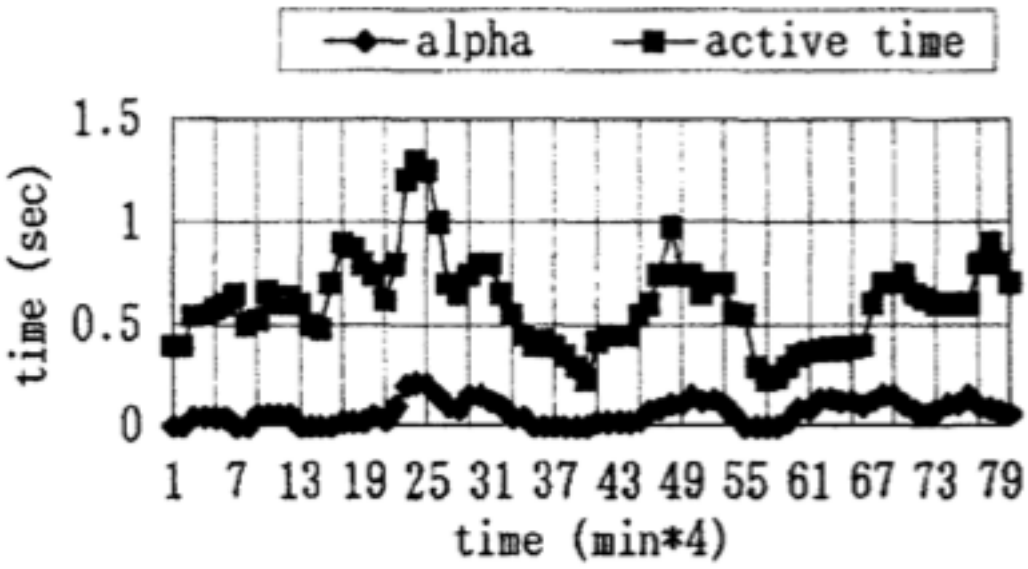

Figura 13 - Relação entre o tempo de aparição de ondas alfa agrupadas e tempo de piscadas

[11] Monitoring Head/Eye Motion for Driver Alertness with One Camera

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=902999

Ano de Publicação: 2000

Autores: Paul Smith, Mubarak Shah, e N. da Vitoria Lobo

Resumo:

Este trabalho descreve um sistema para análise do nível de atenção de um motorista, baseando-se nos movimentos da cabeça e nos seus atributos faciais. O sistema classifica a rotação da cabeça em todas as direções, detecta os olhos e a boca, atribuindo especial atenção ao ritmo de piscadas dos olhos. Os autores se comprometem à confiabilidade de seu trabalho na determinação do nível de atenção atribuído ao trânsito por parte do motorista, mas apesar de acreditarem que seu sistema também pode utilizado para determinar se o motorista está prestes a dormir, dizem sentir falta de uma pesquisa mais aprofundada a respeito do uso dos parâmetros selecionados. Os autores também indicam o uso do tempo em que os olhos permanecem fechados durante uma piscada como um fator determinante da sonolência do indivíduo quando utilizado juntamente com algum outro fator como, por exemplo, a presença de bocejos. Os autores se privam de entrar em detalhes sobre a fisiologia de um motorista em estado de alerta baixo - beirando sonolência -, mas afirmam terem obtido sucesso no uso de técnicas de processamento de imagens utilizando-se dos fatores selecionados para a determinação do nível de atenção do motorista durante o período diurno.

Figura 14 - Tracking da cabeça

[12] Detection of Consciousness Degradation and Concentration of a Driver for Friendly Information Service

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=961722

Ano de Publicação: 2000

Autores: Natsuki Kojima, Kazuhiro Kozuka, Tomoaki Nakano, Shin Yamamoto

Resumo:

Este artigo apresenta uma solução baseada em processamento de imagens para determinar o estado de degradação do nível de consciência de motoristas em trânsito. O nível de degradação de consciência é inferido a partir das mudanças na duração das piscadas dos olhos determinados pelas imagens obtidos dos seus rostos enquanto dirigem. Os autores obtiveram um índice de reconhecimento de piscadas de 95% e o nível de consciência foi estimado a partir das mudanças no tempo em que os olhos permanecem fechados no decorrer do tempo. O trabalho baseia-se na hipótese de que quanto maior a duração de uma piscada, maior o nível de sonolência do indivíduo e, portanto menor o nível de consciência do mesmo. Para validar o emprego da técnica, seus resultados foram comparados ao nível de atividade cerebral em 5 indivíduos. Embora cada indivíduo tenha apresentado diferentes características em relação à sonolência e ao ritmo de piscadas, os autores afirmam que este é bom indicador de sonolência e atribuem estas pequenas diferenças ao fato de serem parâmetros fisiológicos, deixando o tratamento das mesmas para trabalhos futuros.

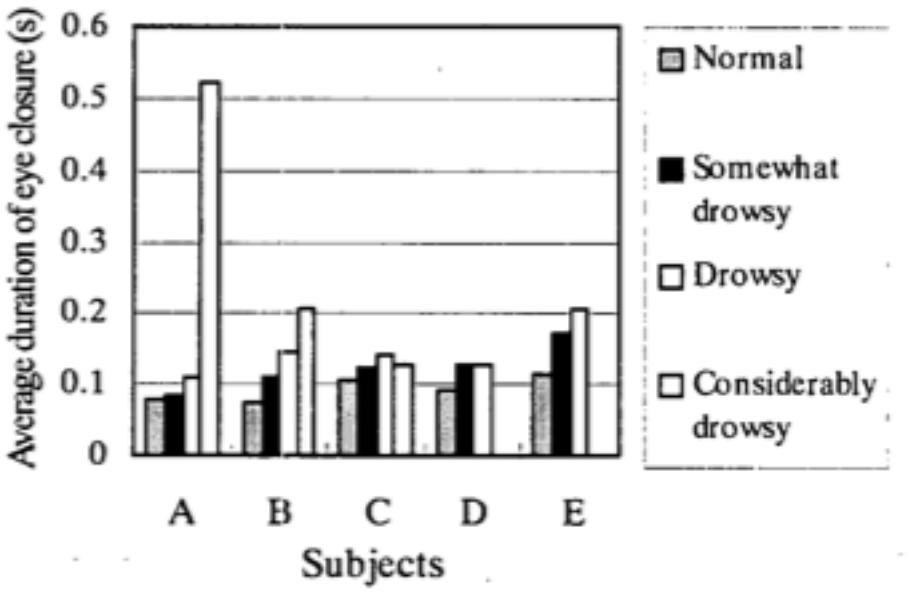

Figura 15 - Duração média de fechamento das pálpebras para cada nível de consciência dos cinco indivíduos

[13] Determining Driver Visual Attention With One Camera

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1260587

Ano de Publicação: 2003

Autores: Paul Smith, Mubarak Shah, Niels da Vitoria Lobo

Resumo:

Este artigo apresenta um sistema para a análise da atenção visual empregada por um motorista durante o trânsito. O sistema se baseia na estimativa de movimentos globais e estatísticas baseadas em cores para a realização do tracking da cabeça e das qualidades faciais apresentadas pelo motorista. Dentre as características faciais, destacam-se como fatores confiáveis o ritmo de piscadas e a oclusão da boca. No que diz respeito a estes parâmetros, são avaliados a frequência de piscadas, o tempo em que os olhos permanecem fechados e a existência de grandes aberturas bocais indicando a presença de bocejos.
Os autores concluem que os parâmetros utilizados acarretam numa boa taxa de acerto no que diz respeito ao diagnóstico de falta de atenção visual por parte do motorista, mas também acarretam em vários falsos positivos devido a detalhes de implementação. Os autores ainda apontam que seria interessante avaliar aspectos comportamentais do motorista como, por exemplo, a taxa de verificação dos retrovisores, a direção do olhar - se o motorista está a olhar para frente - e a taxa de verificação do velocímetro em função do tempo.

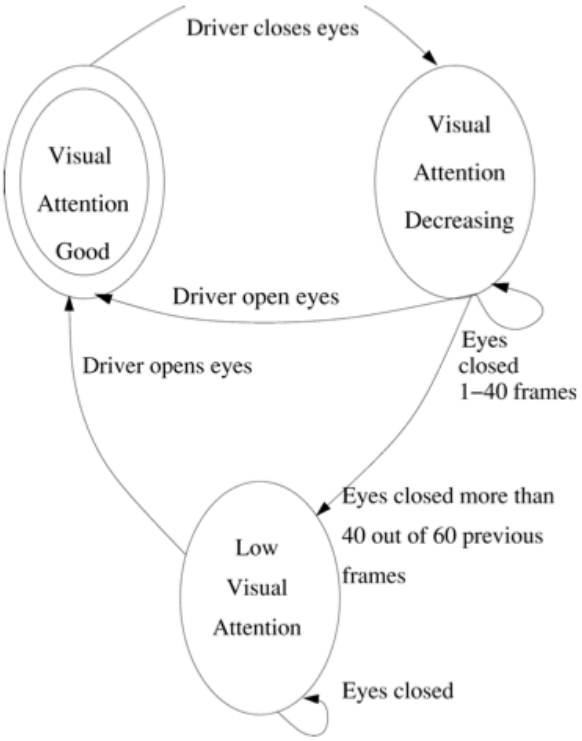

Figura 16 - Autômato finito para motorista abrindo e fechando os olhos

[14] Affordable Visual Driver Monitoring System for Fatigue and Monotony

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1401415

Ano de Publicação: 2004

Autores: Thomas Brandt, Ralf Stemmer, Andry Rakotonirainy

Resumo:

Este trabalho apresenta um sistema de vigilância para monitorar os movimentos da cabeça de um motorista bem como seus padrões de piscadas dos olhos. Baseando-se nestes parâmetros, o sistema é capaz de detectar sintomas de fadiga e monotonia. O objetivo do sistema proposta é diagnosticar o nível de atenção do motorista utilizando recursos de processamento de imagens em um equipamento de baixo custo. Para a determinação da confiabilidade dos parâmetros fisiológicos da solução foram utilizados resultados de diferentes sensores e dispositivos tais quais EEG, EOG, EKG e GSR. Os autores afirmam que já nos primeiros momentos de testes a melhor combinação de sensores para a detecção de monotonia é composta pelo GSR e o EOG. Sendo este último responsável pela captura do ritmo de piscadas, concluindo-se, portanto que este é um dos fatores mais confiáveis na determinação de monotonia e fadiga via processamento de imagens.

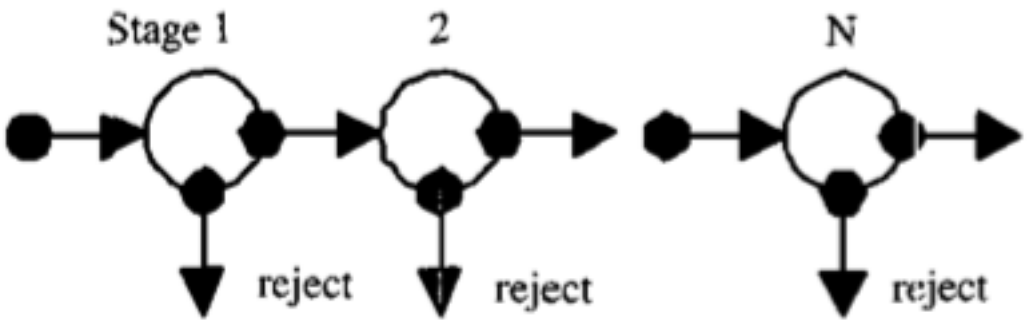

Figura 17 - Boosted cascade com N estágios

[15] Development of Non-contact Real-time Blink Detection System for Doze Alarm

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1491689

Ano de Publicação: 2004

Autores: Tomofurni MIYAKAWA, Hironobu TAKANO e Kiyomi NAKAMURA

Resumo:

Neste trabalho os autores definem um estado chamado "Doze" como o estado onde o nível de consciência cai devido à falta de sono, fadiga ou doença. Segundo os autores, quando o nível de consciência cai, várias mudanças fisiológicas ocorrem, como por exemplo, o aumento da quantidade de piscadas das pálpebras e o tempo em que os olhos permanecem fechados aumenta. Neste estudo foi desenvolvido um sistema de detecção de sonolência livre de contato baseado nestes parâmetros. Este sistema foi validado através do uso de um eletrocefalograma a fim de realizar uma análise da correlação entre o nível de consciência e as piscadas. Os autores apresentam provas concretas baseadas em comparações com eletrocefalogramas de que a solução implementada via processamento de imagens em tempo real de detecção e análise de piscadas apresenta resultados satisfatórios no diagnóstico de sonolência.

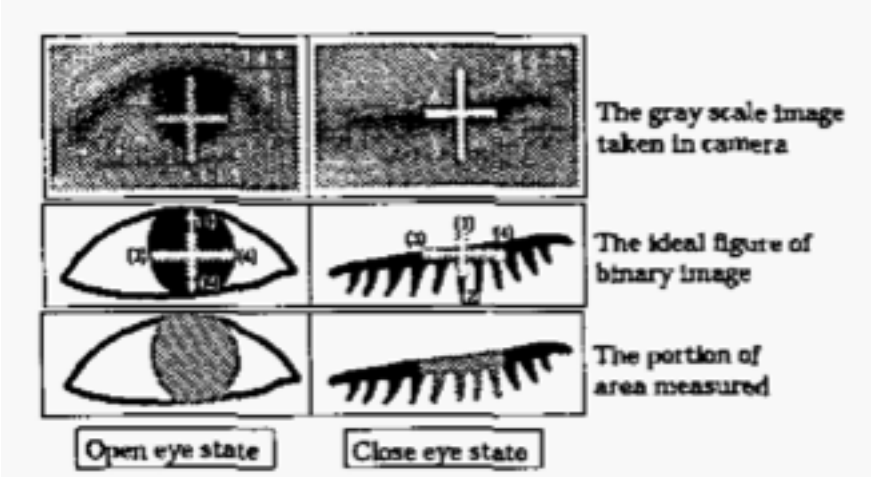

Figura 18 - Área da Iris que é medida tanto com olhos fechados quanto abertos

[16] Hypovigilence Analysis: Open or Closed Eye or Mouth? Blinking or Yawning Frequency?

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1577268

Ano de Publicação: 2005

Autores: A. Benoit, A. Caplier

Resumo:

Este trabalho objetiva a proposição de um método para estimar os estados aberto e fechado dos olhos e da boca e detectar movimentos associados aos mesmos tais quais o piscar dos olhos e o bocejo. O contexto de aplicação deste método diz respeito à detecção de hipervigilância, ou seja, estado de fadiga, sonolência ou falta de atenção por parte do usuário, no caso um motorista ou piloto. Os autores afirmam que o estado de sonolência do indivíduo em análise pode sim ser estimado via observação da frequência ou ritmo de piscadas. O trabalho foca-se no método proposto para detecção dos parâmetros acima citados, deixando de lado os aspectos fisiológicos que compram sua utilidade em relação à aplicação escolhida como estudo de caso pelo artigo. Isto ocasiona no foco em testes de validação e homologação dos algoritmos de detecção do ritmo de piscadas e bocejos, e na consequente falta de testes de validação comprovando a eficácia da solução no que diz respeito ao diagnóstico de hipervigilância.

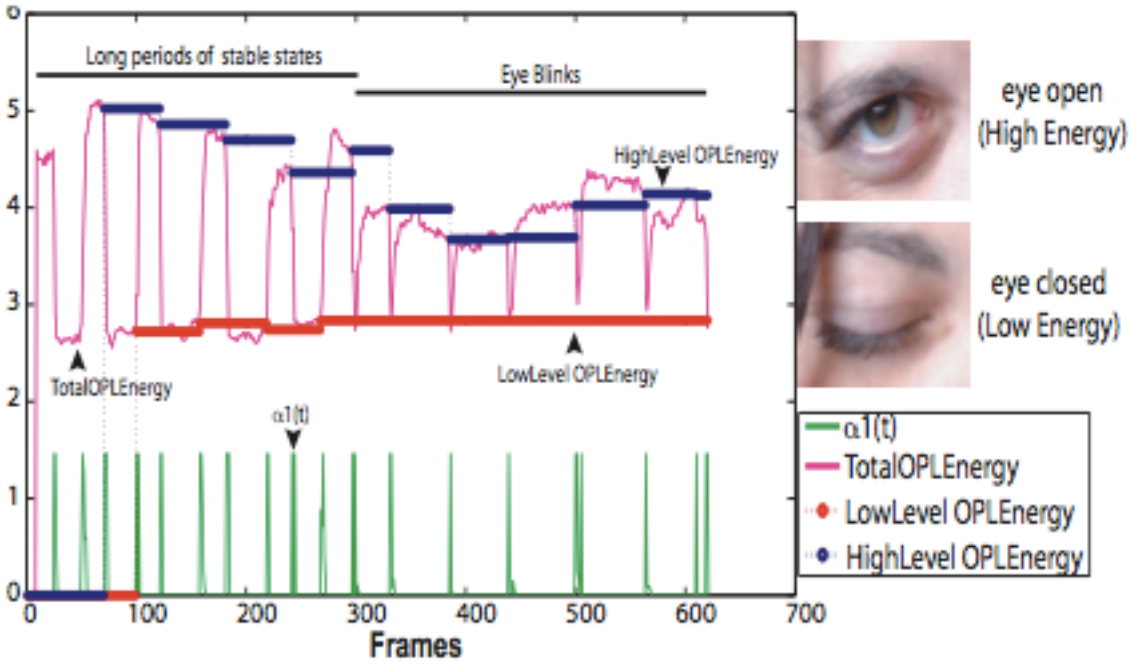

Figura 19 - Evolução temporal da energia OPL em uma sequência de piscares de um olho

[17] Real-Time System for Monitoring Driver Vigilance

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1603553

Ano de Publicação: 2006

Autores: Luis M. Bergasa, Jesús Nuevo, Miguel A. Sotelo, Rafael Barea, e María Elena Lopez

Resumo:

Este trabalho apresenta um protótipo livre de contato baseado em visão computacional para o monitoramento de estado de vigilância de um motorista em tempo real. O protótipo consiste em um sistema de hardware para a capturação de imagens do motorista em tempo real utilizando um iluminador infra-vermelho e a implementação por software para a monitoração de alguns aspectos visuais comportamentais que caracterizam o nível de vigilância do motorista. Dentre os seis aspectos visuais escolhidos, cinco dizem respeito à região ocular sendo três destes relacionados ao ritmo de piscadas: percentual de tempo em que os olhos permanecem fechados, duração do fechamento dos olhos na ocorrência de um piscar de olhos e frequência de piscadas. Para a validação do sistema, e, portanto dos parâmetros fisiológicos escolhidos para embasá-lo, varios experimentos foram realizados em condições de trânsito diurno e noturno em uma via expressa com diferentes usuários. Os experimentos realizados indicam que os fatores relacionados à abertura e fechamento das pálpebras resultam em um percentual de acerto entre 84% e 93%, sendo o restante falsos positivos. Os autores concluem que o uso destes parâmetros é adequado e confiável, mas ainda incompleto para determinar o estado de sonolência com 100% de acerto.

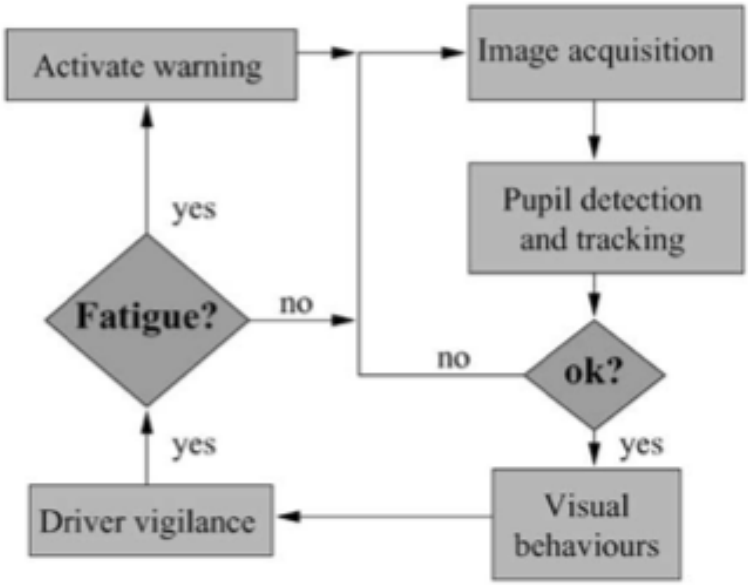

Figura 20 - Arquitetura geral

[18] Efficient Measurement of Eye Blinking under Various Illumination Conditions for Drowsiness Detection Systems

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1698913

Ano de Publicação: 2006

Autores: Ilkwon Park, Jung-Ho Ahn, e Hyeran Byun

Resumo:

Este artigo apresenta uma solução dita eficiente para a medição da frequência do piscar de olhos de um indivíduo sobre várias condições de iluminação, tais quais diurna e noturna, para uso em um sistema de detecção de sonolência. Os autores focam-se na validação dos algoritmos de detecção de piscadas sobre as variadas condições de iluminação onde obtém percentuais de acerto beirando os 100%. Para a determinação do nível de sonolência é utilizado como parâmetro o índice de PERCLOS. Na solução proposta pelo artigo, caso este índice supere os 40%, o usuário é avisado de seu nível de sonolência. Os autores não entram em detalhes sobre a confiabilidade destas medidas no que diz respeito às características fisiológicas ou implicações do aumento da frequência de piscadas ao decorrer do tempo no diagnóstico de sonolência. Afirmam, porém terem obtido bons resultados utilizando o índice PERCLOS.

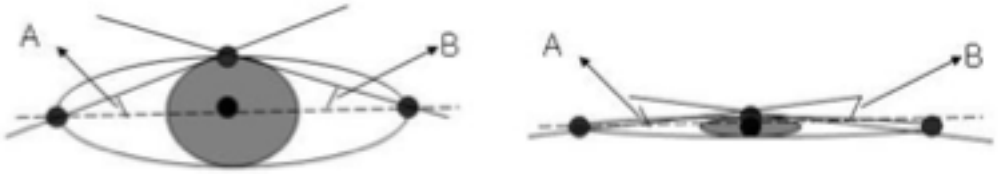

Figura 21 - Medição dos movimentos de um olho: a) um olho aberto e b) um olho fechado

[19] A Pupil Diameter Measurement System for Accident Prevention

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4274098

Ano de Publicação: 2006

Autores: Takayuki SHINODA e Masami KATO

Resumo:

Este trabalho propõe um sistema para a detecção do nível de atenção empregado pelo motorista ao trânsito a fim de prevenir acidentes. Este sistema utiliza como base para a determinação da falta de atenção o tamanho do diâmetro das pupilas em conjunto com parâmetros como a duração das piscadas e a mudança da direção do olhar ao decorrer do tempo. Os autores apresentam fortes indícios da validade do uso destes parâmetros através da exibição de vários experimentos psicológicos comprovados, porém não fornecem nenhuma informação quanto aos testes realizados para validar a solução proposta, não exibindo dados quantitativos sobre quaisquer experimentos realizados e limitando-se a explicar o que cada situação (ritmo de piscadas e diâmetro da pupila) indica sobre o estado de sonolência bem como em quais situações a solução falha.

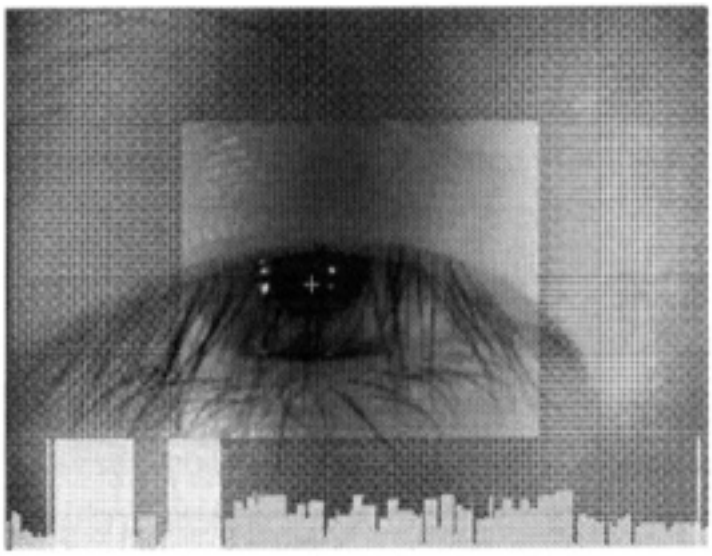

Figura 22 - Exemplo de falha logo após o piscar de um olho

[20] Measurement of Driver's Consciousness by Image Processing - A Method for Presuming Driver's Drowsiness by Eye-Blinks coping with Individual Differences

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4274320

Ano de Publicação: 2006

Autores: Mai Suzuki, Nozomi Yamamoto, Osami Yamamoto, Tomoaki Nakano, e Shin Yamamoto

Resumo:

Este artigo apresenta um sistema para detecção de sonolência em motoristas baseado em ondas geradas a partir de dados obtidos pela captura do ritmo de piscadas do motorista. Mais do que isso os autores utilizam diferenças individuais na detecção da sonolência através de parâmetros obtidos pela onde formada por dados de piscadas individuais. Os autores apresentam indícios da validade do uso de parâmetros como o aumento do tempo em que os olhos permanecem fechados durante uma piscada ou até mesmo a própria frequência do piscar de olhos como fatores confiáveis no diagnóstico de sonolência. Estes indícios são exibidos através de experimentos e possuem formulação matemática, além serem confirmados por trabalho correlatos expostos nas referências do trabalho.

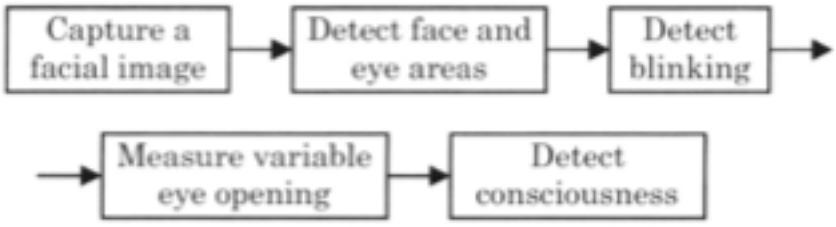

Figura 23 - Procedimento para detecção de sonolência

[21] A Drowsiness and Point of Attention Monitoring System for Driver Vigilance

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04357702

Ano de Publicação: 2007

Autores: Jorge Batista

Resumo:

Este trabalho apresenta um framework para medir o nível de vigilância ou consciência de um motorista em trânsito através da combinação de características faciais com um modelo elíptico do rosto do mesmo. O modelo facial antropométrico é utilizado para a determinação das regiões que os autores acreditam serem as mais importantes do rosto na determinação do estado de sonolência, ou seja, os olhos, a boca e as sobrancelhas. No que diz respeito aos olhos, os parâmetros observados são o ritmo de piscadas e o tempo em que os olhos permanecem fechados. Para a validação dos parâmetros são realizados experimentos com imagens reais em circunstâncias simuladas aleatórias. Os autores apontam o PERCLOS como principal parâmetro para o diagnóstico de sonolência, sendo este o percentual de fechamento das pálpebras sobre a pupila ao longo do tempo, refletindo num fechamento lento das pálpebras ao invés de um piscar de olhos.

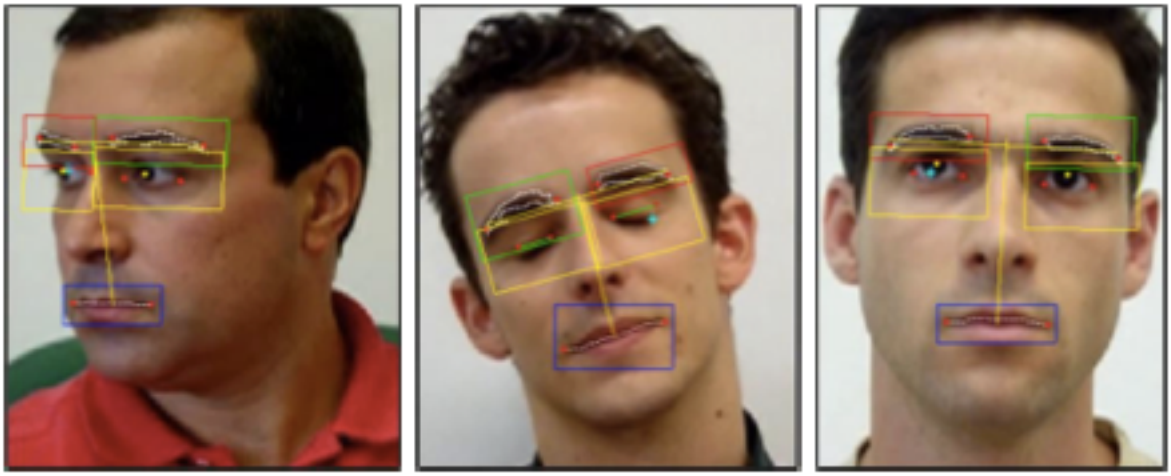

Figura 24 - Features da face localizados em diferentes rostos

[22] A study of classification of the level of sleepiness for the drowsy driving prevention

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4421521

Ano de Publicação: 2007

Autores: Byung-Chan Chang, Jung-Eun Lim, Hae-Jin Kim e Bo-Hyeok Seo

Resumo:

Este trabalho consiste num estudo sobre a classificação do nível de sonolência de um motorista em trânsito. Embora os autores apontem fatores fisiológicos que podem ser obtidos via processamento de imagens através da observação do motorista, como por exemplo, o ritmo de piscada das pálpebras, o tempo que os olhos permanecem fechados, a movimentação da cabeça, a expressão facial e a direção do olhar como fatores confiáveis no diagnóstico de presença da sonolência, os mesmos inferem a impossibilidade do uso destes parâmetros quando o indivíduo em observação está usando óculos. Através da análise de trabalhos correlatos, os autores apontam como um melhor indicador o uso de eletrocefalogramas em conjunto com índices de variação do batimento cardíaco. Estes indicadores necessitam, porém de contato com o motorista, característica não presente nas soluções baseadas na analise de fatores fisiológicos.

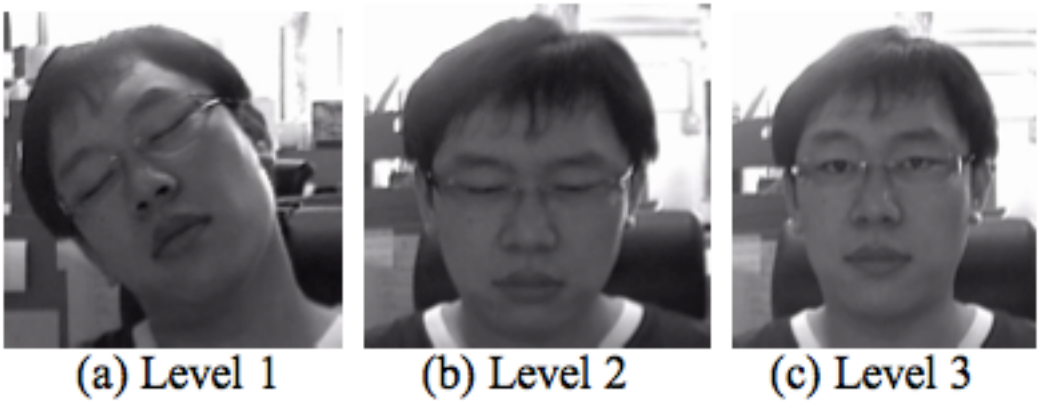

Figura 25 - Categorias de sonolência por imagens de câmera

[23] Using the Active Appearance Model to Detect Driver Fatigue

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4544792

Ano de Publicação: 2007

Autores: Isaam Saeed, Alyoxia Wang, Rajinda Senaratne e Saman Halgamuge

Resumo:

Este trabalho apresenta uma solução para a detecção de fadiga de motoristas através da análise de frames de vídeo. O sistema faz uso do Modelo de Aparência Ativa como forma de captura dos movimentos da face encontrados em um vídeo de entrada. Os autores apresentam uma variação deste modelo que dá foco à região ocular, de forma a determinar a sonolência através do ritmo de piscadas, considerado pelos autores um fator confiável para este diagnóstico. Embora os autores forneçam um fundamentado embasamento teórico ao fazer uso de parâmetros como o PERCLOS, a quantidade de indivíduos utilizados nos experimentos de validação (2) é muita baixa para se obter conclusões definitivas a respeito da solução implementada. Os autores concluem o artigo demonstrando seu contento em relação aos resultados obtidos, mas admitindo a ausência de uma determinação eficaz dos limites entre os estados em que o motorista se apresenta consciente até o momento que se torna sonolento e finalmente chega ao estado onde acidentes de trânsito ocorrem.

Figura 26 - 42 pontos escolhidos para representar a região ocular de um rosto

[24] Estimation of Sleepiness using Frequency Components of Pupillary Response

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4696948

Ano de Publicação: 2008

Autores: Minoru Nakayama, Keiko Yamamoto e Fumio Kobayashi

Resumo:

Este trabalho realiza um estudo sobre a estimativa quantitativa da presença de sono em seres humanos através do uso de componentes de frequência das respostas das pupilas durante a execução dos testes. Embora os autores focalizem seus esforços em relacionar e medir o significado de fatores como o aumento ou diminuição do diâmetro das pupilas como indicadores de sonolência, estes também ressaltam a existência uma relação entre as mudanças no tamanho do diâmetro das pupilas e o ritmo de piscadas das pálpebras, sendo esta última medida utilizada como fator adicional na determinação do diagnóstico de sonolência. O trabalho é concluído pelos autores na apresentação dos resultados obtidos através do experimento com cerca 35 indivíduos, onde foi constatada uma taxa de acerto de 71% dos diagnósticos de sonolência utilizando pontuações SSS (Stanford Sleepiness Score) e VAS (Visual Analogue Scale), chegando aos 94% de acerto quando fazendo uso de ESS (Epworth Sleepiness Score).

[25] Estimation of Drowsiness Level Based on Eyelid Closure and Heart Rate Variability

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5334766

Ano de Publicação: 2009

Autores: Ayumi Tsuchida, Md. Shoaib Bhuiyan e Koji Oguri

Resumo:

Este artigo apresenta um método para estimar o nível de sonolência de um motorista, baseado na observação da variabilidade do batimento cardíaco e do fechamento das pálpebras. Experimentos foram conduzidos em laboratório utilizando um simulador proprietário de direção para induzir a situação de sono no trânsito. Os autores fazem uso do ritmo de piscadas para a classificação do nível de sonolência. A classificação exposta pelos autores define 4 classes, sendo que os primeiros sinais de sonolência começam a aparecer no nível 3 com o surgimento frequente de piscadas lentas, evoluindo o quarto nível com o surgimento de bocejos, atingindo o último nível com o fechamento das pálpebras e o movimento da cabeça recostando-se para trás ou para frente. A aplicação do método proposto nos experimentos realizados em laboratório demonstrou um percentual de acerto de aproximadamente 89%, um resultado satisfatório segundo os autores.

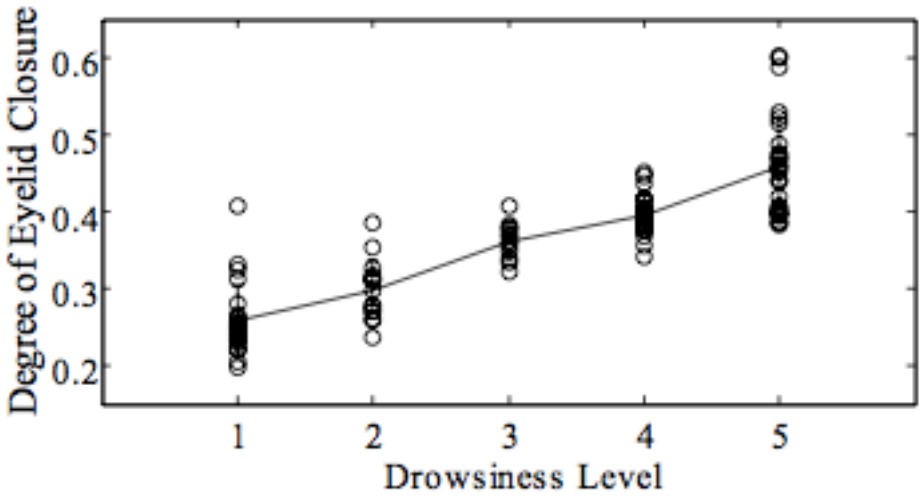

Figura 27 - Um exemplo de relação entre o nível de sonolência e o ângulo do fechamento das pálpebras

2. Inferências sobre detecção de olhos em tempo real

2.1. Revisão Sistemática

Este trabalho tem o intuito de analisar o estado atual das pesquisas relacionadas com a detecção de olhos humanos utilizando tecnicas de processamento de imagem, através de uma revisão sistemática. Neste contexto, será possível analisar a viabilidade de deteção dos olhos sobre algumas variações de iluminação, movimentos de cabeça e em tempo real.

2.1.1. Critério de inclusão e exclusão

Nós examinamos todos os artigos publicados em Inglês na área de processamento de imagens para rastreamento de olhos humanos que estão disponíveis em meio digital, publicados entre Janeiro de 1990 e maio de 2011. Nós limitamos nossa revisão a peer reviewed work, incluindo apenas artigos publicados em revistas ou anais de congressos.

Nós incluímos todo tipo de paper sobre algoritmos de rastreamentos de olhos usando técnicas de processamento de imagens. Por outro lado excluímos:

1. Todo tipo de algoritmo que não tenha como entrada uma imagem ou uma sequência de imagem.
2. Todas pesquisas que são voltadas localizar outras formas que não seja olhos humanos.
3. Todos estudos que não apresentam métricas de desempenho.
4. Todas pesquisas que apresentam algoritmos voltados a síntese em FPGA.
5. Todas pesquisas que tem por objetivo a determinação de onde o olho esta focando.
6. Todos artigos que apresentam uma abordagem que necessita de imagens tiradas com cameras stereos, ou seja trabalhem com “modelos” 3D.
7. Todas as pesquisas que façam aplicação de algoritmos de localização de olhos que não tenha uma descrição do algoritmo utilizado.
8. Todo algoritmo que não apresente um desempenho de tempo real e que não tenham a possibilidade de se fazer otimizações básicas, como implementar em outra linguagem, para atingir o requisito de tempo real.

2.1.2. Fontes de dados e estratégia de busca

Esta revisão utiliza como fonte de dados o IEEEXplore, ACM Digital Library, ISI (Institute for Scientific Information) Web of Science, ScienceDirect e WILEY Interscience Database.
Listam-se abaixo todas as fontes de dados e suas respectivas estratégias de busca.

IEEE XPLORE
((((((Document Title:Eye Tracking) OR Document Title:Eye Detection) AND image processing) NOT Document Title:head) NOT Document Title:estereo) NOT gaze)
Subject: Computing & Processing (Hardware/Software)
Publication Year: 1990 - 2011

ACM Digital Library
((Title:eye and Title:detection) or (Title:eye and Title:tracking)) and (not Title:3D and not Title:estereo and not gaze) AND (pyr >= 1990 AND pyr <= 2011)

ISI (Institute for Scientific Information) Web of Science
Title=(eye detection) OR Title=(eye tracking)
Refined by: General Categories=(SCIENCE & TECHNOLOGY) AND Topic=(image processing) AND Subject Áreas=(COMPUTER SCIENCE) AND Document Type=(ARTICLE)
Timespan=All Years.

ScienceDirect
pub-date > 1989 and TITLE(eyes detection) or TITLE(eye tracking)[All Sources(Computer Science)]

WILEY Interscience
eye detection in Article Titles OR eye tracking in Article Titles NOT estereo in Article Titles NOT head in All Fields between years 1990 and 2011

2.1.3. Identificação e Seleção de Artigos

A busca inicial retornou 375 artigos. Num primeiro momento, títulos e resumos foram rápidamente revisados levando em consideração os critérios de inclusão. Trabalhos irrelevantes ou duplicados foram removidos, e aplicando o criteio de exclusão reduzindo a lista para 23 artigos.

2.1.3.1. Artigos Incluídos

2.1.3.1.1. IEEE

1. A novel approach for real time eye state detection in fatigue awareness system
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5513139

2. A Novel Method For Eye Region Detection In Gray-level Image
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1178981

3. A Real-Time Adaptive Learning Method for Driver Eye Detection
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4700035

4. An Algorithm for real time eye detection in face images
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1334521

5. An Improved Real Time Eye State Identification System in Driver Drowsiness Detection
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4376601

6. Blink Detection and Eye Tracking for Eye Localization
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1273293

7. Communication via Eye Blinks - Detection and Duration Analysis in Real Time
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=990641

8. Corner Sharpening With Modified Harris Corner Detection to Localize Eyes in Facial Images
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5606076

9. Driver drowsiness identification by means of passive techniques for eye detection and tracking
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5729612

10. Dual-state Parametric Eye Tracking
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=840620

11. Efficient eye detection method based on grey intensity variance and independent components analysis
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5639159

12. Eye Detection and Tracking in Video Streams
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1413921

13. EYE DETECTION BASED ON RECTANGLE FEATURES AND PIXEL-PATTERN-BASED TEXTURE FEATURE
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4445995

14. Eye Detection Using Color Information and a New Efficient SVM
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5634520

15. FACE AND EYE DETECTION FOR PERSON AUTHENTICATION IN MOBILE PHONES
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4357512

16. FACE AND EYE TRACKING ALGORITHM BASED ON DIGITAL IMAGE PROCESSING
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=973079

17. Rapid human-eye detection based on an integrated method
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5471399

18. Real-time eye blink detection with GPU-based SIFT tracking
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4228575

19. Real-time Eye Detection in Video Streams
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4667828

20. Two-step Approach for Real-time Eye Tracking with a New Filtering Technique
Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=886044

2.1.3.1.2. ACM

Após a etapa de exclusão os artigos selecionados nesta biblioteca foram descartados devido a estarem em outras bibliotecas.

2.1.3.1.3. ISI

Após a etapa de exclusão os artigos selecionados nesta biblioteca foram descartados devido a estarem em outras bibliotecas.

2.1.3.1.4. Science Direct

21. Tracking and measuring drivers' eyes
Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V09-3VVCMHP-13-1&_cdi=5641&_user=687353&_pii=0262885696010918&_origin=search&_zone=rslt_list_item&_coverDate=08%2F31%2F1996&_sk=999859991&wchp=dGLzVzz-zSkWl&md5=f11efc5e07d9b836b236bab1f5b259ef&ie=/sdarticle.pdf

22. Variance projection function and its application to eye detection for human face recognition
Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V15-3V8TPN7-G-33&_cdi=5665&_user=687353&_pii=S0167865598000658&_origin=search&_zone=rslt_list_item&_coverDate=07%2F31%2F1998&_sk=999809990&wchp=dGLzVlz-zSkzk&md5=a0cbfaf10bf560798d3be980f2a5121f&ie=/sdarticle.pdf

23. Blink detection for real-time eye tracking
Link:http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6WKB-471807N-4-1&_cdi=6902&_user=687353&_pii=S108480450290130X&_origin=search&_zone=rslt_list_item&_coverDate=04%2F30%2F2002&_sk=999749997&wchp=dGLbVzW-zSkWb&md5=d6b1efaa6aa780b3164c3324bdc809e2&ie=/sdarticle.pdf

2.1.3.1.5. WILEY

Após a etapa de exclusão os artigos selecionados nesta biblioteca foram descartados devido a estarem em outras bibliotecas.

2.1.4. Extração e Verificados dos Dados

Cada trabalho selecionado para revisão foi resumido e projetado em uma tabela comparativa, cobrindo os seguintes aspectos:
Ano de publicação: Indica o ano que o artigo foi publicado.
Precisão média obtida: Indica a precisão, em porcentagem, relatada no artigo. Desempenho Médio: Indica quantos milissegundos são necessários para o processamento de um frame.
Tamanho da imagem: Indica quantos pixels de largura e altura as imagens que foram usadas no processo de testes possuem.
Processador: Indica a freqüência do processador utilizado nos testes.
Linguagem: Indica a linguagem de programação utilizada.
Numero de frames: Indica o numero de frames utilizado nos testes.
Detecção de face: Indica se o artigo descreve algum tipo de algoritmo para detecção de face.
Detecção de piscadas: Indica se o artigo descreve algum tipo de algoritmo para detecção de piscadas.

2.2. Tabela Comparativa

IDs	Ano de Publicação	Precisão Média obtida	Desempenho Médio	Tamanho da imagem	Processador	Linguagem	Numero de frames	Detecção de Face	Detecção de piscadas
1	2010	92.68%	40ms	640x480		C/C++	11500	S	S
2	2002	84%	250ms	320x240	800MHz	C/C++	56	N	N
3	2008	97%	38ms	360×288	3000MHz	C/C++	13000	S	S
4	2004	96%	300ms	640x480	1000MHz	C/C++	1423	N	N
5	2007	90%	100ms	320x240	600MHz		918	S	S
6	2003	97%	46ms	768x576	1600MHz		9100		S
7	2001	95%.	36ms	320x240	1000MHz		421	N	S
8	2010		470ms	320x240	2200MHz	MATLAB	620		N
9	2010	97%	142ms	640 x 480	2100MHz		78000		S
10	2000	98%	333ms	640x480	400MHz		500		S
11	2010	98%	100ms		1000MHz		1000		N
12	2004	98%	56ms		2000MHz		4250	S	N
13	2007	96.79%			2400MHz	C	4000	N	N
14	2010	94.92%	153ms	128x128			12776	N	N
15	2007	96%	500ms	320x240	220MHz			S	N
16	2007	91.85%	600ms	640x480	450MHz	C++	999	S	N
17	2010	94.7%.	700ms	320x240	1600MHz	MATLAB	227	S	N
18	2007	97%	40ms	640x480		OpenVIDIA	1,500	S	S
19	2008	92.73%	25ms	320×240	2800MHz	C++		S	N
20	2000		76ms	320x240	175MHz			S	N
21	1996	90%	40ms	256 x 256	80MHz	C++	30000		S
22	1998		400ms	48x30	166MHz			N	N
23	2002	90%	67ms	320x240	450MHz			S	S

2.3. Resumo dos Artigos

A seguir são fornecidos resumos dos artigos classificados de acordo com a aplicação dos critérios de inclusão após a execução da string de busca em cada uma das fontes de dados utilizadas nesta revisão sistemática de literatura.

[1]. A novel approach for real time eye state detection in fatigue awareness system

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5513139

Ano de Publicação: 2010

Autores: H Wang, L. B. Zhou, Y Ying

Resumo:

Este trabalho apresenta uma aplicação de detecção de fadiga usando uma câmera comum e uma nova forma de detectar o estado do olho. O algoritmo se divide em duas partes a detecção da região dos olhos e em seguida a classificação do estado destes. Para a execução da primeira parte é usado o método de aprendizado AdaBoost tendo como fator descriminante as caracteristicas Haar-like. Já no problema de detecção do estado do olho foi utilizado o vetor de características resultantes do Color Correlogram, com bm= dmax=10 resultanto em 1000 características, para fazer o treinamento do Ababoost. Esta abordagem se apresentou robusta e eficiente mesmo em más condições de iluminação e movimentos de cabeça.

TABLE I. ADABOOST ALGORITHM FOR LEARNING CLASSIFIER

- Given p positive and n negative samples (x_1, y_1)…, (x_{p+n}, y_{p+n}) where $y_i = \pm 1$.
- Initialize i-th sample's weight $w_1(i) = 1/(p+n)$, where i = 1, p+ n.
- Weak classifiers are simply defined by finding a threshold with minimal weighted classification error to each single feature of X.
- For k= 1, … , N iterations
  - For each feature, train a weak classifier with minimal weighted error.
  - Choose the weak classifier $h_k(x)$ with lowest minimal weighted

    error $\varepsilon_k = W_k(i) \cdot \sum_{i=1}^{p+n} |h_k(x_i) - y_i|$.
  - Update

    $$W_{k+1}(i) = \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k}, if\ h_k(x_i) = y_i \\ e^{\alpha_k}, h_k(x_i) \neq y_i \end{cases}$$

    where Z_k is a normalization factor of W_{k+1}

    and $\alpha_k = \frac{1}{2} In\left(\frac{1-\varepsilon_k}{\varepsilon_k}\right)$.
- After T iterations, the final strong classifier is given by the sign decision

  function $d(x) = \sum_{i=1}^{N} \alpha_i h_i(x)$.

Figura 28 - Algoritmo para detecção de fadiga

[2]. A Novel Method For Eye Region Detection In Gray-level Image

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1178981

Ano de Publicação: 2002

Autores: Zhiming Liu, Xin He, Jiliu Zhou. Guoqing Xiong

Resumo:

Este artigo descreve um novo método para detecção dos olhos em imagens em tons de cinza. Para a obtenção da região dos olhos este usa o operador de Laplace seguido de uma projeção horizontal Integrante, após obtida as regiões que possivelmente contem um olho é calculada a função de energia, ou seja uma função para mensurar a probabilidade de existir um olho, para cada região então estas informações são parametrizadas num algoritmo de busca genérico para dos olhos com precisão. Este algoritmo, segundo os testes mostrado no artigo, é capaz de obter 4 frames por segundo num Celeron 800Mhz com uma precisão de 84%.

[3]. A Real-Time Adaptive Learning Method for Driver Eye Detection

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4700035

Ano de Publicação: 2008

Autores: Zhang Guang-yuan, Cheng Bo, Feng Rui-jia, Zhang Xi-bo

Resumo:

Este trabalho propõe uma solução para determinar os estado de sonolência do motorista de um veiculo qualquer. Para tal é apresentado um algoritmo composto de dois modos, o modo de aprendizagem e modo de não aprendizagem. No modo de aprendizagem é utilizado diferenciação temporal para obter a região da face em seguida é obtida as bordas da imagens, aplicando regras heurísticas de tamanho do olho na imagem de bordas para obter os candidatos dos olhos, estes candidatos serão usados para formar a base de conhecimento. Quando a base de conhecimento atinge 300 imagens, o algoritmo muda para o modo de não aprendizagem, então é calculado o coeficiente de coincidência do modelo para com a imagem, se nenhum olho for achado o algoritmo volta para o modo de aprendizagem. Após obtida a região dos olhos é calculado o esqueleto dos olhos e calculado a altura do centro da imagem para dizer se os olhos estão abertos ou fechados. Este algoritmo apresentou desempenho de tempo real para o modo de aprendizagem e o de não aprendizagem com uma taxa de acerto maior que 94%, porem todos os testes foram executados em ambiente ideal.

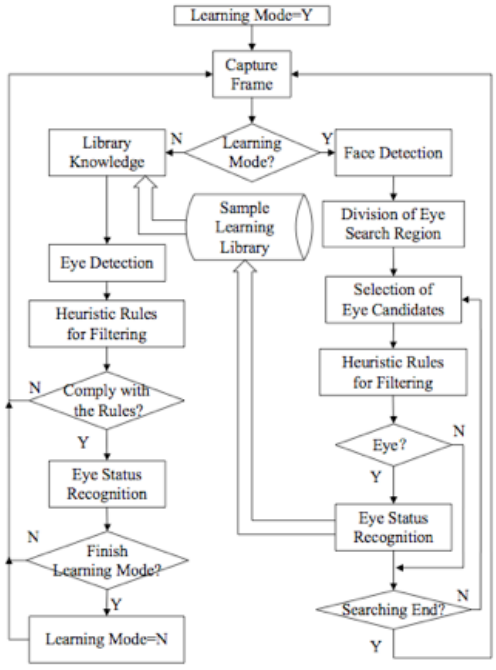

Figura 29 - Diagrama de detecção do olho

[4]. An Algorithm for real time eye detection in face images

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1334521

Ano de Publicação: 2004

Autores: T. D'Orazio, M. Leo, G.Cicirelli, A. Distante

Resumo:

Neste trabalho é proposto um novo algoritmo para detecção do olho que usa informações geométricas da íris para determinar em toda a imagem as regiões candidatas a conter um olho, e então se usa simetria para verificar a existência do olho par e usar este então como critério de parada. Um problema que a utilização do padrão da íris oferece é o fato as pessoas terem tamanho de olhos diferentes, para solver este problema o algoritmo faz a busca usando um intervalo de raio no padrão da íris. A novidade deste trabalho é que o algoritmo funciona em imagens complexas sem restrições de fundo, cor de pele e assim por diante.
Mesmo não tendo restrições de fundo, cor de pele, rotação da cabeça dentre outros este algoritmo apresentou uma ótima e precisão e bom desempenho, atingindo 7.5 frames por segundo em um Pentium III 1GHz.

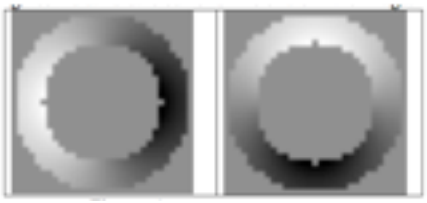

Figura 30 - Mascaras de dimensão que são convolutas com a imagem gradiente

[5]. An Improved Real Time Eye State Identification System in Driver Drowsiness Detection

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4376601

Ano de Publicação: 2007

Autores: Tianyi Hong, Huabiao Qin and Qianshu Sun

Resumo:

Este trabalho propõe um sistema de detecção do estado do olho em tempo real, este foi proposto para ser usada numa aplicação de detecção de sonolência em motorias. O primeiro passo deste algoritmo é a identificação da região da face usando Adaboost usando características haar-like, em seguida é utilizado uma projeção horizontal, sobre a imagem da face, para detectar a região dos olhos. Para a detecção do estados dos olhos, o algoritmo proposto pelo paper, aplica uma função de threshold para separar a imagem em preto em branco, em seguida é utilizado um conjunto de funções complexas para determinar o estado do olho.
O sistema proposto pelo artigo consegue uma taxa de acerto maior que 90% e 10 frames por segundo usando usando um processador de 600MHz em imagens de 320x240 pixels, o que pode ser considerado um ótimo desempenho.

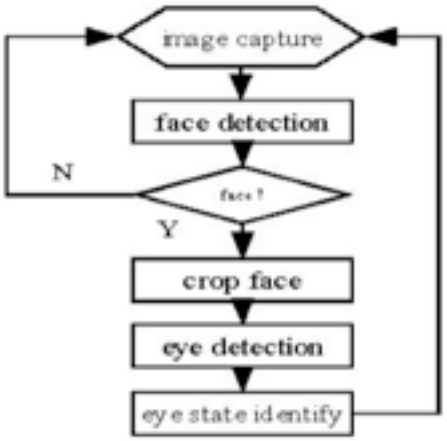

Figura 31 - Diagrama de deteção de sonolência

[6]. Blink Detection and Eye Tracking for Eye Localization

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1273293

Ano de Publicação: 2003

Autores: T.N.Bhaskar, Foo Tun Keat 8: Surendra Ranganath

Resumo:

Neste trabalho é proposto um método de usar diferenciação de frames juntamente com o cálculo do fluxo óptico para detecção do piscar dos olhos. A diferenciação de frames permite a determinação rápida de possíveis regiões em movimento. Se forem detectados, o fluxo óptico é calculado para estas regiões para detectar a direção do movimento, ou seja horizontal ou vertical, caso o movimento seja horizontal este é ignorado, pois o piscar os olhos é horizontal, em seguida os movimentos verticais da cabeça são ignorados baseados nas características de largura e na média dos vetores de velocidade do movimento. Tendo os olhos localizados então esta ão rastreados usando o Kanade Lucas Tomasi (KLT) tracker. Neste artigo foram mostrados testes os quais obtiveram um precisão de 97%, alem do bom nivel de precisão foram obtidos 25 frames por segundo em imagens de 768 x 576 pixels em um processador de 1.6GHz. Uma limitação do esquema proposto, levando em conta só a detecção dos olhos, é o fato de ser necessário um piscar de olhos é para conseguir detectar os olhos que, como foi mostrado pelo artigo, não é um problema quando o que se quer detectar é um movimento.

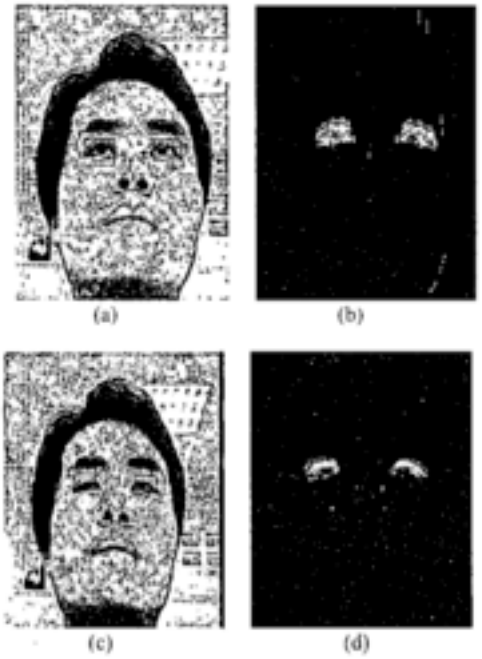

Figura 32 - Resultados por diferenciação de frames

[7]. Communication via Eye Blinks - Detection and Duration Analysis in Real Time

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=990641

Ano de Publicação: 2001

Autores: Kristen Grauman, Margrit Betke, James Gips, Gary R. Bradski

Resumo:

Neste paper é apresentado um método real time que detecta a piscada e mede com precisão a duração da mesma. O sistema se destina a fornecer uma forma alternatica de interação com um computador para permitir que pessoas com deficiências graves a utilização do mesmo. O sistema permite comunicação utilizando padrões de piscadas, tais como seqüências de piscadas longas e curtas que são interpretadas como mensagens semióticas. A localização dos olhos é determinada automaticamente através do movimento de pisca inicial do usuário. Posteriormente, o olho é rastreado porcorrelação ao longo do tempo, e mudas de aparência são automaticamente analisados a fim de classificar o olho como abertos ou fechados em cada frame. Nenhuma inicialização manual, iluminação especial, ou de detecção de prévia do rosto são necessários. O sistema foi testado com jogos interativos e uma ortografia pro-grama. Resultados demonstram a precisão da detecção geral de 95.6% e uma taxa média de 28 frames por segundo.

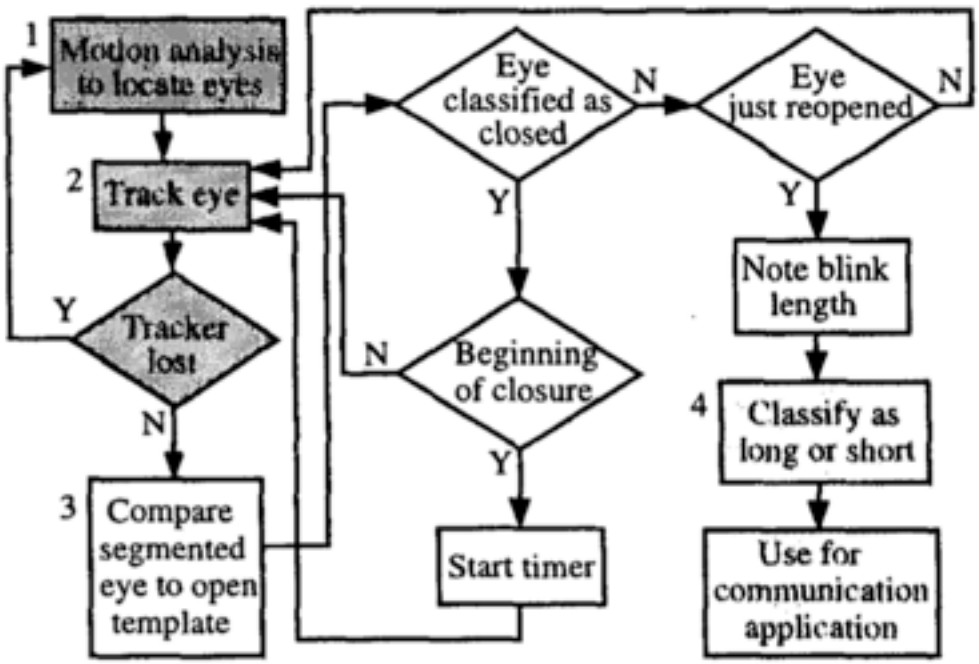

Figura 33 - Overview do sistema

[8]. Corner Sharpening With Modified Harris Corner Detection to Localize Eyes in Facial Images

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5606076

Ano de Publicação: 2010

Autores: Tianyi Hong, Huabiao Qin and Qianshu Sun

Resumo:

Neste trabalho uma nova estratégia baseada na combinação de filtros é sugerido para os localização de olhos em que os filtros são usados para localizar e destacar cantos da região, com intensidade máxima locais referidos aqui como região afiada. O processo de filtragem aqui proposto utiliza como detector de canto o método Harris em combinação com os filtros Homomorphic and Tophat-Bothat.
O uso do filtro Tophat-Bothat melhora a imagem na primeira etapa, resultando em uma região destacada com diferenças no nível de intensidade máxima após o qual a aplicação do filtro Homomorphic extrai a região desejada e suprime outros.
Pela rotulagem morfológica e busca inteligente na imagem binária o par olho é obtido. Então facilmente a caixa delimitadora e centróide de cada caixa é determinado com operadores morfológicos que encontrar propriedades para cada região rotulados na imagem binária.

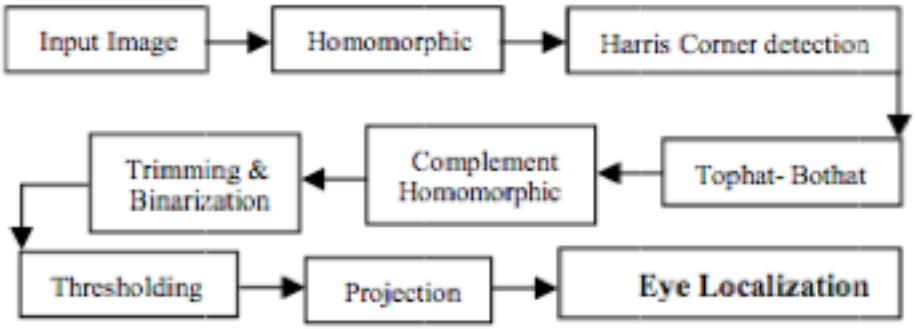

Figura 34 - Diagrama para detecção dos olhos

[9]. Driver drowsiness identification by means of passive techniques for eye detection and tracking

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5729612

Ano de Publicação: 2010

Autores: Andrea Cristiani, Marco Porta, Davide Gandini, Gian Mario Bertolotti.

Resumo:

O sistema apresentado neste paper é baseado no processamento da imagem do rosto do condutor, adquiridos por uma webcam instalada no painel do carro. Inicialmente, o rosto do motorista é identificado dentro de frames adquiridos por meio do algoritmo de Viola-Jones. Falsos positivos são evitados, selecionando, entre os possíveis candidatos, aquela cujo tamanho e posição coincidir com a localização típica e dimensão da cabeça do motorista. Por causa da posição da web cam e para reduzir a carga algoritmo computacional, somente o olho direito é rastreado. Como para o rosto, se reduz a área de interesse para a região dos olhos usando a técnica Viola-Jones. Morfologia humana também é explorada, como os olhos geralmente estão localizados, na vertical, em uma região que ocupa um terço do rosto e colocado um pouco abaixo do limite superior da cabeça. Para identificar com precisão o olho e suas características dentro da região filtrada anteriormente, foi feita uma primeira implementação de um método de detecção AAM olho. Depois de criar uma formação adequada conjunto de imagens de rostos para construir o modelo, cada foto foi anotado manualmente, colocando pontos adequada (marcos) no olho e sobrancelha (Fig. 2b). O algoritmo ICA foi então usado para construir um melhor modelo (Fig.2c), após uma inicialização dos parâmetros do modelo de acordo com a posição e o tamanho da região dos olhos foi encontrada. A análise do estado do olho é realizada somente se a cabeça do piloto estiver numa posição de descanso. A imagem do olho é convertida para tons de cinza e aplicado detecção de bordas. O nível de abertura do olho pode ser avaliado pelo cálculo do valor médio de todas as colunas e compará-lo com o modelo do olho aberto.

Figura 35 - Tracking de um olho

[10]. Dual-state Parametric Eye Tracking

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=840620

Ano de Publicação: 2000

Autores: Zhiming Liu, Xin He, Jiliu Zhou. Guoqing Xiong

Resumo:

Neste trabalho é proposto um método robusto e preciso de tracking dos olhos, capaz de detectar os estados dos olhos, e estimar os parâmetros olho para cada frame em uma seqüência. Inicializado o modelo de olho no primeiro frame o canto interno do olho pode ser rastreado com precisão através do rastreamento features points. Este algoritmo assume que os cantos externos dos olhos estão na linha conectando dois cantos internos dos olhos. Então, os cantos exteriores podem ser obtidos por informação da forma do olho que foram calculadas a no primeiro frame. A borda e intensidade da íris são usadas para detectar os estados do olho. Para um olho aberto, as pálpebras devem ser rastreadas através do feature point tracking. Para um olho fechado, os contornos das pálpebras não precisam ser rastreados. Os resultados experimentais mostram que o método funciona bem para ambos os estados do olho, olho e os parâmetros corretos são recuperados mesmo após a fechar os olhos. Este tracker funciona obustamente idependente de raça, identidade, expressão e movimentos de cabeça.

Figura 36 - Resultados do tracking pelo método proposto no artigo

[11]. Efficient eye detection method based on grey intensity variance and independent components analysis

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5639159

Ano de Publicação: 2010

Autores: M. Hassaballah T, Kanazawa, S. Ido, S. Ido

Resumo:

Neste estudo é apresentado um sistema para localizar os olhos usando Análise de componentes independentes (ICA), que é um algoritmo de aprendizado não supervisionado. Primeiro é utilizado a conbinação da variação de intensidade de tons de cinza na região dos olhos e ICA para detectar uma região em torno dos olhos. Em um segundo passo, informações de intensidade são utilizadas pala localizar o ponto central o olho. O método proposto foi avaliado em diferentes bases de dados XM2VTS, BioID e FERET e os resultados experimentais demonstram melhor desempenho em relação aos métodos existentes. Além disso, uma alta taxa de detecção de 93,3% pode ser alcançada em 600 imagens com óculos.

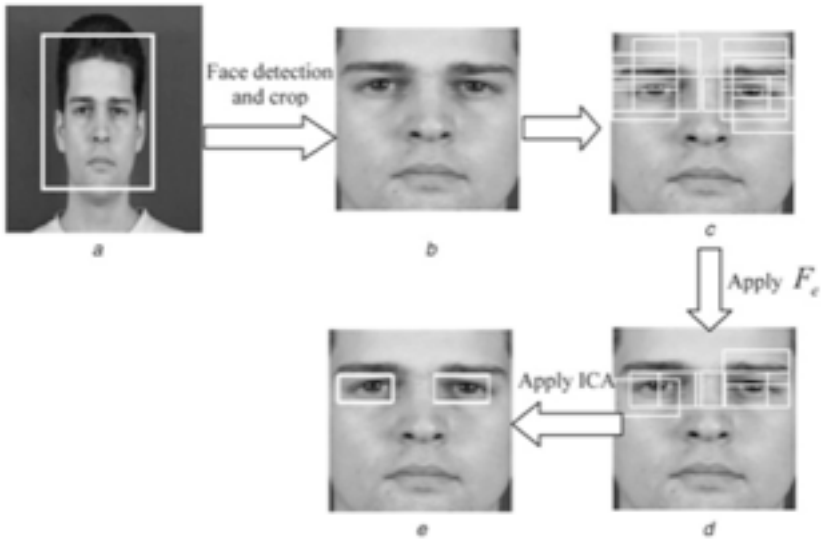

Figura 37 - Diagrama exemplificando a localização dos olhos na imagem

[12]. Eye Detection and Tracking in Video Streams

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1413921

Ano de Publicação: 2004

Autores: Abdolhossein Fathi and Mohammad T.Manzuri

Resumo:

Este trabalho apresenta um novo método para detecção de olho em uma seqüência de imagens de um vídeo. Primeiro é encontrado a região do rosto usando as características de cor da pele, então a localização dos olhos é determinada pela aplicação de três pistas na região do rosto detectado. A primeira sugestão é a intensidade dos olhos e cor, porque a intensidade de cada região do olho é relativamente baixo e sua cor diferente da cor da pele circundante. A sugestão segundo é baseado na posição e tamanho dos olhos. A terceira sugestão é baseada na convolução do proposto eye-variance-filter nos fames que possivelmente contem o olho. Em condições ideais, o desempenho do sistema foi superior a 98%.

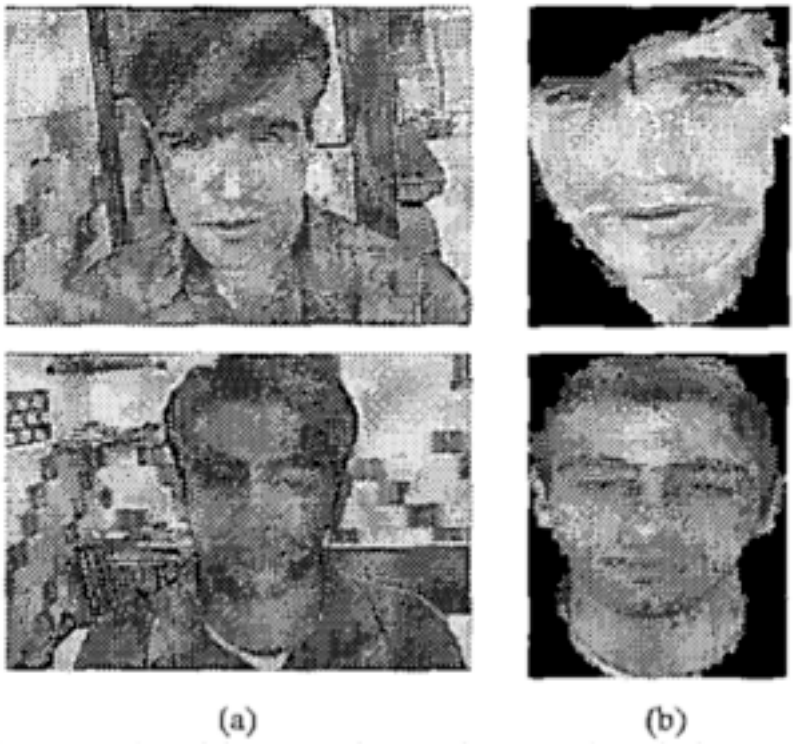

Figura 38 - Resultados da detecção da região da face: a) imagens originais e b) regiões da face detectados

[13]. Eye Detection Based on Rectangle Features and Pixel-Pattern-Based Texture Feature

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4445995

Ano de Publicação: 2007

Autores: Huchuan Lu, Wei Zhang, Deli Yang

Resumo:

Neste trabalho é proposto um novo método para detecção de olho com base em características retângulo e pixel-pattern-based texture feature (PPBTF). Primeiro é executado o classificador AdaBoost características por retângulo é construído para fazer a detecção de olho em uma imagem aproximada da frente facial. En seguida é a imagem resultante é dimensionada para 24×12 para calcular as características de PPBTF, então estas caracteristicas são colocadas no AdaBoost e num classificador SVM para obter uma melhor precisão. Algumas imagens do banco de imagens FERET foram utilizadas no experimento. Neste trabalho também foi mostrado um comparativo entre PPBTF e o LBP, onde o PPBTF mostra-se mais eficiente.

[14]. Eye Detection Using Color Information and a New Efficient SVM

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5634520

Ano de Publicação: 2010

Autores: Zhiming Liu, Xin He, Jiliu Zhou. Guoqing Xiong

Resumo:

Este trabalho apresente um método em tempo real para detecção dos olhos, de forma precisa, usando informações de cor e características wavelet juntamente com a nova e eficiente Máquinas de Vetores de Suporte (eSVM). Em particular, este método consiste em duas etapas: a seleção de candidatos do olho e validação. A etapa de seleção rejeita 99% dos pixels através de uma análise da distribuição da cor dos olhos no espaço de cor YCbCr, enquanto o restante 1% dos pixels são tratados posteriormente pela fase de validação. A etapa de validação se aplica 2D Haar wavelets para a representação da imagem em multi-escala, PCA para a redução de dimensionalidade, e eSVM para detectar o centro de um olho. O eSVM, baseado na idéia de minimizar a margem máxima de amostras classificadas erroneamente, é definido em menos vetores de suporte do que o SVM padrão, podendo assim atingir a velocidade mais rápida detecção. Experiências mostram a viabilidade deste método, que pode processar 6,25 imagens com o tamanho de 128x128 por segundo atingindo uma precisão média na detecção do olho de 94,92%.

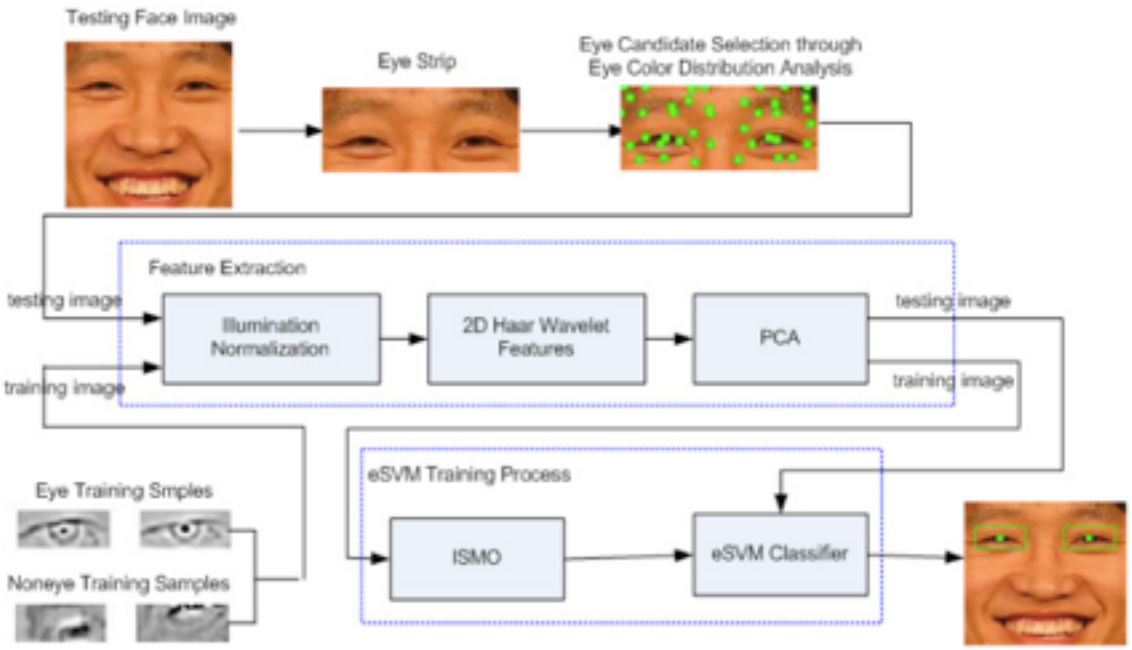

Figura 39 - Arquitetura do sistema para detecção dos ohos

[15]. Face and Eye Detection for Person Authentication in Mobile Phones

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4357512

Ano de Publicação: 2007

Autores: Zhiming Liu, Xin He, Jiliu Zhou. Guoqing Xiong

Resumo:

Este artigo descreve uma apliacação de autenticação utilizando biometria, mais especificamente a face e os olhos. Outro ponto interesante é o fato de este sistema ter sido projetado para rodar num dispositivo móvel. Basicamente ele utiliza caracteristicas Haar-like com AdaBoost para a detecção da face e dos olhos. Já para o sistema de autenticação em si é utilizado Local Binary Pattern (LBP). O algoritmo proposto é capaz de rodar a 2 frames por segundo num Nokia N90.

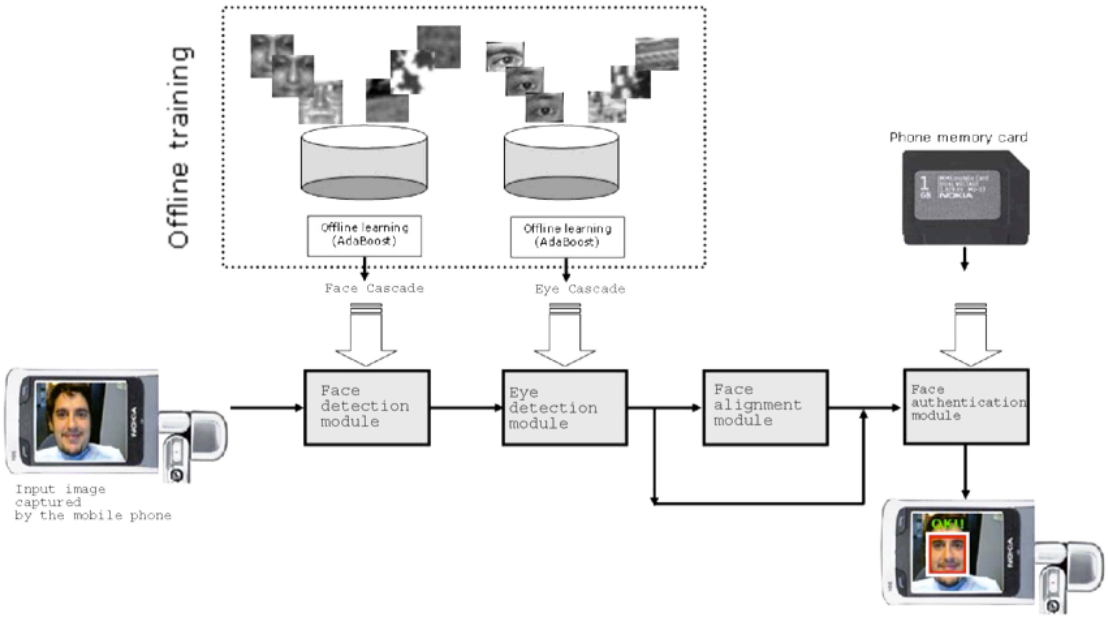

Figura 40 - Diagrama do sistema

[16]. Face and Eye Tracking Algorithm Based on Digital Image Processing

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=973079

Ano de Publicação: 2007

Autores: Claudio A. Perez, Alvaro Palma, Carlos A. Holzmann E Christian Pera

Resumo:

Neste trabalho é propomos um método de eye tracking constituido de cinco fases. Estes incluem, detecção de face grossa e fina, onde encontrará a região olhos de probabilidade máxima, mapa da localização e detecção da pupila e íris. Apenas tons de cinza foram considerados para este cálculo (8 bits). Os algoritmos de detecção de face e olhos foram avaliados em 102 imagens do banco de dados Purdue e 897 imagens de uma seqüência de vídeo. O algoritmo de detecção de rosto chegou a um 99% e 100% de taxa de detecção correta das bases de dados, respectivamente. Sobre o mesmo banco de dados o algoritmo de detecção pupila/íris alcançou 85,3% e 98,4% respectivamente a uma velocidade aproximada de 600ms um processador de 450MHz.

[17]. Rapid human-eye detection based on an integrated method

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5471399

Ano de Publicação: 2010

Autores: Liu xia, Dong yongqing, Li su, Huang chao

Resumo:

Este artigo propoem um método para localizar olhos homanos de forma precisa. O primeiro passo do algoritmo é localizar a parte facial em toda a imagem usando o algoritmo AdaBoost, depois com o método de extração de características localizar a área aproximada de um olho humano, e finalmente,encontrar a exata localização do olho com um método de correspondência de modelo melhorado. O algoritmo proposto integra o método de extração de características e o método de correspondência de modelo, fazendo uso das suas vantagens. O algoritmo de detecção de olhos proposto pode efetivamente reduzir a área de olho de detecção, e melhorar a velocidade de detecção e precisão.
Os resultados experimentais mostram que, sem oculos ou parte do olho coberto, pode-se ser detectar os olhos com rapidez e eficiência. Ao mesmo tempo, o algoritmo proposto também tem alguns inconvenientes. Quando os olhos estão bloqueadas por óculos ou cabelo, o algoritmo proposto executa mal.

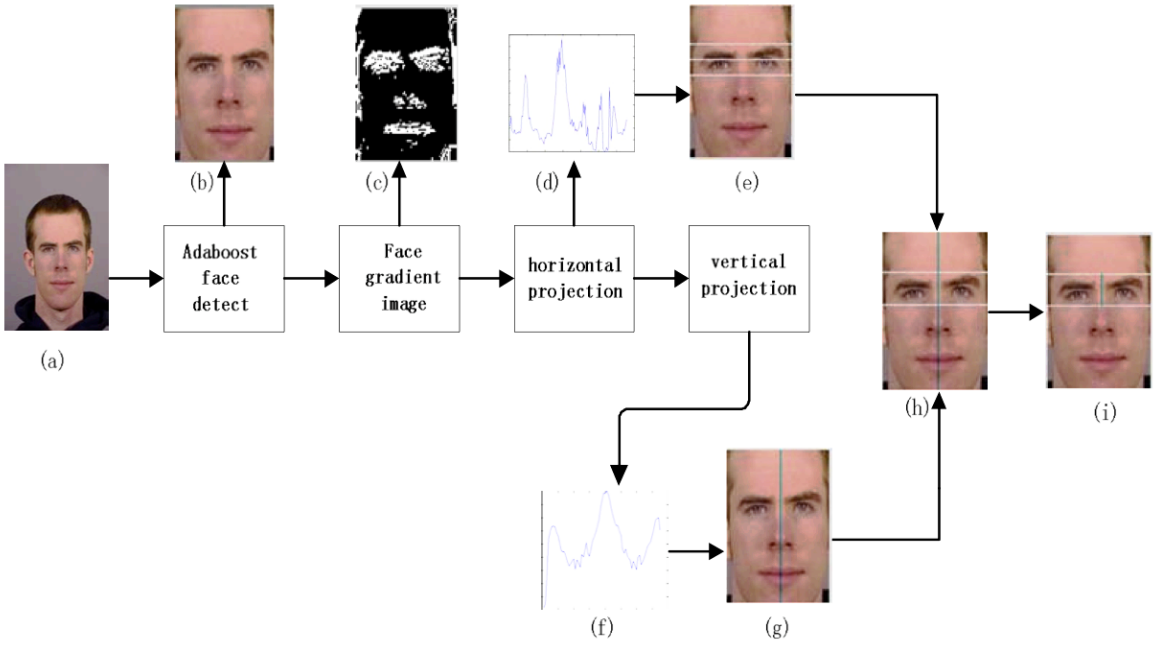

Figura 41 - Diagrama de detecção dos olhos

[18]. Real-time eye blink detection with GPU-based SIFT tracking

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4228575

Ano de Publicação: 2007

Autores: Zhiming Liu, Xin He, Jiliu Zhou. Guoqing Xiong

Resumo:

Este artigo relata a implementação em GPU(OpenVIDIA) de um detector de piscadas em tempo real sobre condições de iluminação próximas ao infravermelho. O piscar de olhos são detectados dentro de regiões de interesse que estão alinhados com os olhos do sujeito na inicialização. O alinhamento é mantido através do tempo pelo tracking dos postos caracteristicos SIFT que são utilizados para estimar a transformação entre a face inicial e a dos próximos frames. A implementação em GPU do algoritmo de extração dos pontos caracteristicos SIFT garante processamento em tempo real. Os pontos característicos são encontrados usando o recurso de escala-invariante de transformação.

Algorithm 1 Algorithm for blink detection. See Figure 2.

1. *Locate motion regions using frame differentiation in each eye roi.*
2. *Threshold the motion regions and keep the better blob based on position, area, angle, density and width/height ratio.*
3. *Repeat 1-2 until candidates in both regions are found.*
4. *Compute optical flow field in the blob regions and extract the vertical and horizontal vectors in the blobs' coordinate system.*
5. *If the dominant motion is downward for both regions, the closing frame is saved otherwise steps (1) to (5) are repeated.*
6. *Repeat steps (1) to (4) with the additional constraint which is that blob position must be close enough to the saved closing position.*
7. *If an opening frame is found within the determined maximum blink length, the blink sequence is kept for further analysis, otherwise the blink hypothesis is rejected and the process restarts from step (1).*

Figura 42 - Algoritmo proposto no artigo

[19]. Real-time Eye Detection in Video Streams

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4667828

Ano de Publicação: 2002

Autores: Kunhui Lin, Jiyong Huang, Jiawei Chen, Changle Zhou

Resumo:

Um esquema de detecção rápida do olho para uso em transmissões de vídeo é apresentado neste artigo. A coerência temporal de quadros seqüenciais foi usado para melhorar significativamente a velocidade de detecção. Primeiro, é utilizado o AdaBoost para localizar um frame que contem os olhos. Em seguida, essas posições dos candidatos são filtrados por padrões geométricos dos olhos humanos. Após cada quadro é detectado, a janela de detecção é atualizada. Em nossos experimentos a taxa de detecção média foi de 92,73% para 320 × 240 vídeos teste de resolução,com uma velocidade de 24.98ms por quadro. Essa velocidade é mais rápida do que a pesquisa anterior, no entanto a taxa de detecção não diminuir drasticamente.

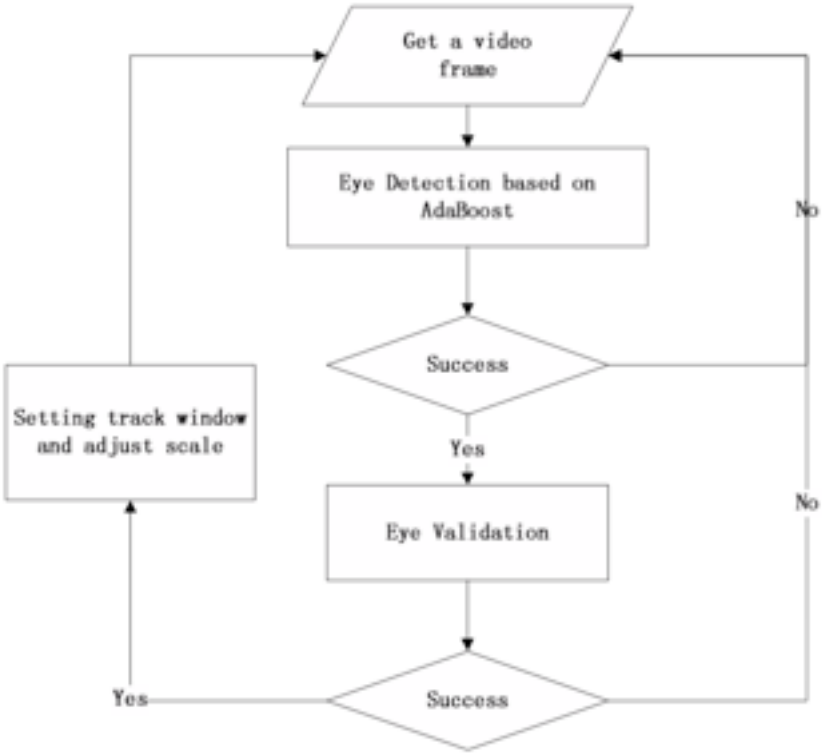

Figura 43 - Overview do sistema

[20]. Two-step Approach for Real-time Eye Tracking with a New Filtering Technique

Link: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=886044

Ano de Publicação: 2000

Autores: Shinjiro Kawato e Jun Ohya

Resumo:

Este artigo propoem uma abordagem em duas fases para rastreamento ocular em transmissões de vídeo. Primeiro é detectado / acompanhado um ponto entre os olhos. Para esta tarefa, é utilizado o filtro Circle-frequency filter que fora proposto anteriormente pelo autor. Assim que que é feita a detecção do ponto entre os olhos é bastante fácil para localizar os olhos, que são as duas pequenas partes mais escuras em cada lado deste ponto.
O sistema foi implementado numa estação de trabalho SGI O2. O tamanho da imagem de vídeo é de 320x240 pixels. O sistema processa imagens em 7 frames por segundo no modo de detecção e 13 frames por segundo no modo de rastreamento sem qualquer hardware especial.

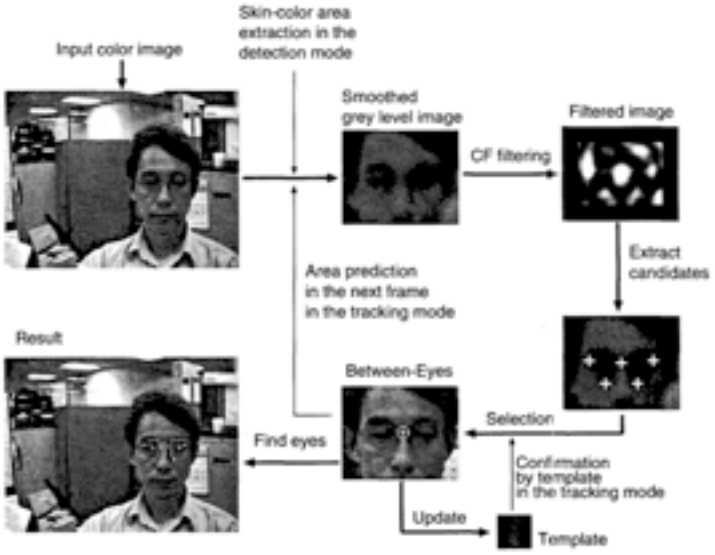

Figura 44 - Overview do processo de detecção e tracking

[21]. Tracking and measuring drivers eyes

Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V09-3VVCMHP-13-1&_cdi=5641&_user=687353&_pii=0262885696010918&_origin=search&_zone=rslt_list_item&_coverDate=08%2F31%2F1996&_sk=999859991&wchp=dGLzVzz-zSkWl&md5=f11efc5e07d9b836b236bab1f5b259ef&ie=/sdarticle.pdf

Ano de Publicação: 1996

Autores: David Tack, Ian Craw

Resumo:

Este paper apresenta um algoritmo para a detecção de sonolência que é divididos em seis módulos como se pode ver na imagem abaixo. O sucesso do loop principal é largamente determinado pelos estágios de inicialização. A inicialização pode ser bom suficiente para uma sessão de condução particular, enquanto um inicialização pobre vai levar rapidamente ao fracasso, e uma re-inicialização. O loop principal demora aproximadamente 40 ms por ciclo, permitindo que 25 imagens por segundo a serem processadas.

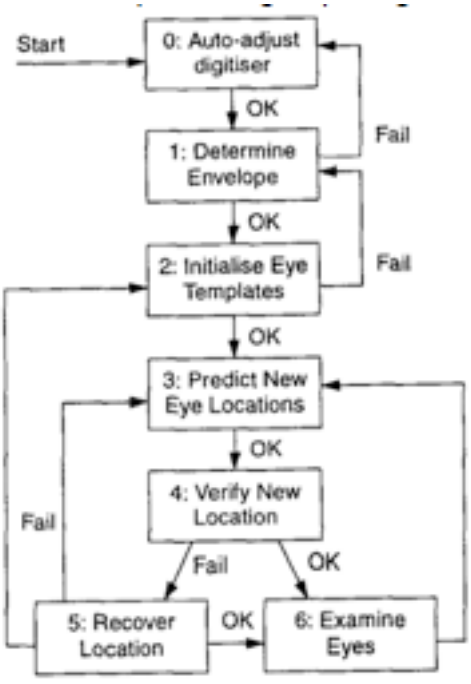

Figura 45 - Diagrama de blocos exibindo a estrutura do sistema

[22]. Variance projection function and its application to eye detection for human face recognition

Link: http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V15-3V8TPN7-G-33&_cdi=5665&_user=687353&_pii=S0167865598000658&_origin=search&_zone=rslt_list_item&_coverDate=07%2F31%2F1998&_sk=999809990&wchp=dGLzVlz-zSkzk&md5=a0cbfaf10bf560798d3be980f2a5121f&ie=/sdarticle.pdf

Ano de Publicação: 1998

Autores: G.C. Feng, P.C. Yuen

Resumo:

A função de projeção da variação é desenvolvida e relatada neste artigo. O VPF proposto pode ser utilizado como uma ferramenta básica e aplicada para reconhecimento de padrões e aplicações de visão computacional. Neste trabalho, a VPF foi aplicada com sucesso na detecção dos olhos e os resultados são encorajadores. Além disso, a complexidade computacional é relativamente baixa. O tempo de computação para extrair um olho de um 48 x30 imagem é inferior a 0,4 segundo em um 166 MHz Pentium, o que o torna adequado para uso em aplicações práticas.

[23]. Blink detection for real-time eye tracking

Link:http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6WKB-471807N-4-1&_cdi=6902&_user=687353&_pii=S108480450290130X&_origin=search&_zone=rslt_list_item&_coverDate=04%2F30%2F2002&_sk=999749997&wchp=dGLbVzW-zSkWb&md5=d6b1efaa6aa780b3164c3324bdc809e2&ie=/sdarticle.pdf

Ano de Publicação: 2002

Autores: T. Morris, P. Blenkhorn and Farhan Zaidi

Resumo:

Este trabalho é motivado pelo objetivo de proporcionar controle de um computador baseado em movimentos de cabeça e olhos com o intuido de possibilitar a utilização deste por pessoas que possuem deficiencias fisicas. O sistema descrito aqui usa espaço-temporal de filtragem, proposto por Turk, e mapas de variância para localizar a cabeça e encontrar os pontos de interesse da região dos olhos. Estes pontos de interesse da região dos olhos são utilizados para fazer o tracking dos olhos usando o algoritmo de rastreamento de Lucas Kanadecom. Com isto foi possivel o tracking de olhos e cabeça com um taxa de 30 frames por segundo e mais de 90 % de acerto sob variadas condições de iluminação para as pessoas de diferentes etnias, com e sem o uso de óculos.

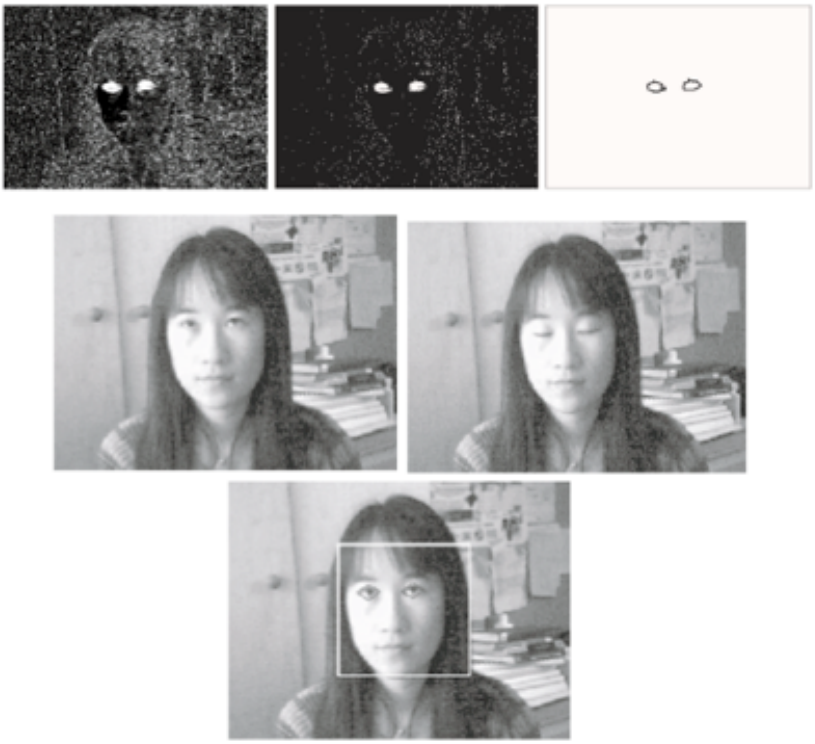

Figura 46 - Exemplos do processo de detecção de sonolência

3. Conclusões

Este relatório técnico compreendeu duas revisões sistemáticas visando levantar qual o estado da arte em relação às metodologias para detecção de sonolência de motoristas em trânsito através do processamento de imagens em tempo real.

Para tanto, 48 artigos de temas correlatos foram selecionados e apresentados, primeiramente abordando quais as soluções propostas em relação ao uso dos mais diversos fatores precursores da sonolência, alguns abordando inclusive a questão de fadiga ou monotonia no trânsito. Em seguida foram apresentados artigos mais relacionados ao tema do ponto de vista computacional, ou seja, realizando um levantamento das técnicas de processamento de imagens que podem ser utilizadas para identificar e fazer proveito dos precursores fisiológicos da sonolência apresentados na primeira parte do relatório.

3.1 Precursores Fisiológicos

As soluções propostas fazem uso dos mais diversos fatores precursores da sonolência, alguns abordando inclusive a questão de fadiga ou monotonia no trânsito. Embora estas solução difiram consideravalmente, todas elas tem um ponto em comum: o fato de fazerem uso de estatísticas relacionadas à frequência do piscar das pálpebras como fator principal ou de grande peso na determinação do diganóstico de sonolência.

Vários trabalhos aqui citados apresentam fundamentos fisiológicos consistentes embasando o uso de tais parâmetros como indicadores precursores da sonolência no trânsito. Com a exceção de um, todos os demais trabalhos apresentam experimentos realizados com diferente número de participantes tanto em situação reais quanto simuladas, validando assim o uso de tais parâmetros.

Concluí-se portanto que parâmetros como a frequência de piscadas dos olhos, o tempo entre piscadas e o tempo em que as pálpebras permanecem fechadas são indicatores confiáveis para diagnosticar o estado de consciência no que diz respeito à sonolência de um motorista em trânsito.

3.2 Técnicas e Algoritmos

As soluções basicamente se dividem em duas, sem aprendizagem de máquina e com aprendizagem de máquina. Dentre os algoritmos de aprendizagem de máquina tem-se destaque para o AdaBoost com Haar-like como fator discriminante, este é principalmente usado para detecção da face ou uma aproximação da região dos olhos. Dentre os algoritmos que não utilizam aprendizagem de máquina dois deles se mostram bastantes usados, um deles é a utilização do espaço-temporal normalmente utilizado para a detecção da face. O outro é a projeção horizontal para achar a região que contem os olhos. Já no problema de detecção dos olhos em si foram apresentadas varias técnicas como VPF (Variance Projection Function), utilização das informações geométricas da íris, Color Correlogram com AdaBoost, utilização do espaço-temporal, entre outras.

Vários trabalhos aqui citados apresentam testes com um numero de imagens consideráveis cobrindo varias situações de iluminação, posição da cabeça e a utilização de óculos, nos quais fora repostados bons níveis de precisão e desempenho tendo inclusive um artigo que apresenta uma solução baseada em um celular N97.

3.3 Vantagens

É sabido que a sonolência ao voltante representa um importante risco nas rodovias, tendo sido reportado por vários estudados que estabeleceram fortes relações entre a sonolência do motorista e acidentes no trânsito.

Um sistema de monitoração de sonolência em tempo real capaz de detectar a falta de atenção visual cedo o suficiente para alertar o motorista sobre seu estado de sonolência, pode diminuir consideravelmente o índice de mortes e/ou gastos da população.

Para tanto, se faz necessário um dispositivo aclopado ao veículo. Entende-se que um dispositivo móvel tal qual um smartphone forneceria esta solução por um custo mais reduzido do que a implantação de tal sistema em um veículo, não sendo necessário realizar nenhum tipo de modificação no carro, nem mesmo a contratação de profissionais especializados.

3.4 Dificuldades e Sugestões

A maioria dos artigos que abordam o tema de detecção de sonolência baseado no processamento de imagens apresentam soluções que executam em um computador pessoal de baixo poder de processamento. Destas grande parte são de tempo real.

Embora isso indique a possibilidade de realização da mesma tarefa em um dispositivo móvel de alto poder de processamento, nenhum artigo abordando a execução de algoritmos visando plataformas móveis apresenta soluções em tempo real.

Este pode ser um indicador da dificuldade de implementação de uma solução de monitoramento de sonolência via processamento de imagens de boa performance em um celular, afinal vários são os passos até que se possa tomar conclusões a respeito da sonolência, sendo necessário localizar a cabeça do motorista na imagem, localizar os olhos dentro desta segunda área e apenas então realizar os cálculos necessários sobre a abertura e fechamento das pálpebras. Todos estes algoritmos quando combinados demandam uma capacidade de processamento que pode inviabilizar uma solução móvel de tempo real.

4. Referências

[1] FLORES, Marco Javier; ARMINGOL, José Maria; LA ESCALERA, Arturo de (Org.). Driver Drowsiness Warning System Using Visual Information for Both Diurnal and Nocturnal Illumination Conditions. Disponível em: <http://delivery.acm.org/10.1145/1930000/1928491/p3-flores.pdf?ip=150.162.246.33&CFID=31652632&CFTOKEN=90441450&__acm__=1309388743_9fb27f3a7b3feb298ac884b62b212420>. Acesso em: 10 jun. 2011.

[2] PAPADELIS, Christos; CHEN, Zhe; KOURTIDOU-PAPADELI, Chrysoula (Org.). Monitoring sleepiness with on-board electrophysiological recordings for preventing sleep-deprived traffic accidents. Disponível em: <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VNP-4P899C7-1-1&_cdi=6184&_user=687353&_pii=S1388245707002945&_origin=search&_zone=rslt_list_item&_coverDate=09%2F30%2F2007&_sk=998819990&wchp=dGLzVlz-zSkWW&md5=9e3cf31aa01c96cf6c840953c078f990&ie=/sdarticle.pdf>. Acesso em: 10 jun. 2011.

[3] BARBATO, Giuseppe; PADOVA, Vittoria De; PAOLILLO, Antonella Raffaella (Org.). Increased spontaneous eye blink rate following prolonged wakefulness. Disponível em: <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6T0P-4M6SBM8-2-1&_cdi=4868&_user=687353&_pii=S0031938406004185&_origin=search&_zone=rslt_list_item&_coverDate=01%2F30%2F2007&_sk=999099998&wchp=dGLzVlz-zSkWA&md5=a441d72f311834bc25802462129f8267&ie=/sdarticle.pdf>. Acesso em: 10 jun. 2011.

[4] NUMATA, Nakaho; KITAJIMA, Hiroki; GOI, Yoshihiro (Org.). Analysis of drivers' behavior before and after crashes in simulated expressway driving to predict sleepiness levels for doze alarm activation. Disponível em: <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V3Y-3V57C42-9-P&_cdi=5743&_user=687353&_pii=S0389430498000113&_origin=search&_zone=rslt_list_item&_coverDate=07%2F01%2F1998&_sk=999809996&wchp=dGLzVlb-zSkWW&md5=9072492c59f24eac8a9180af91a87916&ie=/sdarticle.pdf>. Acesso em: 10 jun. 2011.

[5] CAFFIER, Philipp P.; ERDMANN, Udo; ULLSPERGER, Peter (Org.). Experimental evaluation of eye-blink parameters as a drowsiness measure. Disponível em: <http://www.springerlink.com/content/e6x53ulv1yrm9w3t/fulltext.pdf>. Acesso em: 10 jun. 2011.

[6] CAFFIER, Philipp P.; ERDMANN, Udo; ULLSPERGER, Peter (Org.). The spontaneous eye-blink as sleepiness indicator in patients with obstructive sleep apnoea syndrome-a pilot study. Disponível em: <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6W6N-4FDNBX4-1-5&_cdi=6603&_user=687353&_pii=S1389945704002230&_origin=&_coverDate=03%2F31%2F2005&_sk=999939997&view=c&wchp=dGLbVzW-

zSkWB&md5=04a960a2029cdd4148dae6e6faccf869&ie=/sdarticle.pdf>. Acesso em: 10 jun. 2011.

[7] VERWEY, Willem B.; ZAIDEL, David M. (Org.). Predicting drowsiness accidents from personal attributes, eye blinks and ongoing driving behaviour. Disponível em: <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6V9F-3XH3HFM-D-1&_cdi=5897&_user=687353&_pii=S0191886999000896&_origin=&_coverDate=01%2F01%2F2000&_sk=999719998&view=c&wchp=dGLzVlz-zSkWW&md5=59b37ea4119f75ddc3bb02b81f6e9d0b&ie=/sdarticle.pdf>. Acesso em: 10 jun. 2011.

[8] PADOVA, Vittoria De; BARBATO, Giuseppe; CONTE, Francesca (Org.). Diurnal variation of spontaneous eye blink rate in the elderly and its relationships with sleepiness and arousal. Disponível em: <http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6T0G-4WS2J1D-1-7&_cdi=4862&_user=687353&_pii=S0304394009008568&_origin=&_coverDate=09%2F29%2F2009&_sk=995369998&view=c&wchp=dGLzVzz-zSkWB&md5=4beb7408c4f67da983362409856a374c&ie=/sdarticle.pdf>. Acesso em: 10 jun. 2011.

[9] VINCENT, Victor J.; SUBBARAO, Wunnava (Org.). Software for an expert system for human fatigue analysis. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=95192>. Acesso em: 10 jun. 2011.

[10] FUNADA, Mariko Fujikake; NINOMIJA, Satoki P.; SUZUKI, Satoshi (Org.). On an Image Processing of Eye Blinking to Monitor Awakening Levels of Human Beings. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=652663>. Acesso em: 10 jun. 2011.

[11] SMITH, Paul; SHAH, Mubarak; LOBO, N. da Vitoria (Org.). Monitoring Head/Eye Motion for Driver Alertness with One Camera. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=902999>. Acesso em: 10 jun. 2011.

[12] KOJIMA, Natsuki; KOZUKA, Kazuhiro; NAKANO, Tomoaki (Org.). Detection of Consciousness Degradation and Concentration of a Driver for Friendly Information Service. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=961722>. Acesso em: 10 jun. 2011.

[13] SMITH, Paul; SHAH, Mubarak; LOBO, Niels da Vitoria (Org.). Determining Driver Visual Attention With One Camera. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1260587>. Acesso em: 10 jun. 2011.

[14] BRANDT, Thomas; STEMMER, Ralf; RAKOTONIRAINY, Andry (Org.). Affordable Visual Driver Monitoring System for Fatigue and Monotony. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1401415>. Acesso em: 10 jun. 2011.

[15] MIYAKAWA, Tomofurni; TAKANO, Hironobu; NAKAMURA, Kiyomi (Org.). Development of Non-contact Real-time Blink Detection System for Doze Alarm. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1491689>. Acesso em: 10 jun. 2011.

[16] BENOIT, A.; CAPLIER, A. (Org.). Hypovigilence Analysis: Open or Closed Eye or Mouth? Blinking or Yawning Frequency? Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1577268>. Acesso em: 10 jun. 2011.

[17] BERGASA, Luis M.; NUEVO, Jesús; SOTELO, Miguel A. (Org.). Real-Time System for Monitoring Driver Vigilance. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1603553>. Acesso em: 10 jun. 2011.

[18] PARK, Ilkwon; AHN, Jung-ho; BYUN, Hyeran (Org.). Efficient Measurement of Eye Blinking under Various Illumination Conditions for Drowsiness Detection Systems. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1698913>. Acesso em: 10 jun. 2011.

[19] SHINODA, Takayuki; KATO, Masami (Org.). A Pupil Diameter Measurement System for Accident Prevention. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4274098>. Acesso em: 10 jun. 2011.

[20] SUZUKI, Mai; YAMAMOTO, Nozomi; YAMAMOTO, Osami (Org.). Measurement ofDriver's Consciousness by Image Processing - A Method for Presuming Driver's Drowsiness by Eye-Blinks coping with Individual Differences-. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4274320>. Acesso em: 10 jun. 2011.

[21] BATISTA, Jorge (Org.). A Drowsiness and Point of Attention Monitoring System for Driver Vigilance. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04357702>. Acesso em: 10 jun. 2011.

[22] CHANG, Byung-chan; LIM, Jung-eun; KIM, Hae-jin (Org.). A study of classification of the level of sleepiness for the drowsy driving prevention. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4421521>. Acesso em: 10 jun. 2011.

[23] SAEED, Isaam; WANG, Alyoxia; SENARATNE, Rajinda (Org.). Using the Active Appearance Model to Detect Driver Fatigue. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4544792>. Acesso em: 10 jun. 2011.

[24] NAKAYAMA, Minoru; YAMAMOTO, Keiko; KOBAYASHI, Fumio (Org.). Estimation of Sleepiness using Frequency Components of Pupillary Response. Disponível em:

<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4696948>. Acesso em: 10 jun. 2011.

[25] TSUCHIDA, Ayumi; BHUIYAN, Md. Shoaib; OGURI, Koji (Org.). Estimation of Drowsiness Level Based on Eyelid Closure and Heart Rate Variability. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5334766>. Acesso em: 10 jun. 2011.

[26]. Wang, H.; Zhou, L.B.; Ying, Y.; , "A novel approach for real time eye state detection in fatigue awareness system," Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on , vol., no., pp.528-532, 28-30 June 2010
doi: 10.1109/RAMECH.2010.5513139
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5513139&isnumber=5513125

[27]. Zhiming Liu; Xin He; Jiliu Zhou; Guoqing Xiong; , "A novel method for eye region detection in gray-level image," Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on , vol.2, no., pp. 1118- 1121 vol.2, 29 June-1 July 2002
doi: 10.1109/ICCCAS.2002.1178981
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1178981&isnumber=26481

[28]. Guangyuan Zhang; Bo Cheng; Ruijia Feng; Xibo Zhang; , "A Real-Time Adaptive Learning Method for Driver Eye Detection," Computing: Techniques and Applications, 2008. DICTA '08.Digital Image , vol., no., pp.300-304, 1-3 Dec. 2008
doi: 10.1109/DICTA.2008.43
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4700035&isnumber=4699978

[29]. D'Orazio, T.; Leo, M.; Cicirelli, G.; Distante, A.; , "An algorithm for real time eye detection in face images," Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on , vol.3, no., pp. 278- 281 Vol.3, 23-26 Aug. 2004
doi: 10.1109/ICPR.2004.1334521
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1334521&isnumber=29387

[30]. Tianyi Hong; Huabiao Qin; Qianshu Sun; , "An Improved Real Time Eye State Identification System in Driver Drowsiness Detection," Control and Automation, 2007. ICCA 2007. IEEE International Conference on , vol., no., pp.1449-1453, May 30 2007-June 1 2007
doi: 10.1109/ICCA.2007.4376601
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4376601&isnumber=4376307

[31]. Bhaskar, T.N.; Foo Tun Keat; Ranganath, S.; Venkatesh, Y.V.; , "Blink detection and eye tracking for eye localization," TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region , vol.2, no., pp. 821- 824 Vol.2, 15-17 Oct. 2003
doi: 10.1109/TENCON.2003.1273293
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1273293&isnumber=28486

[32]. Grauman, K.; Betke, M.; Gips, J.; Bradski, G.R.; , "Communication via eye blinks - detection and duration analysis in real time," Computer Vision and Pattern Recognition, 2001. CVPR 2001.

Proceedings of the 2001 IEEE Computer Society Conference on , vol.1, no., pp. I-1010- I-1017 vol.1, 2001
doi: 10.1109/CVPR.2001.990641
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=990641&isnumber=21353

[33]. Lak, M.; Soleimani Yazdi, A.-M.; , "Corner sharpening with modified harris corner detection to localize eyes in facial images," ELMAR, 2010 PROCEEDINGS , vol., no., pp.27-31, 15-17 Sept. 2010
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5606076&isnumber=5606063

[34]. Cristiani, A.; Porta, M.; Gandini, D.; Bertolotti, G.M.; Serbedzija, N.; , "Driver Drowsiness Identification by Means of Passive Techniques for Eye Detection and Tracking," Self-Adaptive and Self-Organizing Systems Workshop (SASOW), 2010 Fourth IEEE International Conference on , vol., no., pp.142-146, 27-28 Sept. 2010
doi: 10.1109/SASOW.2010.30
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5729612&isnumber=5729523

[35]. Ying-li Tian; Kanade, T.; Cohn, J.F.; , "Dual-state parametric eye tracking," Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on , vol., no., pp.110-115, 2000
doi: 10.1109/AFGR.2000.840620
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=840620&isnumber=18088

[36]. Hassaballah, M.; Kanazawa, T.; Ido, S.; , "Efficient eye detection method based on grey intensity variance and independent components analysis," Computer Vision, IET , vol.4, no.4, pp.261-271, Dec. 2010
doi: 10.1049/iet-cvi.2009.0097
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5639159&isnumber=5639156

[37]. Fathi, A.; Manzuri, M.T.; , "Eye detection and tracking in video streams," Communications and Information Technology, 2004. ISCIT 2004. IEEE International Symposium on , vol.2, no., pp. 1258- 1261 vol.2, 26-29 Oct. 2004
doi: 10.1109/ISCIT.2004.1413921
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1413921&isnumber=30638

[38]. Huchuan Lu; Wei Zhang; Deli Yang; , "Eye detection based on rectangle features and pixel-pattern-based texture features," Intelligent Signal Processing and Communication Systems, 2007. ISPACS 2007. International Symposium on , vol., no., pp.746-749, Nov. 28 2007-Dec. 1 2007
doi: 10.1109/ISPACS.2007.4445995
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4445995&isnumber=4445799

[39]. Shuo Chen; Chengjun Liu; , "Eye detection using color information and a new efficient SVM," Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on , vol., no., pp.1-6, 27-29 Sept. 2010
doi: 10.1109/BTAS.2010.5634520
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5634520&isnumber=5634461

[40]. Hadid, A.; Heikkila, J.Y.; Silven, O.; Pietikainen, M.; , "Face and Eye Detection for Person Authentication in Mobile Phones," Distributed Smart Cameras, 2007. ICDSC '07. First ACM/IEEE International Conference on , vol., no., pp.101-108, 25-28 Sept. 2007
doi: 10.1109/ICDSC.2007.4357512
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4357512&isnumber=4357491

[41]. Perez, C.A.; Palma, A.; Holzmann, C.A.; Pena, C.; , "Face and eye tracking algorithm based on digital image processing ," Systems, Man, and Cybernetics, 2001 IEEE International Conference on , vol.2, no., pp.1178-1183 vol.2, 2001
doi: 10.1109/ICSMC.2001.973079
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=973079&isnumber=20974

[42]. Liu Xia; Dong Yongqing; Li Su; Huang Chao; , "Rapid Human-Eye Detection Based on an Integrated Method," Communications and Mobile Computing (CMC), 2010 International Conference on , vol.1, no., pp.3-7, 12-14 April 2010
doi: 10.1109/CMC.2010.115
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5471399&isnumber=5471384

[43]. Lalonde, M.; Byrns, D.; Gagnon, L.; Teasdale, N.; Laurendeau, D.; , "Real-time eye blink detection with GPU-based SIFT tracking," Computer and Robot Vision, 2007. CRV '07. Fourth Canadian Conference on , vol., no., pp.481-487, 28-30 May 2007
doi: 10.1109/CRV.2007.54
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4228575&isnumber=4228510

[44]. Kunhui Lin; Jiyong Huang; Jiawei Chen; Changle Zhou; , "Real-Time Eye Detection in Video Streams," Natural Computation, 2008. ICNC '08. Fourth International Conference on , vol.6, no., pp.193-197, 18-20 Oct. 2008
doi: 10.1109/ICNC.2008.278
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4667828&isnumber=4667783

[45]. Kawato, S.; Ohya, J.; , "Two-step approach for real-time eye tracking with a new filtering technique," Systems, Man, and Cybernetics, 2000 IEEE International Conference on , vol.2, no., pp.1366-1371 vol.2, 2000
doi: 10.1109/ICSMC.2000.886044
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=886044&isnumber=19153

[46]. David Tock, Ian Craw, Tracking and measuring drivers' eyes, Image and Vision Computing, Volume 14, Issue 8, 6th British Machine Vision Conference, August 1996, Pages 541-547, ISSN 0262-8856, DOI: 10.1016/0262-8856(96)01091-8.
(http://www.sciencedirect.com/science/article/pii/0262885696010918)
Keywords: Blink-rate; Eye tracking; Driver monitoring

[47]. G. C. Feng, P. C. Yuen, Variance projection function and its application to eye detection for human face recognition, Pattern Recognition Letters, Volume 19, Issue 9, July 1998, Pages 899-906, ISSN 0167-8655, DOI: 10.1016/S0167-8655(98)00065-8.
(http://www.sciencedirect.com/science/article/pii/S0167865598000658)

Keywords: Face recognition; Eye detection; Variance projection function; Biometric identification

[48]. T. Morris, P. Blenkhorn, Farhan Zaidi, Blink detection for real-time eye tracking, Journal of Network and Computer Applications, Volume 25, Issue 2, April 2002, Pages 129-143, ISSN 1084-8045, DOI: 10.1006/jnca.2002.0130.
(http://www.sciencedirect.com/science/article/pii/S108480450290130X)

Apêndice B – Código Fonte

AdaBoost

```
@interface AdaBoost ()

- (StrongClassifier *)strongClassifierWithFeatures:(int)numberOfFeatures
andTrainSet:(TrainSet *)trainSet;

@end

@implementation AdaBoost

- (CascadedClassifier
*)cascadedClassfiersWithFalsePositiveRateForLayer:(float)falsePositiveRate
detectionRateForLayer:(float)detectionRate
andTargetFalsePositiveRate:(float)targetFalsePositiveRate {
   CascadedClassifier *cascadedClassifier = [[CascadedClassifier alloc] init];

   TrainSet *trainSet = [TrainSetDataSource trainSet];

   while (cascadedClassifier.falsePositiveRate > targetFalsePositiveRate) {
      int numberOfFeatures = 0;

      float curentDetactionRate = 0;
      float curentFalsePossitiveRate = 1.0;
      StrongClassifier *strongClassifier = nil;

      NSMutableArray *newNonEyes = [NSMutableArray array];

      while (!(curentDetactionRate >= detectionRate && curentFalsePossitiveRate <=
falsePositiveRate)) {
         numberOfFeatures ++;
         NSLog(@"numberOfFeatures: %i", numberOfFeatures);
         strongClassifier = [self strongClassifierWithFeatures:numberOfFeatures
andTrainSet:trainSet];

         strongClassifier.threshold = 1.1;


         while (!(curentDetactionRate >= detectionRate && curentFalsePossitiveRate
<= falsePositiveRate) &&
                strongClassifier.threshold > 0.15) {
```

```
            strongClassifier.threshold -= 0.1;

            int nOfHits = 0;
            for (RawImage *image in trainSet.eyes) {
               nOfHits += [strongClassifier evaluate:image];
            }
            curentDetactionRate = ((float)nOfHits) / trainSet.eyes.count;

            int nOfFalsePositives = 0;
            for (ImageToTrain *image in trainSet.nonEyes) {
               if ([strongClassifier evaluate:image]) {
                  nOfFalsePositives ++;
                  [newNonEyes addObject:image];
               }
            }
            curentFalsePossitiveRate = ((float)nOfFalsePositives) /
trainSet.nonEyes.count;

            NSLog(@"detectionRate: %f - falsePositiveRate: %f - threshold:%f   ---  %f -
%f", curentDetactionRate, curentFalsePossitiveRate, strongClassifier.threshold,
falsePositiveRate, detectionRate);
         }
      }

      [cascadedClassifier.strongClassifiers addObject:strongClassifier];
      if (cascadedClassifier.strongClassifiers.count == 1) {
         cascadedClassifier.detectionRate = curentDetactionRate;
         cascadedClassifier.falsePositiveRate = curentFalsePossitiveRate;
      } else {
         cascadedClassifier.detectionRate *= curentDetactionRate;
         cascadedClassifier.falsePositiveRate *= curentFalsePossitiveRate;
      }
      trainSet.nonEyes = newNonEyes;

      NSLog(@"numberOfFeatures: %i - level: %i - falsePositiveRate:%f -
detectionRate: %f", strongClassifier.weekClassifiers.count,
cascadedClassifier.strongClassifiers.count, cascadedClassifier.falsePositiveRate,
cascadedClassifier.detectionRate);
      NSLog(@"\n\n");
   }

   return [cascadedClassifier autorelease];
}

- (StrongClassifier *)strongClassifierWithFeatures:(int)numberOfFeatures
andTrainSet:(TrainSet *)trainSet {
```

```
    StrongClassifier *strongClassifier = [[StrongClassifier alloc] init];
    NSMutableArray *features = [NSMutableArray
arrayWithArray:[WeekClassifierDataSource allWeekClassifiers]];

    double **weights = malloc(numberOfFeatures * sizeof(void *));
    for (int t = 0; t < numberOfFeatures; t++) {
        weights[t] = malloc(trainSet.totalSetLenght * sizeof(double));
    }

    double positiveWeight = 1.0 / trainSet.eyes.count;
    double negaticeWeight = 1.0 / trainSet.nonEyes.count;
    for (int i = 0; i < trainSet.images.count; i++) {
        ImageToTrain *image = [trainSet.images objectAtIndex:i];
        weights[0][i] = image.hasAnEye ? positiveWeight : negaticeWeight;
    }

    for (int t = 0; t < numberOfFeatures; t++) {
        double weightAdded = 0;
        for (int i = 0; i < trainSet.images.count; i++) {
            weightAdded += weights[t][i];
        }
        if (weightAdded == 0.0) {
            weightAdded = 1.0;
        }
        for (int i = 0; i < trainSet.images.count; i++) {
            weights[t][i] = weights[t][i] / weightAdded;
            ImageToTrain *image = [trainSet.images objectAtIndex:i];

            for (WeekClassifier *feature in features) {
                [feature trainWithImage:image];
            }
        }

        WeekClassifier *featureWithLowestError = nil;
        double lowestError = MAXFLOAT;
        for (WeekClassifier *feature in features) {
            feature.error = 0;
            for (int i = 0; i < trainSet.images.count; i++) {
                ImageToTrain *image = [trainSet.images objectAtIndex:i];
                feature.error += weights[t][i] * ABS((image.hasAnEye - [feature
evaluate:image]));
            }

            if (feature.error < lowestError) {
                [featureWithLowestError release];
                featureWithLowestError = [feature retain];
```

```
            lowestError = feature.error;
        }
      }

      if (t < numberOfFeatures -1) {
        for (int i = 0; i < trainSet.images.count; i++) {
          ImageToTrain *image = [trainSet.images objectAtIndex:i];
          int hit = image.hasAnEye == [featureWithLowestError evaluate:image];
          double bt = lowestError / (1 - lowestError);
          weights[t+1][i] = weights[t][i] * pow(bt, hit);
        }
      }

      [features removeObject:featureWithLowestError];//
      [strongClassifier.weekClassifiers addObject:featureWithLowestError];
      strongClassifier.linear += [featureWithLowestError alfaT];
    }

    return [strongClassifier autorelease];
}

- (StrongClassifier *)strongClassifierWithFeatures:(int)numberOfFeatures {
    TrainSet *trainSet = [TrainSetDataSource trainSet];
    return [self strongClassifierWithFeatures:numberOfFeatures
andTrainSet:trainSet];
}

@end
```

Classifier

```
#import "Classifier.h"

@interface Classifier ()
@property (nonatomic, assign) float trainWindow;

@end

@implementation Classifier

@synthesize currentScale = _currentScale;
@synthesize trainWindow = _trainWindow;

- (id)init {
    self = [super init];
    if (self) {
```

```
        self.currentScale = 1;
        _trainWindow = 30.0;
    }

    return self;
}

- (BOOL)evaluate:(RawImage *)image withWindow:(int)window {
    self.currentScale = window / _trainWindow;
    return [self evaluate:image];
}

- (BOOL)evaluate:(RawImage *)image {
    return NO;
}

@end
```

CascadeClassifier

```
#import "CascadedClassifier.h"
#import "StrongClassifier.h"

@implementation CascadedClassifier

@synthesize strongClassifiers = _strongClassifiers;
@synthesize falsePositiveRate = _falsePositiveRate;
@synthesize detectionRate = _detectionRate;


- (id)init {
    self = [super init];

    if (self) {
        _strongClassifiers = [[NSMutableArray alloc] init];
        _falsePositiveRate = 1.0;
        _detectionRate = 1.0;
    }

    return self;
}

- (void)dealloc {
    [_strongClassifiers release];
    [super dealloc];
```

```
}

- (BOOL)evaluate:(RawImage *)image {
   for (StrongClassifier *strongClassifier in _strongClassifiers) {
      strongClassifier.currentScale = self.currentScale;
      if (![strongClassifier evaluate:image]) {
         return NO;
      }
   }

   return YES;
}

- (NSString *)description {
   return [_strongClassifiers description];
}

@end
```

StrongClassifier

```
#import "StrongClassifier.h"
#import "WeekClassifier.h"
#import "SBJson.h"

@interface StrongClassifier ()

@end

@implementation StrongClassifier
@synthesize weekClassifiers = _weekClassifiers;
@synthesize linear = _linear;
@synthesize threshold = _threshold;

- (id)init {
   self = [super init];

   if (self) {
      _weekClassifiers = [[NSMutableArray alloc] init];
      _threshold = 0.5;
   }

   return self;
}

- (void)dealloc {
```

```
    [_weekClassifiers release];

    [super dealloc];
}

- (BOOL)evaluate:(RawImage *)image {
    double evaluation = 0;
    for (WeekClassifier *weekClassifier in _weekClassifiers) {
        weekClassifier.currentScale = self.currentScale;
        evaluation += [weekClassifier alfaT] * [weekClassifier evaluate:image];
    }

    BOOL isPositive = evaluation >= _threshold * _linear;
    return isPositive;
}

+ (id)strongClassifierFromJSON:(NSString *)JSON {
    StrongClassifier *strongClassifier = [[StrongClassifier alloc] init];

    SBJsonParser *jsonParser = [[SBJsonParser alloc] init];
    NSDictionary *strongClassifierAsDic = [jsonParser objectWithString:JSON];
    [jsonParser release];

    strongClassifier.threshold = [[strongClassifierAsDic objectForKey:@"threshold"]
floatValue];
    NSArray *weekClassifiers = [strongClassifierAsDic
objectForKey:@"weekClassifiers"];
    for (NSDictionary *dic in weekClassifiers) {
        WeekClassifier *weekClassifier = [WeekClassifier
weekClassifierFromJSONDic:dic];
        [strongClassifier.weekClassifiers addObject:weekClassifier];

        strongClassifier.linear += [weekClassifier alfaT];
    }

    return [strongClassifier autorelease];
}

- (NSString *)JSON {
    NSMutableString *JSON = [NSMutableString string];

    [JSON appendString:@"{"];
    [JSON appendFormat:@"\"threshold\":\"%f\"", _threshold];

    [JSON appendFormat:@"\"weekClassifiers\":"];
    [JSON appendString:@"["];
```

```
    for (WeekClassifier *weekClassifier in _weekClassifiers) {
        [JSON appendString:[weekClassifier JSON]];
        if (weekClassifier != _weekClassifiers.lastObject) {
            [JSON appendString:@","];
        }
    }
    [JSON appendString:@"]"];
    [JSON appendString:@"}"];

    return JSON;
}

- (NSString *)description {
    return [self JSON];
}

@end
```

WeakClassifier

```
#import "WeekClassifier.h"

@implementation WeekClassifier

@synthesize numberOfX = _numberOfX;
@synthesize numberOfY = _numberOfY;
@synthesize xCoordinates = _xCoordinates;
@synthesize yCoordinates = _yCoordinates;
@synthesize avaragePositive = _avaragePositive;
@synthesize avarageNegative = _avarageNegative;
@synthesize error = _error;


- (id)init {
    self = [super init];
    if (self) {
    }

    return self;
}

- (void)trainWithImage:(ImageToTrain *)image {
    int diffrence = [self diffrenceForImage:image];
    if (image.hasAnEye) {
        if (self.avaragePositive == 0.0) {
            self.avaragePositive = diffrence;
```

```
        } else {
            self.avaragePositive += diffrence;
            self.avaragePositive /= 2;
        }
    } else {
        if (self.avarageNegative == 0.0) {
            self.avarageNegative = diffrence;
        } else {
            self.avarageNegative += diffrence;
            self.avarageNegative /= 2;
        }
    }
}

- (BOOL)evaluate:(RawImage *)image {
    int diffrence = [self diffrenceForImage:image];

    return ABS(self.avaragePositive - diffrence) < ABS(self.avarageNegative -
diffrence);
}

- (void)dealloc {
    free(_xCoordinates);
    free(_yCoordinates);

    [super dealloc];
}

- (NSString *)JSON {
    NSMutableString *JSON = [NSMutableString string];
    [JSON appendFormat:@"{\"featureName\":\"%@\",", self.class];
    [JSON appendFormat:@"\"avaragePositive\":%f,", _avaragePositive];
    [JSON appendFormat:@"\"avarageNegative\":%f,", _avarageNegative];
    [JSON appendFormat:@"\"error\":%f,", _error];

    [JSON appendString:@"\"xCoordinates\":["];
    for (int index = 0; index < self.numberOfX; index++) {
        if (index != 0) {
            [JSON appendString:@","];
        }
        [JSON appendFormat:@"%i", self.xCoordinates[index]];
    }
    [JSON appendString:@"],"];

    [JSON appendString:@"\"yCoordinates\":["];
    for (int index = 0; index < self.numberOfY; index++) {
```

```
        if (index != 0) {
            [JSON appendString:@","];
        }
        [JSON appendFormat:@"%i", self.yCoordinates[index]];
    }
    [JSON appendString:@"]}"];

    return JSON;
}

- (NSString *)description {
    return  [self  JSON];
}

- (int)diffrenceForImage:(RawImage *)image {
    return 999999999;
}

- (double)alfaT {
    double bt = _error / (1.0 - _error);

    double alfaT = log(1.0/bt);

    return alfaT;
}

+ (id)weekClassifierFromJSONDic:(NSDictionary *)dic {
    WeekClassifier *weekClassifier = [[NSClassFromString([dic
objectForKey:@"featureName"]) alloc] init];

    weekClassifier.avaragePositive = [[dic objectForKey:@"avaragePositive"]
doubleValue];
    weekClassifier.avarageNegative = [[dic objectForKey:@"avarageNegative"]
doubleValue];
    weekClassifier.error = [[dic objectForKey:@"error"] doubleValue];

    NSArray *xCoordinates = [dic objectForKey:@"xCoordinates"];
    for (int x = 0; x < weekClassifier.numberOfX; x++) {
        weekClassifier.xCoordinates[x] = [[xCoordinates objectAtIndex:x] intValue];
    }

    NSArray *yCoordinates = [dic objectForKey:@"yCoordinates"];
    for (int y = 0; y < weekClassifier.numberOfY; y++) {
        weekClassifier.yCoordinates[y] = [[yCoordinates objectAtIndex:y] intValue];
    }
```

```
    return [weekClassifier autorelease];
}

@end
```

TrainSetDataSource

```
#import "TrainSetDataSource.h"
#import "ImageToTrain.h"

@implementation TrainSetDataSource

+ (TrainSet *)trainSet {
    TrainSet *trainSet = [[TrainSet alloc] init];

    int numberOfEyes = 99999999;
    int numberOfNonEyes = 40;

    while (trainSet.eyes.count < numberOfEyes) {
        NSString *imageName = [NSString stringWithFormat:@"eye_%i.png",
trainSet.eyes.count];
        UIImage *image = [UIImage imageNamed:imageName];
        if (!image) {
            break;
        }
        ImageToTrain *imageToTrain = [[ImageToTrain alloc] initWithImage:image
andHasAnEye:YES];
        [trainSet.eyes addObject:imageToTrain];
        [trainSet.images addObject:imageToTrain];
        [imageToTrain release];
    }

    NSLog(@"number of eyes: %i", trainSet.eyes.count);

    while (trainSet.nonEyes.count < numberOfNonEyes) {
        NSString *imageName = [NSString stringWithFormat:@"non_eye_%i.png",
trainSet.nonEyes.count];
        UIImage *image = [UIImage imageNamed:imageName];
        if (!image) {
            break;
        }
        ImageToTrain *imageToTrain = [[ImageToTrain alloc] initWithImage:image
andHasAnEye:NO];
        [trainSet.nonEyes addObject:imageToTrain];
        [trainSet.images addObject:imageToTrain];
        [imageToTrain release];
```

```
    }
    NSLog(@"number of nonEyes: %i", trainSet.nonEyes.count);

    return [trainSet autorelease];
}

@end
```

WeakClassifierDataSource

```
#import "WeekClassifierDataSource.h"

#import "ConcentricoTwoRectanglesWeekClassifier.h"
#import "HorizontalThreeRetangleWeekClassifier.h"
#import "HorizontalTwoRetangleWeekClassifier.h"
#import "VerticalThreeRetangleWeekClassifier.h"
#import "VerticalTwoRetangleWeekClassifier.h"
#import "FourRetangleWeekClassifier.h"

static NSMutableArray *allWeekClassifiers;

@interface WeekClassifierDataSource ()

+ (NSArray *)allHorizontalTwoRetangleWeekClassifiers;
+ (NSArray *)allHorizontalThreeRetangleWeekClassifiers;

+ (NSArray *)allVerticalTwoRetangleWeekClassifiers;
+ (NSArray *)allVerticalThreeRetangleWeekClassifiers;

+ (NSArray *)allConcentricoTwoRectanglesWeekClassifiers;
+ (NSArray *)allFourRetangleWeekClassifiers;

@end

@implementation WeekClassifierDataSource

+ (NSArray *)allHorizontalTwoRetangleWeekClassifiers {
    NSMutableSet *features = [NSMutableSet set];

    for (int initialX = 0; initialX + HALF_MIN_FEATURE_SIZE < TRAIN_WINDOW_SIZE;
initialX++) {
        for (int initialY = 0; initialY + MIN_FEATURE_SIZE < TRAIN_WINDOW_SIZE;
initialY++) {
            for (int finalX = initialX + MIN_FEATURE_SIZE; finalX < TRAIN_WINDOW_SIZE;
finalX++) {
```

```
            for (int finalY = initialY + MIN_FEATURE_SIZE; finalY <
TRAIN_WINDOW_SIZE; finalY += 2) {
                int middleY = initialY + (finalY - initialY)/2;
                HorizontalTwoRetangleWeekClassifier *feature =
[[HorizontalTwoRetangleWeekClassifier alloc] initWithInitialX:initialX finalX:finalX
initialY:initialY middleY:middleY finalY:finalY];
                [features addObject:feature];
                [feature release];
            }
        }
    }
  }

  return [features allObjects];
}

+ (NSArray *)allVerticalTwoRetangleWeekClassifiers {
  NSMutableArray *features = [NSMutableArray array];

  for (int initialX = 0; initialX + MIN_FEATURE_SIZE < TRAIN_WINDOW_SIZE;
initialX++) {
    for (int initialY = 0; initialY + HALF_MIN_FEATURE_SIZE <
TRAIN_WINDOW_SIZE; initialY++) {
      for (int finalX = initialX + MIN_FEATURE_SIZE; finalX < TRAIN_WINDOW_SIZE;
finalX++) {
            for (int finalY = initialY + MIN_FEATURE_SIZE; finalY <
TRAIN_WINDOW_SIZE; finalY += 2) {
                int middleX = initialX + (finalX - initialX)/2;
                VerticalTwoRetangleWeekClassifier *feature =
[[VerticalTwoRetangleWeekClassifier alloc] initWithInitialX:initialX
middleX:middleX finalX:finalX initialY:initialY finalY:finalY];
                [features addObject:feature];
                [feature release];
            }
        }
    }
  }

  return features;
}

+ (NSArray *)allHorizontalThreeRetangleWeekClassifiers {
  NSMutableArray *features = [NSMutableArray array];
  int minFeatureHeight = 2*MIN_FEATURE_SIZE;
```

```
   for (int initialX = 0; initialX + MIN_FEATURE_SIZE < TRAIN_WINDOW_SIZE;
initialX++) {
      for (int initialY = 0; initialY + minFeatureHeight < TRAIN_WINDOW_SIZE;
initialY++) {

         for (int finalX = initialX + MIN_FEATURE_SIZE; finalX < TRAIN_WINDOW_SIZE;
finalX++) {
            for (int finalY = initialY + minFeatureHeight; finalY < TRAIN_WINDOW_SIZE;
finalY += 3) {

               int firstMiddleY = initialY + HALF_MIN_FEATURE_SIZE;
               for (;firstMiddleY + MIN_FEATURE_SIZE < finalY; firstMiddleY++) {

                  int secondMiddleY = firstMiddleY + HALF_MIN_FEATURE_SIZE;
                  for (;secondMiddleY + HALF_MIN_FEATURE_SIZE < finalY;
secondMiddleY++) {
                     HorizontalThreeRetangleWeekClassifier *feature =
[[HorizontalThreeRetangleWeekClassifier alloc] initWithInitialX:initialX
finalX:finalX initialY:initialY firstMiddleY:firstMiddleY
secondMiddleY:secondMiddleY finalY:finalY];
                     [features addObject:feature];
                     [feature release];
                  }
               }
            }
         }
      }
   }

   return features;
}

+ (NSArray *)allVerticalThreeRetangleWeekClassifiers {
   NSMutableArray *features = [NSMutableArray array];
   int minFeatureWidth = 2*MIN_FEATURE_SIZE;

   for (int initialX = 0; initialX + minFeatureWidth < TRAIN_WINDOW_SIZE;
initialX++) {
      for (int initialY = 0; initialY + MIN_FEATURE_SIZE < TRAIN_WINDOW_SIZE;
initialY++) {

         for (int finalY = initialY + MIN_FEATURE_SIZE; finalY < TRAIN_WINDOW_SIZE;
finalY++) {
            for (int finalX = initialX + minFeatureWidth; finalX < TRAIN_WINDOW_SIZE;
finalX += 3) {
```

```
            int firstMiddleX = initialX + HALF_MIN_FEATURE_SIZE;
            for (;firstMiddleX + MIN_FEATURE_SIZE < finalX; firstMiddleX++) {

              int secondMiddleX = firstMiddleX + HALF_MIN_FEATURE_SIZE;
              for (;secondMiddleX + HALF_MIN_FEATURE_SIZE < finalX;
secondMiddleX++) {
                VerticalThreeRetangleWeekClassifier *feature =
[[VerticalThreeRetangleWeekClassifier alloc] initWithInitialX:initialX
firstMiddleX:firstMiddleX secondMiddleX:secondMiddleX finalX:finalX
initialY:initialY finalY:finalY];
                [features addObject:feature];
                [feature release];
              }
            }
          }
        }
      }
    }

    return features;
}

+ (NSArray *)allConcentricoTwoRectanglesWeekClassifiers {
    NSMutableArray *features = [NSMutableArray array];

    int minMargin = HALF_MIN_FEATURE_SIZE;
    int minFeatureSize = 1.5 * MIN_FEATURE_SIZE;

    for (int initialX = 0; initialX + minFeatureSize < TRAIN_WINDOW_SIZE; initialX++)
{
      for (int initialY = 0; initialY + minFeatureSize < TRAIN_WINDOW_SIZE;
initialY++) {

        for (int finalX = initialX + MIN_FEATURE_SIZE; finalX < TRAIN_WINDOW_SIZE;
finalX += 2) {
          for (int finalY = initialY + MIN_FEATURE_SIZE; finalY <
TRAIN_WINDOW_SIZE; finalY += 2) {

            for (int horizontalMargin = minMargin; 2*horizontalMargin +
HALF_MIN_FEATURE_SIZE < finalX -initialX; horizontalMargin++) {
              for (int verticalMargin = minMargin; 2*verticalMargin +
HALF_MIN_FEATURE_SIZE < finalY -initialY; verticalMargin++) {
                ConcentricoTwoRectanglesWeekClassifier *feature =
[[ConcentricoTwoRectanglesWeekClassifier alloc] initWithInitialX:initialX
finalX:finalX initialY:initialY finalY:finalY horizontalMargin:horizontalMargin
verticalMargin:verticalMargin];
```

```
                    [features addObject:feature];
                    [feature release];
                }
            }
        }
    }
}

return features;
}

+ (NSArray *)allFourRetangleWeekClassifiers {
NSMutableArray *features = [NSMutableArray array];

for (int initialX = 0; initialX + MIN_FEATURE_SIZE < TRAIN_WINDOW_SIZE;
initialX++) {
    for (int initialY = 0; initialY + MIN_FEATURE_SIZE < TRAIN_WINDOW_SIZE;
initialY++) {
        for (int finalX = initialX + MIN_FEATURE_SIZE; finalX < TRAIN_WINDOW_SIZE;
finalX += 2) {
            for (int finalY = initialY + MIN_FEATURE_SIZE; finalY <
TRAIN_WINDOW_SIZE; finalY += 2) {

                int middleX = (finalX - initialX)/2;
                int middleY = (finalY - initialY)/2;
                FourRetangleWeekClassifier *feature = [[FourRetangleWeekClassifier
alloc] initWithInitialX:initialX middleX:middleX finalX:finalX initialY:initialY
middleY:middleY finalY:finalY];
                [features addObject:feature];
                [feature release];
            }
        }
    }
}

return features;
}

+ (NSArray *)allWeekClassifiers {
if (!allWeekClassifiers) {
    NSArray *allHorizontalTwoRetangleWeekClassifiers =
[WeekClassifierDataSource allHorizontalTwoRetangleWeekClassifiers];
    NSLog(@"allHorizontalTwoRetangleWeekClassifiers: %i",
allHorizontalTwoRetangleWeekClassifiers.count);
```

```
    NSArray *allVerticalTwoRetangleWeekClassifiers = [WeekClassifierDataSource
allVerticalTwoRetangleWeekClassifiers];
    NSLog(@"allVerticalTwoRetangleWeekClassifiers: %i",
allVerticalTwoRetangleWeekClassifiers.count);

    NSArray *allHorizontalThreeRetangleWeekClassifiers =
[WeekClassifierDataSource allHorizontalThreeRetangleWeekClassifiers];
    NSLog(@"allHorizontalThreeRetangleWeekClassifiers: %i",
allHorizontalThreeRetangleWeekClassifiers.count);

    NSArray *allVerticalThreeRetangleWeekClassifiers =
[WeekClassifierDataSource allVerticalThreeRetangleWeekClassifiers];
    NSLog(@"allVerticalThreeRetangleWeekClassifiers: %i",
allVerticalThreeRetangleWeekClassifiers.count);

    NSArray *allConcentricoTwoRectanglesWeekClassifiers =
[WeekClassifierDataSource allConcentricoTwoRectanglesWeekClassifiers];
    NSLog(@"allVerticalThreeRetangleWeekClassifiers: %i",
allConcentricoTwoRectanglesWeekClassifiers.count);

    NSArray *allFourRetangleWeekClassifiers = [WeekClassifierDataSource
allFourRetangleWeekClassifiers];
    NSLog(@"allFourRetangleFeatures: %i",
allFourRetangleWeekClassifiers.count);

    allWeekClassifiers = [NSMutableArray array];
    [allWeekClassifiers
addObjectsFromArray:allVerticalTwoRetangleWeekClassifiers];
    [allWeekClassifiers
addObjectsFromArray:allHorizontalTwoRetangleWeekClassifiers];
    [allWeekClassifiers
addObjectsFromArray:allHorizontalThreeRetangleWeekClassifiers];
    [allWeekClassifiers
addObjectsFromArray:allVerticalThreeRetangleWeekClassifiers];
    [allWeekClassifiers
addObjectsFromArray:allConcentricoTwoRectanglesWeekClassifiers];
    [allWeekClassifiers addObjectsFromArray:allFourRetangleWeekClassifiers];

    NSLog(@"allWeekClassifiers: %i", allWeekClassifiers.count);
  }

  return allWeekClassifiers;
}

@end
```

ConcentricoTwoRectanglesWeekClassifier

```
#import "ConcentricoTwoRectanglesWeekClassifier.h"

@implementation ConcentricoTwoRectanglesWeekClassifier

- (id)init {
   self = [super init];
   if (self) {
      self.numberOfX = 4;
      self.numberOfY = 4;
      self.xCoordinates = malloc(self.numberOfX * sizeof(int));
      self.yCoordinates = malloc(self.numberOfY * sizeof(int));
   }

   return self;
}

- (id)initWithInitialX:(int)initialX finalX:(int)finalX initialY:(int)initialY
finalY:(int)finalY horizontalMargin:(int)horizontalMargin
verticalMargin:(int)verticalMargin {
   self = [self init];
   if (self) {
      self.xCoordinates[0] = initialX;
      self.xCoordinates[1] = finalX;
      self.xCoordinates[2] = initialX + horizontalMargin;
      self.xCoordinates[3] = finalX - horizontalMargin;

      self.yCoordinates[0] = initialY;
      self.yCoordinates[1] = finalY;
      self.yCoordinates[2] = initialY + verticalMargin;
      self.yCoordinates[3] = finalY - verticalMargin;
   }

   return self;
}

- (int)diffrenceForImage:(RawImage *)image {
   int initialX = self.currentScale * self.xCoordinates[0];
   int firstMiddleX = self.currentScale * self.yCoordinates[1];
   int secondMiddleX = self.currentScale * self.yCoordinates[2];
   int finalX = self.currentScale * self.xCoordinates[3];

   int initialY = self.currentScale * self.yCoordinates[0];
```

```
    int middleYOne = self.currentScale * self.yCoordinates[1];
    int middleYTwo = self.currentScale * self.yCoordinates[2];
    int finalY = self.currentScale * self.yCoordinates[3];

    int externRecValue = [image integralPixelAtX:firstMiddleX andY:middleYOne];
    externRecValue -= [image integralPixelAtX:firstMiddleX andY:initialY];
    externRecValue -= [image integralPixelAtX:initialX andY:middleYOne];
    externRecValue += [image integralPixelAtX:initialX andY:initialY];

    int internRecValue = [image integralPixelAtX:finalX andY:finalY];
    internRecValue -= [image integralPixelAtX:finalX andY:middleYTwo];
    internRecValue -= [image integralPixelAtX:secondMiddleX andY:finalY];
    internRecValue += [image integralPixelAtX:secondMiddleX andY:middleYTwo];

    return externRecValue - internRecValue;
}

@end
```

FourRetangleWeekClassifier

```
#import "FourRetangleWeekClassifier.h"

@implementation FourRetangleWeekClassifier

- (id)init {
    self = [super init];

    if (self) {
        self.numberOfX = 3;
        self.numberOfY = 3;
        self.xCoordinates = malloc(self.numberOfX * sizeof(int));
        self.yCoordinates = malloc(self.numberOfY * sizeof(int));
    }

    return self;
}

- (id)initWithInitialX:(int)initialX middleX:(int)middleX finalX:(int)finalX
initialY:(int)initialY middleY:(int)middleY finalY:(int)finalY {
    self = [self init];
    if (self) {
        self.xCoordinates[0] = initialX;
        self.xCoordinates[1] = middleY;
        self.xCoordinates[2] = finalX;
```

```
        self.yCoordinates[0] = initialY;
        self.yCoordinates[1] = middleY;
        self.yCoordinates[2] = finalY;
    }

    return self;
}

- (int)diffrenceForImage:(RawImage *)image {
    int initialX = self.currentScale * self.xCoordinates[0];
    int middleX = self.currentScale * self.yCoordinates[1];
    int finalX = self.currentScale * self.xCoordinates[2];

    int initialY = self.currentScale * self.yCoordinates[0];
    int middleY = self.currentScale * self.yCoordinates[1];
    int finalY = self.currentScale * self.yCoordinates[2];


    int firstRecValue = [image integralPixelAtX:middleX andY:middleY];
    firstRecValue -= [image integralPixelAtX:middleX andY:initialY];
    firstRecValue -= [image integralPixelAtX:initialX andY:middleY];
    firstRecValue += [image integralPixelAtX:initialX andY:initialY];

    int secondRecValue = [image integralPixelAtX:finalX andY:middleY];
    secondRecValue -= [image integralPixelAtX:finalX andY:initialY];
    secondRecValue -= [image integralPixelAtX:middleX andY:middleY];
    secondRecValue += [image integralPixelAtX:middleX andY:initialY];

    int thirdRecValue = [image integralPixelAtX:middleX andY:finalY];
    thirdRecValue -= [image integralPixelAtX:middleX andY:middleY];
    thirdRecValue -= [image integralPixelAtX:initialX andY:finalY];
    thirdRecValue += [image integralPixelAtX:initialX andY:middleY];

    int fourthRecValue = [image integralPixelAtX:finalX andY:finalY];
    fourthRecValue -= [image integralPixelAtX:finalX andY:middleY];
    fourthRecValue -= [image integralPixelAtX:middleX andY:finalY];
    fourthRecValue += [image integralPixelAtX:middleX andY:middleY];

    return firstRecValue + fourthRecValue - secondRecValue - thirdRecValue;
}

@end
```

HorizontalThreeRetangleWeekClassifier

```
#import "HorizontalThreeRetangleWeekClassifier.h"

@implementation HorizontalThreeRetangleWeekClassifier

- (id)init {
   self = [super init];
   if (self) {
      self.numberOfX = 2;
      self.numberOfY = 4;
      self.xCoordinates = malloc(self.numberOfX * sizeof(int));
      self.yCoordinates = malloc(self.numberOfY * sizeof(int));
   }

   return self;
}

- (id)initWithInitialX:(int)initialX finalX:(int)finalX initialY:(int)initialY
firstMiddleY:(int)firstMiddleY secondMiddleY:(int)secondMiddleY finalY:(int)finalY
{
   self = [self init];
   if (self) {
      self.xCoordinates[0] = initialX;
      self.xCoordinates[1] = finalX;

      self.yCoordinates[0] = initialY;
      self.yCoordinates[1] = firstMiddleY;
      self.yCoordinates[2] = secondMiddleY;
      self.yCoordinates[3] = finalY;
   }

   return self;
}

- (int)diffrenceForImage:(RawImage *)image {
   int initialX = self.currentScale * self.xCoordinates[0];
   int finalX = self.currentScale * self.xCoordinates[1];

   int initialY = self.currentScale * self.yCoordinates[0];
   int firstMiddleY = self.currentScale * self.yCoordinates[1];
   int secondMiddleY = self.currentScale * self.yCoordinates[2];
   int finalY = self.currentScale * self.yCoordinates[3];

   int firstRecValue = [image integralPixelAtX:finalX andY:firstMiddleY];
   firstRecValue -= [image integralPixelAtX:finalX andY:initialY];
   firstRecValue -= [image integralPixelAtX:initialX andY:firstMiddleY];
   firstRecValue += [image integralPixelAtX:initialX andY:initialY];
```

```
    int secondRecValue = [image integralPixelAtX:finalX andY:secondMiddleY];
    secondRecValue -= [image integralPixelAtX:finalX andY:firstMiddleY];
    secondRecValue -= [image integralPixelAtX:initialX andY:secondMiddleY];
    secondRecValue += [image integralPixelAtX:initialX andY:firstMiddleY];

    int thirdRecValue = [image integralPixelAtX:finalX andY:finalY];
    thirdRecValue -= [image integralPixelAtX:finalX andY:secondMiddleY];
    thirdRecValue -= [image integralPixelAtX:initialX andY:finalY];
    thirdRecValue += [image integralPixelAtX:initialX andY:secondMiddleY];

    return firstRecValue + thirdRecValue - secondRecValue;
}

@end
```

HorizontalTwoRetangleWeekClassifier

```
#import "HorizontalTwoRetangleWeekClassifier.h"

@interface HorizontalTwoRetangleWeekClassifier ()

- (int)diffrenceForImage:(RawImage *)image;

@end

@implementation HorizontalTwoRetangleWeekClassifier

- (id)init {
    self = [super init];
    if (self) {
        self.numberOfX = 2;
        self.numberOfY = 3;
        self.xCoordinates = malloc(self.numberOfX * sizeof(int));
        self.yCoordinates = malloc(self.numberOfY * sizeof(int));
    }

    return self;
}

- (id)initWithInitialX:(int)initialX finalX:(int)finalX initialY:(int)initialY
middleY:(int)middleY finalY:(int)finalY {
    self = [self init];
    if (self) {
        self.xCoordinates[0] = initialX;
```

```
        self.xCoordinates[1] = finalX;

        self.yCoordinates[0] = initialY;
        self.yCoordinates[1] = middleY;
        self.yCoordinates[2] = finalY;
    }

    return self;
}

- (int)diffrenceForImage:(RawImage *)image {
    int initialX = self.currentScale * self.xCoordinates[0];
    int finalX = self.currentScale * self.xCoordinates[1];

    int initialY = self.currentScale * self.yCoordinates[0];
    int middleY = self.currentScale * self.yCoordinates[1];
    int finalY = self.currentScale * self.yCoordinates[2];

    int firstRecValue = [image integralPixelAtX:finalX andY:middleY];
    firstRecValue -= [image integralPixelAtX:finalX andY:initialY];
    firstRecValue -= [image integralPixelAtX:initialX andY:middleY];
    firstRecValue += [image integralPixelAtX:initialX andY:initialY];

    int secondRecValue = [image integralPixelAtX:finalX andY:finalY];
    secondRecValue -= [image integralPixelAtX:finalX andY:middleY];
    secondRecValue -= [image integralPixelAtX:initialX andY:finalY];
    secondRecValue += [image integralPixelAtX:initialX andY:middleY];

    return secondRecValue - firstRecValue;
}

@end
```

VerticalThreeRetangleWeekClassifier

```
#import "VerticalThreeRetangleWeekClassifier.h"

@implementation VerticalThreeRetangleWeekClassifier

- (id)init {
    self = [super init];
    if (self) {
        self.numberOfX = 4;
        self.numberOfY = 2;
        self.xCoordinates = malloc(self.numberOfX * sizeof(int));
        self.yCoordinates = malloc(self.numberOfY * sizeof(int));
```

```
    }

    return self;
}

- (id)initWithInitialX:(int)initialX firstMiddleX:(int)firstMiddleX
secondMiddleX:(int)secondMiddleX  finalX:(int)finalX initialY:(int)initialY
finalY:(int)finalY {
    self = [self init];
    if (self) {
        self.xCoordinates[0] = initialX;
        self.xCoordinates[1] = firstMiddleX;
        self.xCoordinates[2] = secondMiddleX;
        self.xCoordinates[3] = finalX;

        self.yCoordinates[0] = initialY;
        self.yCoordinates[1] = finalY;
    }

    return self;
}

- (int)diffrenceForImage:(RawImage *)image {
    int initialX = self.currentScale * self.xCoordinates[0];
    int firstMiddleX = self.currentScale * self.xCoordinates[1];
    int secondMiddleX = self.currentScale * self.xCoordinates[2];
    int finalX = self.currentScale * self.xCoordinates[3];

    int initialY = self.currentScale * self.yCoordinates[0];
    int finalY = self.currentScale * self.yCoordinates[1];

    int firstRecValue = [image integralPixelAtX:firstMiddleX andY:finalY];
    firstRecValue -= [image integralPixelAtX:firstMiddleX andY:initialY];
    firstRecValue -= [image integralPixelAtX:initialX andY:finalY];
    firstRecValue += [image integralPixelAtX:initialX andY:initialY];

    int secondRecValue = [image integralPixelAtX:secondMiddleX
andY:self.yCoordinates[1]];
    secondRecValue -= [image integralPixelAtX:secondMiddleX andY:initialY];
    secondRecValue -= [image integralPixelAtX:firstMiddleX andY:finalY];
    secondRecValue += [image integralPixelAtX:firstMiddleX andY:initialY];

    int thirdRecValue = [image integralPixelAtX:finalX andY:finalY];
    thirdRecValue -= [image integralPixelAtX:finalX andY:initialY];
    thirdRecValue -= [image integralPixelAtX:secondMiddleX andY:finalY];
    thirdRecValue += [image integralPixelAtX:secondMiddleX andY:initialY];
```

```
    return firstRecValue + thirdRecValue - secondRecValue;
}

@end
```

VerticalTwoRetangleWeekClassifier

```
#import "VerticalTwoRetangleWeekClassifier.h"

@implementation VerticalTwoRetangleWeekClassifier

- (id)init {
    self = [super init];
    if (self) {
        self.numberOfX = 3;
        self.numberOfY = 2;
        self.xCoordinates = malloc(self.numberOfX * sizeof(int));
        self.yCoordinates = malloc(self.numberOfY * sizeof(int));
    }

    return self;
}

- (id)initWithInitialX:(int)initialX middleX:(int)middleX finalX:(int)finalX
initialY:(int)initialY finalY:(int)finalY {
    self = [self init];
    if (self) {
        self.xCoordinates[0] = initialX;
        self.xCoordinates[1] = middleX;
        self.xCoordinates[2] = finalX;

        self.yCoordinates[0] = initialY;
        self.yCoordinates[1] = finalY;
    }

    return self;
}

- (int)diffrenceForImage:(RawImage *)image {
    int initialX = self.currentScale * self.xCoordinates[0];
    int middleX = self.currentScale * self.xCoordinates[1];
    int finalX = self.currentScale * self.xCoordinates[2];

    int initialY = self.currentScale * self.yCoordinates[0];
```

```
    int finalY = self.currentScale * self.yCoordinates[1];

    float firstRecValue = [image integralPixelAtX:middleX andY:finalY];
    firstRecValue -= [image integralPixelAtX:middleX andY:initialY];
    firstRecValue -= [image integralPixelAtX:initialX andY:finalY];
    firstRecValue += [image integralPixelAtX:initialX andY:initialY];

    float secondRecValue = [image integralPixelAtX:finalX andY:finalY];
    secondRecValue -= [image integralPixelAtX:finalX andY:initialY];
    secondRecValue -= [image integralPixelAtX:middleX andY:finalY];
    secondRecValue += [image integralPixelAtX:middleX andY:initialY];

    return secondRecValue - firstRecValue;
}

@end
```

ImageToTrain

```
#import "ImageToTrain.h"

@implementation ImageToTrain

@synthesize hasAnEye = _hasAnEye;
@synthesize integralImagePixels = _integralImagePixels;

- (id)initWithImage:(UIImage *)pimage andHasAnEye:(BOOL)hasAnEye {
    self = [super initWithUIImage:pimage];
    if (self) {
        self.hasAnEye = hasAnEye;
    }

    return self;
}

- (void)dealloc {
    free(_integralImagePixels);
    [super dealloc];
}

- (int *)integralImagePixels {
    if (!_integralImagePixels) {
        _integralImagePixels = malloc(self.width * self.height * sizeof(double));
    }
    return  _integralImagePixels;
}
```

```
@end
```

AdaBoostClassifier

```
#import "AdaBoostClassifier.h"
#import "UIImage+Resize.h"

static int imageIndex = 0;


@interface AdaBoostClassifier ()

@property (nonatomic, retain) Classifier *classifier;

@end

@implementation AdaBoostClassifier
@synthesize classifier = _classifier;

- (id)initWithClassifier:(Classifier *)classifier {
   self = [super init];
   if (self) {
      self.classifier = classifier;
   }

   return self;
}

- (void)saveImage:(UIImage *)image {
   NSData *imageData = UIImagePNGRepresentation(image);
   NSString *documentsDirectory =
[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES) objectAtIndex:0];
   NSString *fileName = [NSString stringWithFormat:@"%i.png", imageIndex];
   NSString *fullPathToFile = [documentsDirectory
stringByAppendingPathComponent:fileName];
   [imageData writeToFile:fullPathToFile atomically:YES];

   imageIndex++;
}

- (NSArray *)rectsWithEyeInRawImage:(RawImage *)rawImage {
   NSMutableArray *rectsWithEye = [NSMutableArray array];
   int maxWindow = 80;
   for (int x = 0; x + maxWindow < rawImage.width; x+=2) {
      for (int y = 0; y + maxWindow < rawImage.height; y+=2) {
```

```
        RawImage *subImage = [[RawImage alloc]
initWithFatherRawImage:rawImage andSubImageOrigin:CGPointMake(x, y)];

        int currentWindow = 20;
        while (currentWindow <= maxWindow) {
          if ([_classifier evaluate:subImage withWindow:currentWindow]) {
            CGRect rectWithEye = CGRectMake(x, y, currentWindow,
currentWindow);
            [rectsWithEye addObject:[NSValue valueWithCGRect:rectWithEye]];

            NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
            UIImage *image = [rawImage.image croppedImage:rectWithEye];
            //image = [image resizedImage:CGSizeMake(30, 30)
interpolationQuality:kCGInterpolationLow];
            [self saveImage:image];
            [pool drain];
          }

          currentWindow += 4;
        }
        [subImage release];
      }
    }

    return rectsWithEye;
}




- (void)dealloc {
    self.classifier = nil;
    [super dealloc];
}

- (NSString *)description {
    return [NSString stringWithFormat:@"%@", _classifier];
}

@end
```