Cizenando Morello Bonfá

Um Sistema de Reconhecimento Facial em Vídeo Baseado em uma Implementação Multithread do Algoritmo TLD

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC-Rio.

Orientador: Profº. Raul Queiroz Feitosa

Rio de Janeiro
novembro de 2013

Cizenando Morello Bonfá

Um Sistema de Reconhecimento Facial em Vídeo Baseado em uma Implementação Multithread do Algoritmo TLD

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Engenharia Elétrica da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada

Prof. Raul Queiroz Feitosa
Orientador
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Alberto Barbosa Raposo
Departamento de Informática – PUC-Rio

Prof. Bruno Feijó
Departamento de Informática – PUC-Rio

Prof. Gilson A. O. Pedro da Costa
Departamento de Engenharia Elétrica – PUC-Rio

Prof. José Eugênio Leal
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 05 de novembro de 2013

Cizenando Morello Bonfá

Graduou-se em Engenharia Elétrica na Universidade Federal do Espírito Santo em 2010. Atualmente é Oficial Primeiro-Tenente Engenheiro Eletrônico da Marinha do Brasil onde trabalha no Programa de Desenvolvimento do Submarino Nuclear.

Ficha Catalográfica

Bonfá, Cizenando Morello

Um sistema de reconhecimento facial em vídeo baseado em uma implementação multithread do algoritmo TLD / Cizenando Morello Bonfá ; orientador: Raul Queiroz Feitosa– 2013.

102 f. : il. (color.) ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2013.
Inclui bibliografia

1. Engenharia elétrica – Teses. 2. Rastramento. 3. Reconhecimento facial. 4. Estimativa de pose. 5. Multiprocessamento. 6. Detecção de face. 7. Qualidade de Imagem. I. Feitosa, Raul Queiroz. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

Kathy, palavras não descrevem minha gratidão.

Agradecimentos

À Deus, pela vida.

Aos meus pais, Cizenando e Graça, por sempre estarem ao meu lado, apoiando minhas decisões por mais difíceis que fosse, para que hoje eu pudesse estar aqui.

Meu cunhado Hercílio pela ajuda na época das vacas magras.

Minha madrinha Ester, que sempre se orgulhou mim e hoje é um anjo que guia meus passos.

Meu orientador Raul, por sua paciência e disponibilidade de adaptar-se ao meu horário de trabalho.

Ao CNPq e a PUC-Rio, pela bolsa no período de mestrado.

Ao grande amor da minha vida, minha esposa Katharine, sem a qual esta dissertação dificilmente teria sido finalizada. Obrigado.

Resumo

Bonfá, Cizenando Morello; Feitosa, Raul Queiroz (Orientador) **Um Sistema de Reconhecimento Facial em Vídeo Baseado em uma Implementação Multithread do Algoritmo TLD.** Rio de Janeiro, 2013. 102p. Dissertação de Mestrado - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

A identificação facial em vídeo é uma aplicação de grande interesse na comunidade cientifica e na indústria de segurança, impulsionando a busca por técnicas mais robustas e eficientes. Atualmente, no âmbito de reconhecimento facial, as técnicas de identificação frontal são as com melhor taxa de acerto quando comparadas com outras técnicas não frontais. Esse trabalho tem como objetivo principal buscar métodos de avaliar imagens em vídeo em busca de pessoas (rostos), avaliando se a qualidade da imagem está dentro de uma faixa aceitável que permita um algoritmo de reconhecimento facial frontal identificar os indivíduos. Propõem-se maneiras de diminuir a carga de processamento para permitir a avaliação do máximo número de indivíduos numa imagem sem afetar o desempenho em tempo real. Isso é feito através de uma análise da maior parte das técnicas utilizadas nos últimos anos e do estado da arte, compilando toda a informação para ser aplicada em um projeto que utiliza os pontos fortes de cada uma e compense suas deficiências. O resultado é uma plataforma *multithread*. Para avaliação do desempenho foram realizados testes de carga computacional com o uso de um vídeo público disponibilizado na AVSS (*Advanced Video and Signal based Surveillance*). Os resultados mostram que a arquitetura promove um melhor uso dos recursos computacionais, permitindo um uso de uma gama maior de algoritmos em cada segmento que compõe a arquitetura, podendo ser selecionados segundo critérios de qualidade da imagem e ambiente onde o vídeo é capturado.

Palavras-chave

Rastreamento; Reconhecimento Facial; Estimativa de Pose; Multiprocessamento; Detecção de Face; Qualidade da Imagem.

Abstract

Bonfá, Cizenando Morello; Feitosa, Raul Queiroz (Advisor). **A Face Recognition System for Video Sequences Based on a Multithread Implementation of TLD.** Rio de Janeiro, 2013. 102p. MSc. Dissertation – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Face recognition in video is an application of great interest in the scientific community and in the surveillance industry, boosting the search for efficient and robust techniques. Nowadays, in the facial recognition field, the frontal identification techniques are those with the best hit ratio when compared with others non-frontal techniques. This work has as main objective seek for methods to evaluate images in video to look for people (faces), assessing if the image quality is in an acceptable range that allows a facial recognition algorithm to identify the individuals. It's proposed ways to decrease the processing load to allow a maximum number of individuals assessed in an image without affecting the real time performance. This is reached through analysis of most the techniques used in the last years and the state-of-the-art, compiling all information to be applied in a project that uses the strengths of each one and offset its shortcomings. The outcome is a multithread platform. Performance evaluation was performed through computational load tests by using public videos available in AVSS (Advanced Video and Signal based Surveillance). The outcomes show that the architecture makes a better use of the computational resources, allowing use of a wide range of algorithms in every segment of the architecture that can be selected according to quality image and video environment criteria.

Keywords

Tracking; Face Recognition; Pose Estimation; Multithreading; Face Detection; Image Quality.

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

AMM	Active Appearance Model
ASM	Active Shapel Model
BB	Bounding Box
dN	Distância dos seguimentos aos exemplos negativos
dP	Distância dos seguimentos aos exemplos positivos
KLT	Kanade Lucas Tomasi Tracker
LBP	Local Binary Pattern
LK	Lucas Kanade
NCC	Normalized Cross Correlation
nccN	NCC do seguimento sob o conjunto de exemplos negativos
nccP	NCC do seguimento sob o conjunto de exemplos positivos
NN	Nearest Neighbour
QI	Qualidade da Imagem
TLD	Tracking Learning Detection
WPA	Wavelet Packet Analysis

“Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas graças a Deus, não sou o que era antes”.
(Martin Luther King)

1
INTRODUÇÃO

A identificação facial em vídeo é uma aplicação de grande interesse na comunidade cientifica e na indústria de segurança. Esse interesse vem impulsionando a busca por técnicas mais robustas e eficientes computacionalmente. (Zhao et al., 2003).

Como uma das mais bem sucedidas aplicações de análise de imagem, a identificação facial tem recebido recentemente bastante atenção, especialmente nos últimos anos. Pelo menos duas razões explicam esta tendência: a primeira é a ampla gama de aplicações comerciais e de segurança, e a segunda é a disponibilidade de tecnologias viáveis depois de 30 anos de pesquisa (Lu, 2003). Apesar de os atuais sistemas de reconhecimento terem alcançado certo nível de maturidade, seu sucesso é limitado pelas condições impostas por muitas aplicações reais, como por exemplo, a dificuldade de reconhecimento de imagens de faces adquiridas em um ambiente externo com mudanças de iluminação e ou de pose. Isso se deve ao fato de a grande maioria dos identificadores faciais serem construídos para reconhecimento de faces frontais e com iluminação uniforme, uma vez que a maioria das bases de dados disponíveis conta somente com imagens nessa pose fotografadas num ambiente controlado (com iluminação corretiva).

Hoje, existem métodos biométricos de identificação pessoal mais seguros, como por exemplo, a análise de digitais e *scanner* de íris e retina, porém, é importante ressaltar que esses métodos dependem da cooperação dos participantes, ao passo que um sistema de identificação facial tem o potencial de operar mesmo sem a cooperação ou conhecimento do participante. A configuração de um sistema básico de reconhecimento facial é mostrada na Figura 1.

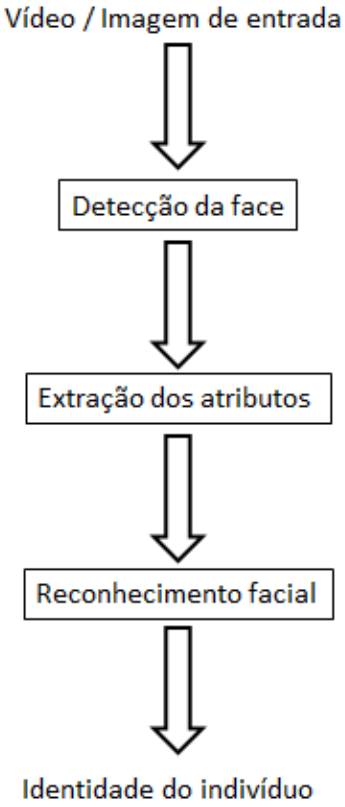


Figura 1: Configuração de um sistema básico de identificação em vídeo

Ao longo dos últimos 8 anos, muita pesquisa tem se concentrado em reconhecimento facial em vídeo (Xu, 2006). Para aplicações tais como carteiras de motoristas, devido à natureza controlada do processo de aquisição de imagem, o problema de detecção da imagem da face é relativamente simples. Há contudo aplicações mais desafiadoras, como por exemplo, o reconhecimento facial em uma imagem de uma multidão em movimento num aeroporto. Em tais casos, se somente uma imagem frontal estiver disponível, a localização automática e detecção da face podem envolver grandes desafios. Os problemas de iluminação e pose são duas questões particularmente difíceis.

Tendo em vista tais dificuldades, este trabalho propõe uma arquitetura paralela de um sistema automático de reconhecimento facial em sequência de vídeo. O sistema procura aproveitar os avanços da tecnologia do reconhecimento facial a partir de imagens frontais. Basicamente o sistema procura imagens frontais de faces e quando as encontra, ativa um algoritmo de reconhecimento concebido para imagens faciais estáticas frontais. O sistema proposto tem a capacidade de buscar a identidade de mais de um alvo simultaneamente, ou seja,

se houver três alvos o sistema seguirá a face de cada um aplicando o algoritmo de reconhecimento somente nas amostras de imagem que apresentarem qualidade adequada(nesse caso, pose frontal).

1.1. Motivação

O crescimento do crime organizado e do terrorismo internacional tem marcado a história da humanidade na última década. No Brasil a criminalidade alcançou níveis alarmantes especialmente nos grandes aglomerados urbanos. Em consequência, o poder público no Brasil segue a tendência internacional de investir em tecnologia para a formulação e operacionalização de uma política de segurança pública não apenas repressiva, mas também preventiva.

Esta conjuntura tem impelido em todo o mundo o aumento do número de sistemas de vídeo monitoramento. Nestes sistemas um arranjo de câmeras estrategicamente instaladas em locais com grande afluência de pessoas, reúne imagens que são visualizadas em centrais de vigilância, onde agentes de segurança podem monitorar vários locais ao mesmo tempo. A tecnologia de vídeo-monitoramento traz consigo benefícios notáveis em comparação com a vigilância presencial: amplia o espaço vigiado, diminui a exposição de agentes de segurança ao risco, agiliza a intervenção, inibe o ato delituoso e reduz consequentemente a criminalidade.

Conforme o número de câmeras de vigilância aumenta em locais públicos torna-se cada vez menos viável avaliar visualmente o grande volume de dados gerados por estes dispositivos na forma de vídeos. A expansão do vídeo monitoramento exige, portanto, o desenvolvimento de técnicas automáticas de análise de imagens em vídeo que permitam a identificação automática de indivíduos suspeitos, principalmente por meio de imagens faciais.

Atualmente, no âmbito de reconhecimento facial, as técnicas de identificação frontal são as com melhor taxa de acerto quando comparadas com outras técnicas não frontais (Phillips et al., 2001). Além disso, a grande maioria das bases de dados disponíveis para identificação conta somente com imagens frontais. Por esse motivo é de grande interesse a identificação da pose de faces em imagens em vídeo, para aplicar os recursos de processamento em faces com maior

probabilidade de reconhecimento usando as técnicas atuais de identificação frontal.

1.2. Objetivo Geral

Esse trabalho tem como objetivo principal propor e avaliar uma arquitetura de um sistema paralelo para o reconhecimento facial a partir de imagens de vídeo.

1.3. Objetivo Específico

São objetivos específicos desta dissertação:

- Construir um protótipo completo da arquitetura proposta, que incorpore funções de detecção, rastreamento, avaliação de qualidade e reconhecimento de imagens faciais em vídeo.
- Investigar métodos alternativos, propor, implementar e avaliar o desempenho de uma solução para o rastreamento de imagens faciais em vídeo.

1.4. Contribuições do Trabalho

O presente trabalho faz uma análise do estado da arte das técnicas de reconhecimento facial em vídeo, com ênfase nas técnicas de rastreamento de faces.

O trabalho propõe ainda uma estrutura para o sistema de reconhecimento facial em vídeo, o que inclui a definição funcional de seus módulos e da comunicação entre estes numa arquitetura paralela de memória compartilhada, tipicamente, numa máquina de múltiplos núcleos.

Contribuição igualmente importante é o protótipo em software que implementa a arquitetura proposta. Trata-se de uma plataforma *multithread* que permite o teste de algoritmos alternativos em cada um dos módulos que compõem o sistema, constituindo uma ferramenta importante para a continuação da pesquisa sobre o tema.

1.5. Organização do Trabalho

No capítulo 2 é realizada a revisão bibliográfica, onde é apresentada parte das publicações recentes sobre os algoritmos e técnicas que foram estudas ao longo desse trabalho.

No capítulo 3 é apresentado de forma mais detalhada os algoritmos e a teoria que embasa cada uma das técnicas utilizadas nessa dissertação.

No capítulo 4 é esboçada a estrutura geral do sistema, com as relações em cada nó e as informações compartilhadas e trocadas por elas.

No capítulo 5, por sua vez, é realizada uma avaliação de desempenho do programa protótipo com os atualmente disponíveis para realização de comparações.

As conclusões e algumas propostas para projetos futuros a fim de complementar e melhorar o desempenho do projeto são apresentados no Capítulo 6.

Por fim, são apresentados o sumário e a bibliografia utilizada nessa dissertação.

2
REVISÃO BIBLIOGRÁFICA

Detecção de faces em imagens é um passo chave em inúmeras aplicações de visão computacional, como reconhecimento facial. Essa é uma difícil tarefa em analise de imagens, devido principalmente à ampla variabilidade intra-classe resultado da influência das condições do ambiente e das várias poses assumidas pela face.

O primeiro passo em qualquer sistema de processamento da face é detectar a localização em imagens onde faces estão presentes. Essa tarefa é desafiadora devido à variabilidade em escala, localização, orientação, e pose. Expressão facial, oclusão e condições de iluminação também modificam a aparência geral das faces.

Entre os métodos de detecção de faces, aqueles baseados em algoritmos de aprendizado tem atraído muita atenção recentemente e têm demonstrado excelentes resultados.

Métodos de detecção de faces em imagens podem ser classificadas em várias categorias. Entre as mais relevantes estão as abordagens que se baseiam em atributos invariantes, métodos de correspondência de modelos e baseadas em aparência.

As baseadas em atributos invariantes visam encontrar descritores que não têm seus valores modificados mesmo quando há variação em pose, ponto de vista, ou condições de iluminação. Feições da face como sobrancelhas, olhos, nariz, boca e linha do cabelo são comumente extraídas usando detectores de borda. Sirohey (1993) propôs um método de localização para segmentar a face de um plano de fundo desordenado. Ele usa um detector proposto por Canny (1986) e heurísticas para remover e agrupar bordas de modo a somente aquelas no contorno da face sejam preservadas.

Yow e Cipolla (1997) apresentaram um método baseado em atributos que usa uma grande quantidade de evidências da imagem para determinar a presença da face, como pontos de interesse, bordas, variação de intensidades. Han *et al.*

(2004) desenvolveram uma técnica baseada em morfologia para extrair o que eles chamam de segmentos análogos a olhos.

Han (1998) afirma que os olhos e sobrancelhas são as feições mais salientes e estáveis da face humana, sendo portanto úteis para detecção.

A textura da face humana, por ter características singulares, também pode ser usada para discriminar a face de outros objetos numa imagem. Augusteijn e Skujca (1993) usam métodos que inferem a presença da face através das texturas possuem a vantagem de serem capazes de detectar faces em poses extremas ou que possuem barba ou óculos.

A cor da pele também carrega bastante informação útil para os detectores e tem se provado ser um atributo eficaz, como usado por Graf *et al.* (1996). Apesar de diferentes pessoas terem diferentes cores de pele, muitos estudos tem mostrado que a maior diferença está em suas intensidades em vez de suas crominâncias.

Muitos sistemas de cores foram usados, entre eles o RGB por Jebara e Pentland (1998), YcrCb por Chai e Ngan (1998), YES por Saber e Tekalp (1998), HSV por Saxe e Foulds (1996), CIE XYZ por Chen *et al*.(1996), entre outros.

Nos métodos de correspondência muitos padrões de uma face são armazenados para descrever a face como um todo. A correlação entre uma imagem de entrada e os padrões armazenados são computados para realizar a detecção.

Yuille *et al.*(1992) usaram estruturas deformáveis para modelar os atributos faciais que se ajustavam a um modelo elástico. Nesta abordagem os atributos da face são descritas por modelos parametrizados. Uma função de energia é definida para vincular bordas, picos e vales da imagem de entrada aos parâmetros correspondentes no modelo.

Nos métodos baseados em aparência os modelos são aprendidos de um conjunto de imagens de treinamento que possui variabilidade representativa do aspecto facial. Esses modelos aprendidos são então usados para detecção.

Redes Neurais têm sido aplicadas com sucesso em muitos problemas de reconhecimento de padrões. Como a detecção facial pode ser tratada como um problema de classificação binária, várias redes neurais foram propostas. A vantagem em usá-las para esse propósito advém da praticidade de treinar um sistema para identificar padrões complexos existentes em faces. Feraud e Bernier (2002) propuseram um método de detecção usando redes neurais auto

associativas. A ideia é baseada no trabalho de Kramer (1991) que mostra que uma rede auto associativa com cinco camadas é capaz de realizar uma análise de componentes principais não-linear.

O primeiro framework de detecção de objetos a fornecer altas taxas de detecção em tempo real foi proposto em 2001 por Viola e Jones. Apesar de poder ser treinado para detectar uma variedade de objetos, foi motivado inicialmente para tratar o problema de detecção de faces. Nesse framework são usados atributos do tipo *Haar-like* selecionados pelo algoritmo *AdaBoost*, que tem seu processamento acelerado pela seleção de atributos simples e combinação de vários classificadores fracos em cascata para gerar um classificador forte.

O tempo de treinamento dos classificadores usados por Viola e Jones, que pode passar de semanas, é o maior ponto fraco desse sistema. Em Treptow e Zell (2004) é usado um algoritmo evolucionário junto com o *AdaBoost* para encontrar atributos que fornecerão melhores classificadores. O algoritmo evolucionário substitui a busca exaustiva sobre todos os atributos possíveis de modo que mesmo conjuntos muito grandes podem ser buscados em um tempo razoável.

Outra técnica que usa uma avaliação em cascata similar é proposta por Romdhani, *et al.* (2004) A principal diferença está no conjunto de atributos selecionados pela função de classificação e no algoritmo de treinamento usado para selecioná-los. Em vez disso, para selecionar e treinar os classificadores Romdhani, S. *et al.* (2004) usam uma máquina de vetor suporte conhecida para gerar classificadores com performance de generalização garantida.

Embora os algoritmos anteriores possuam um bom desempenho em tempo real, eles não são robustos com relação à rotação do tipo *roll* nas faces.

Satyanadh e Vijayan (2004) abordam essa questão. Eles usam um algoritmo detector baseado na textura da face que faz uso de vetores formados por atributos extraídos da imagem por WPA *(Wavelet Packet Analysis*) que são invariantes a escala e rotação para classificar o objeto em face ou não-face. A técnica pode ser dividida em dois grandes módulos. O primeiro inclui um pré-processamento para ressaltar e restaurar as cores naturais em imagens tiradas em ambientes escuros ou com iluminação variável, a localização da pele usando uma transformada não-linear do espaço de cor YcbCr e extração da região candidata a face. O segundo módulo calcula a distância entre o vetor de atributos de uma face protótipo e da região candidata a face e as classificam em faces ou não-faces.

Singh *et al.* (2003) afirmam que a cor é um importante atributo das faces humanas. O processamento da cor é muito mais rápido que o processamento de outros atributos faciais e sob certa iluminação é invariante à orientação. Usa os algoritmos *Skin Color Based Face Detection in RGB Color Space; Skin Color Based Face Detection in YcbCr Color Space* e *Skin Color Based Face Detection in HSI Color Space*. A partir da combinação do resultado das regiões detectadas pelos três algoritmos, a região da textura da face é extraída, indicando a posição da face.

Henry e Takeo (2004) também usam coeficientes das *Wavelets* para compor os atributos extraídos de segmentos de imagem, no entanto são usados vários classificadores, cada um especializado num objeto com determinada orientação. Cada um desses classificadores, também treinados através do *AdaBoost*, determinam se o objeto está presente ou não numa determinada janela dentro da imagem. Cada um desses classificadores é baseado em estatísticas de partes localizadas. De cada parte é extraída um subconjunto de coeficientes da *wavelet* correspondente. Esses coeficientes codificam informações de localização no espaço, frequência e orientação.

Robustez contra tentativa de burlar os sistemas de detecção facial, com o uso de fotografias, são abordadas no trabalho de Li *et al*.(2004) que promove a detecção de faces vivas, usando informações de estrutura e movimento do rosto, baseado na análise de Fourier do espectro de uma única face ou de uma sequência. O trabalho afirma que as componentes de alta frequência das fotos devem ser menores que aquelas nas faces reais, bem como seu desvio padrão.

No trabalho de Han *et al*. (1998) um método de modelagem do background da imagem baseado em GMM (*Gaussian Mixture Models*) é usado para gerar uma máscara binária do fundo da imagem, isolando os segmentos de imagem que contém pessoas. Os pixels dessas regiões isoladas são aplicados numa rede neural treinada para detecção de faces.

Li e Zhang (2004) afirmam que um classificador aprendido pelo *AdaBoost* é subótimo em termos da taxa de erro. Partindo desse principio, propôs um algoritmo de treinamento alternativo, o *FloatBoost*, para criar classificadores de faces em cascata para várias poses que integram um detector facial em pirâmide, que promove o processamento mais fino da imagem nos últimos estágios, com os segmentos de imagem mais promissores.

Como comentado anteriormente, existe um sério problema devido à variabilidade intra-classe de conjuntos de dados de faces em várias poses, que é muito maior que a variabilidade de conjuntos que contém somente faces frontais. Apesar de a arquitetura *AdaBoost* de detector piramidal de Yan Wang *et al*. ser capaz de lidar com esse problema, a complexidade inserida nesse contexto leva a uma alta carga computacional e ao surgimento de *Over-fitting* no treinamento. *Over-fitting* foi discutido por Nakamura *et al.* (2004), mas abordagens mais robustas ainda são necessárias.

Rastreamento tem como objetivo estimar o movimento do objeto. Rastreadores normalmente assumem que o objeto é visível em toda sequência de vídeo. Eles exploram a correspondência temporal entre os quadros.

Várias abordagens para rastreamento foram desenvolvidas, entre elas as que faziam uso de modelos estatísticos Edward *et al.* (1998) e baseados em exemplos Toyama e Blake (2001)

Hager e Belhumeur (1998) usaram um modelo paramétrico para rastreamento. *Active Appearance Model* (AAM) foi introduzido por Cootes *et al*. (1998). Essa técnica usa um modelo estatístico da forma e aparência em escalas de cinza do objeto de interesse.

Cor e forma são pistas importantes para rastreamento, e são base de muitos métodos. Um método de rastreamento robusto de face baseado no algoritmo de condensação é combinado com informações da textura da pele e forma da face (Hyung-Soo e Daijin, 2007).

Rastreamento em tempo real é proposto por Comaniciu *et al.* (2006) através do CAMSHIFT (*Continuously Adaptive Mean Shift*) e filtro de Yao e Gao (2001). Eles propuseram um algoritmo de rastreamento de face baseado na transformada croma da pele e lábios.

Outro rastreador em tempo real é o TLD (*Tracking Learning Detector*), que é um rastreador de objetos baseado em modelos proposto por Kalal *et al.* (2011). Esse método alcança o estado-da-arte em precisão e desempenho. Ele utiliza um detector com modelo atualizado em tempo real e um rastreador baseado no que foi proposto por Tomasi e Kanade (1991).

Muitos fatores influenciam o reconhecimento facial, um dos mais importantes é a qualidade da imagem. Entre os maiores desafios estão a iluminação e a pose das faces. É muito difícil para um sistema realizar a

identificação quando a variação de luz é muito grande, pois muitos desses sistemas dependem do padrão formado pelas intensidades dos pixels no processamento. Várias abordagens foram propostas para lidar com esses problemas, entre elas o uso da análise de componentes principais para a eliminação das primeiras componentes principais das faces dos indivíduos.

Métodos que fazem uso de modelos da aparência utilizam métricas baseadas em imagem para verificar a semelhança da cabeça de um indivíduo a um conjunto de exemplos com poses conhecidas, atribuindo a imagem analisada a pose da face que tem maior semelhança com ela. Essas comparações podem ser feitas com correlação cruzada normalizada em várias resoluções da imagem (Beymer, 1994) e através do erro quadrático médio de uma janela deslizante.

Niyogi e Freeman (1996) propuseram um sistema que possui a vantagem de ser facilmente expandido, pois os novos exemplos podem simplesmente ser inseridos no conjunto de análise sem treinamento, entretanto isso se torna uma desvantagem quando o conjunto fica muito grande, tornando o processamento muito pesado. Outra desvantagem desse método é sua degradação nos casos de imprecisão na localização da face e do fato de o sistema ser somente capaz de estimar poses num conjunto discreto. Uma proposta para solucionar esses problemas é treinar um conjunto de máquinas de vetor suporte para localizar a face e estimar a pose (Ng e Gong, 2002).

A medida da pose também pode ser realizada através do uso de arranjos de detectores, cada um treinado para detectar de forma eficiente uma determinada pose (Osuna, *et al*, 1997), (Viola e Jones, 2004). Esses detectores são similares aos que usam modelos de aparência por operarem diretamente sobre um segmento de imagem. Em vez de comparar uma imagem a um grande conjunto de modelos individuais a imagem é avaliada por um detector treinado com muitos exemplos por um algoritmo de aprendizagem supervisionada. Uma primeira tentativa de por em prática essa ideia usava três maquinas de vetor suporte para três ângulos discretos de *yaw* (Huang *et al.*, 2007).

Um sistema mais recente, proposto por Zhang *et al.* (2006), usa cinco classificadores treinados através do algoritmo *FloatBoost* operando numa configuração multicâmera de campo profundo. Uma das vantagens desse método é que não é necessária uma etapa de localização da face, visto que o detector indica a localização da face numa determinada pose na imagem. Entretanto o

treinamento desses detectores é muito dispendioso, podendo levar semanas para se completar. Além disso, as exigências computacionais aumentam linearmente com o número de detectores usados, dificultando a implementação de um sistema em tempo real capaz de detectar várias poses.

A estimativa da pose também pode ser realizada com o uso de grafos-modelo flexíveis que formam uma estrutura que deve ser ajustada sobre a face do indivíduo com auxilio de uma métrica baseada em pixels para comparar as imagens. Esse modelo, ajustado a face, da como resposta uma determinada pose. Essa técnica permite comparações no nível de feições em vez de comparações da aparência global.

Os vértices desse modelo são associados a um conjunto de descritores de atributos das faces, como por exemplo, pontos fiduciais (olhos, nariz, boca, orelha, etc.), que são utilizados num processo iterativo para deformar a estrutura com o intuito de encontrar a distância mínima entre os atributos da face analisada e as de cada grafo modelo. A pose atribuída à face é aquela associada ao grafo com maior similaridade.

Lades *et al*. (1993) utilizam *Gabor jets* como descritores de cada ponto fiducial, extraídos de várias faces de vários indivíduos em múltiplas poses. Da mesma forma que as técnicas anteriores, a pose estimada é discreta, exigindo muitos grafos modelo para realizar uma estimativa mais fina.

Outro modelo flexível que evoluiu para estimativa de pose foi *Active Shape Models* (ASMs) de Cootes *et al.* (2001), que são modelos estatísticos da forma de objetos que, da mesma forma, podem iterativamente se deformar para se ajustar a um exemplo do objeto numa imagem. As formas são restringidas a uma modelo de distribuição de pontos, que permite a estrutura deformar somente para maneiras vistas em um conjunto de exemplos de treinamento rotulados. Uma evolução do algoritmo anterior é o *Active Appearance Model* do próprio T. Cootes *et al.* (2001) que inclui informações de textura para aumentar a precisão da estimativa.

Métodos baseados no posicionamento relativo dos pontos fiduciais na imagem fornecem uma estimativa continua de valores para poses (Wang e Sung, 2007) com o uso de uma câmera. Com múltiplas câmeras ao redor da cabeça, o ângulo *yaw* pode ser estimado com a maioria dos modelos baseados na cor da pele (Canton-Ferrer *et al.,* 2007). Essa técnica pode ser incrementada para uso em

vídeo através de rastreadores, que buscam os pontos quadro a quadro (Zhu e Fujimura, 2003).

Uma combinação dos métodos apresentados anteriormente também é possível, e visam compensar as deficiências dos sistemas funcionando individualmente (Hu *et al.* 2006), (Bartlett, 1997) alcançando melhor performance para imagens sob diferentes condições de iluminação. Sua hipótese era de que as primeiras componentes principais capturavam somente variações devido à iluminação. Entretanto isso interferia o reconhecimento sob condições normais. Outra abordagem feita por Chen *et al.* (2006) era de utilizar a transformada discreta da função cosseno sobre a imagem da face para suprimir variações da iluminação no domínio logaritmo ao mesmo tempo que ressalta outras feições do rosto como olhos, sobrancelhas, nariz e boca, que são pontos da face que carregam muita informação da identidade do indivíduo. Wang, H. *et al.* (2004) tinham o objetivo de adquirir atributos das faces invariantes à iluminação para um grupo de imagens do mesmo sujeito. Georghiades *et al*. (2001) propôs um método para lidar com as variações de iluminação pelo uso de um cone de iluminação. Esta técnica também trata sombreamento e múltiplas condições de iluminação. A principal desvantagem desse método é que o conjunto treinamento exige mais que três imagens alinhadas por pessoa.

Enquanto sistemas de reconhecimento facial dependem principalmente de imagens estáticas, há um significante interesse em desenvolver sistemas robustos que aceitem vídeo como dados de entrada. Reconhecimento facial em vídeo tem atraído interesse devido a difusão de câmeras de vigilância de baixo custo. Entretanto, imagens faciais em vídeo estão frequentemente em poses não-frontais e estão sujeitas a mudanças substanciais da iluminação, degradando a performance da maioria dos sistemas comerciais. Duas características distintivas estão disponíveis num vídeo: i) múltiplos quadros do mesmo sujeito e ii) informação temporal. Múltiplos quadros garantem a variação de pose, permitindo a seleção adequada de quadros de boa qualidade. Informação temporal em vídeo é considerada como a informação embutida na dinâmica de movimento facial em vídeo. No entanto, é dificil determinar se há alguma informação relacionada a identidade no movimento facial: mais trabalho precisa ser feito para utilizar a informação temporal (Lee *et al.* 2003), (G. Aggarwal *et al.* 2004), (Zhou *et al.* 2004).

A dificuldade de reconhecimento de face em vídeo depende da qualidade das imagens de face em termos de pose, variações de iluminação, oclusão e resolução. O grande número de quadros de vídeo, também aumenta a carga computacional. Ao contrário das imagens estáticas 2D, vídeos geralmente contém vários individuos numa seqüência de quadros. A maioria dos detectores faciais em tempo real são capazes de detectar várias faces numa dada imagem (Viola e Jones, 2001). A detecção e reconhecimento simultâneos podem ser realizados associando cada face no quadro corrente com imagens da face observadas em quadros anteriores. Problemas de baixa resolução foram abordados através da adaptação de super-resolução baseada em melhoria de imagem (Keren *et al.* 1988).

Roy-Chowdhury e Xu (2006) desenvolvem um sistema de reconhecimento facial a partir de sequências de vídeo que é robusto a variação de pose e condições de iluminação. Se baseia em resultados teóricos que integram os efeitos de movimento, iluminação e forma para gerarem um modelo facial, e a partir dele capturar uma imagem adequada para a identificação.

Como a face é um objeto 3D diferentes fontes de iluminação podem gerar várias condições de iluminação e sombreamento. Estudos vêm sendo desenvolvidos em busca de atributos faciais que são robustas contra variações de iluminação e formas de compensar essa variação através de conhecimento prévio da localização das fontes luminosas (Qing *et al.*, 2006), (Chen *et al*., 2006). Esses métodos proporcionam imagens da face visualmente aprimorada depois de normalizar a iluminação e mostram melhorias na precisão do sistema de reconhecimento de até 100%.

Expressão facial é um tipo de variação intra-classe que também interfere na identificação. Há algumas abordagens baseadas em atributos locais (Martinez, 2002) e em modelos 3D (Lu e Jain, 2008), (Kakadiaris *et al*., 2007) projetados para lidar com essa questão.

3
ALGORITMOS APLICADOS

3.1.
Viola-Jones

O algoritmo proposto por Paul Viola e Michael Jones (Viola; Jones, 2001) é um método para detecção de objetos capaz de processar imagens de forma rápida e alcançar altas taxas de detecção. Ele utiliza uma nova representação de imagens chamada “imagens integrais”, que nada mais é do que uma matriz com as mesmas dimensões da imagem de entrada cujos elementos são a soma dos valores dos pixels acima e à esquerda daquela posição. Além disso, é usado um algoritmo de aprendizado baseado no *AdaBoost* que seleciona um pequeno número dos mais relevantes atributos de um grande conjunto de atributos e gera classificadores eficientes. Na Figura 2 são mostrados três classificadores fracos dispostos em cascata. O primeiro classificador fraco recebe todos os segmentos de imagem para processamento, e segundo seus critérios de classificação, elimina uma parcela com baixa probabilidade de conter o objeto de interesse. O segundo classificador fraco recebe os segmentos não descartados pelo primeiro e realiza outra classificação seguindo outros critérios. O mesmo procedimento é realizado do segundo para o terceiro classificador. Esse processamento em cascata segue por quantas camadas forem necessárias para alcançar a precisão desejada.

Usando um método para combinar classificadores cada vez mais complexos em cascata é possível que regiões de fundo da imagem sejam rapidamente descartadas enquanto é disponibilizado um tempo maior de processamento em regiões mais prováveis de conter objetos. A cascata pode ser vista como um mecanismo que descarta regiões com baixa probabilidade de conterem instâncias dos objetos de interesse.

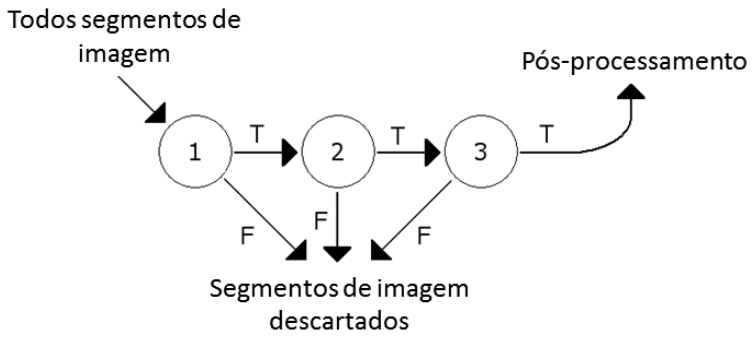


Figura 2: Classificadores fracos em cascata.

Há muitas razões para usar atributos ao invés de pixels diretamente. A mais importante é que atributos descrevem melhor as classes de objetos presentes numa imagem. Outra razão é que sistemas baseados em atributos operam muito mais rápido que um sistema baseado em pixels.

Os atributos empregados pelo método de detecção são a soma dos valores dos pixels da imagem dentro de áreas retangulares. Esses atributos se assemelham às funções da base de *Haar*. No entanto, como os atributos usados pelo método de Viola-Jones envolvem mais do que uma área retangular, eles geralmente são mais complexos. A Figura 3 ilustra quatro tipos de atributos usados no detector. O valor de qualquer atributo é sempre simplesmente a soma dos valores dos pixels dentro dos retângulos claros subtraída da soma dos valores dos pixels dentro dos retângulos escuros. Com o uso da representação de imagens chamada imagem integral, atributos obtidos de regiões retangulares da imagem podem ser avaliadas em tempo constante, o que confere ao método uma considerável vantagem em termos de tempo de processamento quando comparado a métodos alternativos. Como cada área retangular em um atributo é sempre adjacente a pelo menos um outro retângulo, conclui-se que qualquer atributo calculado a partir de duas áreas retangulares pode ser calculado com seis referências ao ponteiro da imagem integral, qualquer atributo calculado a partir de três áreas retangulares pode ser computado com oito referências ao ponteiro da imagem integral e qualquer atributo calculado a partir de quatro áreas retangulares pode ser computado com apenas nove referências ao ponteiro da imagem integral (Viola; Jones, 2001).

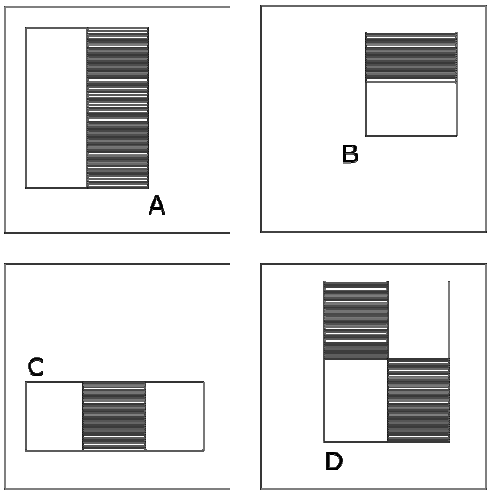


Figura 3: Tipos de features usado pelo Viola-Jones.

Portanto é possível criar classificadores para detectar instâncias de objetos na imagem utilizando atributos como os mostrados na Figura 3 em várias escalas diferentes, calculados a partir de imagens integrais, para encontrar regiões na imagem que obedecem a uma série de limiares que são obtidos durante o treinamento do classificador.

3.2. TLD

O *Tracking-Learning-Detector* (TLD) é um rastreador baseado no conceito de aprendizado em tempo real, onde o objeto a ser rastreado é aprendido à medida que assume novas formas e poses (Kalal et al., 2010). Por exemplo, o módulo de *tracking* é baseado no muito conhecido Lucas-Kanade *Tracker* (Tomasi; Kanade, 1991), que é usado em rastreamento de pontos esparsos numa imagem. O módulo de *detection* faz uso de classificadores baseados em árvores e correlação normalizada. Todos esses métodos foram unidos para implementar um rastreador que alcança o estado-da-arte na área, o TLD.

A informação adquirida de forma independente pelos blocos de *tracking* e *detection* são fundidas e uma posição mais refinada do que a que seria entregue por esses blocos individualmente é gerada. O bloco *learning* é o responsável por gerar exemplos de treinamento e treinar os classificadores presentes no *detection*, mas não exerce influência direta sobre o *tracking*, pois a implementação utilizada nesse trabalho não faz uso de nenhuma informação que possa atualizar essa

estrutura do rastreador. Um diagrama simplificado do TLD é mostrado na Figura 4.

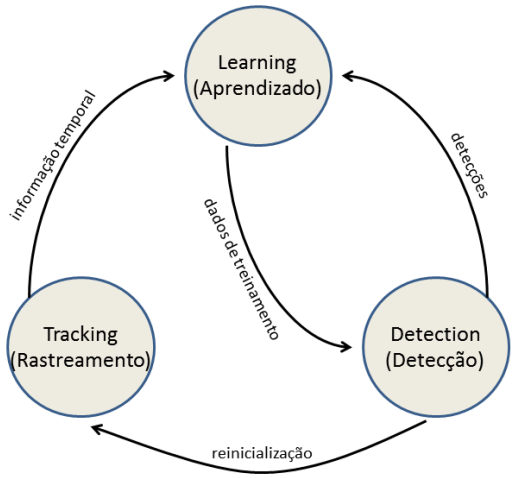


Figura 4: Diagrama simplificado do sistema de rastreamento.

A didática escolhida para apresentar o algoritmo será a de apresentar a sequência de ações tomadas ao longo da execução do programa, apresentando alguns conceitos à medida que forem necessários para o entendimento da parte do algoritmo sendo explicado. Esta explicação visa o entendimento do sistema como um todo, e não como um conjunto de blocos com interface bem definida. Isso permitirá que alguns detalhes pertencentes à interface dos blocos sejam completamente entendidos, uma vez que sua explicação depende da apresentação de partes de algoritmos pertencentes a mais de um bloco.

Outra representação do diagrama de blocos da Figura 4 é o da Figura 5. Essa será a representação em que será baseada a explicação adiante.

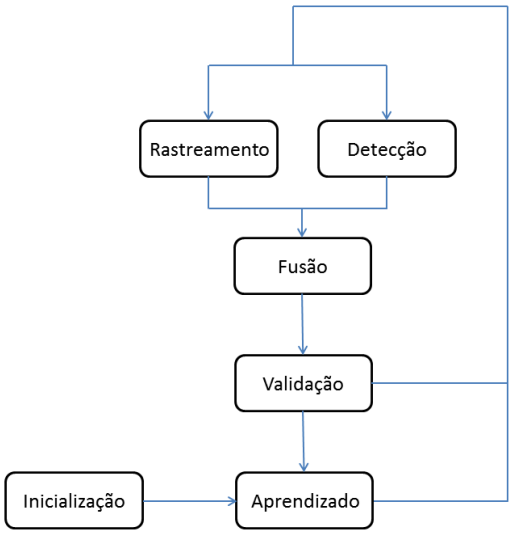


Figura 5: Diagrama de fluxo do algoritmo TLD.

A primeira ação realizada no código do TLD é a inicialização de todos os parâmetros fixos. Dentre os mais importantes estão o tamanho mínimo do objeto, tamanho do segmento (*patch*) normalizado usado pelo NN, limiares para o classificador baseado em pequenas árvores (*ferns*) e NN (*Nearest Neighbour*), número de atributos (*features)*, número de transformações (*warps*) e sobreposição para o segmento ser considerado positivo ou negativo.

O "tamanho mínimo do objeto" garante a não aceitação de objetos muito pequenos como objetos para rastreamento, uma vez que diminuem a eficácia do algoritmo. O "tamanho do segmento" normalizado está diretamente ligado com a precisão, processamento e quantidade de memoria utilizada pelo programa. Esses segmentos são utilizados pelo classificador NN utilizando o cálculo de semelhança com NCC. Grandes dimensões aumentam a precisão do resultado, mas tomam mais tempo de processamento e memória do computador, e dimensões muito reduzidas diminuem a qualidade dos resultados. A escolha que melhor atende esses requisitos foi feita empiricamente, assim como foi feita para a maioria dos parâmetros de configuração. O limiar para a média aritmética dos classificadores baseados em árvores é o valor mínimo de confiança para determinado segmento seguir na cadeia de processamento do detector. O limiar usado no classificador NN, que é o último estágio do detector do sistema, é responsável por determinar se o segmento tem confiança suficiente para ser considerado como sendo o objeto ou parte dele. Esses limiares são ajustados

empiricamente, pois dependem do ambiente de onde serão obtidos os quadros do vídeo e da precisão desejada.

O número de atributos influencia diretamente a capacidade de discriminação do classificador baseado em árvore, uma vez que quanto mais atributos forem utilizados pelas árvores, maior será a precisão da descrição do segmento. No entanto, à medida que o número de atributos utilizados aumenta, a tendência é que a carga de processamento aumente. Logo a escolha do número de atributos deve levar em consideração a relação acurácia x desempenho requerida para o classificador. O *número de warps* indica a quantidade de transformações de um segmento a fim de gerar exemplos sintéticos para treinamento do detector baseado em árvores (pequenas rotações, suavização, ruído, empenamentos e deslocamentos aplicados nos segmentos). Por fim o percentual de sobreposição de uma *Bounding Box* sobre o segmento atualmente considerado como objeto é utilizada para decidir se essas novas regiões devem ser rotuladas como objeto (positivas) ou background (negativas).

Todos esses parâmetros do TLD serão explicados com mais detalhes à medida que o algoritmo for sendo apresentado.

Os principais procedimentos a serem realizados no TLD são o treinamento inicial do detector e a inicialização da estrutura do TLD (dentre os itens estão os apresentados previamente) e, feito isso, o processamento de cada quadro. As outras tarefas são secundárias e não são exclusivas do TLD, como inicialização do hardware de vídeo e exibição de resultados. Desse modo, o assunto será subdividido em dois tópicos: Inicialização do detector e *tracker* e Processamento de quadro.

3.2.1. Inicialização do detector e *tracker*

O primeiro passo realizado nessa etapa do TLD é a inicialização da estrutura da função do LK. Ao fim desse procedimento é acionada uma função que gera, a partir das dimensões do objeto encontrado pelo detector de faces (que nesse sistema tem sempre dimensões quadradas), uma grade de *Bounding Boxes* de várias escalas e posições dentro da região do quadro da imagem. Uma *Bounding*

Box nada mais é que uma região retangular que delimita um segmento de imagem num quadro. Um exemplo é mostrado na Figura 6.

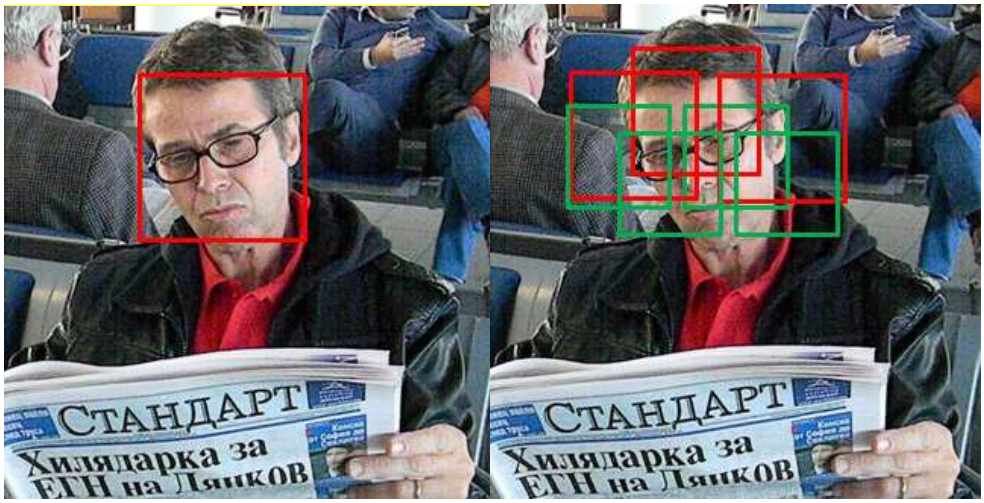


Figura 6: Exemplo inicial (esquerda) e grade de BB's gerada (direita).

Essa grade de *Bounding Boxes* é importante para o TLD, pois reparte as regiões da imagem de modo que a cada quadro, o segmento de imagem delimitado por ela possa ser avaliado num classificador quanto ao seu rótulo (objeto ou background) e quanto a seu posicionamento relativo à última posição válida do objeto (criação de exemplos de treinamento). Além disso, essa grade permanece inalterada ao longo do processo, ou seja, independente do quadro que estiver sendo exibido, essas *Bounding Boxes* permanecem nas mesmas posições e com mesmas escalas, não sendo, portanto, recalculadas em nenhum outro momento durante a execução do programa. Vale ressaltar que na implementação do sistema apresentado nesse trabalho, a grade gerada pelo primeiro objeto face detectado é compartilhada por todos os alvos rastreados pelo TLD, promovendo um maior compartilhamento de informações e diminuição de dados redundantes.

Após a criação da grade são geradas as coordenadas de onde serão extraídas informações das *Bounding Boxes*, compondo os chamados atributos como mostrado na Figura 7.

Figura 7: *Bounding Box* com atributos distribuídos.

As coordenadas são geradas aleatoriamente no início do programa e mantidas ao longo da execução. Suas posições dependem da escala da BB, ou seja, a posição da coordenada de um pixel dentro da BB usado no atributo é proporcional às dimensões dessa BB. Isso permite que um determinado atributo seja adaptado a qualquer BB em qualquer escala.

Os atributos utilizados nesse trabalho são do tipo *2bit binary patterns* que codificam a orientação do gradiente dentro do segmento avaliado, como mostrado na Figura 8.

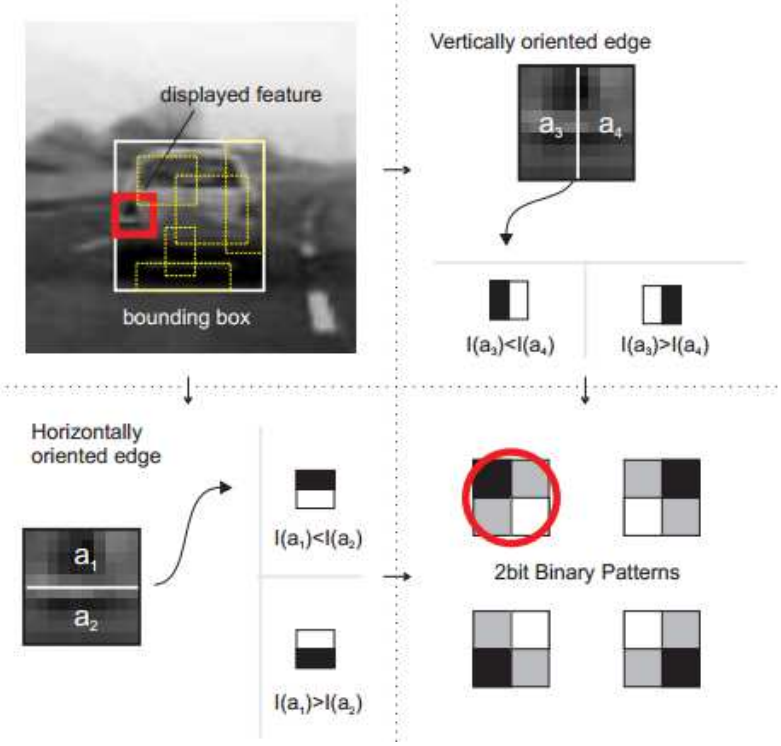


Figura 8: Procedimento para criação dos 2bit Binary Patterns.

Os atributos são calculadas da seguinte forma: dada uma coordenada de um atributo (destacado em vermelho na Figura 8) é extraído um segmento da imagem dentro dessa BB e é avaliado o gradiente na direção horizontal e vertical. O gradiente em cada direção é representado por um bit, ou seja, um atributo é representado por dois bits nessa codificação, o que possibilita que ela adquira uma das quatro formas possíveis como mostrado na Figura 8.

Como será mostrado mais adiante esses atributos são usados pelo classificador baseado em árvores para atribuir uma probabilidade da BB ser o objeto, ou pelo menos parte dele.

Criados os atributos (mais especificamente as coordenadas de onde serão tiradas as informações para criá-las) esses dados são passados para as *ferns* (árvores) juntamente com a grade de BB*'s* para prepará-las para o treinamento do classificador baseado em árvores.

O passo seguinte é o calculo do percentual de área sobreposta de cada BB gerada no grid sobre a BB inicial (a BB encontrada pelo detector de faces) como mostrado na Figura 9.

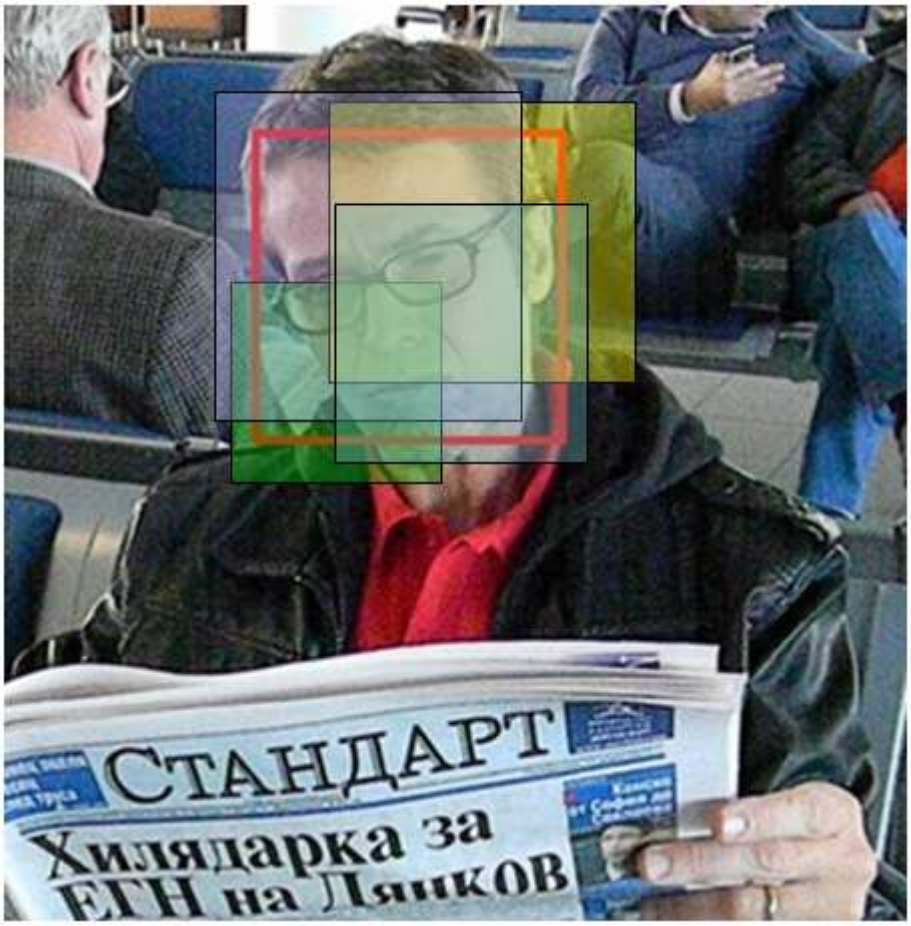


Figura 9: A sobreposição é dada pela razão da área de interseção sobre a área total das BB's.

A BB do *grid* com maior sobreposição é utilizada para atualizar as coordenadas da BB inicial gerada pelo detector de faces. Isso torna o resultado do processamento dos quadros posteriores mais precisos, uma vez que a BB utilizada nos cálculos esta dentro do conjunto gerado no grid. Esse percentual também é importante na criação dos exemplos positivos e negativos de treinamento para os classificadores *ferns* e NN do bloco detector.

Para criação dos exemplos positivos são selecionadas aleatoriamente 10 BB*'s* com sobreposição maior que 0,6. Cada uma dessas BB*'s* sofre 20 *warps,* que são transformações no segmento da imagem para criação de exemplos sintéticos, com o intuito de complementar os exemplos positivos de treinamento para os classificadores *ferns*. Os *warps* geram 200 exemplos (10 BB*'s* x 20 *warps* = 200 exemplos). A variância desse segmento de imagem é mantida na memória para ser usada como um limiar de escolha de BB*'s* promissoras pelo detector, ou seja,

BB’*s* muito uniformes (variância menor que o limiar configurado) são desconsideradas pelo detector para as etapas seguintes.

Para a criação dos exemplos negativos são selecionadas aleatoriamente 100 BB’*s* com sobreposição menor que 0,2 e com variância maior que o limiar definido anteriormente. Essas BB’*s* não sofrem *warps*, uma vez que há uma abundância de exemplos negativos disponíveis, não sendo necessário, portanto, criação de exemplos sintéticos, como feito para os exemplos positivos.

Para o treinamento do classificador NN são reescalados os segmentos de imagem limitados pelas BB’*s* para as dimensões 15x15 para comporem os exemplos de treinamento deste classificador.

De posse dos exemplos, são treinadas as *ferns* e os NN. A parte do detector mostrada na Figura 10 é um classificador *fern.*

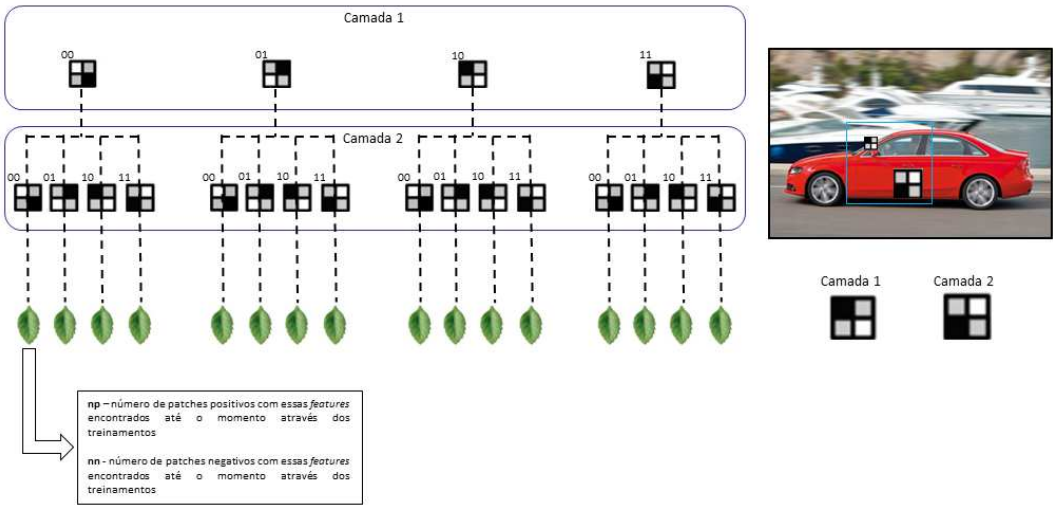


Figura 10: Classificador *fern* de duas camadas.

Cada exemplo de treinamento das *ferns* é usado para criar uma estatística dentro do classificador. O Treinamento funciona da seguinte maneira: dado um exemplo positivo, seus atributos são extraídos para cada uma das *ferns,* como mostrado na Figura 11.

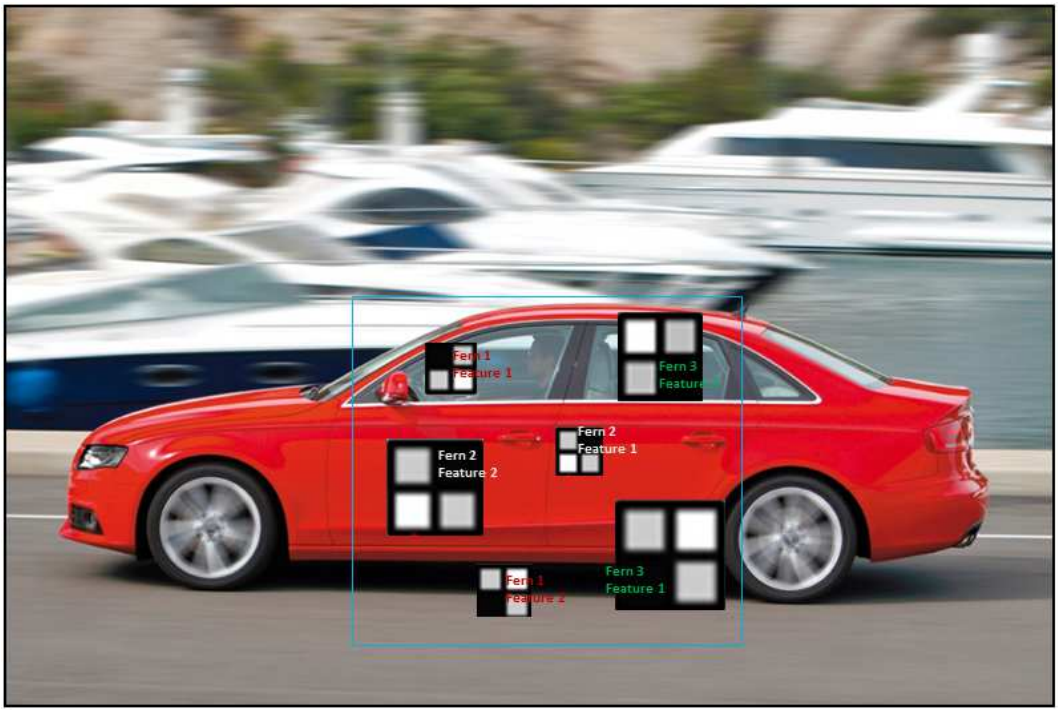


Figura 11: Atributos extraídos de uma BB.

Esses números em binário, resultado da extração e codificação dos atributos, indicam, no treinamento, qual folha de uma *fern* no classificador deve ter seu peso alterado. Os pesos de cada folha indicam a probabilidade *a posteriori* de um determinado segmento de imagem com aquela caracterização ser ou fazer parte do objeto. Ela é obtida através da razão do número de classificações positivas com aquele atributo sobre a soma do número de classificações positivas e negativas como mostrado na Equação 1.

$$\Pr(obj) = \frac{p}{n+p} \qquad (1)$$

A mesma ideia é usada na classificação. Nesse caso os pesos não são alterados, mas usados para calcular a probabilidade do segmento de imagem se tratar do objeto de interesse. Conforme enfatizado no início do texto, as explicações serão melhor desenvolvidas no ponto onde são aplicadas, para melhor compreensão por parte do leitor. Neste caso, o procedimento será mais bem explicado na subsecção “detecção” da seção “processamento do quadro”.

Para ilustrar, considere uma *fern* que usa 2 atributos para descrever um segmento de imagem. Sabendo que cada atributo é codificado com 2 bits (4 possível arranjos de gradientes) o número de bits usados é 4, resultando numa árvore de 16 folhas. Cada uma dessas folhas indica uma caracterização do segmento da imagem, ou seja, cada folha é representada por 4 bits. Durante o treinamento, ao ser obtida a respectiva representação do segmento, que é um número de 4 bits, a folha correspondente da *fern* que está sendo treinada recebe um incremento na variável de número de segmentos positivos ou negativos, dependendo do rótulo do segmento usado no treinamento. Durante uma classificação esses números são consultados para formar a probabilidade de o segmento ser objeto, usando a Equação 1.

A sequência de ações num treinamento é mostrada na Figura 12. Se o segmento limitado pela BB for positivo o classificador sofrerá uma alteração de modo a aumentar a probabilidade do segmento de imagem com essas características ser considerado como objeto.

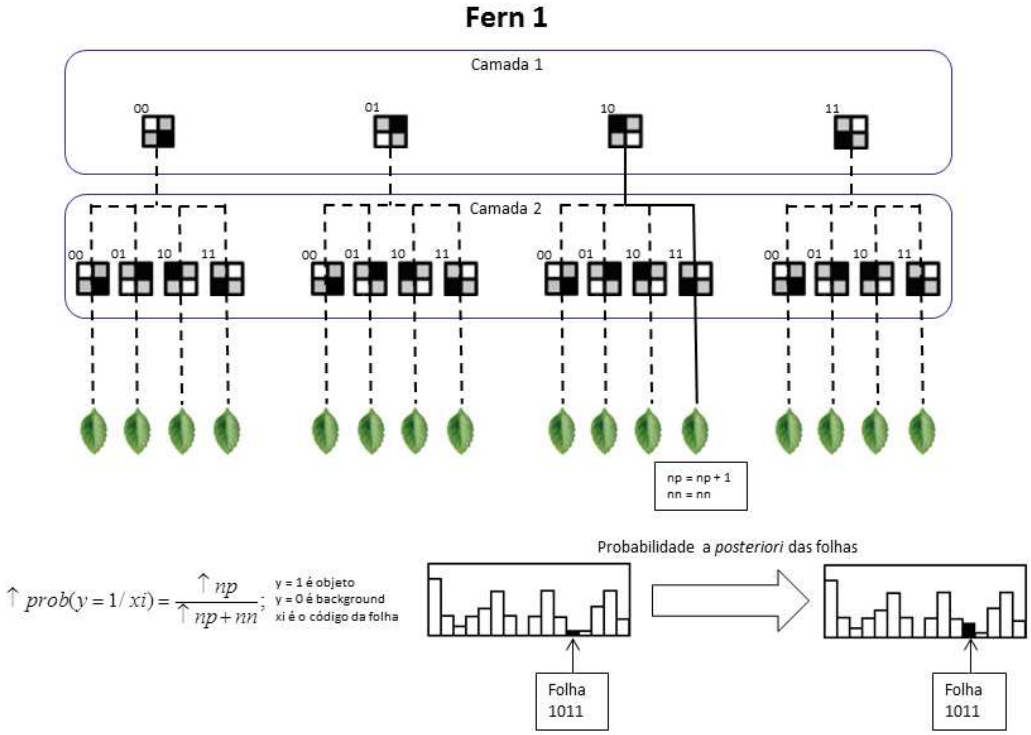


Figura 12: Treinamento da *fern* com um exemplo positivo.

Se o segmento limitado pela BB for positivo o classificador sofrerá uma alteração de modo a diminuir a probabilidade do segmento de imagem com essas características ser considerado como objeto, como a mostrado na Figura 13.

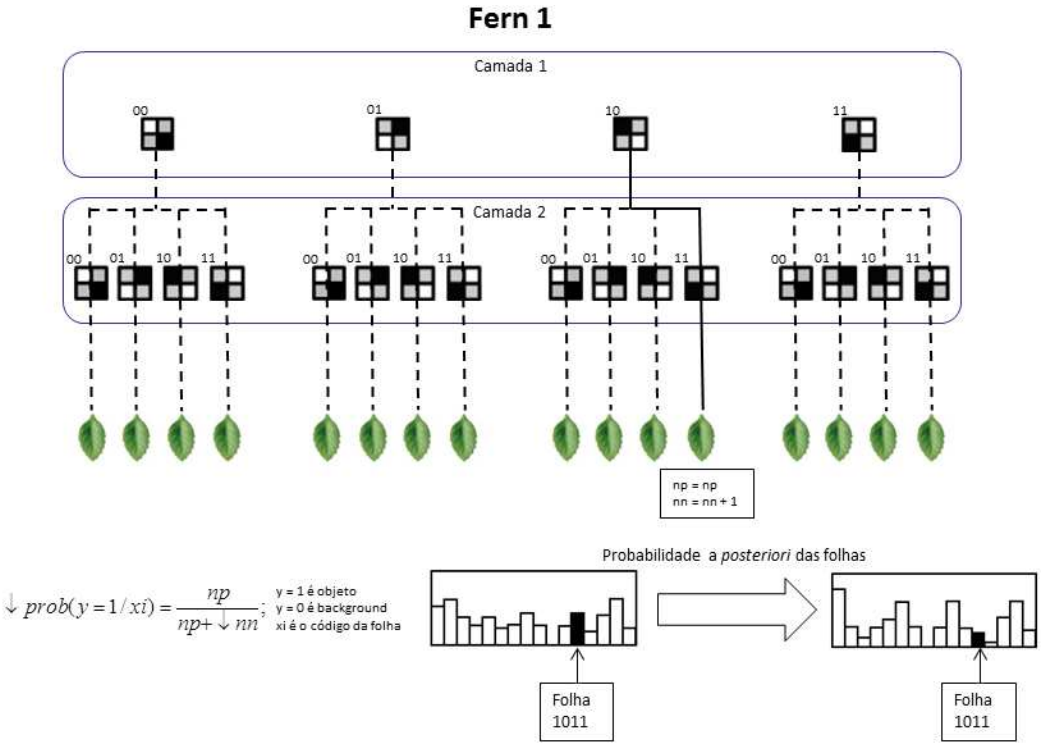


Figura 13: Treinamento da *fern* com um exemplo negativo.

O procedimento para treinamento do bloco NN do detector, como mostrado na Figura 14, é mais simples.

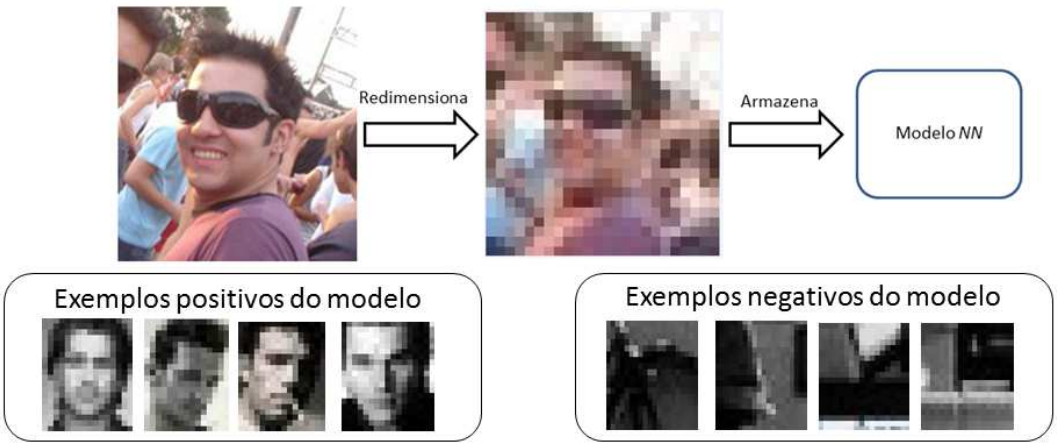


Figura 14: Redimensionamento do exemplo e inserção no modelo.

Os exemplos de treinamento, que são simplesmente os segmentos de imagem reescalados para 15x15 pixels, são selecionados para serem inseridos no modelo do objeto. Essa seleção segue o simples critério de que o novo exemplo, positivo ou negativo, deve conter bastante nova informação sobre o objeto de interesse, sendo avaliado da seguinte maneira:

Inicialmente é calculada a correlação de cada um dos exemplos com os segmentos positivos e negativos existentes no modelo. Isso é feito através da correlação normalizada de cada um dos exemplos com todos os segmentos existentes no modelo, tanto positivos quanto negativos.

A partir das correlações calculadas é estimada a confiança do segmento exemplo em relação ao modelo. Ela é obtida através da maior correlação encontrada com os segmentos positivos e negativos do modelo. Quanto maior a correlação entre segmentos, mais parecidos são. Isso significa que para se ter uma medida de distância entre um segmento exemplo e um dentro do modelo devem ser usadas as seguintes equações:

$$dN = 1 - \max(nccN); \text{ e} \quad (2)$$

$$dP = 1 - \max(nccP). \quad (3)$$

A Equação 2 indica uma estimativa de distância de um segmento exemplo para os segmentos negativos existentes no modelo, e a equação 3 indica uma estimativa de distância de um segmento exemplo para os segmentos positivos existentes no modelo. Essa equação é facilmente interpretada: quanto mais parecidos são os segmentos, maior a correlação. Entretanto a distância é menor, o que implica numa relação inversamente proporcional como evidenciado nas equações acima.

Com essas medidas de distância é criado uma medida de similaridade relativa (confiança), que é dada pela Equação 4.

$$\text{Confiança} = \frac{dN}{dN + dP} \quad (4)$$

A equação acima mostra que a confiança cresce à medida que a distância do segmento avaliado aos segmentos negativos contidos no modelo aumentam e a distância aos segmentos positivos do modelo diminuem.

Com os segmentos exemplos devidamente rotulados e acompanhados por suas respectivas confianças, é realizada a seleção dos que serão inseridos no modelo do objeto.

Caso sejam os primeiros segmentos positivos ou negativos, são automaticamente inseridos, do contrário, são verificadas as seguintes condições:

- Se a confiança do segmento é menor que um limiar configurado e o rótulo indica que o exemplo é positivo, adiciona o segmento ao conjunto de segmentos positivos do modelo.

- Se a confiança do segmento é maior que um limiar configurado e o rótulo indica que o exemplo é negativo, adiciona o segmento ao conjunto de segmentos negativos do modelo.

As condições acima garantem que os segmentos inseridos trazem informações que buscam complementar o modelo do objeto. O rótulo tem precedência sobre a confiança, visto que foi atribuído usando o conhecimento espacial do objeto, ou seja, foi rotulado segundo os critérios de sobreposição das BB*'s* do grid sobre a BB atual do objeto rastreado.

Sendo assim, a condição para que o segmento exemplo seja inserido ao modelo é que haja contradição entre o classificador NN e o rótulo do exemplo. Isso garante que esse exemplo agrega mais informação ao modelo.

3.2.2. Processamento do quadro

Terminada a inicialização do *tracker* a etapa seguinte é o processamento propriamente dito do quadro, que pode ser subdividido em quatro partes:

- Rastreamento (*tracking*)
- Detecção (*detection*)
- Integração (*location estimatior*)
- Aprendizado (*learning*)

3.2.2.1. Rastreamento (*tracking*)

Para efetuar o rastreamento é gerado um *grid* de pontos sobre a BB atual do objeto, como mostrado na Figura 15.

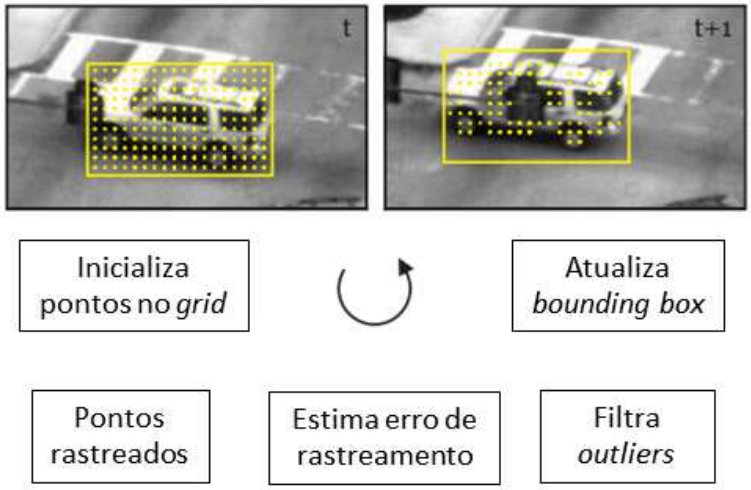


Figura 15: Rotina usada no algoritmo do módulo de rastreamento.

Na implementação desse trabalho é escolhida um grid de 10x10 pontos. Todos os pontos, após serem dispostos na região de interesse, são rastreados utilizando o algoritmo de rastreamento Lucas-Kanade (Tomasi e Kanade, 1991), e, a partir da nova posição dos pontos no quadro seguinte, é estimado o *forward-backward error*, que é calculado como mostrado na Figura 16.

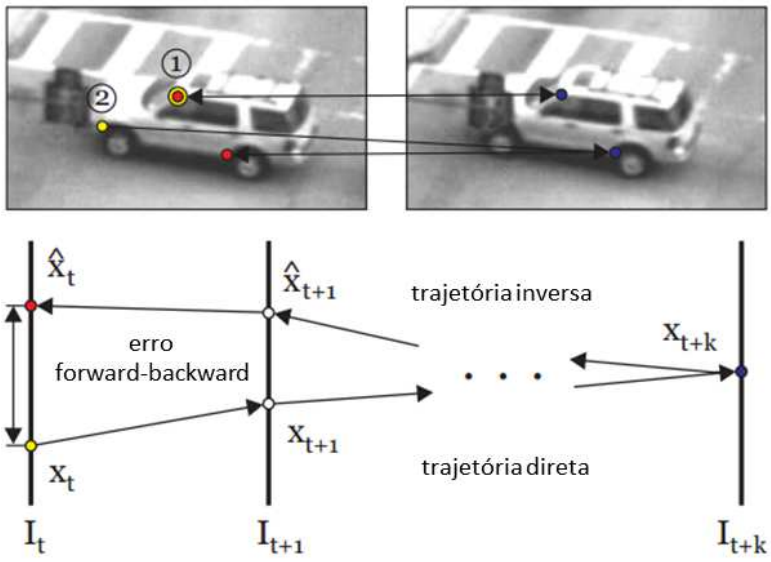


Figura 16: Método para cálculo do forward-backward error.

Como pode ser visualizado na Figura 16, um determinado ponto é rastreado no sentido direto e inverso numa sequência de dois ou mais quadros através do algoritmo explicado anteriormente.

A distância euclidiana entre a posição original do ponto (em amarelo) e a posição do ponto depois de ser rastreado no sentido direto e inverso (em vermelho) é o que chamamos de *forward-backward error* (Kalal *et al*., 2010). Esse procedimento é aplicado a todos os pontos no grid, fornecendo uma estimativa de erro de cada ponto rastreado.

Adicionalmente é aplicada a correlação normalizada sobre a região dos pontos nos quadros subsequentes. Uma alta correlação indica que o ponto tem grande chance de ter sido rastreado corretamente, visto que é esperado que a região em torno do ponto permaneça muito parecida em quadros consecutivos. Bem parecida porque há duas coisas impedem que a correlação normalizada seja máxima: o erro no rastreamento do ponto e a mudança de pose do objeto onde o ponto esta localizado.

Com as duas estimativas apresentadas, é feita uma filtragem, com o intuito de retirar os *outliers* do conjunto. Os pontos cujo *forward-backward error* for menor que a mediana do conjunto de pontos do grid e cuja correlação normalizada for maior que a mediana da correlação desse mesmo conjunto são considerados pontos confiáveis para a estimativa da nova BB. O grupo de pontos resultantes é usado então para estimar as dimensões e posição da BB no quadro seguinte. A nova posição é estimada através do deslocamento médio dos pontos desse conjunto, como mostrado na Figura 17.

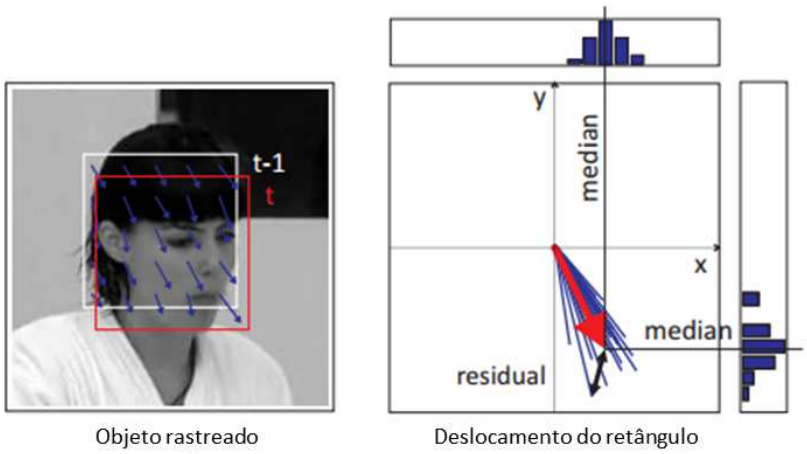


Figura 17: Estimativa de nova posição da BB pelo deslocamento médio de um conjunto de pontos.

Já a nova escala da BB (novas dimensões) é estimada através da razão da distância média entre pontos da BB em um quadro e a distância média entre pontos da BB no quadro subsequente.

Com a nova BB em mãos é feita a transformação desse segmento de imagem para o uso no NN, ou seja, o segmento de imagem é reescalado para as dimensões 15x15 pixels. Feito isso é calculada a confiança da BB com relação ao modelo do objeto disponível. Essa confiança será útil na avaliação dos quadros subsequentes.

3.2.2.2. Detecção (*detection*)

O processo de detecção é realizado pelo processamento das BB's do grid em cascata, ou seja, são usados classificadores mais simples nas primeiras etapas a fim de reduzir o número de segmentos a serem processados pelos blocos que demandam maior tempo de processamento. Isso está ilustrado na Figura 18.

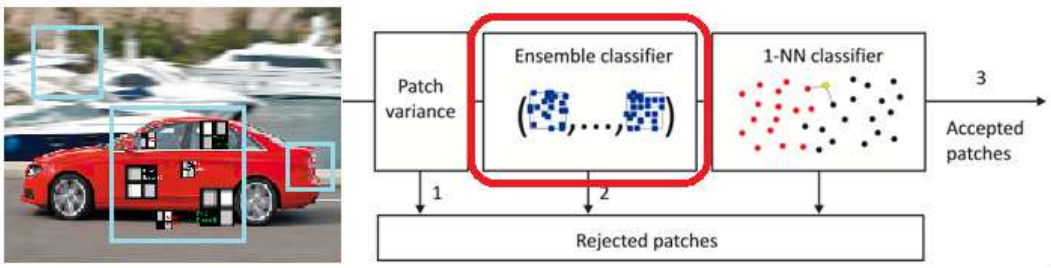
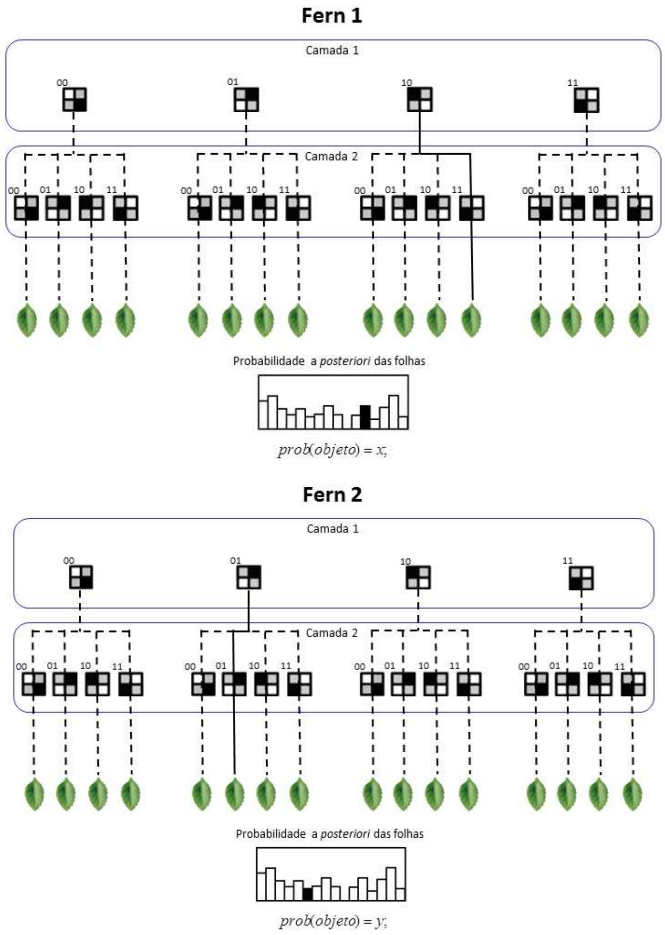


Figura 18: Blocos do detector do sistema.

O primeiro bloco do detector é o de medida de variância. Esse bloco rejeita os segmentos de imagem com variância abaixo de um limiar, ou seja, muito uniformes. Esse limiar é definido a partir da variância do primeiro exemplo do objeto, feita na inicialização. As BB's não rejeitadas têm sua confiança calculada pelo classificador *fern*. A estrutura de três classificadores *fern* de duas camadas está ilustrado na Figura 19.



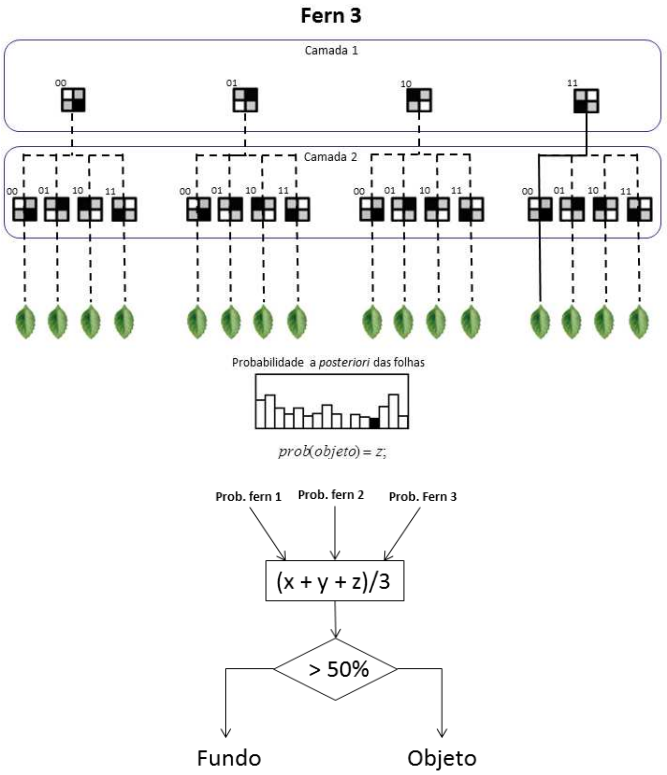


Figura 19: Estrutura de um classificador composto de 3 *fern*s de 2 camadas.

Na Figura 19 é demostrado como é feito o cálculo para determinar se o segmento da imagem é objeto ou não. As probabilidades estimadas por cada *fern* são buscadas na árvore de acordo com os atributos encontrados e são unidas por média aritmética, o que implica num peso igual de decisão para cada uma. Se a probabilidade da média das três for maior que 50% a BB é considerada objeto, em caso negativo é considerado fundo de imagem.

As BB′*s* cujas confianças estejam acima do limiar definido são processadas no próximo estágio, o classificador NN. Como muitas BB′*s* sobrepostas passam para esse estágio, são selecionadas as 100 mais confiáveis para esse bloco.

O segmento de imagem limitado por cada uma das BB′*s* é reescalado para as dimensões 15x15 pixels e suas confianças por similaridade relativa são calculadas. Além disso, é setada uma *flag* que indica se o segmento é altamente correlacionado com os exemplos positivos ou negativos do modelo. As BB′*s* classificadas como positivas pelo detector são aquelas cujas confianças estejam acima do limiar definido na inicialização.

Após ser realizado o rastreamento e detecção, o próximo passo é a integração dos resultados para estimar a nova BB.

3.2.2.3. Integração (location estimator)

Com os resultados do detector e rastreador disponíveis é feita a fusão dos resultados para estimar as novas coordenadas da BB do objeto.

A nova estimativa da BB do objeto depende principalmente de duas coisas: se o rastreador retornou uma estimativa da BB e se o detector encontrou regiões que podem ser o objeto de interesse. De certa forma o detector tem uma precedência maior que o rastreador, visto que é treinado ao longo da execução do programa.

Caso o detector retorne BB*'s*, um algoritmo de *clusterização* hierárquica é aplicado para agrupá-las, cujo parâmetro de medida de distância é baseado na sobreposição espacial das BB*'s*. Para cada *cluster*, uma nova BB é estimada a partir da média de todas as outras, e sua nova confiança é a maior das confianças dentro do cluster correspondente. Isso está ilustrado na Figura 20.

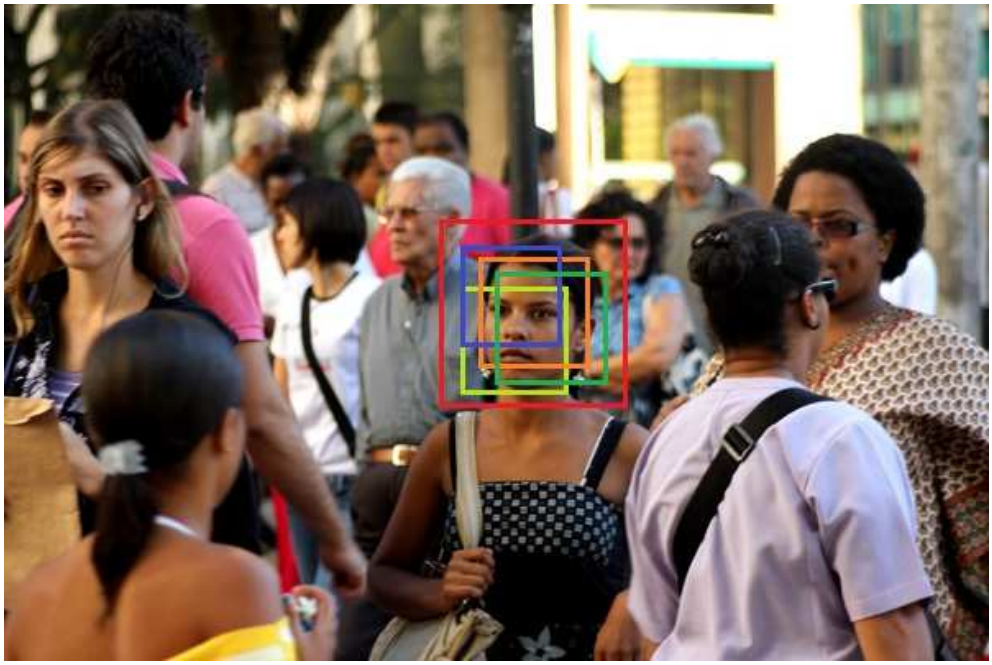


Figura 20: Cluster de BB's utilizado na estimativa da posição real do objeto.

O algoritmo de *clusterização* é realizado da seguinte maneira: primeiro é calculada a sobreposição, par a par, de todas as BB*'s* entregues pelo detector. A partir de uma BB é procurada a mais próxima (por meio da medida de distância citada anteriormente). Se a distância for menor que um certo limiar configurado,

essa BB é colocada no mesmo *cluster*, e além disso, se uma BB já estiver em um outro *cluster*, os dois *clusters* são unidos. Por outro lado, se a sobreposição for maior que o limiar configurado as BB*'s* são colocadas em clusters diferentes. Esse procedimento é aplicada a todas as BB*'s*.

Feito isso, são procurados os *clusters* cujas correspondentes BB*'s* equivalentes possuem uma sobreposição menor que um limiar configurado e confiança maior que a BB estimada pelo rastreador, indicando que a BB estimada por ele está errada.

Se existir somente um *cluster* sob essas condições o rastreador é reinicializado, substituindo a confiança e a BB pela calculada no cluster.

Caso não exista somente um *cluster* nessas condições são selecionadas as BB*'s* do detector que possuem sobreposição com a BB do rastreador acima de um determinado limiar. Essas BB*'s* (do rastreador e do detector) são usadas num refinamento da posição atual do objeto por meio de uma média ponderada das coordenadas, com peso 10 para a BB do rastreador e peso 1 para as BB*'s* do detector.

Outra possibilidade, avaliada quando não ocorrer nenhuma das situações descritas anteriormente, é que o rastreador não possua uma estimativa da BB do objeto. Nesse caso são utilizadas somente as informações provenientes do detector. Caso o detector possua detecções do objeto é realizada a clusterização. Se o resultado for somente um cluster, independente da confiança e da sobreposição com o estimado pelo rastreador (já que não existe) o rastreador é reiniciado, substituindo a confiança e a BB pela calculada no cluster.

Por fim, se o rastreador possuir uma resposta confiável, ou seja, se o rastreador possuir uma estimativa para a BB e não houver *cluster* ou, se houver, existir no mínimo dois (independente da confiança dos *clusters*), é aplicado o algoritmo de aprendizado descrito a seguir.

3.2.2.4. Aprendizado (*learning*)

O primeiro passo no aprendizado é reescalar o segmento limitado pela BB do objeto para as dimensões 15x15 pixels. A partir desse segmento redimensionado é verificada a confiança do segmento do objeto com o modelo

através do NN. Se o segmento possuir uma confiança abaixo de um valor mínimo, for muito correlacionado com os exemplos negativos do modelo ou possuir uma variância abaixo da mínima, o processo de aprendizado é cancelado.

Caso esses critérios sejam atendidos o aprendizado segue com o cálculo da sobreposição da BB atual do objeto com as BB*'s* do grid para criação dos exemplos positivos de treinamento do detector, tantos os exemplos positivos para treinamento do classificador *fern* quanto para o classificador NN. A criação desses exemplos é feita da mesma forma que na inicialização, mudando apenas o número de *warps* e os parâmetros de transformação do segmento.

Os exemplos negativos de treinamento do classificador *fern* são criados pela seleção dos segmentos de imagem limitados pelas BB*'s* no grid que possuem uma sobreposição abaixo de um limiar (definido na inicialização do sistema) e cuja confiança do classificador *fern* seja maior que zero. A confiança maior que zero garante que sejam usados exemplos recorrentes, ou seja, que são normalmente processados no detector, rejeitando aqueles que possuem pouca informação sobre o objeto. Isso é necessário para reforçar a estatística da correspondente folha no classificador.

Os exemplos negativos para o classificador NN são aqueles segmentos de imagem classificados pelo primeiro bloco do detector (classificador *fern*) como sendo objeto, mas que devido a sua pequena sobreposição com a BB atual do objeto devem ser reclassificadas como exemplos negativos do objeto. Esses exemplos negativos carregam mais informação que outros exemplos negativos pelo fato de o classificador tê-los considerado como positivos, sendo necessários para treinar o detector para não cometer novamente o mesmo erro.

Com os exemplos positivos e negativos dos classificadores *fern* e NN em mãos, o modelo é atualizado, seguindo os mesmos critérios utilizados na inicialização do TLD.

3.3. KLT

O algoritmo Lucas-Kanade (LK) (Tomasi e Kanade, 1991), que é base do módulo de *tracking* do TLD, foi uma tentativa de produzir um rastreamento denso de pontos. Como o método é facilmente aplicado a um subconjunto dos pontos na

imagem de entrada, ele se tornou uma importante técnica esparsa (*tracking* de poucos pontos).

O algoritmo pode ser aplicado em um contexto esparso, pois se baseia somente em informações locais que são derivadas de algumas pequenas janelas ao redor de cada ponto de interesse.

A desvantagem de usar pequenas janelas locais no Lucas-Kanade é que grandes movimentos podem mover o ponto para fora do local da janela e assim se tornar impossível para o algoritmo encontrá-lo. Este problema levou o desenvolvimento do algoritmo LK "piramidal", que rastreia o ponto começando do mais alto nível da imagem em pirâmide (menor detalhe) e continua a busca nos níveis mais baixos (detalhes finos). Rastrear com o uso de imagens em pirâmide permite que grandes movimentos sejam pegos pelas janelas locais.

A ideia do LK baseia-se em três hipóteses:

- Constância de brilho: Um pixel de uma imagem de um objeto na cena não pode mudar em aparência à medida que se move de um frame para o outro. Para imagens em escala de cinza significa assumir que o brilho de um pixel não muda à medida que é seguido de um frame para outro.
- Persistência temporal ou "pequenos movimentos": O movimento da imagem de um trecho da superfície muda lentamente com o tempo. Na prática, isto significa que incrementos temporais são rápidos o bastante com relação à escala de movimento na imagem que o objeto não se move muito de um frame para outro.
- Coerência espacial: Pontos vizinhos numa cena pertencem à mesma superfície, tem movimento similar, e projetam para pontos próximos no plano da imagem.

A primeira hipótese, constância de brilho, é apenas a exigência de que os pixels em um trecho rastreado pareçam o mesmo ao longo do tempo. Isso é expresso pela Equação 5.

$$f(x,t) = I(x(t),t) = I(x(t+dt),t+dt) \qquad (5)$$

Essa hipótese é bem simples e significa que a intensidade do pixel rastreado - *I(x(t), t)* - não exibe mudanças ao longo do tempo, ou seja:

$$\frac{\partial f(x,t)}{\partial t} = 0 \qquad (6)$$

A segunda hipótese, persistência temporal, em suma significa que movimentos são pequenos de um frame para o outro. Em outras palavras, pode-se ver esta mudança como aproximar uma derivada de intensidade em relação ao tempo. Para entender as implicações desta hipótese, primeiro considere o caso de uma única dimensão espacial.

Neste caso pode-se começar com a equação de consistência de brilho, substituindo a definição do brilho *f(x, t)* enquanto levando em consideração a dependência implícita de *x e t, I(x(t), t),* e então aplicar a regra da cadeia para derivadas parciais. Isso gera:

$$\frac{\partial I}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial I}{\partial t} = 0 \qquad (7)$$

$$I_x v + I_t = 0 \qquad (8)$$

onde I_x é a derivada espacial através da primeira imagem, I_t é a derivada entre imagens ao longo do tempo, e v é a velocidade procurada. Assim chega-se a uma simples equação para a velocidade de *optical flow* em um caso unidimensional:

$$v = -\frac{I_t}{I_x} \qquad (9)$$

Para ilustrar o problema considere a Figura 21, que mostra uma borda - consistindo de um valor alto na esquerda e um baixo valor à direita - que esta se movendo para direita ao longo do eixo x. O objetivo é identificar a velocidade v na qual a borda esta se movendo, como mostrado na parte superior da Figura 21.

Na parte mais baixa da figura é possível ver que a medida dessa velocidade é apenas uma composição da taxa de subida da borda ao longo do tempo e da taxa de deslocamento dessa borda (que seria a taxa variação espacial do ponto).

A Figura 21 revela outro aspecto da formulação do *optical flow*: as hipóteses não são completamente verdadeiras. Isto é, o brilho da imagem não é realmente estável e os incrementos no tempo (que são determinados pela câmera) frequentemente não são tão rápidos em relação ao movimento como seria desejado. Desta forma, a solução para a velocidade não é exata. Entretanto, se a resposta estiver próxima o bastante da real, é possível alcançá-la por iteração.

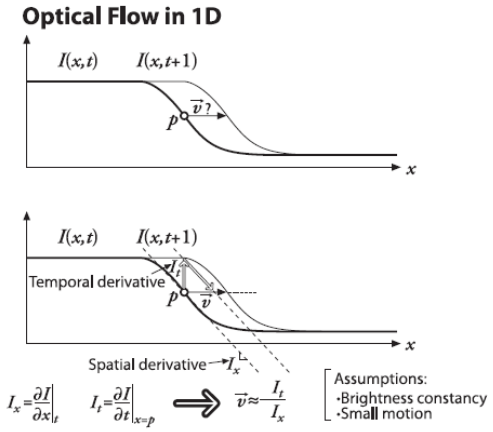


Figura 21: Ilustração do Optical Flow em uma dimensão.

A abordagem apresentada ilustra o caso unidimensional, entretanto a explicação para a abordagem bidimensional, que é a utilizada nas imagens, pode ser facilmente derivada a partir desse exemplo.

3.4. LBP

A técnica conhecida como Padrões Binários Locais (LBP – do inglês *Local Binary Patterns*) (Ahonen et al., 2006) é uma técnica para codificação de texturas em escala de cinza utilizada no algoritmo de reconhecimento facial. O código da textura em torno de um pixel é calculado através de um valor binário atribuído a cada pixel da imagem, formando uma vizinhança linear de raio R ao redor do pixel principal posicionado em uma região central, como mostrado na Figura 22. A partir desta matriz, é realizada uma comparação dos valores dos pixels vizinhos

com o valor do pixel central, atribuindo 0 aos valores do vizinhos inferiores e 1 aos superiores, gerando um vetor de 0s e 1s para cada ponto.

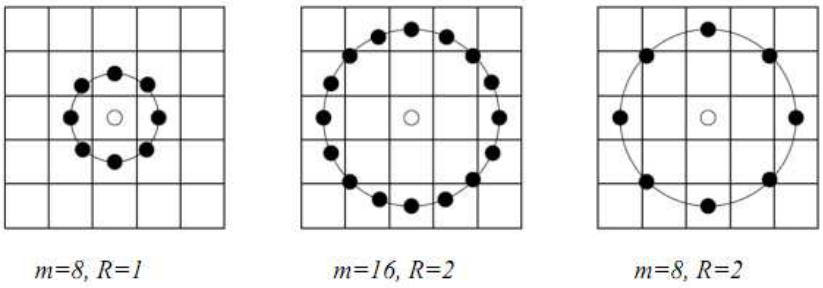


Figura 22: Três exemplos de vizinhanças usadas para definir uma textura e calcular um Local Binary Pattern (LBP).

Esses vetores são usados então para criar uma nova imagem cujas intensidades são dadas em escala de cinza por esses vetores, como mostrado na Figura 23 para um m = 8 e R = 2.

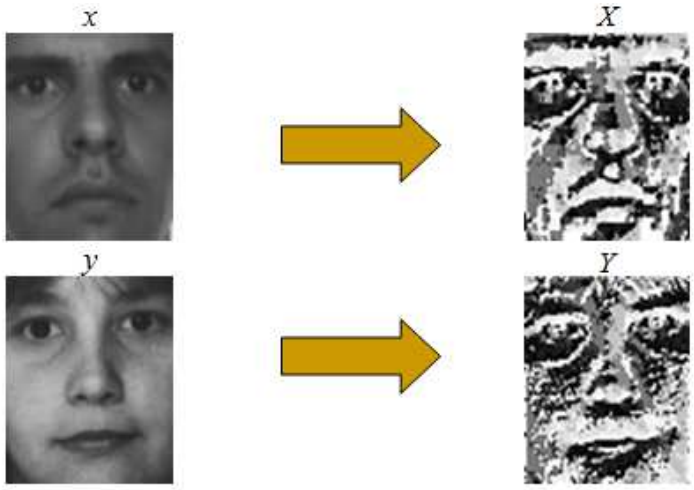


Figura 23: Representação da face usando LBP.

Com a representação LBP das imagens, nesse caso faces, a imagem é repartida por uma matriz n x n predefinida pelo algoritmo de reconhecimento facial e é computado o histograma para cada uma dessas regiões, como mostrado na Figura 24.

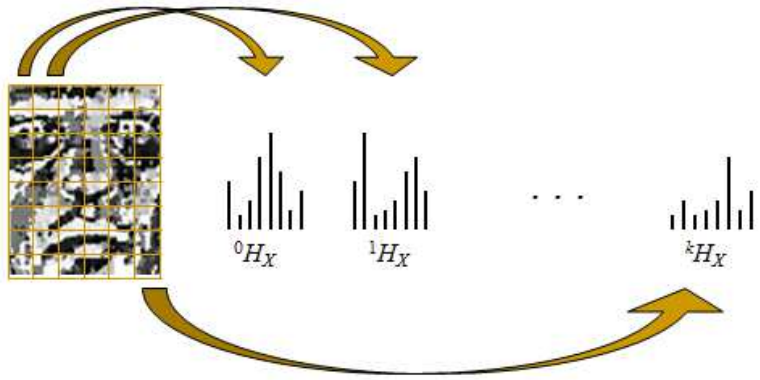
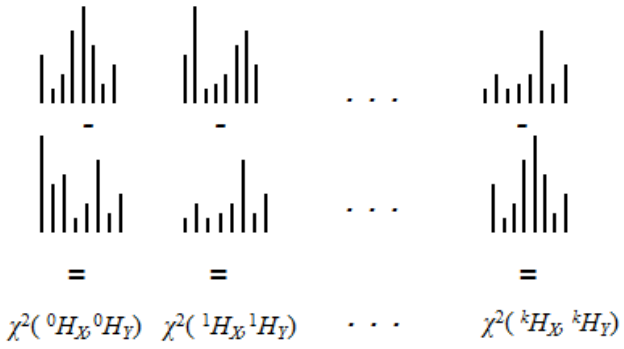


Figura 24: Histogramas da face LBP.

Feito isso é calculado a distancia χ^2 entre blocos correspondentes de duas imagens de faces, possibilitando uma comparação entre faces para reconhecimento, como mostrado na Figura 25. Essa comparação é feita por um limiar que é gerado pela soma dessas distâncias.

$$\chi^2\left({}^bH_X, {}^bH_Y\right) = \sum_c \frac{\left[{}^bH_X(c) - {}^bH_Y(c)\right]^2}{{}^bH_X(c) + {}^bH_Y(c)}$$



$$\chi^2\left(H_X, H_Y\right) = \sum_b w_b \chi^2\left({}^bH_X, {}^bH_Y\right)$$

Figura 25: Cálculo de similaridade.

Para melhorar a acurácia do método, é computada uma soma ponderada dessas distâncias dos blocos, dando maior peso para as regiões da face que possuem maior significância para o reconhecimento, como por exemplo, os olhos e a boca e menor peso para as regiões menos significativas como as bochechas.

4
SISTEMA PROPOSTO

O sistema proposto tem uma arquitetura modular, que facilita sua manutenção e a atualização do código.

4.1.
Descrição Geral da Arquitetura

O sistema é dividido basicamente em quatro blocos, que operam sobre várias *threads*, que são linhas de processamento paralelas. Cada um desses blocos realiza um dos algoritmos apresentados no capitulo 3, como ilustrado na Figura 26.

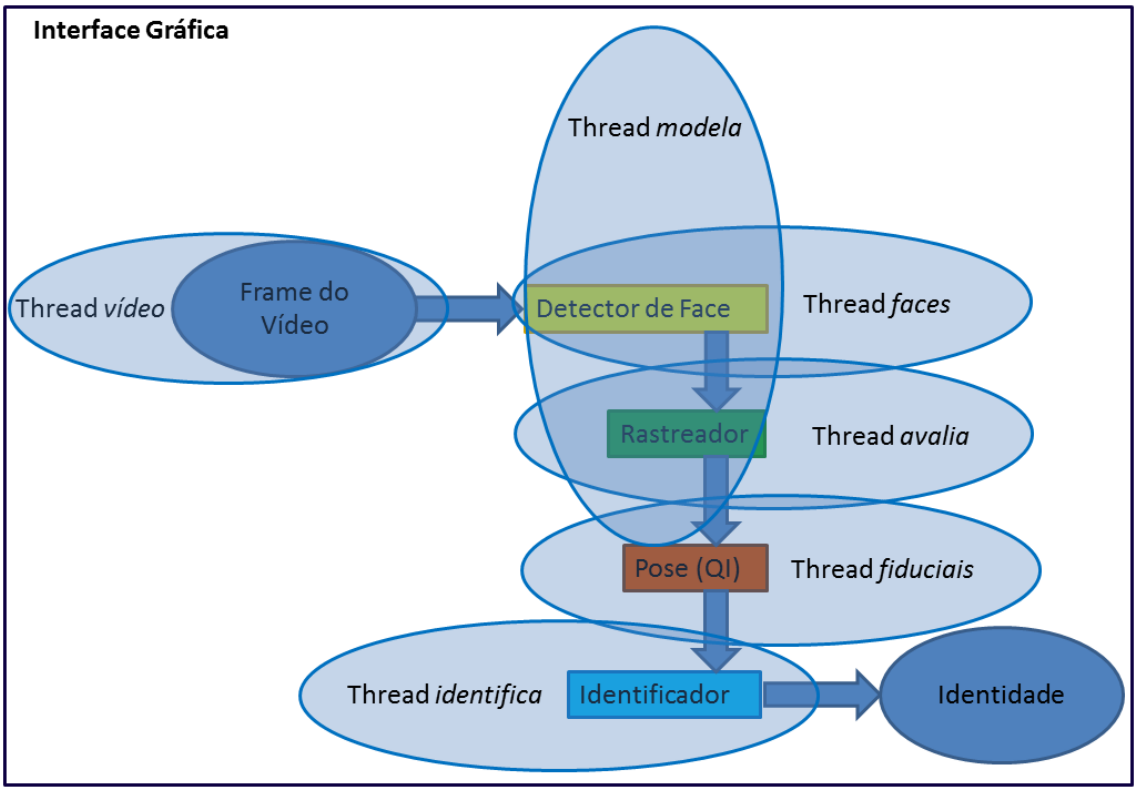


Figura 26: Visão geral do sistema.

No primeiro bloco, identificado na Figura 26 como “detector de face”, é usado um detector de objetos previamente treinado para encontrar faces, baseado no método de Viola-Jones. O segundo bloco, o “rastreador”, é baseado no algoritmo de modelagem de objeto em tempo de execução, o *Tracking-Learning-Detection*. O bloco seguinte avalia a qualidade da imagem (QI), identificado na Figura 26 como “pose” e que, no caso desta dissertação, tão somente avalia a pose da face baseando-se na posição relativa de três pontos fiduciais (olhos e nariz). Por fim, há o bloco “identificador”, que realiza o algoritmo de reconhecimento baseado em LBP.

Ressalte-se novamente que o escopo do trabalho se restringe à arquitetura do sistema e à etapa de rastreamento com vistas à eficiência computacional e não tem como objetivo, portanto, a taxa de reconhecimento do sistema como um todo.

Propõem-se sete *threads* (linhas de processamento gerenciadas independentemente) assim denominadas: *vídeo*, *faces*, *fiduciais*, *modelo*, *avalia*, *identifica e limpa*, sendo que esta última não é apresentada na Figura 26, pois atua sobre todos blocos. Cada uma das *threads* é responsável pelo processamento de um dos blocos no sistema.

4.2. Variáveis mais Importantes

Antes de iniciar a descrição mais profunda do sistema convém apresentar as variáveis utilizadas na implementação:

- faces – Contém as coordenadas da última ocorrência de uma face num quadro de vídeo (BB). Cada elemento dessa variável representa uma face que está sendo rastreada pelo sistema, sendo atualizada quadro a quadro.
- fc_model – Contém as novas instâncias de imagens faciais encontradas no vídeo que devem ser enviadas à *thread* de modelagem.
- modelos – Contém os modelos dos objetos utilizados pelo rastreador do sistema.

- poses – Contém um índice da qualidade de cada imagem facial seguida pelo rastreador. No presente sistema só a informação relativa à pose da face na imagem é armazenada.
- reconhecido – Contém variáveis lógicas utilizadas pelas *threads* para determinar se o objeto já foi ou não reconhecido pelo processamento de um quadro precedente.
- identidades – Contém os nomes dos indivíduos identificados pelo sistema até o instante corrente.
- vídeo – Contém os quadros processados que são capturados pelo dispositivo de entrada de vídeo.

Ao invés de se criar uma única variável contendo todas as informações do sistema, as informações foram separadas em variáveis independentes, como apresentado anteriormente. Essa abordagem permite o acesso simultâneo de múltiplas variáveis, o que não seria possível se todas estivessem concentradas numa única estrutura de dados. Ainda assim essas variáveis são compartilhadas pelas *threads*. Nesse caso os acessos para leitura e escrita das variáveis são controlados por um mecanismo de sincronização, a fim de impedir erros de disputa das variáveis pelas *threads*.

Cada elemento armazenado na variável carrega informação sobre uma instância de um determinado objeto de interesse no vídeo. Isso significa que a detecção de uma instância de um objeto (face rastreada) implica em definir o conteúdo das variáveis que lhe dizem respeito. Como as variáveis são independentes (são referenciadas por diferentes ponteiros de memória), um segundo mecanismo de sincronização é realizado para manter as correspondências entre os elementos das variáveis quando algum objeto é removido pela *thread limpa*. Este procedimento será melhor explicado quando for apresentado o algoritmo desta *thread*.

Uma ilustração da estrutura e relacionamento das variáveis é mostrada na Figura 27. Nessa figura cada caixa corresponde a uma das variáveis do sistema e as setas que as conectam indica a relação de correspondência de cada um de seus elementos com os das outras variáveis.

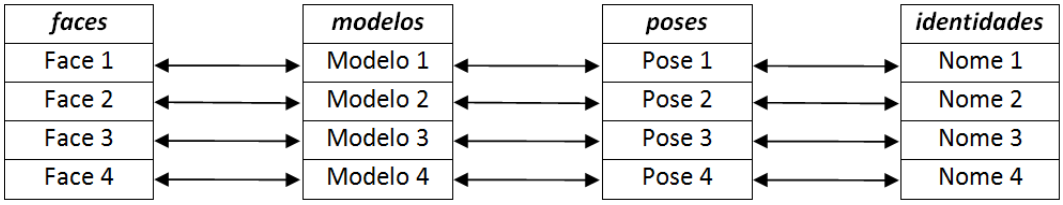


Figura 27: Estrutura de armazenamento dos dados.

Deste ponto em diante descrevem-se as *threads* e a interação entre elas. Como comentado anteriormente são utilizadas 7 *threads* que têm as seguintes funções principais:

- *vídeo* – Responsável por distribuir o fluxo de vídeo entre as *threads*.
- *faces* – Busca novas instâncias de imagens faciais no vídeo e avalia sua originalidade antes de as armazenar para modelagem.
- *modela* – Responsável pela modelagem do objeto que será rastreado nos quadros seguintes do vídeo.
- *avalia* – Os modelos criados pela *thread* "modela" são utilizados nesta *thread*, que atualiza a posição de cada instância de imagem facial no vídeo.
- *fiduciais* – *Thread* de medida de qualidade da imagem, que fornece um índice relacionado à probabilidade de sucesso de uma eventual tentativa de identificação de uma instância de face.
- *identifica* – Extrai o segmento de imagem do alvo para posterior identificação por essa *thread*.
- *limpa* – Instâncias de faces que não são mais detectadas ou que foram erroneamente inseridas pela thread faces são excluídas por essa thread, que é também responsável pela sincronização dessa ação.

A Figura 28 apresenta a interação entre as *threads* e as variáveis que compartilham.

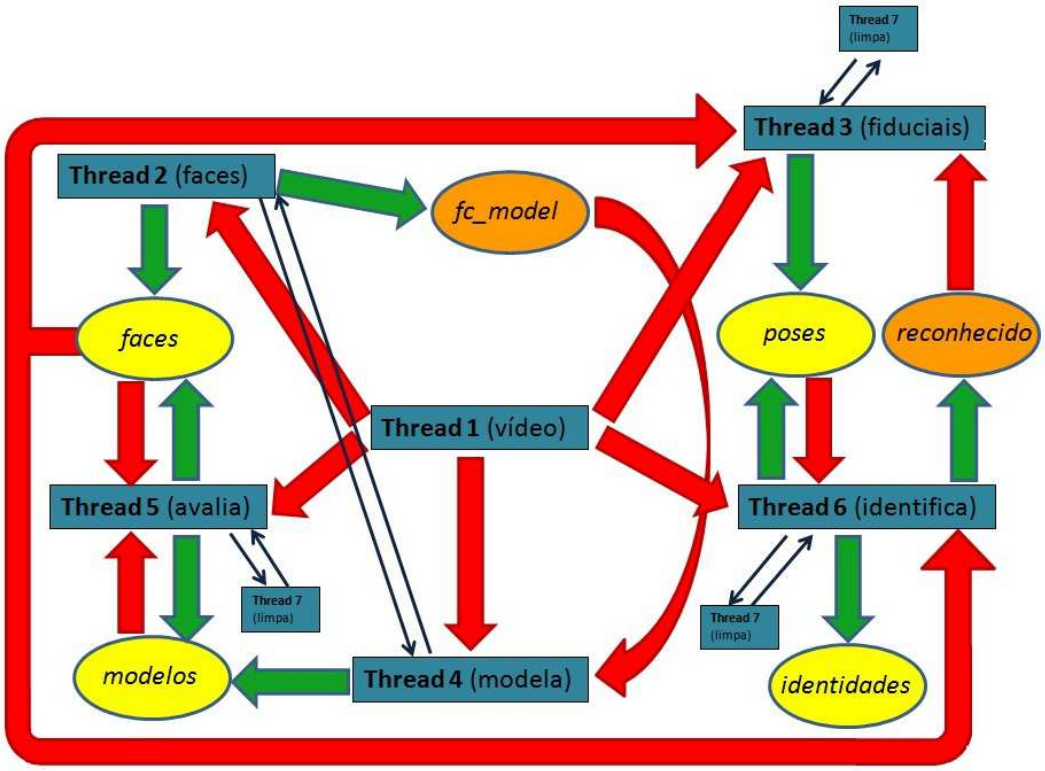


Figura 28: Fluxo de dados do sistema.

As setas em vermelho indicam as operações de leitura, e apontam para a *thread* que a está realizando. As setas em verde indicam as operações de escrita e apontam para a variável cujo conteúdo é alterado. As setas delgadas na cor azul simbolizam os eventos de sincronismo entre as *threads*.

A Figura 28 mostra que o papel da *thread* 1 (*vídeo*) é distribuir os quadros do vídeo para as demais *threads* que os processam. Isso é realizado através da variável "vídeo", objeto na qual a *thread* 1 é inicializada. A *thread 2* (*faces*) detecta novas instâncias putativas dos objetos e armazena as informações a elas relativas na variável "faces", onde estão contidas as coordenadas de cada instância, e escreve na variável "fc_model", onde estão armazenadas as novas instâncias a serem modeladas pela *thread 4* (*modela*). Esta última gera os modelos que serão utilizados pela *thread 5* (*avalia*) para atualizar a posição das faces rastreadas.

As coordenadas das instâncias de faces serão utilizadas pela *thread 3* (*fiduciais*) para extrair o segmento de imagem que contém uma imagem facial e calcular sua pose. As poses de todas as faces são armazenadas na variável "poses", que serve como índice de qualidade para a *thread 6* (*identifica*) que

estima a probabilidade de sucesso na identificação da face na imagem. Caso haja sucesso na identificação o nome do indivíduo é armazenado na variável "identidades" e a variável "reconhecido" correspondente assume o valor "verdadeiro". Em caso de identificação negativa a *flag* se mantém "falsa", permitindo que a *thread* 3 continue os cálculos de estimativa de pose. A *thread* 7 (*limpa*) é responsável por retirar das variáveis as informações referentes a instâncias duplicadas ou que não ocorrem no vídeo passado um tempo máximo pré-definido. Essa *thread* recebe sinais enviados via eventos para sincronizar a sua ação de exclusão de dados das instâncias na estrutura de armazenamento (conjunto formado por todas variáveis compartilhadas pelas *threads*), com a de atualização realizada pelas *threads*. Os sinais de controle e as variáveis diretamente acessadas pela *thread limpa* são mostradas na Figura 29.

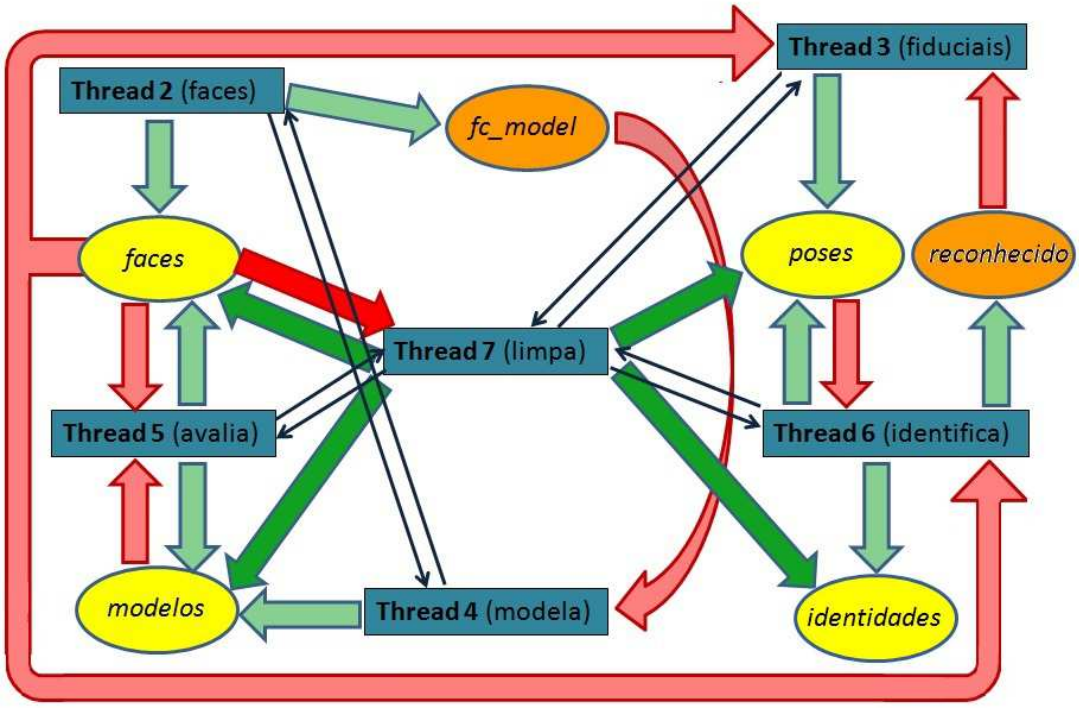


Figura 29: Sinais de controle utilizados pela *thread limpa*.

A *thread limpa* é responsável pela manutenção do sistema, impedindo que informações desnecessárias sejam mantidas desnecessariamente em memória, o que diminuiria o desempenho do programa ao longo do tempo. A informação relativa a uma instância, composta por um elemento de cada uma das variáveis, é acessada simultaneamente pelas *threads*, impedindo que sua remoção seja feita a

qualquer momento. É necessário manter a consistência das informações, pois a remoção de um elemento de uma variável compromete os dados correspondentes à mesma instância de face das demais variáveis. Um exemplo do problema é ilustrado na Figura 30.

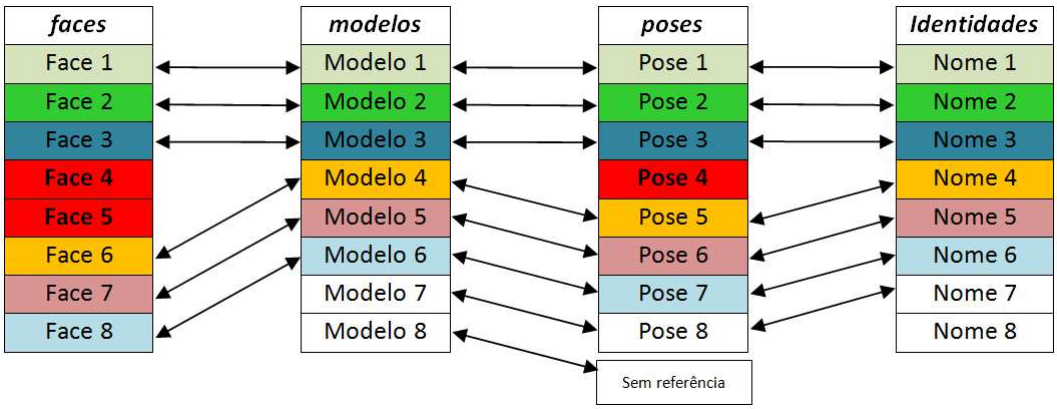


Figura 30: Problemas de sincronismo da estrutura de dados.

O método utilizado para busca de informações de uma instância é baseada nos índices dos elementos dentro das variáveis. Dessa forma, o segundo elemento de “faces” (primeira coluna na Figura 30) refere-se à mesma instância que o segundo elemento de “modelo”, “pose” e “identidade”.

Suponha que a sequência de remoção dos dados nas variáveis seja “faces”, “poses”, “modelos”, “identidades” e que em um determinado momento a configuração mostrada na Figura 30 ocorra. Nessa configuração os dados das “Faces 4 e 5” estão excluídos da variável “faces”, mas suas informações correspondentes nas outras variáveis ainda não foram removidas. Nesse caso as *threads* que buscam as informações da instância de uma face com “Face 6” iriam erroneamente obter o “Modelo 4”, “Pose 5” e “Nome 4”. Por esse motivo a leitura e a escrita simultânea das variáveis durante o momento de exclusão de uma instância devem ser impedidas com o uso de eventos de sincronismo.

Como discutido anteriormente, todas as operações das *threads* são realizadas simultaneamente, visando o aumento de desempenho do sistema de reconhecimento em vídeo. Nos subtópicos seguintes serão apresentados os algoritmos das *threads* citadas.

4.3. Descrição das Threads

4.3.1. vídeo

A *thread vídeo* concentra a captura e distribuição dos quadros do vídeo. Esta *thread* se faz necessária devido à existência de mais de uma linha de processamento que necessita do acesso a essas informações.

Um objeto de sincronização foi criado para permitir que somente uma *thread* por vez tenha acesso à variável onde é armazenado o quadro obtido pelo objeto que faz a captura da câmera de vídeo. Nessa variável, além do quadro bruto, são armazenados vários quadros pré-processados, que são necessários por algumas *threads* do sistema. O acesso às informações é exemplificado pela Figura 31. No laço desta *thread* é verificado inicialmente se o sistema ainda está em execução. Em caso positivo um quadro é capturado pelo dispositivo de entrada de vídeo e processado para ser disponibilizado para as outras *threads* através da variável "vídeo". O processamento do quadro fornece para as *threads* os quadros em formato de imagens integrais, imagens integrais ao quadrado, em cores e em preto e branco, suavizadas e refletidas com relação ao eixo y.

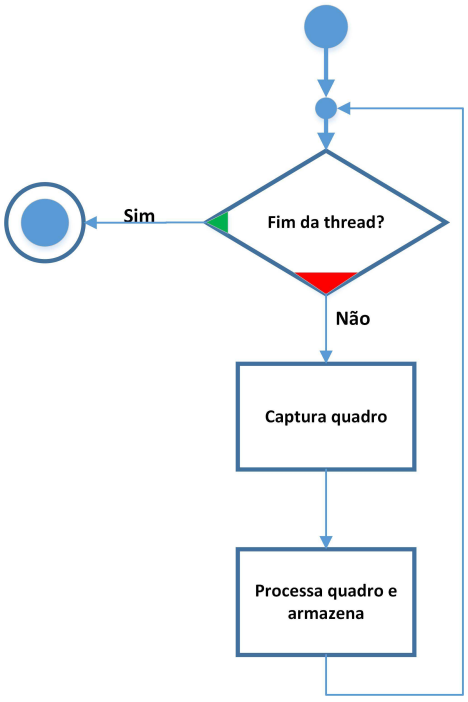


Figura 31: Algoritmo thread *vídeo*.

Esta *thread* é inicializada dentro da variável objeto “vídeo” no momento que esta é instanciada. Dentro dela são armazenados os quadros e as funções para obtenção do quadro e seu número de sequência.

4.3.2. faces

A *thread faces* tem o papel de detectar novas instâncias de faces a cada quadro capturado pela câmera. Ela faz uso das variáveis de “vídeo” (leitura) e “faces” (escrita), além dos eventos de comunicação com a *thread modela*. O algoritmo simplificado deste processo é mostrado na Figura 32.

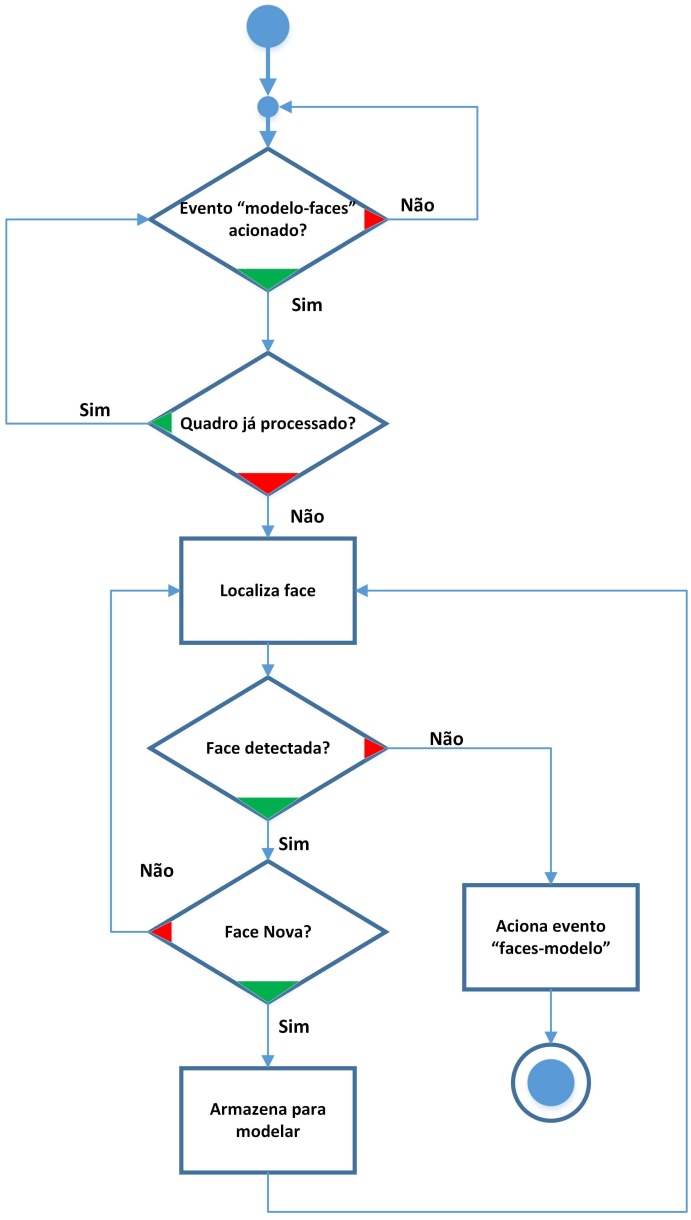


Figura 32: Algoritmo thread *faces*.

O laço dessa *thread* aguarda o acionamento do evento oriundo da *thread modela* antes de executar. Esse evento está previamente acionado no primeiro laço.

Com o evento acionado, o laço segue para a verificação do número de sequência do quadro. Esse número permanece armazenado na memória da *thread* enquanto esta estiver em execução, permitindo consultas posteriores. Caso o quadro possua o mesmo número do último quadro processado o laço reinicia e repete essa verificação até que ela retorne o valor “falso”. Encontrado um novo quadro procede-se à detecção de faces (bloco “Localiza face”), que se baseia no algoritmo de Viola-Jones. Caso uma face seja detectada, é executada a função de verificação de originalidade, que testa se a face detectada ainda não é uma das contidas na variável “faces”, em outras palavras, se não foi detectada anteriormente. Essa função baseia-se na área da sobreposição entre a região encontrada e as regiões das instâncias já encontradas e, portanto, registradas nessa variável. Se a sobreposição estiver acima de um limiar configurado, essa nova face é descartada, caso contrário é guardada numa variável provisória, a “fc_model”, que é compartilhada com a *thread modela*. Essa sequência de instruções é executada até que não seja mais encontrada nenhuma face, momento no qual é acionado o evento “faces_modela” para liberar o laço de execução da *thread modela*. Esse evento não é necessário para o funcionamento do sistema e visa somente ponderar o envio de novos dados para modelagem uma vez que a técnica de medida de originalidade de faces não é muito adequada para esse sistema, pois enquanto a *thread modela* está criando o modelo a ser usado pelo rastreador, a *thread faces* continua a considerar aquela instância previamente encontrado como novo alvo.

4.3.3. modela

Essa *thread* tem o objetivo de criar descrições de todas as instâncias novas de faces detectadas pela *thread faces.* Essas descrições são utilizadas pela *thread avalia* para realizar o rastreamento quadro a quadro. A *thread modela* faz uso das variáveis de “vídeo” (leitura), “faces” (leitura) e “modelos” (escrita), além dos

eventos de comunicação com a *thread faces*. O algoritmo simplificado é mostrado na Figura 33.

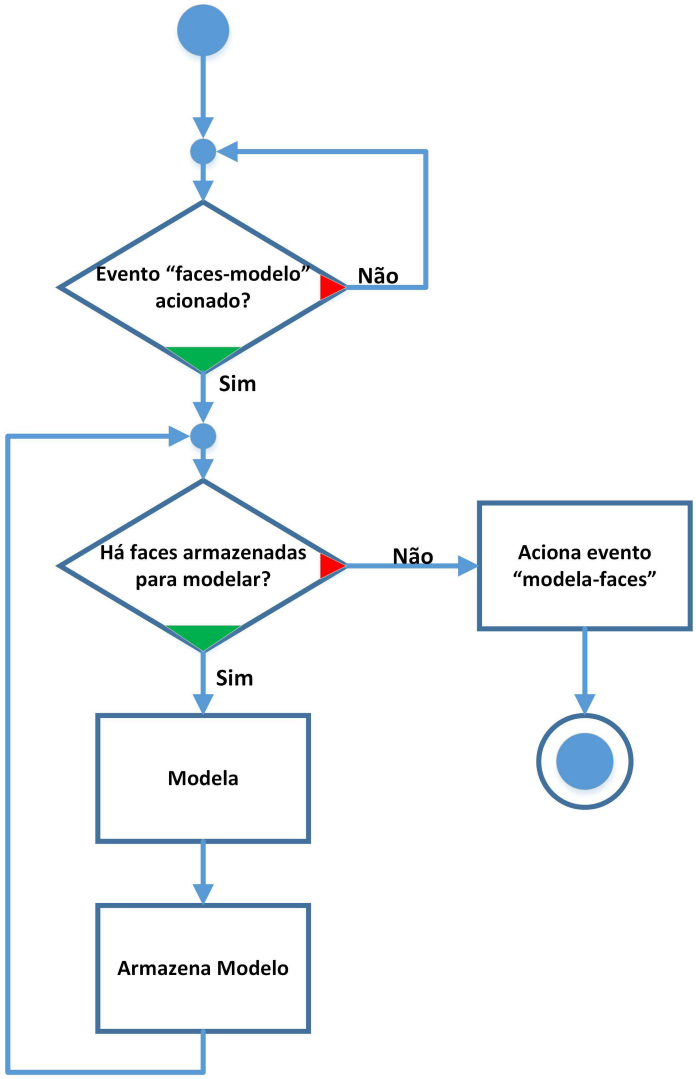


Figura 33: Algoritmo thread *modela*.

O laço dessa *thread* aguarda o acionamento do evento oriundo da *thread faces* antes de executar. Com o evento acionado o laço segue para a verificação da existência de faces para descrever. Caso haja alguma, são criados e inicializados os classificadores *fern* e NN do bloco “Detecção” do TLD correspondente à face modelada. O algoritmo é executado conforme apresentado no Capítulo 3 (seção 3.2.) e a descrição é armazenada na variável “modelos”. Esse laço é repetido até que todas as faces presentes na variável “fc_model” tenham sido processadas, momento no qual é acionado o evento “modela_faces” para solicitar à *thread faces* novas instâncias de faces para serem descritas.

4.3.4. avalia

A *thread avalia* é responsável pelo rastreamento dos indivíduos na sequência de vídeo. Ela realiza os algoritmos necessários para estimar as novas coordenadas dos elementos registrados na variável “faces” *e* treinar o modelo através da coleta de novos dados de treinamento para o detector e rastreador do modelo de cada instância. Ela faz uso das variáveis ”vídeo” (leitura), “faces” (leitura e escrita) e “modelos” (leitura e escrita), além dos eventos de comunicação com a *thread limpa*. Uma representação simplificada do algoritmo é apresentada na Figura 34.

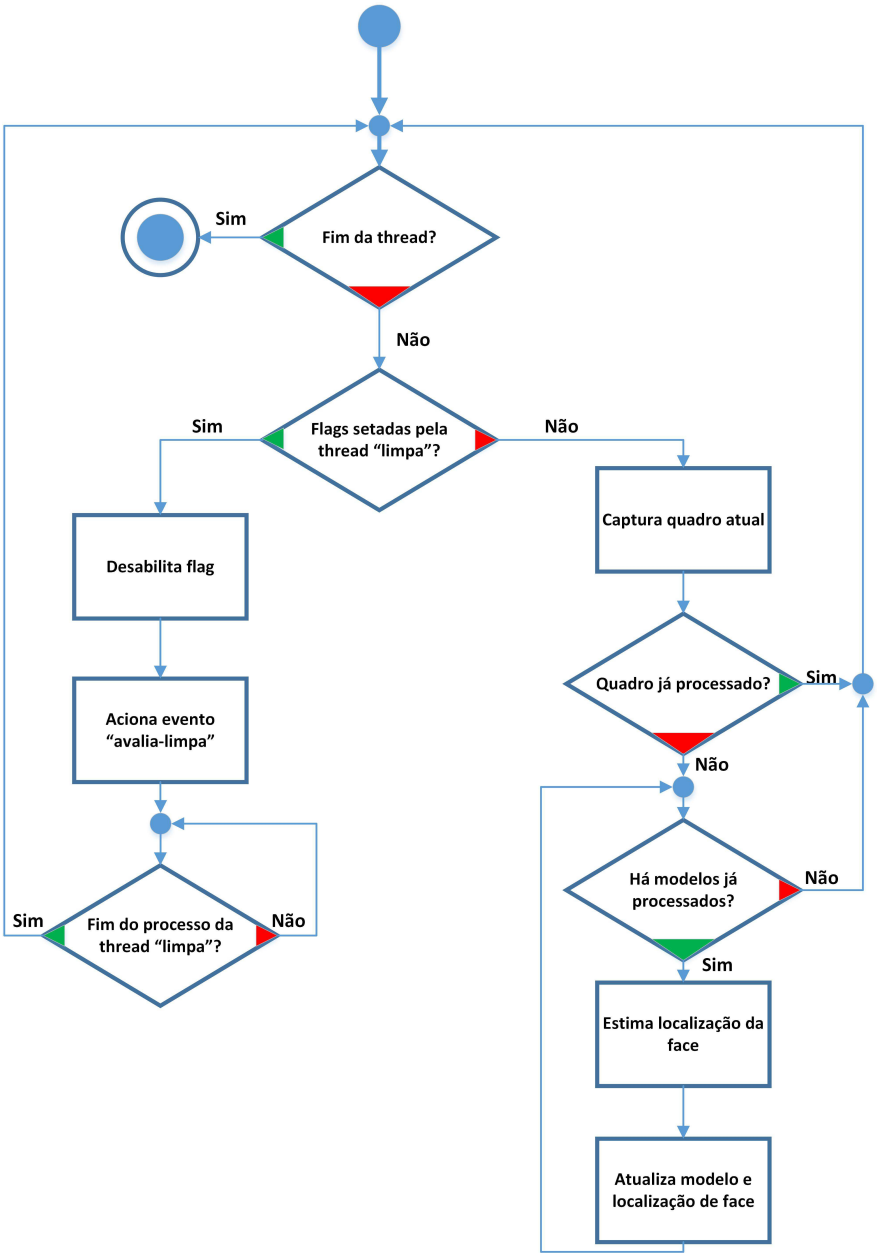


Figura 34: Algoritmo thread *avalia*.

Observe que antes de iniciar o laço de processamento é verificado se a *flag* de finalização da *thread* foi ativada. Em caso negativo é verificada se a *thread limpa* acionou a *flag* utilizada para sinalizar um procedimento de exclusão de dados. Se a *flag* receber o valor "verdadeiro" a *thread* aciona o evento "avalia_limpa" e aguarda o acionamento do evento "limpa_avalia" pela *thread limpa*. Se a *flag* permanecer com o valor "falso" é acionada a função de captura de quadro, que retorna também o número de sequência do mesmo.

Caso o quadro possua o mesmo número do último processado o laço reinicia, senão, o laço segue para verificação de existência de modelos a serem avaliados. Para cada modelo é estimada a correspondente nova localização da face através da aplicação do algoritmo descrito no Capítulo 3 (seção 3.2.2.). Vale lembrar que a próxima localização é dependente da anterior. Concluída esta etapa o modelo é treinado através da aplicação do algoritmo descrito no Capítulo 3 (seção 3.2.2.4.) e a nova posição do alvo é armazenada na variável “faces”. Ao fim desse processo um novo quadro é capturado e são realizados os cálculos para essa nova imagem.

4.3.5.
fiduciais

Um nome mais sugestivo para essa *thread* seria *qualidade*, uma vez que é realizada uma medida de qualidade da imagem facial com o intuito de selecionar as instâncias de faces presentes no vídeo que apresentem qualidade adequada para identificação. A única medida de qualidade calculada é a pose da face no quadro, avaliada empiricamente através do ângulo formado pelas retas formada pelos pontos dos olhos e nariz detectados pelo algoritmo Viola-Jones apresentado no Capitulo 3 (seção 3.1.). Entretanto a estrutura da *thread* está pronta para receber algoritmos de medida de qualidade mais sofisticados. O resultado do cálculo é armazenado na variável “pose” que é posteriormente usada pela *thread identifica* para selecionar quais indivíduos o sistema tentará identificar. Essa *thread* faz uso das variáveis de “vídeo” (leitura), “faces” (leitura), “identidades” (leitura) e “poses” (escrita), além dos eventos de comunicação com a *thread limpa*. O algoritmo simplificado é mostrado na Figura 35.

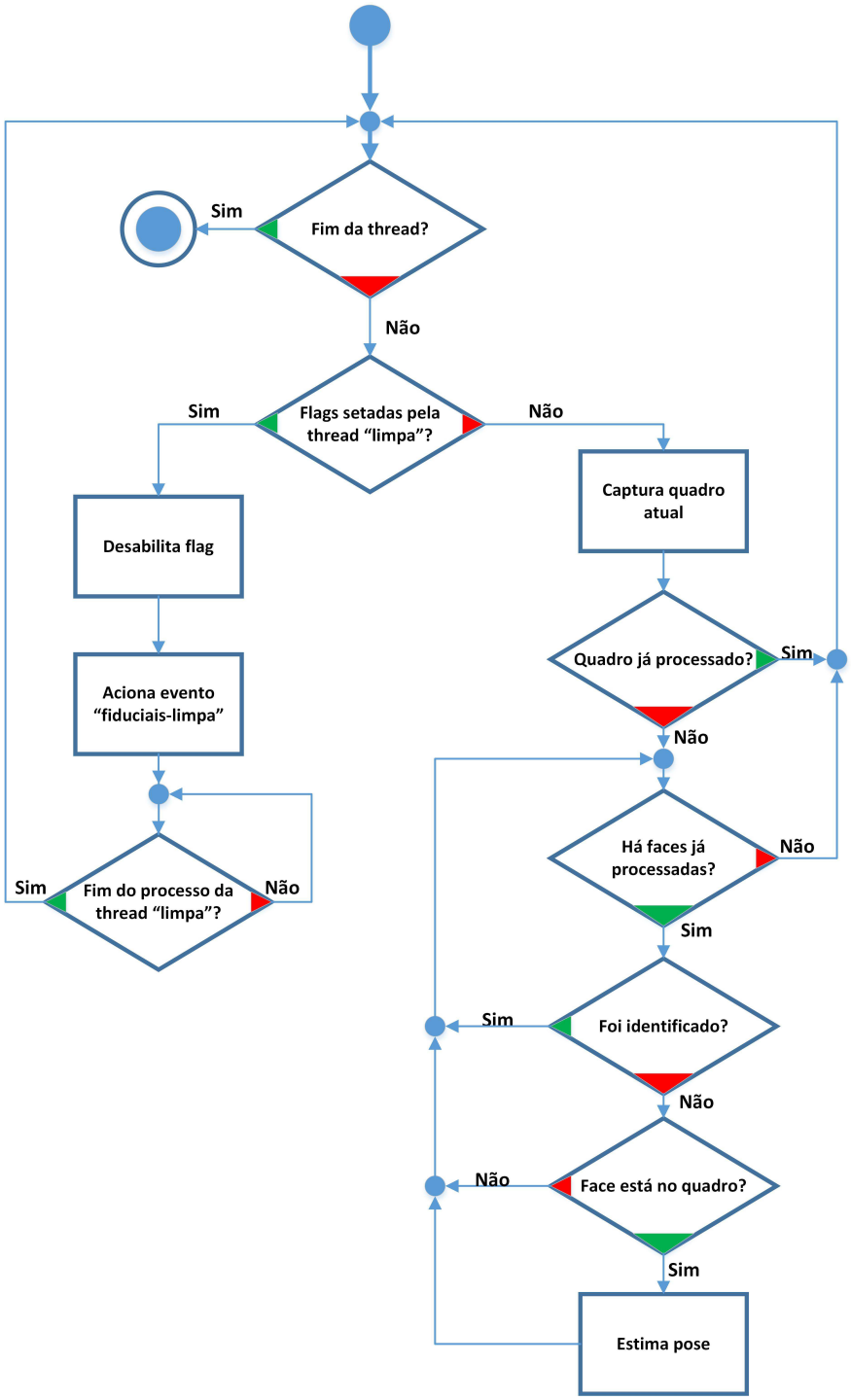


Figura 35: Algoritmo thread *fiduciais*.

Nota-se na Figura 35 que antes de iniciar o laço de processamento é verificado se a *flag* de finalização da *thread* assumiu o valor "verdadeiro", em caso negativo é verificada se a *thread limpa* acionou a *flag* utilizada para sinalizar um procedimento de exclusão de dados. Caso a *flag* receba o valor "verdadeiro" a *thread* aciona o evento "fiduciais_limpa" e aguarda o acionamento do evento

“limpa_fiduciais” pela *thread limpa*. Caso contrário, se a *flag* permanecer com o valor “falso” é acionada a função de captura de quadro, que retorna também o número de sequência do mesmo.

Se o quadro possuir o mesmo número do último processado o laço reinicia. Caso contrário o laço segue para verificação de existência de faces a serem avaliadas na variável “faces”. Para cada instância de face verifica-se se foi identificada. Caso esteja identificada o laço segue para verificar a próxima. No caso de sua identidade não estiver feita é avaliado se a face está atualmente no quadro do vídeo (através da informação armazenada na variável “faces” pela *thread avalia*), pois existe a possibilidade de oclusão por objeto ou de o alvo ter se retirado da área de vigilância da câmera.

Se todas as condições anteriores forem satisfeitas e o individuo estiver presente no quadro a sua pose é calculada e armazenada na variável “poses”. Ao fim desse processo um novo quadro é capturado e são realizados os cálculos para essa nova imagem.

4.3.6. identifica

Esta *thread* é responsável pela identificação dos indivíduos na sequência de vídeo. Um único algoritmo de identificação é utilizado nesse sistema, no entanto a estrutura da *thread* está pronta para receber uma ampla gama de algoritmos de reconhecimento facial. O resultado desta *thread* é armazenado na variável “identidades” que é usada pela *thread principal* para exibição na interface gráfica. Essa *thread* faz uso das variáveis de ”vídeo” (leitura), “faces” (leitura), “identidades” (escrita) e “poses” (leitura e escrita), além dos eventos de comunicação com a *thread limpa*. O algoritmo simplificado é mostrado na Figura 36.

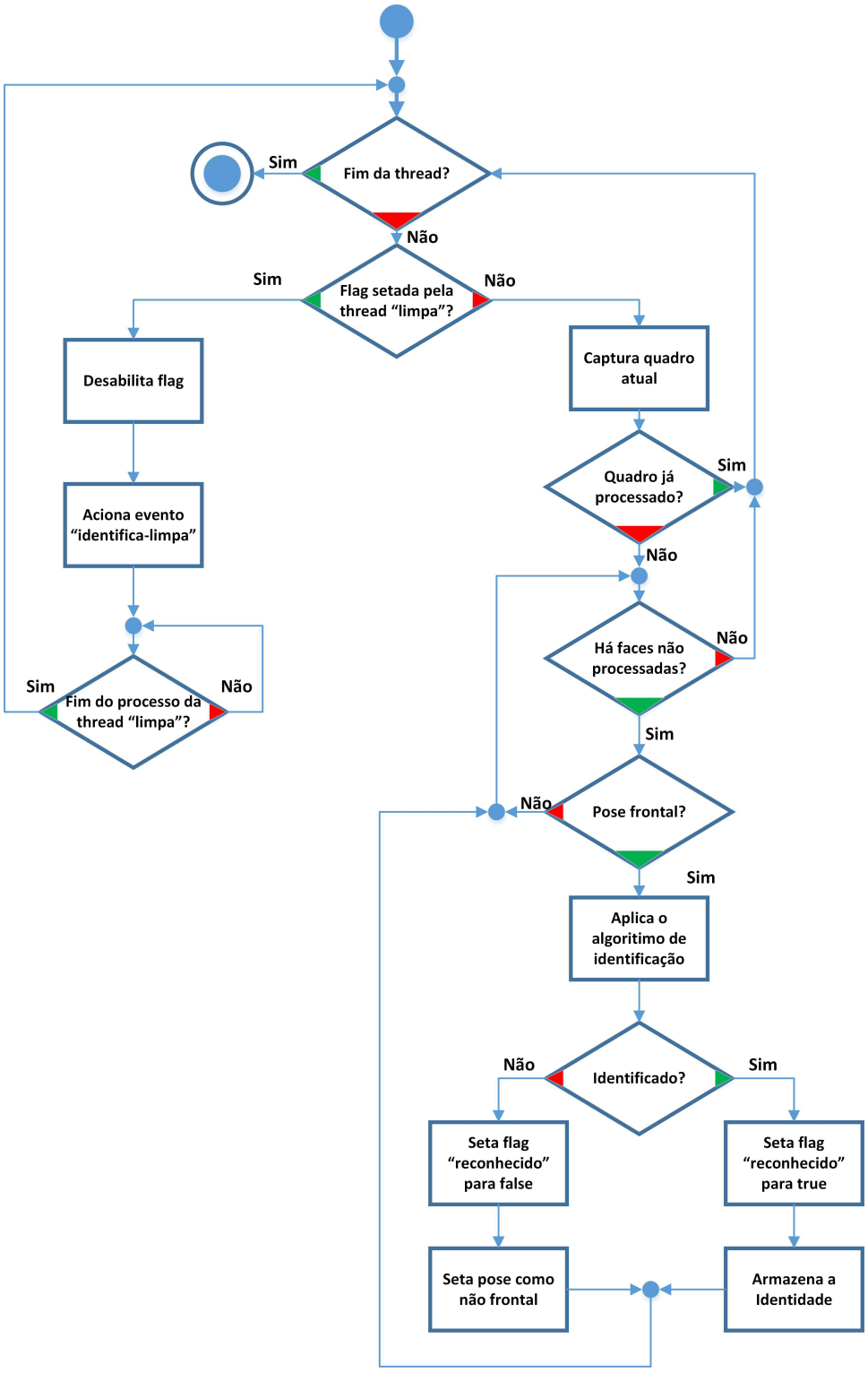


Figura 36: Algoritmo thread *identifica*.

A Figura 36 mostra que antes de iniciar o laço de processamento é verificado se a *flag* de finalização da *thread* assumiu o valor "verdadeiro", em caso negativo é verificada se a *thread limpa* acionou a *flag* utilizada para sinalizar

um procedimento de exclusão de dados. Se a *flag* for setada *true* a *thread* aciona o evento “identifica_limpa” e aguarda o acionamento do evento “limpa_identifica” pela *thread limpa.* Se a *flag* permanecer com o valor “falso” é acionada a função de captura de quadro, que retorna também o número de sequência do mesmo. Caso o quadro possua o mesmo número do último processado o laço reinicia, senão o laço segue para verificação de existência de instâncias de faces a serem avaliadas. Para cada instância de face é verificado se sua pose é frontal, segundo a avaliação conduzida pela *thread fiduciais*. Em caso afirmativo o algoritmo de identificação (apresentado na seção 3.4. do Capítulo 3) é aplicado sobre o a imagem da face extraída do quadro do vídeo.

Nesse ponto da *thread* podem ser inseridos outros algoritmos de reconhecimento facial. Caso o indivíduo tenha sido identificado uma *flag* dentro estrutura da variável “identifica” correspondente ao indivíduo recebe o valor “verdadeiro”. Feito isso o nome obtido da base de dados do reconhecedor facial é armazenado na variável identifica e a pose correspondente na variável “poses” recebe o valor “falso”. Essa última operação impede que seja tentada uma nova identificação da face antes que a *thread fiduciais* reavalie a pose.

Caso a face não tenha sido identificada a *flag* dentro estrutura da variável “identifica” correspondente ao individuo é setada para *false*, bem como sua “pose”. Ao fim desse processo um novo quadro é capturado e são realizados os cálculos para essa nova imagem.

4.3.7. limpa

Essa *thread* tem a função de excluir todos os dados que não são mais úteis ao sistema. Dentre os dados inúteis estão os que se referem aos indivíduos que estão fora de cena por um longo período e os dados duplicados decorrentes de falhas no processo de inclusão de novas instâncias faciais.

Para solucionar o primeiro problema são usados contadores de existência para cada um dos rastreamentos das instâncias, que ao excederem dado limite provocam a atribuição do valor “verdadeiro” para uma *flag* que força sua exclusão. Os limiares para esses contadores são configurados levando em consideração o ambiente em que o dispositivo de vídeo opera, pois dependendo do

campo de visão capturado por ele, o tempo médio de permanência de um indivíduo na cena varia. O segundo problema - dados duplicados devido a falhas no processo de inclusão de novos indivíduos - é resolvido simplesmente calculando a área de interseção de cada região rastreada com as outras: áreas de interseção acima de um limiar configurado indicam alvos duplicados.

Para realização de sua tarefa essa *thread* faz uso de todas variáveis para escrita e da variável “faces” para leitura, além dos eventos de comunicação com as outras *threads* do sistema. Uma ilustração simplificada do algoritmo é mostrada na Figura 37.

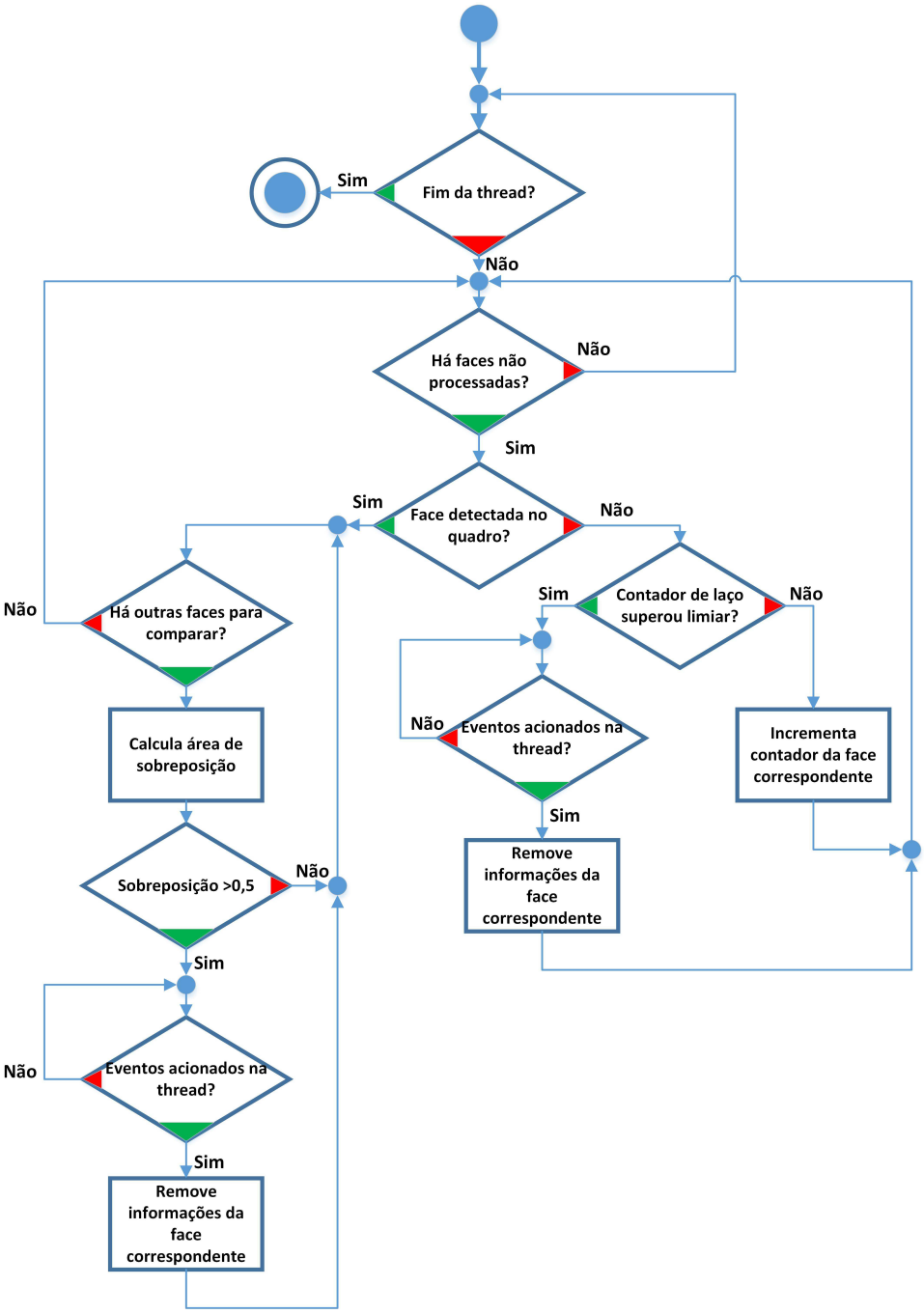


Figura 37: Algoritmo thread *limpa*.

Da mesma maneira que nas outras *threads*, antes de iniciar o laço de processamento é verificado se a *flag* de finalização da *thread* foi ativada, em caso negativo é avaliado se existem dados de indivíduos nas variáveis. Essa informação é obtida pela leitura do número de elementos presentes na variável “faces”. Caso o indivíduo não tenha sua face detectada no quadro, é verificado se o seu contador superou um limiar configurado. Esse contador é sempre incrementado quando é percorrido um laço sem que o indivíduo seja detectado no quadro (pela *thread*

avalia) e zerado assim que ele é novamente encontrado. Se o contador superar o valor máximo pré-definido os dados referentes a esse indivíduo devem ser excluídos. No entanto, primeiro é feita uma solicitação pela *thread limpa* às *threads* que fazem uso das variáveis para evitar erros de referência.

No caso da instância facial avaliada estar presente no quadro, os dados desse indivíduo poderão ainda ser excluídos se estiverem duplicados. O critério para seleção de quais dados devem ser excluídos é o de que os mais antigos devem ser preservados, uma vez que foram treinados por mais tempo e portanto possuem melhores representações do alvo.

Similarmente ao que ocorre para faces não detectadas nos quadros, a exclusão das informações deve ser precedida de uma solicitação pela *thread limpa*, usando os sinais de controle, às *thread* que fazem uso das variáveis da estrutura de dados.

4.3.8. principal

Na *thread principal* são criadas e inicializadas todas as *threads* descritas anteriormente. A inicialização consiste em passar um ponteiro de uma estrutura que contém todos os objetos que serão compartilhados (eventos e variáveis) para as *threads* e disparar o laço de processamento de cada uma.

A função principal desta *thread* é a captura e exibição dos resultados das outras threads, sua finalização correta e desalocamento de memória utilizado.

5
AVALIAÇÃO DE DESEMPENHO

Como exposto no capítulo introdutório, esta dissertação visa propor uma arquitetura paralela para o reconhecimento facial em vídeo. A base de dados, que nos casos práticos mais desafiadores, é muito grande, deve ser consultada somente se uma imagem facial com qualidade suficiente para que o reconhecimento for encontrada.

Neste ponto, é importante salientar que ao longo de todo o processo de pesquisa bibliográfica não foi encontrada na literatura, até o presente momento, uma proposta alternativa, impossibilitando uma comparação direta de desempenho. De qualquer modo, uma avaliação do sistema completo em termos de taxa de reconhecimento foge ao escopo deste trabalho, uma vez que algumas das *threads* implementadas no protótipo tiveram apenas o propósito de compor um sistema completo, sem maiores preocupações quanto ao desempenho. Desta forma, a análise experimental descrita a seguir limitou-se à avaliação de estresse, ou seja, da carga computacional associada, de modo a identificar potenciais gargalos que evidenciem os benefícios da presente abordagem relativamente a implementações que não fazem uma avaliação prévia da qualidade de imagem facial antes de submetê-la ao algoritmo de reconhecimento.

Os testes foram realizados num notebook com CPU Intel Core I7-2630QM e 4GB de memória RAM e fizeram uso de um vídeo público disponibilizado para avaliação de sistemas em vídeo apresentados na conferência internacional AVSS (*Advanced Video and Signal based Surveillance*), mais especificamente o “motinas_multi_face”, que conta com vários indivíduos com o intuito de mostrar a capacidade de processamento de múltiplos alvos paralelamente no sistema. Um quadro do vídeo é mostrado na Figura 38.

Figura 38: AVSS motinas_multi_face.

Foram realizadas as seguintes medidas:

- Tempo de processamento do sistema contendo somente o identificador facial.
- Tempo de processamento usando *multithread* (as informações de todas *threads* são condensadas num só gráfico, considerando sempre o tempo de processamento da *thread* mais lenta).

Cada uma dessas medidas foi realizada com 3 bases de dados de tamanhos diferentes do algoritmo de reconhecimento facial, com o intuito de mostrar a variação do tempo de processamento – e consequentemente o número de quadros/s – devido à aplicação desse algoritmo no vídeo. As bases possuem os tamanhos de 50, 350 e 500 indivíduos. O resultado da primeira medida é mostrado nas Figuras 39 a 44.

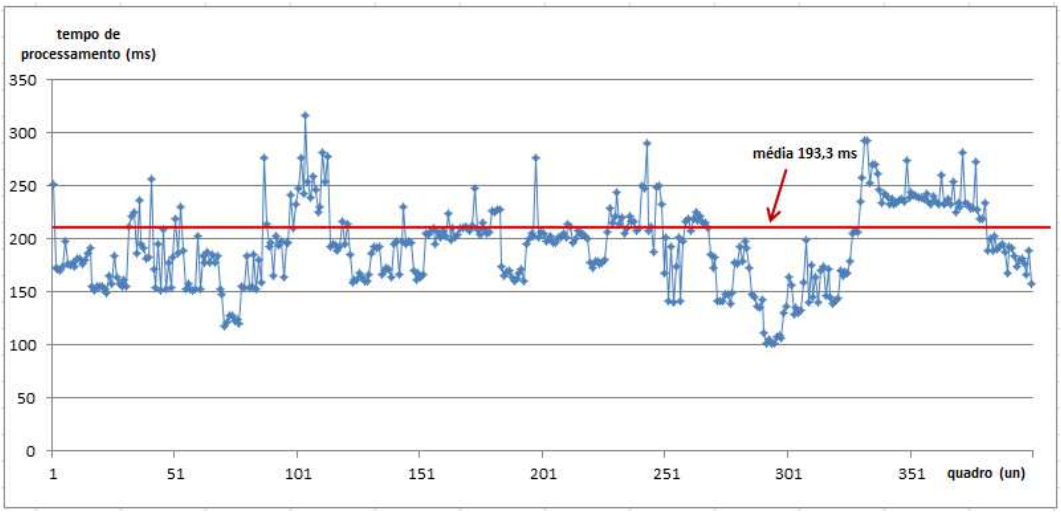


Figura 39: Variação do tempo de processamento para uma base de 50 indivíduos no sistema contendo somente o identificador facial.

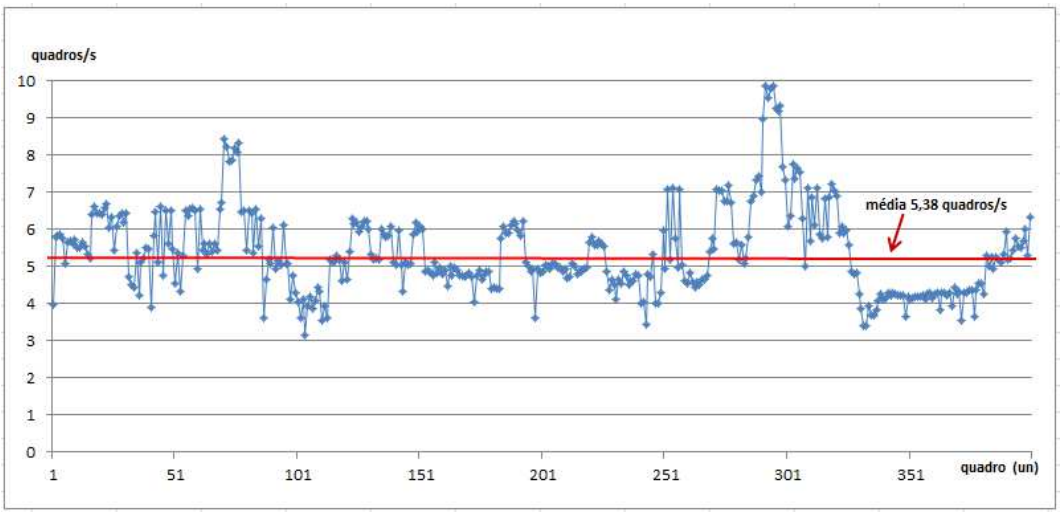


Figura 40: Número de quadros por segundo para uma base de 50 indivíduos no sistema contendo somente o identificador facial.

Nas Figura 40 é mostrado o número de quadros por segundo num sistema baseado somente no identificador facial, que alcança uma taxa de processamento de 5,38 quadros/s em média numa base de 50 indivíduos.

Para uma base de 350 indivíduos esse mesmo sistema alcança uma taxa de 4,08 quadros/s, que equivale a um tempo médio de processamento de 258,96 ms, como mostrado nas Figuras 41 e 42.

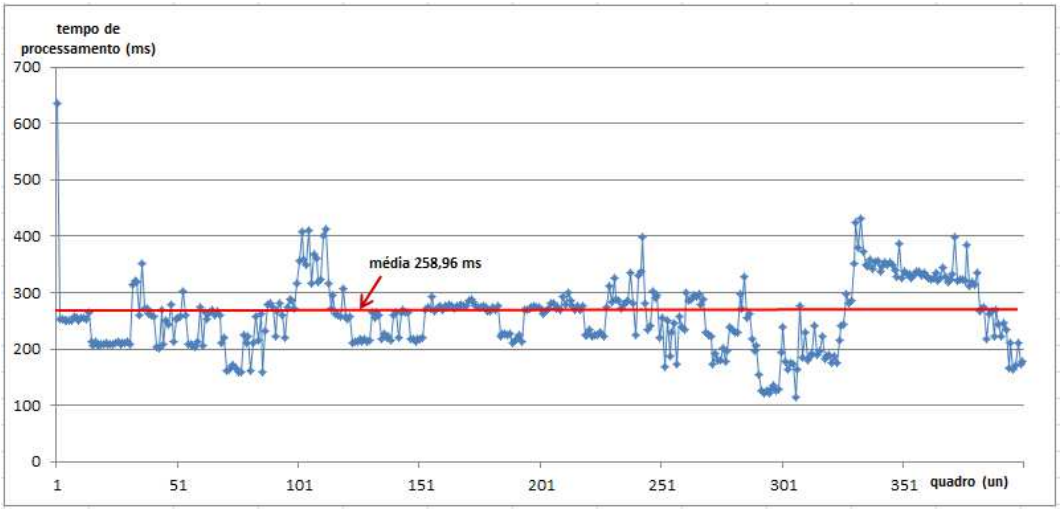


Figura 41: Variação do tempo de processamento para uma base de 350 indivíduos no sistema contendo somente o identificador facial.

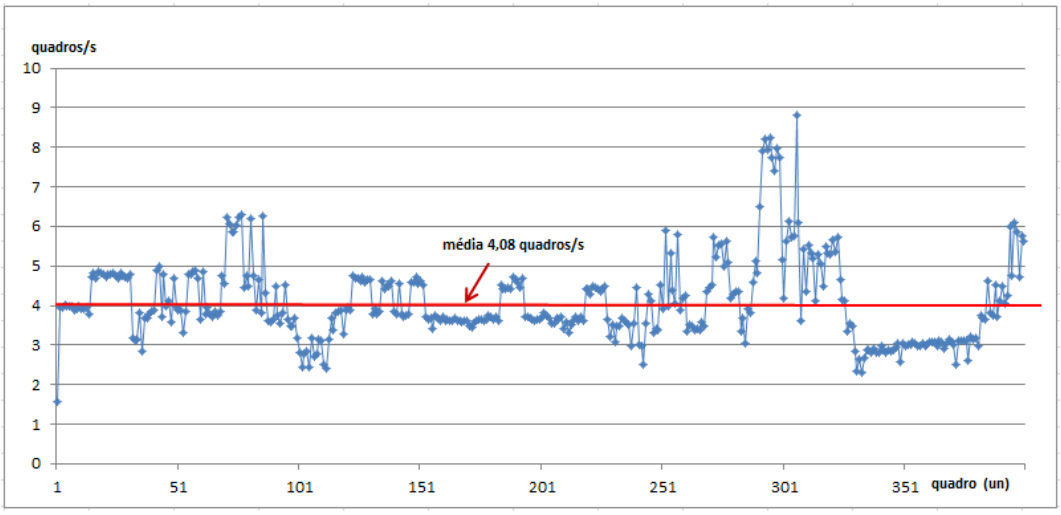


Figura 42: Número de quadros por segundo para uma base de 350 indivíduos no sistema contendo somente o identificador facial.

Numa base de 500 indivíduos o sistema alcança uma taxa de 3,54 quadros/s, que equivale a um tempo médio de processamento de 303,43 ms, como mostrado nas Figuras 43 e 44.

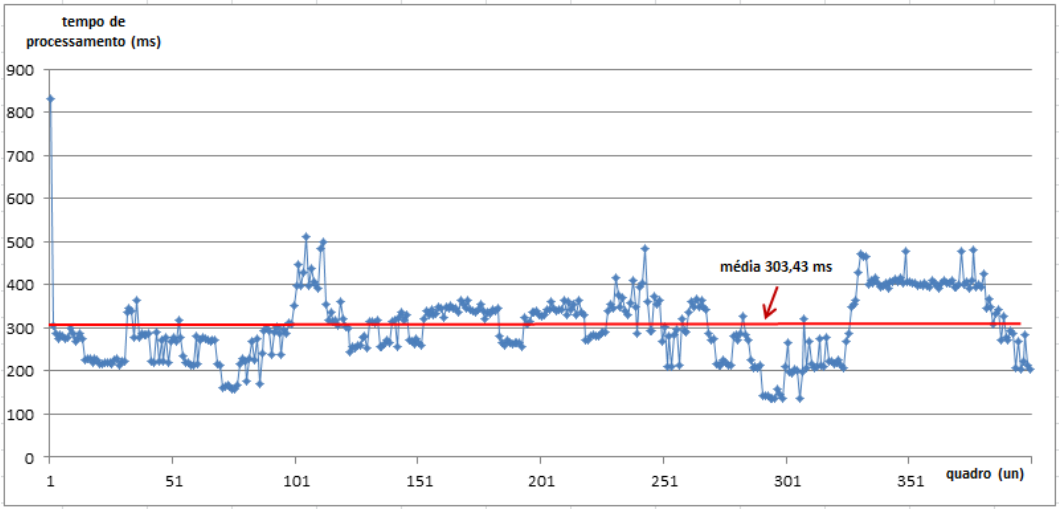


Figura 43: Variação do tempo de processamento para uma base de 500 indivíduos no sistema contendo somente o identificador facial.

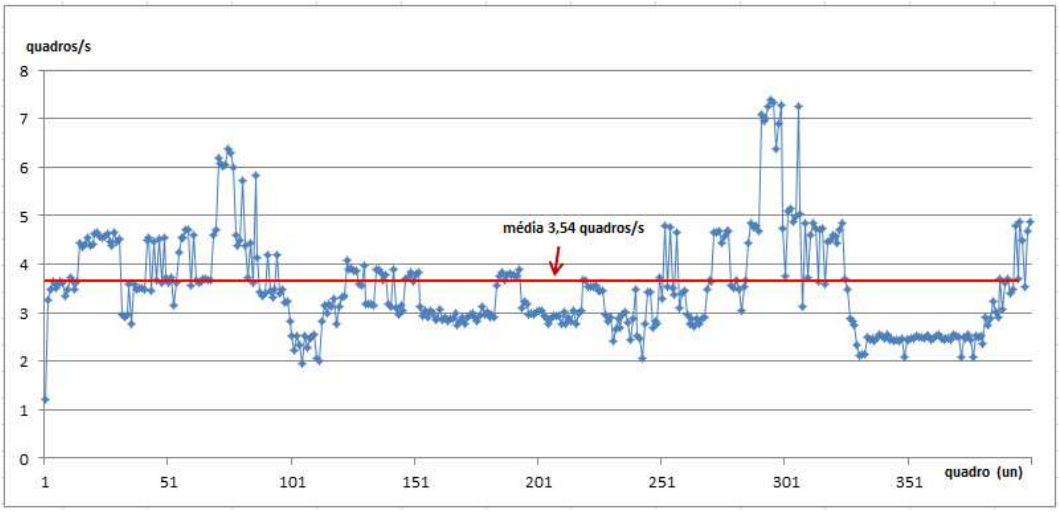


Figura 44: Número de quadros por segundo para uma base de 500 indivíduos no sistema contendo somente o identificador facial.

Nas Figura 46 é mostrado o número de quadros por segundo num sistema baseado em *multithread* com seleção de alvos com maior probabilidade de serem identificados, que alcança uma processamento de 8,04 quadros/s em média numa base de 50 indivíduos.

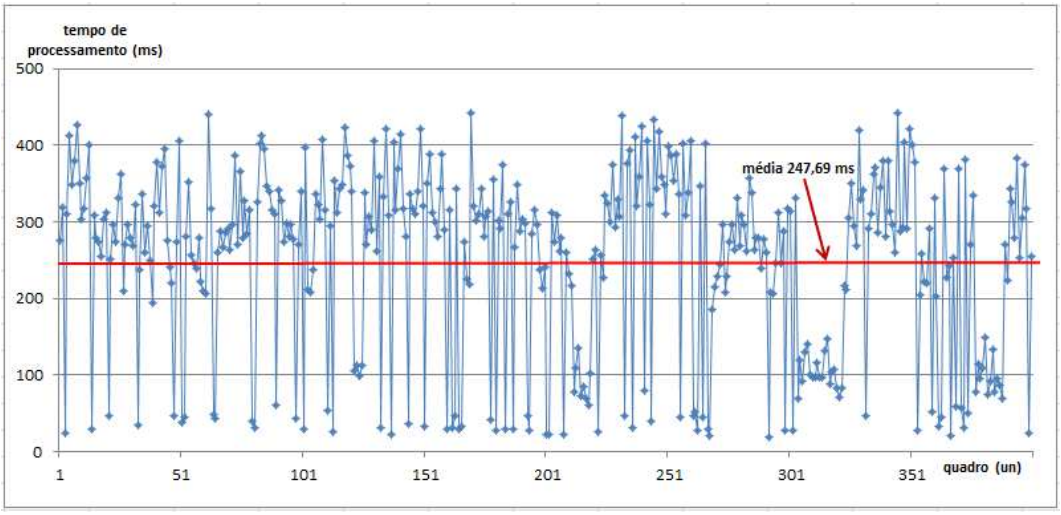


Figura 45: Variação do tempo de processamento para uma base de 50 indivíduos no sistema *multithread*.

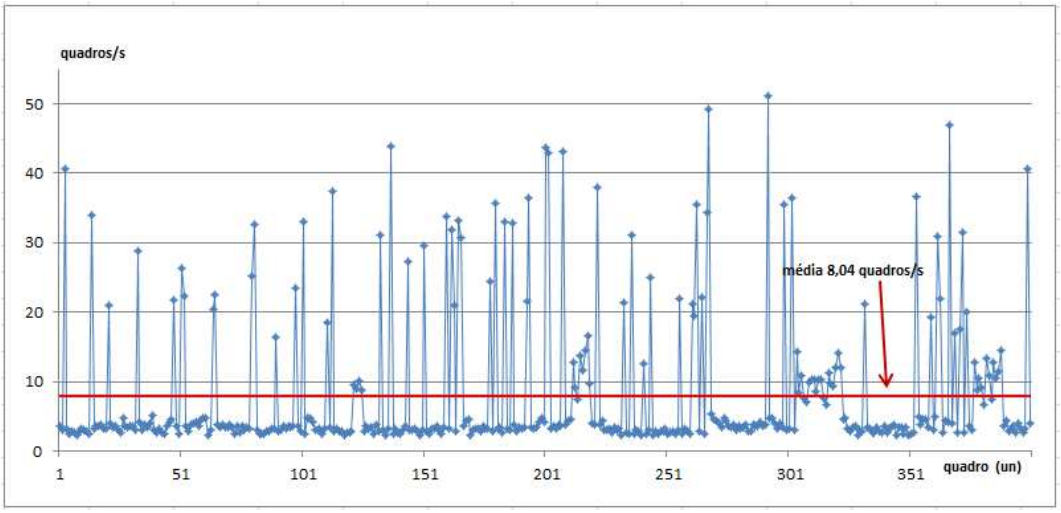


Figura 46: Número de quadros por segundo para uma base de 50 indivíduos no sistema *multithread*.

Para uma base de 350 indivíduos esse mesmo sistema alcança uma taxa de 7,77 quadros/s, que equivale a um tempo médio de processamento de 239,09 ms, como mostrado nas Figuras 47 e 48.

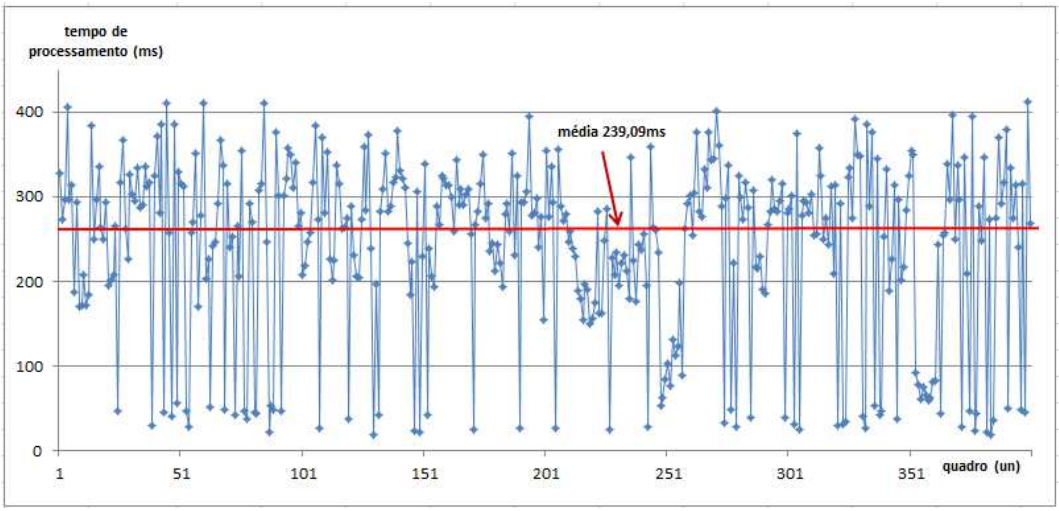


Figura 47: Variação do tempo de processamento para uma base de 350 indivíduos no sistema *multithread*.

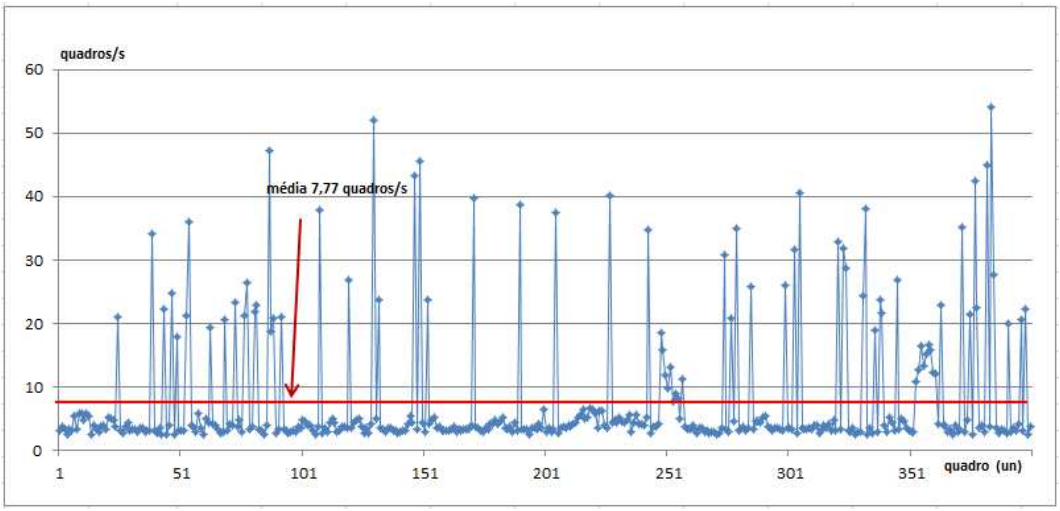


Figura 48: Número de quadros por segundo para uma base de 350 indivíduos no sistema *multithread*.

Numa base de 500 indivíduos o sistema alcança uma taxa de 7,47 quadros/s, que equivale a um tempo médio de processamento de 242,07 ms, como mostrado nas Figuras 49 e 50.

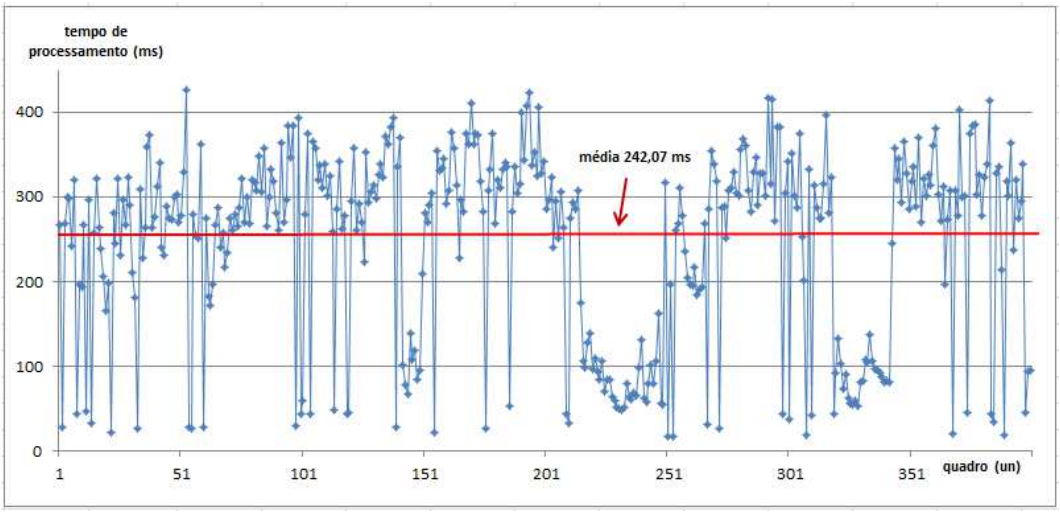


Figura 49: Variação do tempo de processamento para uma base de 500 indivíduos no sistema *multithread*.

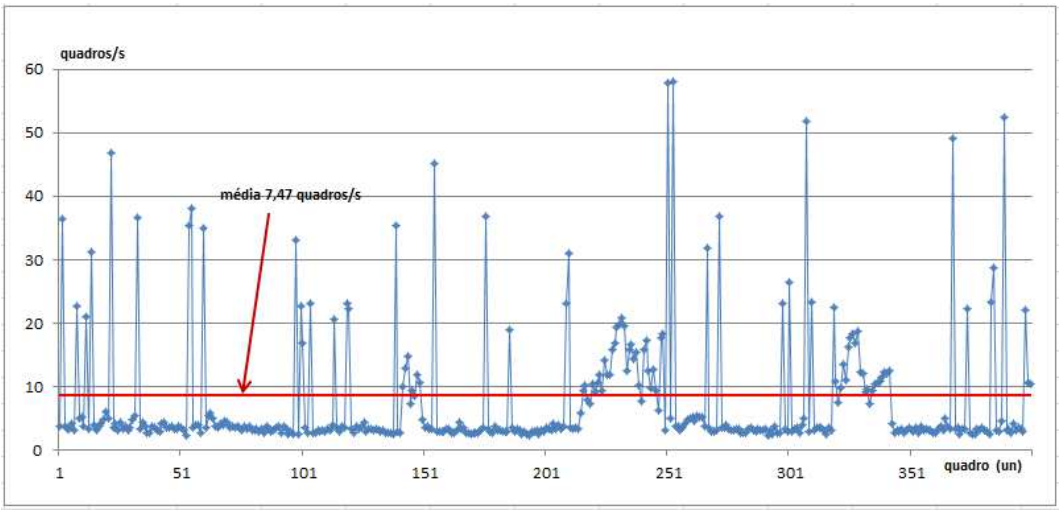


Figura 50: Número de quadros por segundo para uma base de 500 indivíduos no sistema *multithread*.

A Figura 51 mostra um quadro do vídeo processado pelo protótipo, onde os quatro indivíduos são rastreados após a identificação.

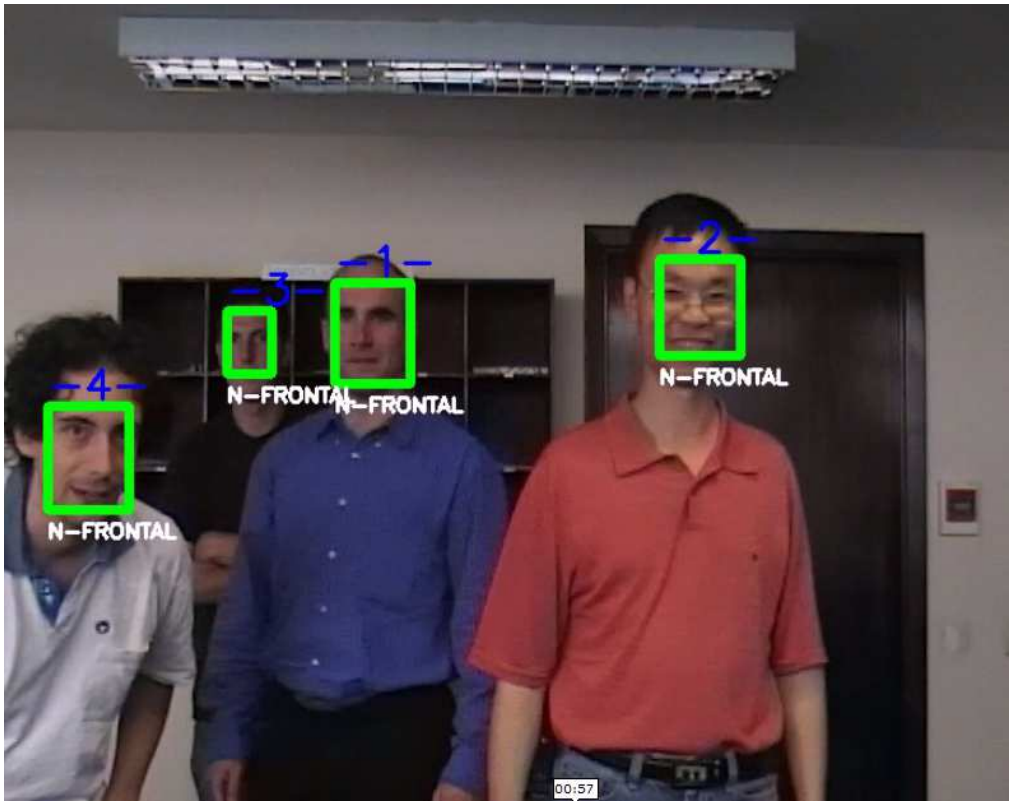


Figura 51: Vídeo processado.

Melhores resultados podem ser alcançados com o ajuste adequado dos parâmetros do sistema, principalmente nos módulos de detecção de face e rastreamento, que são os algoritmos mais executados ao longo de todo o processo.

Na Figura 52 é mostrado o número de quadros por segundo ao longo de todo o processamento do vídeo na *thread avalia*. Nesse último teste foi limitado o número de quadros entregues pela *thread* de vídeo em 20 quadros/s de modo a simular um dispositivo de captura de vídeo com taxa de captura máxima.

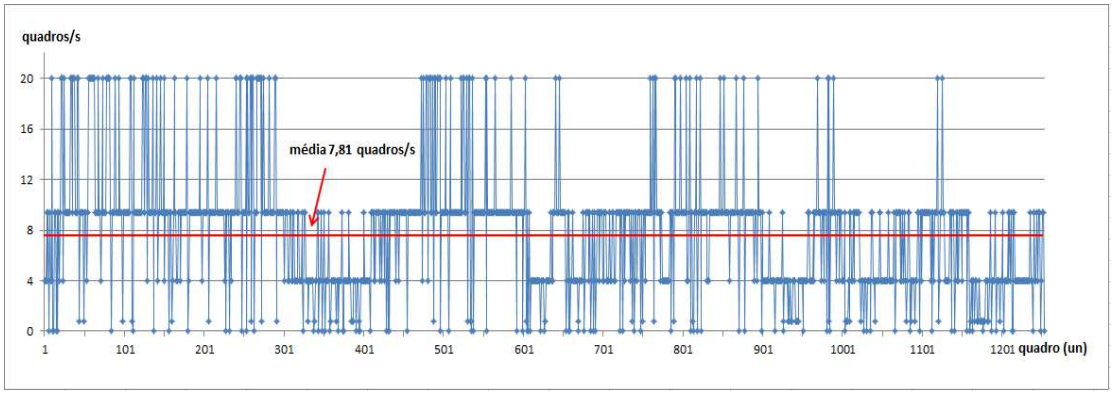


Figura 52: Taxa de quadros ao longo de toda a execução da *thread avalia*.

A Figura 53, por sua vez, monstra a taxa média de quadros por segundo na *thread faces*. A velocidade dessa *thread* depende muito da precisão ajustada no algoritmo de detecção de faces. Quanto maior a precisão, a tendência é que a taxa média de quadros por segundo diminua.

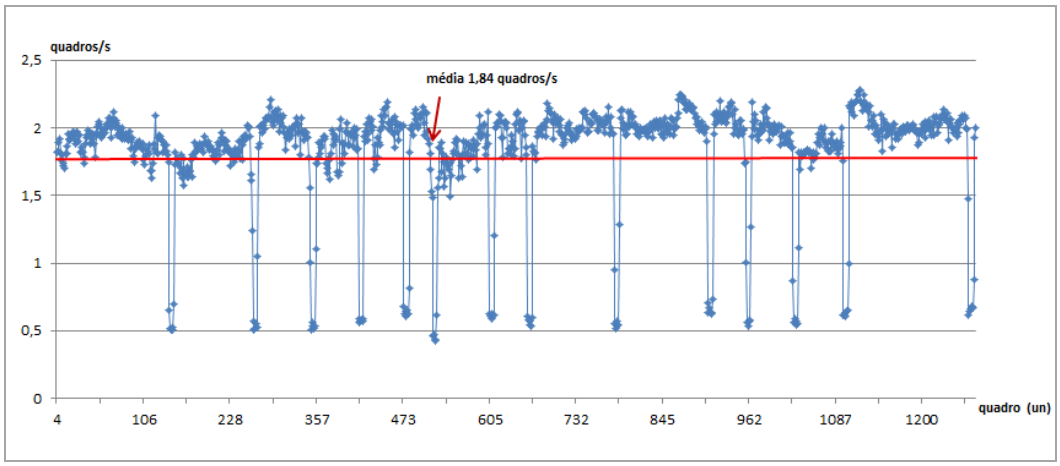


Figura 53: Taxa de quadros ao longo de toda a execução da *thread faces*.

A Figura 54 e 55 mostram o momento em que as *threads fiduciais* e *identifica* são dispensadas do processamento dos quadros, uma vez que a *thread identifica* realizou com sucesso a identificação dos indivíduos presentes no vídeo.

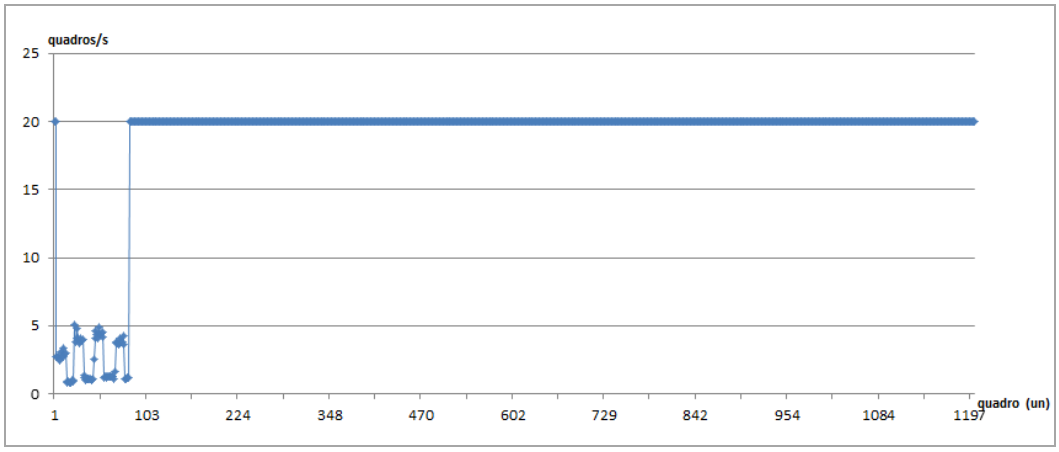


Figura 54: Taxa de quadros ao longo de toda a execução da *thread fiduciais*.

Sempre que a *thread fiduciais* avalia a imagem de uma face como frontal a *thread identifica* tenta identifica-la. Quando feito com sucesso, a *thread fiduciais* deixa de avaliar a pose dessa face. Se todas as faces avaliadas como frontais forem

identificadas, tanto a *thread fiduciais* quanto a *thread identifica* não realizam mais cálculos, visto que isso só é feito enquanto houver faces não identificadas.

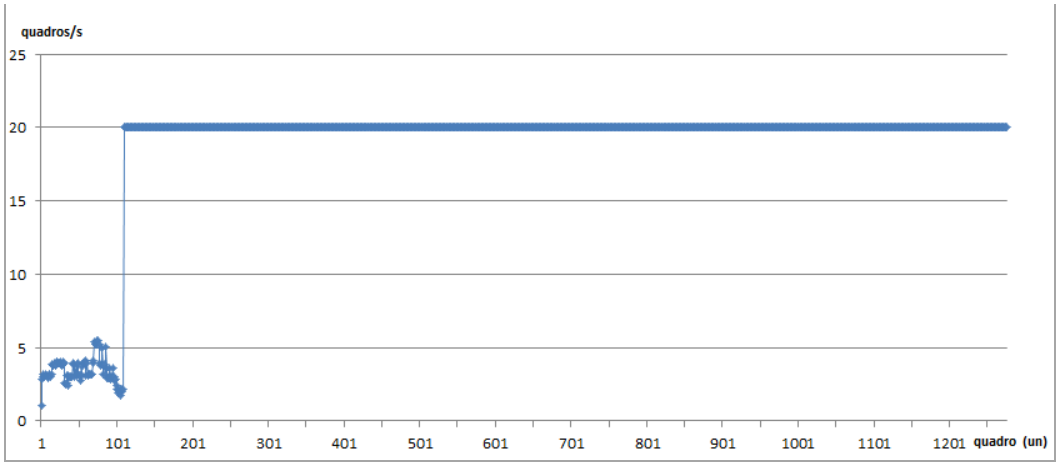


Figura 55: Taxa de quadros ao longo de toda a execução da *thread identifica*.

6
CONCLUSÕES E TRABALHOS FUTUROS

O protótipo do sistema de reconhecimento em vídeo foi implementado com sucesso. Fez-se uso do algoritmo Viola-Jones para detecção do objeto de interesse, no caso a face do individuo, e de uma implementação do algoritmo de rastreamento TLD (*Tracking-Learning-Detector*), adaptado para operar num sistema *multithread* para rastreamento de vários alvos. A informação fornecida por esses dois módulos é a localização do objeto de interesse ao longo do vídeo. Essa localização serve de base para a aplicação de algoritmos de avaliação de qualidade de imagem (pose, iluminação, foco, contraste, etc) que seria utilizado para selecionar quais objetos (faces) tem maior perspectiva de sucesso numa identificação por um determinado algoritmo de reconhecimento. No presente protótipo o único parâmetro de qualidade de imagem avaliado é a pose da face. Esse parâmetro é o mínimo necessário para que seja possível a aplicação algoritmo de reconhecimento, visto que se baseia em imagens faciais frontais. A técnica para a estimativa da pose é bem simples, consistindo na localização de alguns pontos fiduciais (olhos e nariz) e verificação de sua posição relativa.

Durante a avaliação de desempenho fica evidente que o tempo de processamento aumenta à medida que a base de dados que contém indivíduos aumenta. No sistema baseado somente o identificador facial, onde não é feita uma seleção dos alvos com qualidade suficiente para serem identificados, a taxa de quadros por segundo vai de 5,38 a 3,54 quadros/s (variação de 34%). Já no sistema baseado em *multithread* com seleção adequada de alvos a taxa de quadros por segundo vai de 8,04 a 7,47 quadros/s (variação de 7%). Essa menor sensibilidade à variação do número indivíduos na base é um dos pontos fortes dessa abordagem, e é devido à não aplicação do algoritmo de reconhecimento quando a qualidade da face não é boa o suficiente para o algoritmo de reconhecimento. Além disso, com essa abordagem, é possível manter a identidade

do alvo ao longo da sequência de vídeo com uso do algoritmo de rastreamento, dispensando novas identificações a cada quadro para um mesmo indivíduo.

Algumas sugestões que podem ser implementadas em trabalhos futuros são:

- Compartilhar informação entre os estágios finais correspondentes do TLD (*Bounding Boxes* identificadas em um modelo já podem ser descartadas do outro);
    - Como visto no Capítulo 4 o estágio final do processamento de um modelo entrega um conjunto de BB*'s* identificadas como o objeto de interesse. Essas BB *'s* podem ser excluídas do processamento do próximo modelo, diminuindo progressivamente a região a ser avaliada pelo TLD, e consequentemente aumentando a velocidade de execução.

- Inserção do bloco de medida de qualidade de imagem (expressão, pose, iluminação, background, etc.);
    - Um dos principais objetivos desse sistema é a seleção inteligente de segmentos de imagem que possuam faces com qualidade suficiente para serem identificadas. Esse projeto futuro sugere o desenvolvimento de um bloco que avalie outros parâmetros de qualidade além da pose, como expressão, iluminação, fundo da imagem, foco, etc., a fim de tornar o bloco mais criterioso na escolha de faces com maior probabilidade de identificação.

- Inserir uma medida de pose mais precisa (algoritmo ASM);
    - Uma medida de pose mais precisa implica numa medida de qualidade mais precisa. Esse indicador é um dos mais importantes, sendo necessária uma atenção especial ao cálculo dessa medida. Melhores estimativas de pose podem ser alcançadas com o uso de *Active Shape Models* (ASM) (Cootes *et al.*, 2001).

- Inserir um bloco de extração de background, para diminuir área a ser processada;

    - Uma sugestão para aumentar o desempenho do sistema é diminuindo a área da região a ser processada pelas etapas posteriores. Isso pode ser alcançado através de técnicas de extração de background, uma vez que a tendência da câmera é de ficar fixa com um plano de fundo inalterado ao longo do tempo (Moreira et al., 2013).

- Usar o sistema TLD para implementar novos sistemas identificadores baseados em movimento;

    - Algumas técnicas de reconhecimento facial fazem uso do movimento do alvo de interesse para realizarem a identificação. Partindo desse principio, um dos algoritmos dentro do bloco de reconhecimento pode fazer uso da precisão e robustez do *tracker* do sistema (Moreira et al., 2013).

- Utilizar o sistema para construir um modelo 3D da face.

    - O sistema permite a ampliação da base de dados de um indivíduo identificado, uma vez que são extraídos vários padrões e atributos durante a execução para treinamento de um modelo da face rastreada. Entre elas, poses diferentes da face do indivíduo. Isso abre possibilidades para implementação de algoritmos de reconhecimento mais eficazes que fazem uso dessas informações extras do indivíduo.

7
REFERÊNCIAS BIBLIOGRÁFICAS

AGGARWAL, G.; ROY-CHOWDHURY, A. K.; CHELLAPPA, R. A system identiffcation approach for video-based face recognition. In **Proceedings, 17° International Conference On Pattern Recognitio n**, volume 4, pages 175 -178, 2004.

AHLBERG J. , “An Active Model for Facial Feature Tracking,” **EURASIP Journal on Applied Signal Processing**, pp. 566-571, 2002.

AHONEN, T.; HADID, A.; PIETIKÄINEN, M. Face Description with Local Binary Patterns: Application to Face Recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 2006, vol 28, n°12, pp 2037-2041.

AUGUSTEIJN, M.F.; SKUJCA ,T.L. Identification of Human Faces through Texture-Based Feature Recognition and Neural Network Technology. **Proceedings IEEE Conf. Neural Networks**, pp. 392-398, 1993.

BARTLETT, M.; STEWART, T. 1997. **Viewpoint invariant face recognition using independent Component analysis and attractor networks**. In M. Mozer, M. Jordan, & T. Petsche, Eds.,Advances in Neural Information Processing Systems 9. Cambridge, MA: MIT Press 817-823.

BEYMER, D. Face recognition under varying pose. in **Proceedings IEEE Conf. Computer Vision and Pattern Recognition**, 1994, pp. 756–761.

BRADSKI, G., KAEHLER, A. **Learning OpenCV**, 1 Ed., Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

BREIMAN, L. **Random forests**. ML, 45(1):5–32, 2001.

CANNY, J. A Computational Approach to Edge Detection. **IEEE Trans. Pattern Analysis and Machine Intelligence**, vol. 8, no. 6, pp. 679-698, June 1986.

CANTON-FERRER, C., CASAS, J.; PARDA'S, M. Head. Pose Detection Based on Fusion of Multiple Viewpoint Information: Multimodal

Technologies for Perception of Humans**. Proceedings First Int'l Workshop Classification of Events, Activities and Relationships**, pp. 305-310, 2007.

CARLO T.; TAKEO K. Detection and Tracking of Point Features, **International Journal of Computer Vision**, 1991.

CHAI, D; NGAN, K.N. Locating Facial Region of a Head-and-Shoulders Color Image**, Proceedings Third Int'l Conf. Automatic Face and Gesture Recognition**, pp. 124-129, 1998.

CHEN, Q.; WU, H.; YACHIDA, M. Face Detection by Fuzzy Matching. **Proceedings Fifth IEEE Int'l Conf. Computer Vision**, pp. 591-596, 1995.

CHEN, T.; WOTAO, Y.; XIANG, S. Z.; COMANICIU, D.; HUANG, T. S. Total variation models for variable lighting face recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence** , 28(9):1519{1524, 2006.

CHEN, W.; ER, M. J.; WU, S. Illumination compensation and normalization for robust face recognition using discrete cosine transform in logarithm domain. **IEEE Transactions on Systems**, Man and Cybernetics - Part B: Cybernetics, 36(2), 458-466,2006.

COMANICIU, D.; RAMESH, V.; MEER, P. Real-time tracking of non-rigid objects using mean shift. In **Proceedings of the CVPR**, Hilton Head Island, S.C., U.S.A., 2000. Vol. 2, pp. 142-149.

COOTES, T.; EDWARDS, G.; TAYLOR, C. "Active Appearance Models," **IEEE Trans. Pattern Analysis and Machine Intelligence**, vol. 23, no. 6, pp. 681-685, June 2001.

COOTES, T.; WALKER, K.; TAYLOR, C. View-based active appearance models. In **Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition**, pages 227-232, 2000.

COOTES, T.F.; EDWARDS, G.J.; TAULOR, C.J. Active Appearance Models, **Proceedings Fifth European Conf. Computer Vision**, H. Burkhardt and B. Neumann, eds., vol. 2, pp. 484-498, 1998.

COOTES,T.; EDWARDS G.; TAYLOR, C. Active Appearance Models. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 23, 2001, pp 681-685.

COOTES,T.; TAYLOR, C.; COOPER, D.; GRAHAM, J. **Active Shape Models –Their Training and Application**, Computer Vision and Image Understanding, vol. 61, no. 1, pp. 38-59, 1995.

CRAW, I.; TOCK, D.; BENNETT, A. Finding face features. In **Proceedings European Conference on Computer Vision**, p. 92-96, 1992.

EDWARD, C.J.; TAYLOR, C.J.; COOTES, T.F. Learning to Identify and Track Faces in an Image Sequence. **Proceedings Int'l Conf. Automatic Face and Gesture Recognition**, pp. 260-265, 1998.

EVERINGHAM, M.; ZISSERMAN, A. Identifying individuals in video by combinig 'generative' and discriminative head models, in **Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV '05)**, vol. 2, pp. 11031110, Beijing,China, October 2005.

FÉRAUD ,R.; BERNIER, O. **Ensemble and Modular Approaches for Face Detection: A Comparison.** Advances in Neural Information Processing Systems 10, M.I. Jordan, M.J. Kearns, and S.A. Solla, eds., pp.472-478, MIT Press, 1998.

FÉRAUD, R.; BERNIER, O.; VILLET, J.-E.; COLLOBERT, M. A Fast and Accurate Face Detector Based on Neural Networks. **IEEE Trans. Pattern Analysis and Machine Intelligence**, vol. 22, no. 1, pp. 42-53, Jan. 2001.

FÉRAUD, R.; **Neural Networks and Estimation for Face Detection. Face Recognition: From Theory to Applications**, H. Wechsler, eds., vol. 163, pp. 424-432, 1998.

GEORGHIADES, A.; KRIEGMAN, D.; BELHUMEUR, P. From few to many: Generative models for recognition under variable pose and illumination. **IEEE Transactions Pattern Analysis and Machine Intelligence**, 40, 643-660. 2001.

GORODNICHY, D. O. On importance of nose for face tracking. In **Proceedings IEEE International Conference on Automatic Face and Gesture Recognition** , pages 181–186, May 2002.

GRAF, H.P.; CHEN ,T.; PETAJAN ,E.; COSATTO, E. Locating Faces and Facial Parts, **Proceedings First Int'l Workshop Automatic Face and Gesture Recognition**, pp. 41-46, 1995.

GRAF, H.P.; COSATTO, E. D.; GIBBON, KOCHEISE, M.; PETAJAN, E. “Multimodal System for Locating Heads and Faces,” **Proceedings Second Int'l Conf. Automatic Face and Gesture Recognition**, pp. 88-93, 1996.

HAGER, G.D.; BELHUMEUR, P.N. Efficient Region Tracking with Parametric Models of Geometry and Illumination. **IEEE Transactions on Pattern Analyvsis and Machine Intelligence**, vol. 20, 1998, pp 1025-1039.

HAN, C.-C.; LIAO, H.-Y.M.; YU , K.-C.; CHEN, L.-H. Fast Face Detection via Morphology-Based Pre-Processing. **Proceedings Ninth Int'l Conf. Image Analysis and Processing**, pp. 469-476, 1998.

HAN, M.; SETHI,A.; GONG, Y. A detection-based multiple object tracking method, In **Proceedings Int. Conf. Image Process. (ICIP)**, pages 3065-3068, Singapore, Oct. 2004.

HENRY S.; TAKEO K. **Object detection using the statistics of parts**. IJCV. P. 151–177, 2004. Live face detection based on the analysis of Fourier spectra (2004).

HSU, R.-L.; MOTTALEB, M. A.; JAIN, A. K. Face detection in color images. **Proceedings IEEE Transactions on Pattern Analysis and Machine Intelligence**, 24(5):696-706, 2002.

HU, L.; Chen, Y. Z.; ZHANG, H. “Estimating face pose by facial asymmetry and geometry,” in **Proceedings IEEE Int’l. Conf. Automatic Face and Gesture Recognition**, 2004, pp. 651–656.

HUANG, J., SHAO, X.; WECHSLER, H. Face pose discrimination using support vector machines (SVM). in **Proceedings Int’l. Conf. Pattern Recognition**, 1998, pp. 154–156.

HYUNG-SOO L.; DAIJIN K.; **Robust face tracking by integration of two separate trackers: Skin Color and facial shape**, Pattern Recognition 40 (2007) ,pp:3225 – 3235.

JEBARA, T.S.; PENTLAND, A. Parameterized Structure from Motion for 3D Adaptive Feedback Tracking of Faces. **Proceedings IEEE Conf. Computer Vision and Pattern Recognition**, pp. 144-150, 1997.

JEBARA, T.S.; RUSSEL, K.; PENTLAND, A. Mixtures of Eigenfeatures for Real-Time Structure from Texture. **Proceedings Sixth IEEE Int'l Conf. Computer Vision and Pattern Recognition**, pp. 128-135, 1998.

KAKADIARIS, I. A.; PASSALIS, G.; TODERICI, G.; MURTUZA, N.; . LU, Y.; KARAMPATZI- AKIS, N.; THEOHARIS.; T. Three-dimensional face recognition in the presence of facial expressions: An annotated deformable model approach. **IEEE Transac- tions on Pattern Analysis and Machine Intelligence** , 29(4):640{649, 2007.

KALAL , Z., MATAS, J., MIKOLAJCZYK, K., Forward-Backward Error : Automatic Detection of Tracking Failures, in **ICPR**, 2010.

KALAL , Z., MATAS, J., MIKOLAJCZYK, K., **Online learning of robust object detectors during unstable tracking**, 3rd On-line Learning for Computer Vision Workshop 2009, Kyoto, Japan, IEEE CS.

KALAL , Z., MATAS, J., MIKOLAJCZYK, K., P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints, 23rd **IEEE Conference on Computer Vision and Pattern Recognition, CVPR,** June 13-18, San Francisco, CA, USA, 2010.

KALAL , Z., MATAS, MIKOLAJCZYK, K., Forward-Backward Error: Automatic Detection of Tracking Failures, **International Conference on Pattern Recognition**, 23-26 August, 2010, Istambul, Turkey.

KALAL, Z.; MATAS, J.; MIKOLAJCZYK, K. **Online learning of robust object detectors during unstable tracking.** On-line Learning for Computer Vision Workshop, 2009.

KEREN, D.; PELEG, S.; BRADA. R. Image sequence enhancement for super-resolution image sequence enhancement. In **Proceedings IEEE Conference on Computer Vision and Pattern Recognition**, pages 742-746, 1988.

KRAMER, M.A.; **Nonlinear Principal Component Analysis Using Autoassociative Neural Networks**. Am. Inst. Chemical Eng. J., vol. 37, no. 2, pp. 233-243, 1991.

LADES, M., J. C. Vorbr¨uggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. W¨urtz, and W. Konen, Distortion invariant object recognition in the dynamic link architecture. **IEEE Trans. Comput**., vol. 42, pp. 300–311, 1993.

LEE, K.; HO, J.; YANG, M.; KRIEGMAN. D.; Video-based face recognition using probabilistic appearance manifolds. In **Proceedings IEEE Conference on Computer Vision and Pattern Recognition** , pages 313-320, 2003.

LEE,K. C. ; HO, J.; YANG, M. H.; KRIEGMAN, D . Video-based face recognition using probabilistic appearance manifolds. In **Proceedings of IEEE Int. Conf. on Computer Vision and Pattern Recognition** , pages 313–320, 2003.

LI, J.; WANG, Y.; TAN,T.; JAIN, A. Live Face Detection Based on the Analysis of Fourier Spectra, Biometric Technology for Human Identification, **Proceedings SPIE**, vol. 5404, pp. 296-303, 2004.

LI, S. Z. .; ZHANG, Z. Q. . FloatBoost Learning and Statistical Face Detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, VOL. 26, NO. 9, September, 2004.

LI, S. Z.; JAIN, A. K;. **Handbook of Face Recognition.** Springer-Verlag, Secaucus, NJ, 2005.

LU, X. **Image Analysis for Face Recognition**, 2003, disponível em: <http://www.cse.msu.edu/ lvxiaogu/publications/ImAna4FacRcg lu.pdf>, acessado em 20 jan. 2013.

LU; X.; JAIN, A. K. Deformation modeling for robust 3d face matching. **IEEE Transactions on Pattern Analysis and Machine Intelligence** , p. 1346-1357, 2008.

MARTINEZ, A. M.; Recognizing imprecisely localized, partially occluded, and expression variant faces from a single sample per class. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 24(6):748-763, 2002.

MOREIRA, G., FEIJÓ, B., LOPES, H., FEITOSA, R. Q. Real-time Object Tracking in High-Definition Video Using Frame Segmentation and Background Integral Images, In: XXVI SIBGRAPI CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES, 2013, Arequipa. **Proceeding. Arequipa: Universidad Católica San Pablo**, 2013.

MURPHY-CHUTORIAN, E., TRIVEDI, M., “Head Pose Estimation in Computer Vision: A Survey”, **IEEE Transactions on Pattern Analysis and Machine Intelligence**.

NAKAMURA, M.; NOMIYA, H.; UEHARA, K. Improvement of boosting algorithm by modifying the weighting rule. **Annals of Mathematics and Artificial Intelligence**, 41:95-109,2004.

NG , J.; GONG, S. **Composite support vector machines for detection of faces across views and pose estimation**. Image and Vision Computing, vol. 20, no. 5-6, pp. 359–368, 2002.

NIYOGI, S.; FREEMAN, W. Example-based head tracking. in **Proceedings Int'l. Conf. Automatic Face and Gesture Recognition**, 1996, pp. 374–378.

O'TOOLE,A. J.; ROARK, D. A.; ABDI, H. **Recognizing moving faces: A psychological and neural synthesis.** Trends in Cognitive Science , 6:261–266, 2002.

OJALA, T., PIETIKAINEN, M., MAENPAA, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 24(7), 971-987. 2002.

OSUNA, E., FREUND, R, and GIROSI, F., Training support vector machines: An application to face detection. in **Proceedings IEEE Conf. Computer Vision and Pattern Recognition**, 1997, pp. 130–136

PHILLIPS, P.; GROTHER, P.; MICHEALS,R. J.; BLACKBURN, D. M.; TABASSI,E.; BONE, J. M. **Face recognition vendor test 2002 results**. Technical report, 2003

PHILLIPS,P.J. ; MOON, H.; RAUSS, P.; RIZVI,S.A. **The FERET Evaluation Methodology for Face-Recognition Algorithms.** Computer Vision and Pattern Recognition, pp. 137-143, 2001.

QING, L.; SHAN, S.; CHEN, X.; GAO, W. Face recognition under varying lighting based on the probabilistic model of gabor phase. In **Proceedings International Conference on Pattern Recognition** , pages 1139-1142, 2006.

ROMDHANI, S., TORR, P., SCHOELKOPF, B., BLAKE, A.: Efficient face detection by a cascaded support-vector machine expansion. **Proceedings of The Royal Society A** 460 (2004) 3283–3297.

ROY-CHOWDHURY, A.; XU, Y. **Pose and Illumination Invariant Face Recognition Using Video Sequences, Face Biometrics for Personal**

Identification: Multi-Sensory Multi-Modal Systems, Springer-Verlag, pp. 9-25, 2006.

SABER, E.; TEKALP, A.M. **Frontal-View Face Detection and Facial Feature Extraction Using Color, Shape and Symmetry Based Cost Functions**. Pattern Recognition Letters, vol. 17, no.8, pp. 669-680, 1998.

SATYANADH G., AND VIJAYAN K. A, **Face Detection Technique Based on Rotation Invariant Wavelet Features.** International Conference on Information Technology: Coding and Computing (ITCC'04), Volume 2, April 5-7, 2004, Las Vegas, Nevada, USA.

SAXE, D.; FOULDS, R. "Toward Robust Skin Identification in Video Images," **Proceedings Second Int'l Conf. Automatic Face and Gesture Recognition**, pp. 379-384, 1996.

SINGH, S.K.; CHAUHAN, D.S.; VATSA, M.; SINGH, R. A robust skin color based face detection algorithm, Tamkang **Journal of Science and Engineering** 6(4) 2003 227-234.

SIROHEY, S.A. **Human face segmentation and identification**. Technical Report CS-TR-3176, Univ. Of Maryland, 1993.

T.F. Cootes, C.J. Taylor, D. Cooper, and J. Graham, **Active Shape Models - Their Training and Application**. Computer Vision and Image Understanding, vol. 61, no. 1, pp. 38-59, Jan. 1995. L.C. Jain et al., publ. CRC Press, ISBN 0-8493-2055-0, Chapter 11, pp. 355-396, (1999).

TOMASI, C., KANADE, T., **Detection and Tracking of Point Features**. Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.

TOYAMA , K.; BLAKE, A. Probabilistic Exemplar Based Tracking in a Metric Space. **Proceedings Int'l Conf. Computer Vision**, vol. 2, pp. 50-57, 2001.

TREPTOW, A.; ZELL, A. Combining adaboost learning and evolutionary search to select features for real-time object detection. in **Proceedings of the IEEE Congress on Evolutionary Computation**, Portland, Oregon, vol. 2. IEEE, 2004, pp. 2107–2113.

VIOLA, P. A; JONES, M. J . Robust real-time face detection. **International Journal of Computer Vision**, 57(2):137-154, 2004.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. in **Proceedings IEEE Conf. Computer Vision and Pattern Recognition**, 2001, pp. 511–518.

WANG, H., LI, S. Z., WANG, Y. Face recognition under varying lighting conditions using self quotient image. In **Proceedings of the IEEE international conference on automatic face and gesture recognition** pp. 819-824, 2004.

WANG, J.-G.; SUNG, E. **Enhancement of 3D Head Pose Estimated by Point at Infinity**. Image and Vision Computing, vol. 25, no. 12, pp. 1864-1874, 2007.

WISKOTT, L.; FELLOUS, J.; KRUGER, N., VON DER MALSBURG, C., Face Recognition by Elastic Bunch Graph Matching, In **Intelligent Biometric Techniques in Fingerprint and Face Recognition**, eds.

WU,J., BRUBAKER, S. C., MULLIN, M. D. , REHG, J. M. Fast asymmetric learning for cascade face detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence** , 30(3):369{382, 2008.

YAN W.; YANGHUA L.; LINMI T.; GUANGYOU X. , Real-time multi-view face detection and pose estimation in video stream. 18th International Conference on Pattern Recognition , pp: 354 – 357,2006.

YANG, M.; KRIEGMAN, D.; AHUJA, N. Detecting faces in images: A survey. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 24(1):34-58, 2002.

YAO, H.; GAO; W. **Face locating and tracking method based on chroma transform in color images**. Signal Processing Proc. 2000 , vol 2. pp. 1367-1371, 2000.

YOW, K.C; CIPOLLA, R. A Probabilistic Framework for Perceptual Grouping of Features for HumanFace Detection. **Proceedings Second Int'l Conf. Automatic Face and Gesture Recognition**, pp. 16-21, 1996.

YOW, K.C; CIPOLLA, R. **Feature-Based Human Face Detection**. Image and Vision Computing, vol. 15, no.9 ,pp. 713-735, 1997.

YUILLE, A.; HALLINAN, P.; COHEN, D. **Feature Extraction from Faces Using Deformable Templates.** Int'l J. Computer Vision, vol. 8, no. 2, pp. 99-111, 1992.

ZHANG, Z., HU, Y., LIU M.,HUANG, T., **Head pose estimation in seminar room using multi view face detectors**. in Multimodal Technologies for Perception of Humans, Int'l. Workshop Classification of Events Activities and Relationships, CLEAR 2006, ser. Lecture Notes in Computer Science, R. Stiefelhagen.

ZHAO, W.; R. CHELLAPPA. **Robust face recognition using symmetric shape-from-shading**. Technical Report, Center for Automation Research, University of Maryland, 1999.

ZHAO, W; CHELLAPPA, R; PHILLIPS,P.J.; ROSENFELD,A. **Face Recognition: A Literature Survey**. ACM Computing Surveys, Vol. 35, No. 4, December 2003, pp. 399–458.

ZHOU, S.; KRUEGER, V.; CHELLAPPA, R. **Probabilistic recognition of human faces from video**. Computer Vision and Image Understanding , 91:214-245, 2003.

ZHU, Y.; FUJIMURA, K. 3D Head Pose Estimation with Optical Flow and Depth Constraints. **Proceedings Fourth Int'l Conf. 3-D Digital Imaging and Modeling**, pp. 211-216, 2003.