# Utilização de Tecnologia Adaptativa na Detecção da Direção do Olhar

**Hemerson Pistori**
Universidade Católica Dom Bosco, Depto. Engenharia de Computação,
Campo Grande, Brasil, 79117-900
Universidade de São Paulo, Escola Politécnica,
São Paulo, Brasil, 05508-900
pistori@ec.ucdb.br

**João José Neto**
Universidade de São Paulo, Escola Politécnica,
São Paulo, Brasil, 05508-900
joao.jose@poli.usp.br

**Eduardo Rocha Costa**
Universidade de São Paulo, Escola Politécnica,
São Paulo, Brasil, 05508-900
eduardo.rocha@poli.usp.br

## Resumo

Este artigo descreve um protótipo de um sistema cuja interface com o usuário é feita através da detecção da direção do olhar. Embora existam soluções similares disponíveis no mercado, estes produtos geralmente exigem um hardware de alto custo e arranjos físicos não triviais. O protótipo a que se refere o presente trabalho realiza uma solução não intrusiva e de baixo custo, utilizando técnicas de aprendizagem computacional baseadas em dispositivos adaptativos.

**Palavras chave:** Visão Computacional, Interface Homem-Máquina, Aprendizagem Computacional, Dispositivos Adaptativos.

## Abstract

This paper presents a system prototype with an eyes-gaze driven user interface. Although similar commercial solutions are available, they frequently require sophisticated and expensive hardware and environment arrangements. In this work a low cost prototype is described which features a non-intrusive and cheap approach by using machine learning techniques based on adaptive automata.

**Keywords:** Computational Vision, Human-Computer Interaction, Eye-Driven Man-Machine Interfaces, Machine Learning and Adaptive Devices.

## 1 Introdução

O baixo custo dos dispositivos de captura de imagens, juntamente com o aumento significativo na capacidade de processamento dos computadores pessoais da atualidade, abrem espaço para o desenvolvimento de novos tipos de interface homem-máquina, utilizando técnicas de visão computacional. Uma alternativa que tem se mostrado bastante promissora [10] consiste em detectar, através de uma ou mais câmeras filmadoras, a direção do olhar do usuário que se encontra em frente ao monitor. Esta informação pode ser utilizada tanto de maneira passiva, como, por exemplo, para registrar as imagens de uma página da internet que mais chamam a atenção de determinados usuários, quanto de maneira ativa, quando as informações são utilizadas para controlar o que está sendo apresentado na tela [23]. Estudos indicam que objetos apresentados na tela de um computador podem ser apontados com maior velocidade e comodidade com os olhos do que com um mouse [10].

Um benefício imediato da aplicação de técnicas de detecção da direção do olhar é a possibilidade de melhorar significativamente a capacidade de interação dos portadores de deficiências motoras, permitindo que até pessoas cujos movimentos se limitam ao globo ocular tenham a possibilidade de utilizar computadores. Sistemas completos que possibilitam este tipo de interface já estão disponíveis no mercado, no entanto, a maioria deles exige a utilização de dispositivos especiais, como capacetes, emissores e detectores de sinais infravermelho [1] e câmeras de foco automático (*gaze-camera*) [22].

No presente trabalho demonstra-se, na prática, a viabilidade da utilização de algoritmos de detecção do olhar em um sistema composto de um computador pessoal comum (Pentium II 450Mhz, 128 Mb de Ram, 8Gb de HD) e uma câmera de baixo custo (WebCam Creative) posicionada acima do monitor (ver figura 1). Para isto, implementamos um jogo muito simples, o jogo da velha (*Tic-Tac-Toe*), que pode ser jogado utilizando apenas os olhos. A solução utilizada basea-se em técnicas de visão computacional para localização e extração de atributos na região dos olhos e em técnicas de aprendizagem computacional para a determinação da direção do olhar. A aprendizagem é implementada com o auxilio de um dispositivo adaptativo baseado em árvores de decisão [7], o qual além de apresentar taxas de acerto próximas ao das redes neurais com *backpropagation*, permite que erros de classificação possam ser reparados dinamicamente a partir de novas instâncias de treinamento.

Figura 1: Ambiente em que os experimentos foram realizados

O desenvolvimento do protótipo foi feito em Java, e exigiu a integração de três pacotes de software: um ambiente para processamento de imagens, o ImageJ; uma biblioteca para controle de dispositivos de captura de sinais temporais (*time-based devices*), o Java Media Framework; e uma biblioteca de apoio ao desenvolvimento e testes de algoritmos de aprendizagem computacional e mineração de dados (*Data Mining*), o WEKA. A utilização desses pacotes, além de facilitar a manutenção e integração futura com outros sistemas, possibilitou uma diminuição drástica no tempo de implementação do protótipo (1 mês - 2 homens/mês).

O restante deste artigo está organizado da seguinte forma: a próxima seção apresenta algumas estratégias utilizadas atualmente para detecção da direção do olhar. A seção 3 é uma introdução à tecnologia de aprendizagem utilizada em nosso protótipo. O desenvolvimento do protótipo, juntamente com a descrição das ferramentas utilizadas e criadas neste projeto são apresentas nas seções 4, 5 e 6. Alguns dos resultados obtidos, bem como algumas propostas para prosseguimento desse projeto podem ser encontrados na última seção do presente trabalho.

## 2 Técnicas de Detecção da Direção do Olhar

Os primeiros sistemas a permitirem a interação homem-máquina através de movimentos da face ou dos olhos foram caracterizados por serem ou *intrusivos* ou *expansivos* [22]. Sistemas *intrusivos* exigem que o usuário

"vista" equipamentos especiais, como óculos ou capacetes, enquanto os *expansivos* utilizam dispositivos não-convencionais, como por exemplo, diodos emissores de raios infravermelhos, que facilitam a detecção das pupilas humanas [3], e câmeras filmadoras especiais, com foco automático.

Recentemente, a queda nos custos de dispositivos de captura de imagens, como as populares *WebCams*, impulsionou a busca por sistemas não-intrusivos de detecção da direção do olhar. A direção do olhar pode ser estimada tanto pela observação dos movimentos da cabeça, como um todo, quanto pelo movimento do globo ocular e da íris [22]. Estas estimativas podem ser obtidas por técnicas algébricas de inferência do foco do olhar a partir de características especificas da face projetada em 2D [9, 22], ou por aprendizagem computacional [1, 21].

Stiefelhagen [21] relata ter obtido uma precisão entre 1.3 e 1.9 graus em um sistema que permite o controle do ponteiro do mouse através do olhar. Em seus experimentos foram utilizadas 4000 imagens de pessoas acompanhando um cursor que se deslocava sobre a tela de um monitor. Recortes destas imagens contendo apenas a região da face envolvendo os olhos, o nariz e a boca, junto com a posição do cursor na tela, alimentaram uma rede neural artificial do tipo *feed-forward* com 3 níveis, utilizando a técnica de *backpropagation* para treinamento. Embora precisões de até 0.75 graus possam ser obtidas usando outros métodos [1], as técnicas que utilizam redes neurais, além de não dependerem de dispositivos especiais, podem ser utilizadas em diversos tipos de ambiente e iluminação, o que em geral não ocorre com as outras técnicas. O problema com o enfoque baseado em redes neurais "tradicionais" é que o aprendizado não é incremental, o que dificulta a correção de erros, no modelo aprendido, durante a fase de utilização do sistema.

Seguindo também a linha não intrusiva, Gee e Cipolla [9], descrevem um sistema que infere a direção do olhar a partir da reconstituição de um modelo simplificado da cabeça humana, a partir da imagem projetada. Neste modelo, busca-se valorizar a posição relativa dos olhos, boca e nariz, que, segundo os autores, são atributos que variam menos em relação a diferentes sujeitos do que, por exemplo, as orelhas, que podem estar cobertas, ou semicobertas, por cabelos.

Wang e Sung [22] modelam a direção do olhar como sendo a direção da normal ao plano de suporte de cada íris. Aproximando a forma da íris através de circunferências e assumindo raios, distâncias focais e fatores de escala conhecidos, é possível estimar a normal em questão a partir da projeção, elíptica, destes dois círculos. A manipulação algébrica deste problema oferece, no entanto, duas soluções para cada íris. Wang e Sung propõe a escolha das normais mais próximas entre si para resolver este impasse, e apresentam diversos resultados empíricos que sustentam a utilidade desta heurística. A detecção das duas elipses que correspondem ao contorno da íris na imagem projetada não é uma tarefa trivial, principalmente se for executada em tempo real. Tanto a estratégia de Wang e Sung, quanto a de Gee e Cipolla, exigem uma alta capacidade de processamento, o que dificulta a construção de soluções de baixo custo.

Transformadas de Hough são ferramentas poderosas na detecção de linhas e círculos a partir de imagens cujas bordas foram previamente extraídas ou realçadas [12]. De modo geral, as transformadas trabalham em um domínio definido pelos possíveis parâmetros da equação que descreve o ente geométrico em questão. No caso de retas, descritas pela equação $y = ax + b$, temos um domínio bidimensional, com dois eixos ortogonais, para os parâmetros $a$ e $b$. Cada ponto neste novo "espaço" determina uma possível reta e a existência desta reta na imagem original é calculada a partir de uma varredura nos pontos que participam das bordas detectadas. Para cada um destes pontos, o algoritmo de detecção deve varrer uma das dimensões do espaço de Hough e calcular o valor do outro parâmetro, incrementando um acumulador associado a cada ponto deste espaço. Ao final, os pontos no espaço de Hough com maiores valores em seus acumuladores são os que representam as retas mais prováveis, na imagem original.

Teoricamente, as transformadas de Hough podem ser utilizadas na detecção de qualquer figura geométrica definida por equações paramétricas. No entanto, seu cálculo torna-se impraticável quando as equações são não-lineares e envolvem mais de 3 parâmetros, como é o caso das elipses. McLaughlin [12] propõe uma melhoria para o algoritmo original, melhoria esta que consiste em utilizar um sistema de 3 equações lineares na descrição de elipses, e uma varredura, por amostragem informada, sobre os pontos que constituem as bordas da imagem original. Esta nova técnica denomina-se Transformada de Hough Aleatorizada (*Randomized Hough Transform*). Nossos experimentos indicaram, no entanto, que mesmo esta versão otimizada não pode ser executada em tempo real em equipamentos que não sejam de última geração. Outras técnicas para detecção de elipses envolvem aproximações pelo método dos mínimos quadrados [11] e filtros de Kalman [18].

## 3 Dispositivos Adaptativos

Uma questão recorrente na área da teoria da computação, principalmente em relação aos formalismos utilizados na representação de problemas e algoritmos, é a busca do equilíbrio entre expressividade e usabilidade. Máquinas de Turing, por exemplo, são formalismos altamente expressivos. No entanto, a utilização direta deste formalismo na solução de problemas reais é muito desconfortável. Por outro lado, máquinas de estados

finitos são fáceis de usar, mas possuem um poder de expressão muito mais restrito, o que prejudica a sua aplicação direta na solução de boa parte dos problemas reais mais complexos.

Uma alternativa para um balanceamento aceitável entre expressividade e usabilidade é o emprego de técnicas adaptativas [14, 15, 16], que permitem aumentar a expressividade de um formalismo convencional sem afetar a sua usabilidade. Considerando um formalismo qualquer como sendo um dispositivo computacional, podemos definir tal extensão através da associação, às regras que definem o dispositivo inicial, de um conjunto de outras regras, que dotam tal dispositivo da capacidade de alterar dinamicamente sua própria estrutura, sem modificar fundamentalmente sua sintaxe e semântica originais. Embora tais técnicas adaptativas tenham sido ainda pouco exploradas, diversos exemplos de utilização bem sucedida, na solução de problemas reais, podem ser encontradas em [2, 6, 5]. Entre os formalismos que já foram utilizados em pesquisas com tecnologias adaptativas podemos destacar: autômatos, cadeias de Markov, "*Statecharts*" e Tabelas de Decisão [1].

A aplicação da tecnologia adaptativa a um mecanismo subjacente, formalizado através de árvores de decisão, resulta no dispositivo adaptativo denominado árvore de decisão adaptativa. Este dispositivo pode então ser utilizado para modelar um algoritmo de aprendizagem que simplesmente representa as instâncias de treinamento (armazenadas como vetores de atributos [13]) de um problema qualquer como uma espécie de árvore de prefixos, cada nível da qual representando um atributo. Este algoritmo, acrescido de um módulo estatístico utilizado para classificar instâncias não previstas na árvore de decisão, e de um outro módulo capaz de reordenar os atributos da árvore com base no ganho de informação (*information gain* [19]) global de cada atributo, deu origem ao algoritmo denominado AdapTree. Este foi justamente o mecanismo de aprendizagem utilizado neste trabalho, principalmente pela capacidade incremental, que permite a absorção de novos exemplos de treinamento, durante a execução. Uma descrição mais detalhada deste algoritmo pode ser encontrada em [7].

## 4 Ferramentas Auxiliares

Constatamos durante o desenvolvimento deste projeto que há disponibilidade de pacotes de excelente qualidade escritos em Java, com código aberto e gratuitos, tanto na área de visão computacional, quanto na de aprendizagem de máquina. Embora predominem códigos abertos em C e C++, a portabilidade da linguagem C e suas derivadas está muito restrita ao "núcleo ANSI", o que se torna um problema quando os produtos a serem desenvolvidos envolvem dispositivos não convencionais e interfaces gráficas [17]. Uma vez que o baixo custo do produto final é uma das metas deste projeto, e que a portabilidade é uma ferramenta importante para a redução de custos, optamos pela utilização exclusiva de soluções baseadas em Java. Principalmente como uma forma de promover e retribuir o esforço da comunidade desenvolvedora de software livre, e mesmo não sendo comum para um artigo científico, descreveremos brevemente, a seguir, as ferramentas utilizadas.

### 4.1 ImageJ

Para implementar e testar técnicas de visão computacional e processamento digital de imagens, utilizamos o pacote ImageJ, que é uma versão multiplataforma, ainda em desenvolvimento, do software NIH Image, para Macintosh. Entre os recursos oferecidos pelo pacote destacamos a disponibilidade, com programas-fonte abertos, de diversos algoritmos para: manipulação dos mais variados formatos de arquivo de imagens, detecção de bordas, melhoria de imagens, cálculos diversos (áreas, médias, centróides) e operações morfológicas. Esse software disponibiliza também um ambiente gráfico que simplifica a utilização de tais recursos, além de permitir a extensão através de *plugins* escritos em Java. Um outro fator importante para a sua escolha é a existência de uma grande comunidade de programadores trabalhando em seu desenvolvimento, com novos plugins sendo disponibilizados freqüentemente. Um desses plugins, o *HoughCircles*, foi desenvolvido pelo nosso grupo durante a execução deste projeto e está disponível na página do ImageJ [2].

### 4.2 Waikato Environment for Knowledge Analysis

O terceiro pacote utilizado neste projeto foi o WEKA[3] (Waikato Environment for Knowledge Analysis) [24, 4], também escrito em Java e com programas-fonte abertos. O WEKA é um ambiente bastante utilizado em pesquisas na área de aprendizagem de máquina, pois oferece diversos componentes que facilitam a implementação de classificadores e agrupadores (*clustering tools*). Além disto, esse ambiente permite que novos algoritmos sejam comparados a outros algoritmos já consolidados na área de aprendizagem, como é o caso dos algoritmos C4.5, Backpropagation, KNN e Naive Baeyes, entre outros. Podemos com esse

[1] http://www.pcs.usp.br/~lta
[2] http://rsb.info.nih.gov/ij/
[3] http://www.cs.waikato.ac.nz/ ml/weka/

pacote obter facilmente resultados estatísticos comparativos da execução simultânea de diversos programas de aprendizagem em domínios como, por exemplo, o reconhecimento de caracteres, o reconhecimento de imagens e o diagnóstico médico. O AdapTree foi implementado utilizando esta biblioteca, especificamente a partir da especialização de uma classe que implementa o algoritmo ID3 [19], um algoritmo clássico de indução de árvores de decisão.

### 4.3 Java Media Framework

A captura de imagens em tempo real é feita através do pacote *Java Media Framework*[4], da Sun Microsystems. Além da portabilidade, uma das principais vantagens deste pacote é a possibilidade da integração de soluções baseadas na detecção da direção do olhar, em páginas da Internet. Para utilizar o ImageJ, em conjunto com o JMF, criamos um plugin, o *CameraReader*, que permite a aplicação de filtros, desenvolvidos com auxílio do pacote ImageJ, diretamente aos quadros (*frames*) capturados por uma câmera filmadora. É importante ressaltar que embora o JMF seja distribuido gratuitamente, ele não é um software livre.

## 5 Desenvolvimento

Para realizar alguns dos nossos experimentos com interfaces baseadas na direção do olhar, implementamos um simples jogo da velha. Existem apenas nove regiões de interesse para o usuário neste jogo: as posições do tabuleiro onde podemos marcar os círculos ou as cruzes. Fazendo com que a apresentação visual do tabuleiro ocupe grande parte do monitor, conseguimos relaxar o grau de precisão na detecção da direção do olhar. Além disto, é bastante natural a transposição da interface por mouse para a interface pelo olhar.

Inicialmente, existe um período de aprendizagem, em que usuário deve olhar para cada uma das nove regiões do tabuleiro. A captura de imagens de treinamento é iniciada e finalizada clicando-se o mouse. Depois que uma quantidade pré-definida de exemplos é capturada, o sistema infere uma primeira árvore de decisão, e o usuário pode começar a jogar "com o olhar". Uma jogada é identificada quando o usuário mantém o olhar em uma região do tabuleiro por alguns segundos. Caso o sistema não indique a região correta, o usuário pode acionar o módulo de aprendizagem, clicando duas vezes na região apropriada, enquanto olha. Como no modo de treinamento inicial, o primeiro clique inicia a captura de exemplos, enquanto o segundo a termina. Isso ilustra uma das vantagens de um algoritmo incremental de aprendizagem, como o AdapTree, que permite que o modelo inferido (no caso, uma árvore de decisão), possa ser dinamicamente modificado em parte, e praticamente em tempo real.

Por questões de eficiência, o módulo de aprendizagem não trabalha diretamente sobre a imagem obtida, mas sobre um conjunto de parâmetros dela extraídos pelo módulo de processamento digital de imagens. A figura 2 apresenta a relação entre o módulos desenvolvidos: aprendizagem (AdapTree), extração de atributos (PDI) e jogo da velha (Velha); as bibliotecas auxiliares: Weka, ImageJ, JMF; e o usuário, que se comunica com o sistema através do mouse e da câmera filmadora. A extração de atributos pode ser dividida nas três fases que serão descritas nas próximas seções: pré-processamento da imagem, delimitação da região dos olhos e cálculo de atributos.

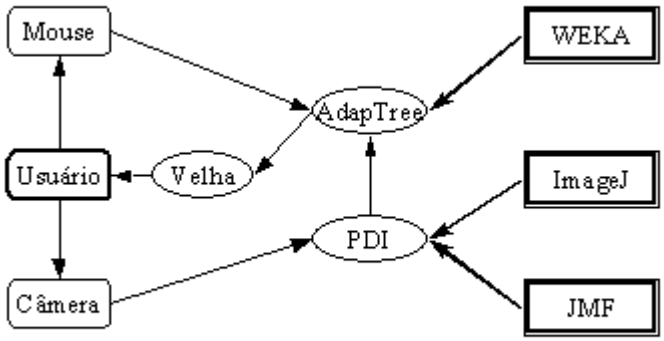


Figura 2: Três módulos principais do protótipo: AdapTree, PDI e Velha

[4] http://java.sun.com/products/java-media/jmf/

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

Tabela 1: Matriz de convolução para Detecção de Bordas

### 5.1 Pré-Processamento

A figura 3 ($a$) mostra a imagem sem nenhum processamento, tal como é extraída por meio de uma câmera WebCam Creative, posicionada acima do monitor. O contraste e o brilho da imagem obtida pela câmera são ajustados apropriadamente para facilitar o pré-processamento: pouco contraste e alto brilho causam um efeito de suavização na imagem. O primeiro passo consiste em transformar a imagem de colorida, RGB, para tons de cinza, seguindo a fórmula $Cinza = (Red * 0.299 + Green * 0.587 + Blue * 0.114)$. Aplicamos em seguida um detector de bordas verticais, cujo núcleo é mostrado na tabela 1. O resultado da aplicação deste filtro é apresentado na figura 3 ($b$). Finalmente, binarizamos a imagem aplicando uma técnica de threshold iterativo, descrita inicialmente por Ridler e Calvard [20], e cuja implementação está disponível no pacote ImageJ. A imagem resultante é mostrada na figura 3 ($c$).

($a$) ($b$) ($c$) ($d$)

Figura 3: Detecção da Região dos Olhos
(a) Original Suavizada (b) Bordas Verticais (c) Threshold Iterativo (d) Região dos Olhos

### 5.2 Delimitação da Região dos Olhos

Na detecção da região dos olhos busca-se marcar, na imagem previamente processada, uma região retangular, contendo apenas os olhos humanos, como mostra a figura 3 ($d$). O algoritmo responsável por esta tarefa baseia-se em 4 heurísticas: (1) existe uma borda vertical facilmente detectável entre íris e esclera (parte branca do olho) [22], (2) as bordas existentes nas regiões imediatamente superiores (testa) e inferiores (maças da face) são bem mais tênues que as dos olhos e somem quase que completamente durante o pré-processamento, (3) a distância entre os dois olhos, quando vistos de frente, pode ser limitada inferior e superiormente e (4) os olhos mantêm-se razoavelmente alinhados em relação ao eixo horizontal.

Embora seja possível encontrar, facilmente, casos em que as heurísticas acima possam ser invalidadas (e.g. franjas sobre a testa, cabeça inclinada), nos testes efetuados com 3 diferentes pessoas, os resultados obtidos foram encorajadores. Um fator importante dessas heurísticas, com exceção da primeira, é que usuário pode, na maioria das vezes, adequar-se facilmente às restrições por elas impostas, mudando, por exemplo, o penteado do cabelo, afastando-se ou aproximando-se da câmera e não inclinando demasiadamente a cabeça para os lados.

A implementação dessas heurísticas ocorre da seguinte forma: primeiramente, utilizamos a heurística (2) para retirar diversas bordas verticais que aparecem no fundo da imagem e no contorno da face. Essa operação é efetuada através da aplicação de um operador morfológico, que varre a imagem, limpando os pontos que não possuem uma região relativamente vazia abaixo ou acima de si, e seu resultado é ilustrado na figura 3 ($d$). Nos experimentos realizados, os valores exatos do tamanho, posição e quantidade mínima de pontos destas regiões foram determinados empiricamente. Na fase seguinte, testamos dois a dois os pontos que restaram e escolhemos os dois primeiros que satisfazem as heurísticas (3) e (4), utilizando para isto mais algumas constantes que foram determinadas através de medições no conjunto de imagens de treinamento (distâncias e inclinações máximas e mínimas).

### 5.3 Cálculo de Atributos

O cálculo de atributos é feito apenas para a região dos olhos, que é dividida através de um reticulado $2 \times 2$. Para cada uma destas sub-regiões, calculamos o total de pixels (massa) e o centro de massa (média dos

| | AdapTree | C4.5 | Backpropagation |
|---|---|---|---|
| Taxa de Acerto (50) | 92.17% | 91.64% | 100% |
| Taxa de Acerto (250) | 95.18% | 95.18% | 100% |
| Taxa de Acerto (Diferentes Contextos) | 68% | 79.33% | 92% |
| Tempo Médio Aprendizagem | 0.36s | 0.33s | 42.23s |

Tabela 2: Resultados Comparativos de Algoritmos de Aprendizagem

valores $x, y$ de cada pixel na região). Somando-se a esses 12 valores: a massa, o centro de massa e a variância em relação ao centro de massa globais, temos 17 atributos que são efetivamente utilizados pelo módulo de aprendizagem, além, é claro, de um valor indicando a posição do tabuleiro que está sendo focada pelo usuário no momento da captura da imagem.

## 6 Experimentos

Realizamos três tipos de experimentos com o protótipo criado. No primeiro experimento, estimamos a taxa de acerto do módulo de delimitação da região dos olhos, utilizando 3 indivíduos diferentes. Enquanto os indivíduos olhavam para diversas regiões da tela, imagens com as regiões dos olhos delimitadas (como na figura 3 ($d$)) eram mostradas na tela e gravadas em disco. Observando essas imagens, é possível indicar, por inspeção, em quais delas a região dos olhos foi corretamente delimitada. A taxa de acerto obtida gira em torno de 90%, sendo que a retro-alimentação, oferecida pelas imagens delimitadas, pode ser utilizada para aumentar esta taxa. Assim, observando as situações em que o sistema erra, o usuário pode ajustar sua posição diante do monitor, para "facilitar" o trabalho do módulo de detecção da região dos olhos.

No segundo tipo de experimento, o usuário simplesmente utiliza o protótipo para treinar e jogar. Os primeiros resultados indicam que precisamos no mínimo de 250 exemplos de treinamento para obtermos uma taxa de acerto razoável durante o jogo. Embora a obtenção dos exemplos seja uma tarefa bastante simples para o usuário (olhar e clicar), constatamos que o sistema ainda é muito dependente da forma exata em que estes exemplos são extraídos. Por exemplo, se o usuário treina o sistema de manhã e vai jogar à noite, a taxa de acerto cai significativamente, e o sistema precisa ser retreinado.

Para verificar de forma menos subjetiva o desempenho do algoritmo de aprendizagem, coletamos imagens de algumas seções de utilização do software e criamos arquivos de treinamento no formato utilizado pelo Weka. Com estes arquivos é possível obter, automaticamente, diversas estimativas de desempenho. Comparamos o AdapTree com os algoritmos C4.5, Backpropagation. Os resultados são apresentados na tabela 2, para arquivos de treinamento com 50 e 250 exemplos e utilizando o método de comparação *RandomSplit*, do Weka, com 10 repetições e um corte de 66%. A tabela mostra também a taxa de acerto quando os arquivos de teste e de treinamento são extraídos em contextos diferentes. É importante ressaltar que o AdapTree é o único, dentre os três comparados, capaz de absorver novas instâncias de treinamento de forma eficiente, sem que haja necessidade de efetuar novo treinamento.

## 7 Conclusões

Neste trabalho foram apresentadas algumas táticas de solução para o problema da detecção da direção do olhar a partir de imagens da face. Foi apresentado também um protótipo de um sistema que permite a interface homem-máquina utilizando imagens capturadas por uma câmera comum, posicionada sobre a tela de um monitor e capturando imagens frontais do usuário. Podemos constatar através deste protótipo que é possível acompanhar a região dos olhos em tempo real, utilizando equipamentos comuns de processamento digital.

As heurísticas utilizadas para detecção da região dos olhos se baseiam em fatores que apresentam baixo índice de variação entre faces de diferentes pessoas, como é o caso do forte contorno existente entre íris e a esclera, a distância média entre as duas pupilas e a relativa suavidade das regiões superior e inferior aos olhos (testa e maçãs da face). Para determinar a região dos olhos utilizando estas heurísticas, foram empregados filtros de detecção de borda vertical, threshold e operadores morfológicos. O algoritmo baseia-se na hipótese de que o usuário se apresenta em uma posição predefinida em relação ao monitor e à câmera (entre 35cm e 55cm). Essa posição pode ser obtida facilmente observando o erro na detecção da região dos olhos, que é apresentado em tempo real para usuário, o qual pode então realizar ajustes em sua posição física, até encontrar a região correta (quando os erros atingem um patamar aceitável).

Os primeiros experimentos com a utilização de transformadas de Hough para a detecção da projeção elíptica da íris indicaram que seria muito difícil obter os resultados indicados por Wang. Parece, a princípio,

que os bons resultados obtidos por Wang devem-se muito à utilização de uma câmera de foco automático, centrada na íris e em zoom. Por isso, optamos por realizar experimentos com aprendizagem de máquina, como sugerido por Stiefelhagen [21] e Baluja [1], que relatam terem obtido boa precisão na detecção do ponto focado, utilizando apenas uma câmera. A diferença de nossa técnica reside no mecanismo incremental de aprendizagem, baseado na tecnologia adaptativa, o que permite ao usuário corrigir eventuais erros constatados, mesmo após a fase de treinamento inicial.

Um resultado indireto deste projeto foi a constatação do bom desempenho e da qualidade das bibliotecas de código aberto, escritas em Java. Além da facilidade de reutilização e portabilidade inerentes à programação orientada a objetos e à linguagem Java, pudemos integrar, de forma relativamente fácil, três complexos pacotes, um de processamento digital de sinais, outro de controle de mídias baseadas em tempo (time-based) e o terceiro para aprendizagem computacional.

Atualmente, a detecção da região dos olhos não reaproveita qualquer informação de processamentos anteriores. Poderíamos, por exemplo, diminuir o espaço de varredura em busca da posição atual dos olhos armazenando conhecimento sobre a posição anterior. Estudar o impacto e viabilidade deste tipo de estratégia nos parece ser uma interessante linha de pesquisa para os trabalhos seguintes.

Outros pontos que podem ser explorados posteriormente e que podem vir a melhorar bastante a eficácia da detecção da região dos olhos incluem a utilização de algoritmos de detecção de regiões de pele humana [8] e a introdução de uma fase de calibração, em que características específicas do usuário possam ser identificadas. Em relação à fase de aprendizagem, seria interessante explorar novos atributos e modelos baseados em técnicas sintáticas de reconhecimento. Além disso, pode-se iniciar a seguir um processo de sofisticação dos algoritmos para que sejam aplicados em problemas mais complexos e que exijam maior precisão, como por exemplo, um jogo de damas. No jogo de damas, poderemos começar a explorar alternativas para os movimentos de arraste e clique do mouse, utilizando técnicas estudadas na área de Interfaces Homem-Máquina (HCI).

Embora muito ainda possa ser feito para melhorar o protótipo aqui apresentando, acreditamos que estes primeiros experimentos ajudaram a mostrar a viabilidade de novas interfaces baseadas na detecção da direção do olhar. É importante ressaltar que todo o projeto está sendo desenvolvido em plataformas de baixo custo, e utilizando software de programas-fonte abertos, o que acreditamos poder resultar em soluções mais acessíveis do que as disponíveis atualmente no mercado.

## Referências


[1] Shumeet Baluja and Dean Pomerleau. Non-intrusive gaze tracking using artificial neural networks. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 753–760. Morgan Kaufmann Publishers Inc., 1994.

[2] B. A. Basseto and J. J. Neto. A stochastic musical composer based on adaptative algorithms. *Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação. SBC-99.*, 3:105–130, Julho 1999.

[3] C. Colombo and A. D. Bimbo. Interacting through eyes. *Robotics and Autonomous Systems*, 19:359–368, 1997.

[4] S. J. Cunningham and G. Holmes. Developing innovative applications of machine learning. *Proc. of Southeast Asia Regional Computer Confederation Conference*, 1999.

[5] Ricardo Luis de Azevedo da Rocha. *Um Método de Escolha Automática de Soluções Usando Tecnologia Adaptativa.* PhD thesis, Escola Politécnica, Universidade de Sao Paulo, São Paulo, Brasil, Julho 2000. [in portuguese].

[6] Carlos Eduardo Dantas de Menezes. *Um Método para a Construção de Analisadores Morfológicos, Aplicado à Lingua Portuguesa, Baseado em Autômatos Adaptativos.* PhD thesis, Escola Politécnica, Universidade de Sao Paulo, São Paulo, Brasil, Julho 2000. [in portuguese].

[7] H. Pistori e J. J. Neto. Adaptree - proposta de um algoritmo para indução de Árvores de decisão baseado em técnicas adaptativas. *Conferência Latino Americana de Informática - CLEI 2002*, Novembro 2002.

[8] L. Enrique Sucar G. Gomez, M. Sanchez. On selecting an appropriate colour space for skin detection. *MICAI 2002: Advances in Artificial Intelligence, Second Mexican International Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence*, pages 69–78, 2002.

[9] A. H. Gee and R. Cipolla. Determining the gaze of face in images. Technical Report CUED/F-INFENG/TR 174, Cambridge University, Trumpington Street, Cambridge CB2 1PZ, England, 1994.

[10] R. Jacob. Eye tracking in advanced interface design. *In Bareld,W. and Furness, T. (Eds.), Advanced Interface Design and Virtual Environments*, pages 258–288, 1995.

[11] A. Fitzgibbon M. Pilu and R. Fisher. Ellipse-specific direct least-square fitting. *IEEE International Conference on Image Processing*, September 1996.

[12] R. A. McLaughlin. Randomized hough transform: Improved ellipse detection with comparison. *Pattern Recognition Letters*, 19(3):299–305, 1998.

[13] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[14] J. J. Neto. Solving complex problems efficiently with adaptative automata. *Conference on Implementation and Application of Automata - CIAA 2000*, July 2000.

[15] J. J. Neto. Adaptative rule-driven devices - general formulation anda case study. *Conference on Implementation and Application of Automata - CIAA 2001*, July 2001.

[16] J. J. Neto and C. A. B. Pariente. Adaptative automata - a reduced complexity proposal. *Conference on Implementation and Application of Automata - CIAA 2002*, pages 161–170, July 2002.

[17] H. Pistori. Portabilidade de aplicativos com interface gráfica. *Anais da Mostra Científica do XXXIII Congresso Internacional de Informática e Telecomunicações - SUCESU 2000*, 2000.

[18] J. Porill. Fitting ellipses and predicting confidence envelopes using a bias corrected kalman filter. *Image and Vision Computing*, 8(1):37–41, 1990.

[19] J. R. Quinlan. Learning decision tree classifiers. *ACM Computing Surveys*, 28(1), 1996.

[20] T. W. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *IEEE transactions on Systems, Man and Cybernetics*, August 1978.

[21] R. Stiefelhagen, J. Yang, and A. Waibel. Tracking eyes and monitoring eye gaze. *Proceedings of the Workshop on Perceptual User Interfaces (PUI'97).*, pages 98–100, 1997.

[22] J. G. Wang and E. Sung. Gaze determination via images of irises. *Image and Vision Computing*, 19(12):891–911, October 2001.

[23] C. Ware and H. Mikaelian. *An Evaluation of an Eye Tracker as a Device for Computer Input*. Elsevier, 1987.

[24] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.