Comparação de Classificadores para a Estimação de Posicionamento de Cabeça

Weverson Vieira do Nascimento
Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal do Pará
Belém-PA, Brasil, 66075-110
E-mail: weverson@ufpa.br

Ronaldo de Freitas Zampolo
Instituto de Tecnologia
Universidade Federal do Pará
Belém-PA, Brasil, 66075-110
E-mail: zampolo@ufpa.br

***Resumo*—Alvo de diversas pesquisas, a estimação de posicionamento de cabeça tem por objetivo determinar a posição de um indivíduo em uma determinada cena. Estratégias de análise de vídeo para estimação de posição de cabeça que funcionem com webcams convencionais têm um papel central em sistemas de rastreamento ocular de baixo-custo. Neste trabalho, os autores comparam três classificadores (baseados em kernel principal component analysis, KPCA; linear discriminant analysis, LDA; e support vector classification, SVC) para estimar a posição de cabeça em uma base de imagens contendo poses discretas. A extração de características de entrada dos classificadores utiliza transformada wavelet de Gabor, cujos aspectos conceituais básicos são apresentados no texto. Testes utilizando uma base de dados disponível publicamente na internet foram realizados. Os resultados mostram que os classificadores possuem acurácia superior a 70% para número total de classes não superior a 15. À medida em que o número de classes aumenta, o desempenho dos classificadores deteriora significativamente: para 93 classes, por exemplo, as acurácias medidas foram de 34%, 33% e 43% para os classificadores baseados em KPCA, LDA, e SVC, respectivamente.**

***Palavras-chave*—Estimação de pose de cabeça. Wavelet de Gabor. KPCA. LDA. SVC.**

I. Introdução

O olhar é um comportamento comum do ser humano que muitas vezes exprime seu foco de atenção. Por conta disso, tornou-se grande o interesse em desenvolver técnicas para estudar o olhar humano [1]. O *gaze-tracking* (rastreio de olhar) objetiva estimar a direção do olhar de um indivíduo e está intimamente relacionado com o *eye-tracking* (rastreio de olho), cuja finalidade é acompanhar o movimento dos olhos em uma imagem.

A estimação de pose (ou de posicionamento) de cabeça é, juntamente com o *eye-tracking*, parte integrante de um sistema de *gaze-tracking*. Um exemplo de representação dos componentes de um sistema de rastreio de olho/olhar está representado na Figura 1. A primeira etapa consiste no pré-processamento da imagem de entrada (correção de contraste, conversão para escala de cinza e redimensionamento, por exemplo) visando a detecção e segmentação da região da face do usuário [2]. Depois, a face segmentada é usada tanto para o rastreio do olho, como também para a estimação da pose de cabeça. Ambas informações permitem calcular a direção de olhar. O interesse deste trabalho é na etapa de estimação de pose, que é explicada na seção seguinte.

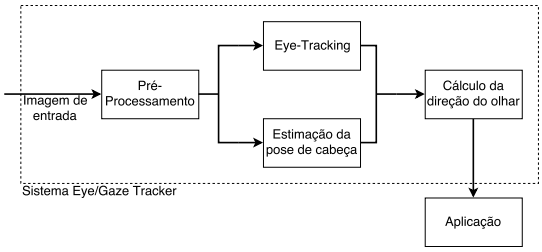

Figura 1: Componentes típicos de um sistema de eye/gaze-tracking. Fonte: o autor

As diversas técnicas de estimação de pose de cabeça têm por objetivo determinar automaticamente a posição da cabeça/face de um individuo presente em uma cena. A Figura 2 mostra os três ângulos que definem o posicionamento de cabeça [1]: o ângulo *pitch* (também chamado de *tilt*) diz respeito ao movimento cima/baixo da cabeça, considerando a rotação em torno de um eixo na horizontal; o ângulo *yaw* (também chamado de *pan*) caracteriza o movimento esquerda/direita da cabeça, considerando a rotação em torno de um eixo na vertical; e o ângulo *roll* é representa o movimento esquerda/direita, considerando fixos os ângulos *pitch* e *yaw* [3].

A técnica proposta por Ji [4] estima a orientação espacial da face humana em tempo real, usando uma câmera monocular e iluminação infravermelha para auxiliar na operação de detecção de pupila. A abordagem também emprega filtragem de Kalman na predição da localização da pupila com base na informação de quadros de vídeo anteriores. Tu, Fu, e Huang [5] usam pirâmides Laplacianas e treinam um modelo tensorial para estimar a posição de cabeça. A estratégia segmenta a face com base na cor de pele, e usa a ponta do nariz como referência para o corte das imagens. Miyama e Matsuda [6] detectam a face usando o método de Viola e Jones [2], adotam o modelo de movimento afim para rastrear a face ao longo do vídeo, e estimam a pose de cabeça avaliando as coordenadas dos olhos e boca em relação ao nariz. Procedimento de detecção de objetos, combinado com o uso da *wavelet* de Haar, determinam tais coordenadas. Chen et al. [7] treinam um interpolador não linear para estimar a pose

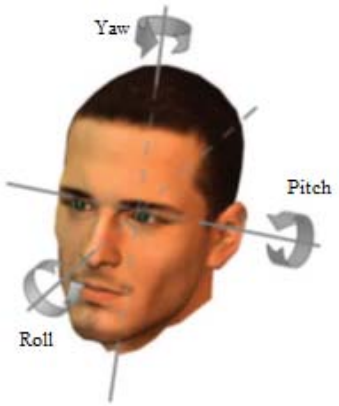

Figura 2: Graus de liberdade do movimento de cabeça. Fonte: [1]

de cabeça, com o auxílio de operadores gradiente *manifold*. Fu e Huang [8] usam grafos de vizinhança e projeções com grafos incorporados supervisionados para determinar a posição da cabeça: as imagens são projetadas em novo subespaço incorporado de dimensões menores que a original e a pose é estimada usando um classificador de vizinhança.

Este trabalho explora variações da técnica de duas etapas proposta por Wu e Trivedi [9]. A primeira das etapas, a de interesse neste trabalho, consiste na extração de características, mediante Transformada *Wavelet* de Gabor seguida de projeção em subespaço para redução de dimensionalidade.

II. Referenciais Teóricos

A. Extração de características com a Transformada Wavelet de Gabor

Diversas técnicas para extração de características empregam análise no domínio da frequência. Apesar do seu relativo sucesso, tais técnicas são incapazes de guardar informação a respeito da localização espacial [9], motivando o uso de Transformadas *Wavelet* (dentre as quais, a de Gabor) [10]. Diferente dos sinais senoidais que formam a base de decomposição da Transformada de Fourier, as transformadas *wavelet* são baseadas em pequenas ondas, chamadas de *ondaletas* ou *wavelets*, que possuem frequência variada e comprimento limitado (suporte finito) [11]. A Figura 3 ilustra a diferença entre as duas transformadas: ela exibe janelas de tempo-frequência de (a) uma função trem de impulsos, (b) uma base senoidal da transformada de Fourier, e (c) uma base de uma transformada *wavelet*, onde a largura e a altura de cada retângulo definem as características de espectrais e espaciais, respectivamente. É possível observar que em (a) não há discriminação da informação da frequência, apesar de existir a informação espacial. Em (b) ocorre o contrário, onde existe a discriminação da informação na frequência, mas não a do tempo. Em (c) tanto a informação espacial quanto a espectral estão presentes. Diversas aplicações multimídia utilizam a extração de características com *wavelet* de Gabor, tais como reconhecimento facial, classificação de textura de imagem e indexação de imagens [12].

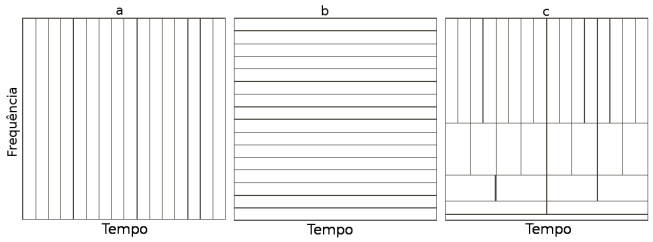

Figura 3: Janelas de tempo-frequência: (a) trem de impulsos, (b) Transformada de Fourier e (c) Transformada *Wavelet*. Fonte: [11]

Conforme exposto por Günther [13], pode-se definir uma *wavelet* mãe por:

$$\psi_{\vec{\kappa}}(\vec{x}) = \frac{\vec{\kappa}^2}{\sigma^2} e^{-\frac{\kappa^2 x^2}{2\sigma^2}} \left[e^{j\vec{\kappa}^\intercal \cdot \vec{x}} - e^{-\frac{\sigma^2}{2}} \right], \tag{1}$$

onde a parte fora do colchete é uma função de envelope Gaussiano com desvio padrão σ, que restringe a onda plana e torna a *wavelet* de Gabor localizável tanto no domínio espacial quanto no da frequência. A primeira parcela dentro do colchete é a onda plana de valor complexo, e a segunda parcela é a componente DC. A família de *wavelets* de Gabor é dada por:

$$\psi_{\vec{\kappa},\phi}(\vec{x}) = \kappa^2 \cdot \psi(\kappa Q(\phi_\nu)^\intercal \cdot \vec{x}), \tag{2}$$

onde $\vec{\kappa} = (\kappa_\zeta \cos(\phi_\nu), \kappa_\zeta \sin(\phi_\nu))$ é a frequência espacial central em coordenadas polares e

$$Q(\phi_\nu) = \begin{pmatrix} \cos\phi_\nu & -\sin\phi_\nu \\ \sin\phi_\nu & \cos\phi_\nu \end{pmatrix}. \tag{3}$$

No domínio do processamento digital, a *wavelet* de Gabor precisa ser discreta e finita, passando a ser definida pelos seguintes parâmetros [14]:

$$\Gamma = (\nu_{max}, \zeta_{max}, \kappa_{max}, \kappa_{fac}, \sigma), \tag{4}$$

onde ν_{max} é a quantidade de direções, ζ_{max} é a quantidade de escalas, κ_{max} é a frequência mais alta da *wavelet*, κ_{fac} é a distância logarítmica entre duas escalas da *wavelet* e σ é a resolução espacial, que mantém fixo o número de oscilações da *wavelet* de Gabor para que seu comprimento de onda tenha o mesmo desvio padrão do envelope Gaussiano de valor $\sigma_{eff} = \frac{\sigma}{k}$. Por padrão, adota-se $\sigma = 2\pi$, levando o termo referente da componente DC ao desaparecimento ($e^{\frac{-\sigma^2}{2}} \approx 10^{-9}$). As direções podem ser obtidas por

$$\phi_\nu = \frac{\nu 180^\circ}{\nu_{max}}, \qquad \{\nu = 0, ..., \nu_{max} - 1\}, \quad \nu \in \mathbb{Z} \tag{5}$$

com frequência central κ_ζ dada por

$$\kappa_\zeta = \kappa_{max} \kappa_{fac}^{\zeta}, \qquad \{\zeta = 0, ..., \zeta_{max} - 1\}, \quad \zeta \in \mathbb{Z}. \tag{6}$$

A Transformada *Wavelet* de Gabor pode, então, ser definida pela convolução da imagem I com a família de *wavelets* de Gabor:

$$G_{\nu,\zeta}(x, y) = I(x, y) * \psi_{\kappa,\zeta}(x, y), \tag{7}$$

onde $G_{\nu,\zeta}(x,y)$ é o resultado da convolução da imagem com a *wavelet* na direção ν e na escala ζ, e $*$ representa a operação de convolução. Apenas as magnitudes são usadas para representação de características, visto que a fase sofre muitos problemas de desalinhamento [15].

A transformada *wavelet* de Gabor é usada para processar cada imagem de entrada. Uma família de *wavelets* sem a componente DC é utilizada para tornar o sistema insensível ao brilho médio. A técnica adota uma análise multirresolução, onde cada par escala-direção constitui uma resolução [16]. Para cada resolução, a imagem resultante é classificada com o uso de *Kernel Principal Component Analysis* (KPCA), *Linear Discriminant Analysis* (LDA), e *Support Vector Classification* (SVC).

B. Redução de dimensionalidade com Kernel PCA

O PCA é a base para análise de dados multivariados que extrai os padrões dominantes dos dados de entrada [17]. É utilizado em várias aplicações, incluindo computação gráfica e visão computacional. Define-se o PCA como a transformação para diagonalizar a matriz de covariância do vetor de dados $\mathbf{x}_k$, com $k = 1, ..., N, \mathbf{x}_k \in \mathbb{R}^N, \sum_{k=1}^{N} \mathbf{x}_k = 0$. A expressão 8 mostra um estimador para a matriz de covariância:

$$\mathbf{C} = \frac{1}{N} \sum_{j=1}^{N_c} \mathbf{x}_j \mathbf{x}_j^{\intercal}, \tag{8}$$

onde $N = \sum_{c=1}^{\ell} N_c$, ℓ é o número de classes e N_c é o número de amostras na classe c.

As novas coordenadas na base de autovetores são chamadas de componentes principais e o subespaço da projeção PCA é um *span* dos principais autovetores da matriz de covariância. Selecionando os autovetores em ordem decrescente dos autovalores associados, minimiza-se a perda de informação na representação dos dados, quando há o descarte de componentes principais. A seleção de quantidades diferentes de autovetores resulta em diferentes níveis de representação. Como o PCA é uma transformação linear, ele não é capaz de lidar com dados não lineares (dados de classes distintas que não podem ser linearmente separados), e por isso uma abordagem generalizada para dados não lineares é necessária. Para isso, primeiro os dados não lineares são mapeados em um espaço F:

$$\Phi : \mathbb{R}^N \to F, \; x \mapsto X, \tag{9}$$

e depois o PCA é aplicado normalmente nos dados projetados em F (dados linearizados). Assumindo que $\sum_{k=1}^{N} \Phi(\mathbf{x}_k) = 0$ (vetores possuem média nula), a nova matriz de covariância é dada por

$$\bar{\mathbf{C}} = \frac{1}{N} \sum_{j=1}^{N_c} \Phi(\mathbf{x}_j) \Phi(\mathbf{x}_j)^{\intercal}. \tag{10}$$

Para facilitar a execução do produto, é feito uso de funções de *kernel*, que já são conhecidas do SVM [18]. Conforme exposto por Schölkopf, Smola e Müller [19], dada a nova função de covariância é necessário encontrar autovalores λ e autovetores $\mathbf{V}$ que solucionem a equação $\lambda \mathbf{V} = \bar{C}\mathbf{V}$. Inserindo a equação 10 nota-se que o espaço de solução de $\mathbf{V}$ está no *span* de $\Phi(\mathbf{x}_1), ..., \Phi(\mathbf{x}_n)$, ou seja:

$$\mathbf{V} = \sum_{i=1}^{N_c} \alpha_i \Phi(\mathbf{x}_i). \tag{11}$$

Considerando que existem coeficientes $\alpha = [\alpha_1, ...\alpha_\ell]$ que satisfaçam a equação 11, definindo uma matriz de *kernel* K $N \times N$ como

$$K := (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) \tag{12}$$

e considerando o sistema equivalente

$$\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot \bar{C}\mathbf{V}) \quad \forall \quad k = 1, ..., N \tag{13}$$

é possível substituir as equações 10 e 11 e a matriz K na equação 13 para chegar em

$$N\lambda K\alpha = K^2\alpha \tag{14}$$

e finalmente as soluções são encontradas resolvendo

$$N\lambda\alpha = K\alpha. \tag{15}$$

A projeção no subespaço KPCA revela uma variedade de estruturas dos dados e por conta disso uma melhor generalização dos mesmos pode ser atingida. Segundo Ham et al. [20], a escolha de diferentes *kernels* pode levar a um melhor aprendizado das variedades de estruturas pelo KPCA. Dentre os *kernels* mais conhecidos podemos citar os polinomiais na forma $k(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x} \cdot \mathbf{y} + \Theta)^d$, o sigmoide $k(\mathbf{x}, \mathbf{y}) = tanh(\gamma \mathbf{x}^T \mathbf{y} + \Theta)$ e a *radial basis function* (RBF) dada por $k(\mathbf{x}, \mathbf{y}) = exp(-\gamma ||\mathbf{x} - \mathbf{y}||^2)$. Θ e γ são parâmetros de cada *kernel*, e uma heurística comum para γ é $\gamma = 1/n_f$ com n_f sendo o número de características [21]. Para Θ a heurística varia de acordo com o *kernel* escolhido. A Figura 4 mostra uma ideia básica de uso do *kernel* no mapeamento de dados não lineares para lineares mostrado na equação 9.

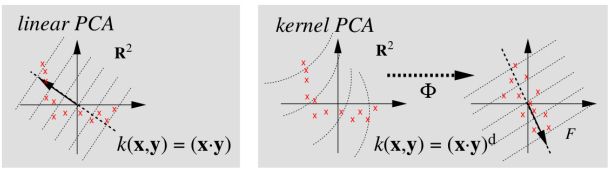

Figura 4: Ideia básica do *Kernel* PCA. Fonte: [19]

C. Linear Discriminant Analysis

Semelhante ao KPCA, o LDA também é uma técnica de redução de dimensionalidade. Entretanto, além do objetivo de encontrar uma projeção que torne os dados de uma classe o mais compacto possível, o LDA tenta separar tanto quanto for possível os dados uma classe da outra. Como mostrado por Tharwat et al. [22], três etapas são necessárias para projetar os dados originais em um espaço dimensional menor:

(a) Calcular a variância entre os elementos de cada classe;
(b) Calcular a variância entre classes;

(c) Criar o espaço dimensional que minimiza o item (a) e maximiza o item (b).

A variância entre os elementos de cada classe representa a diferença entre a média e as amostras de cada classe. Um objetivo do LDA é diminuir essa diferença para que os dados fiquem mais compactados. A variância entre os elementos de cada classe é dada por

$$\mathbf{S}_W = \sum_{c=1}^{\ell} \sum_{i=1}^{N_c} (\mathbf{x}_{c,i} - \boldsymbol{\mu}_c)(\mathbf{x}_{c,i} - \boldsymbol{\mu}_c)^\intercal \tag{16}$$

onde $\mathbf{x}_{c,i}$ representa o elemento i na classe c e

$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbf{x}_{c,i}. \tag{17}$$

A variância entre classes define a distância de separação entre classes distintas. A sua equação é dada por

$$\mathbf{S}_B = \sum_{c=1}^{\ell} N_c (\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^\intercal \tag{18}$$

onde

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i. \tag{19}$$

O termo $(\boldsymbol{\mu}_c - \boldsymbol{\mu})(\boldsymbol{\mu}_c - \boldsymbol{\mu})^\intercal$ representa a distância de separação entre a média da classe i dada por $\boldsymbol{\mu}_i$ e a média total $\boldsymbol{\mu}$. A projeção $\mathbf{W}$ é encontrada através do critério de Fisher, que maximiza o chamado coeficiente de Rayleigh, dado por

$$arg\ \max_{\mathbf{W}} \frac{\mathbf{W}^\intercal \mathbf{S}_B \mathbf{W}}{\mathbf{W}^\intercal \mathbf{S}_W \mathbf{W}} \tag{20}$$

que equivale ao problema de autodecomposição

$$\mathbf{S}_B \mathbf{w}_i = \lambda_i \mathbf{S}_W \mathbf{w}_i, \tag{21}$$

podendo ser reescrito como

$$\mathbf{S}_W^{-1} \mathbf{S}_B \mathbf{w}_i = \lambda_i \mathbf{w}_i. \tag{22}$$

A solução desse problema são os autovalores λ e os autovetores $\mathbf{V}$ da matriz de transformação $\mathbf{W} = \mathbf{S}_W^{-1}\mathbf{S}_B$. A quantidade de autovetores escolhida define o quão reduzido será o novo espaço [23].

Tanto no caso do KPCA quanto do LDA os autovalores representam a magnitude dos autovetores, e estes formam uma base para o espaço em questão. Autovetores com os maiores autovalores carregam mais informações dos dados originais, por isso na maioria das implementações ocorre uma ordenação, para que autovetores $\mathbf{V}$ com os p autovalores mais altos sejam selecionados como constituintes da base do espaço $\mathbf{V_p}$, enquanto os outros autovetores $(\mathbf{V_{p+1}}, \mathbf{V_{p+2}}, ..., \mathbf{V_M})$ são desprezados.

D. Support Vector Classification

O SVM também foi escolhido para testes por se tratar de um algoritmo conhecido para resolver problemas de diversas áreas da ciência e engenharia. O SVM classifica usando uma função linear induzida pelos exemplos de treinamento e que maximiza a distância entre os dados mais próximos de classes distintas [24]. A Figura 5 exibe o exemplo de um hiperplano separando duas classes distintas identificadas pelas cores azul e vermelho. Todas as linhas representam funções que separam os dados de classes distintas, porém, a linha verde é a que maximiza a separação das classes (hiperplano de separação ideal). O espaço utilizado é gerado com o uso de *kernels* como na sessão sobre KPCA.

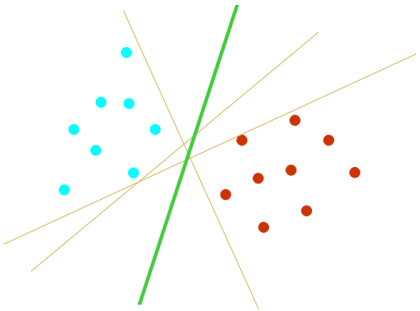

Figura 5: Hiperplano de separação ideal. Fonte: [24]

Conforme Nazemi e Dehghan [25], dado um conjunto de entradas e saídas $D = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N), x_i \in \mathbb{R}^n$ o SVC precisa encontrar a solução do seguinte problema:

$$\min_{\alpha} \frac{1}{2} \alpha^\intercal Q \alpha - e^\intercal \alpha$$

$$\text{sujeito a } \begin{cases} y^\intercal \alpha = 0 \\ 0 \leq \alpha_i \leq Ce, \quad i = 1, ..., n, \end{cases} \tag{23}$$

onde $\alpha \in \mathbb{R}^N$, $C > 0$ é uma constante que controla o equilíbrio entre maximização da margem do hiperplano e minimização dos erros; e é um vetor totalmente preenchido com 1s; e Q é uma matriz $n \times n$ semidefinida com $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, sendo $K(x_i, x_j)$ o *kernel*. A função de decisão do classificador é dada por:

$$f(x) = sgn(\sum_{i=1}^{N} y_i \alpha_i K(x_i, x) + \rho), \tag{24}$$

onde α_i é uma solução otimizada para o problema da equação 23 e ρ é a média de todos os vetores em S dada por

$$\rho = \frac{1}{N_s} \sum_{s \in S} (y_s - \sum_{m \in S} a_m y_m K(x_m, x_s)), \tag{25}$$

sendo S um conjunto de vetores de suporte cujos índices satisfazem $0 < \alpha_i \leq Ce$.

III. Implementação da comparação de classificadores

A. Escolhas para implementação

Para a implementação foi escolhida a linguagem de programação *Python* por ser uma das linguagens mais utilizadas em processamento de imagens [26], e também por ser de gratuita. Com a escolha do *Python*, foi necessária a utilização de bibliotecas compatíveis com a linguagem, sendo que todas as bibliotecas escolhidas são de código aberto. Para o processamento matemático é utilizada a biblioteca *NumPy*, parte integrante do conjunto de bibliotecas *SciPy* [27]. Foi utilizada a biblioteca *Bob's Gabor wavelet routines* para aplicação da transformada *wavelet* de Gabor, cuja implementação é realizada conforme mostrado nos referenciais teóricos [28]. Os classificadores KPCA, LDA e SVC foram implementados utilizando a biblioteca *Scikit-Learn*, que fornece diversos recursos para a aplicação de técnicas de aprendizado de máquina utilizando *Python* [26]. Para o pré-processamento foi utilizado o *OpenCV* [29] e para a exibição gráfica dos resultados foi utilizada a *Matplotlib*, que também faz parte do conjunto de bibliotecas *SciPy* [30].

Dentre as bases de dados pesquisadas foi escolhida a criada para o trabalho de Gourier, Hall e Crowley [31], que foi tornada pública e, por conta disso, possui diversos outros trabalhos que a utilizam em suas implementações, sendo este um dos motivos para sua escolha. Além disso, essa base de dados possui as imagens já rotuladas, no momento de sua aquisição, nos ângulos *pitch* e *yaw* (o ângulo *roll* não é considerado nesta base de dados). A base de dados conta também com um código para navegação entre os arquivos, escrito em linguagem C, que foi adaptado para utilização com a linguagem *Python*.

B. Pré-Processamento, protótipos e classificação

Além das imagens propriamente ditas, o conjunto de dados também dispõe de um arquivo de texto associado a cada imagem contendo as coordenadas da face na imagem [31]. Essa informação é utilizada para pular a etapa de detecção de face, que necessitaria de uma implementação como a do classificador em cascata proposto por Viola e Jones [2]. Após isso, as imagens são redimensionadas para um tamanho fixo de 67x55 *pixels* e depois são convertidas para tons de cinza, tornando-as, assim, prontas para a extração de características com a *wavelet* de Gabor.

Uma *wavelet* mãe com 6 escalas e 8 direções foi utilizada, gerando assim 48 resoluções, e em cada resolução são extraídas a magnitude dos coeficientes complexos resultantes da transformação. Como cada resolução tem como saída uma imagem de tamanho 67×55, para a construção do vetor de características é necessário que a imagem seja vetorizada, ou seja, convertida para o vetor de tamanho 3685, onde cada índice representa uma característica. Para o KPCA o reconhecimento da classe verdadeira é feito utilizando a técnica de casamento, onde cada classe é representada por um vetor de características protótipo [11]. Cada classe c possui um protótipo $\mathbf{m}_c^k$ na resolução k, onde cada protótipo é dado por

$$\mathbf{m}_c^k = \frac{1}{Nc}\sum_{i=1}^{Nc}\mathbf{U}_k\mathbf{x}_{c,i},\ k = 1, ..., 48,\ c = 1, ..., \ell, \qquad (26)$$

sendo $\mathbf{x}_{c,i}$ o vetor de características referente à imagem da pessoa i na classe c e $\mathbf{U}_k$ é o conjunto de autovetores da projeção realizada pelo KPCA na resolução k. Os 10 autovetores com maiores autovalores foram utilizados nesta implementação. No teste, considerando $\mathbf{y}^k$ a projeção KPCA da imagem de teste na resolução k, a sua classe correspondente é determinada pela distância euclidiana desta com o protótipo na sua resolução e com cada classe c:

$$d_c^k = ||\mathbf{y}^k - \mathbf{m}_c^k|| \qquad (27)$$

com a norma euclidiana sendo definida como $||\mathbf{a}|| = (\mathbf{a}^T\mathbf{a})^{\frac{1}{2}}$ e considerando a classe escolhida como a que possui a menor distância euclidiana. Como mostrado por Shen e Bai [32], outras medidas de distância podem ser utilizadas para eventualmente melhorar o desempenho, como a similaridade cosseno. É feita, então, uma votação, onde a classe com mais aparições dentre todas as k resoluções é definida como a classe da imagem de teste. Empates são resolvidos arbitrariamente. Para o SVC e o LDA são considerados diretamente os resultados das predições de cada classificador e feita a comparação entre a classe prevista e a classe real. No caso do KPCA e do SVC o *kernel* utilizado é o RBF, com $\gamma = 1/nf$.

IV. Experimentos e resultados

A. Experimento

O conjunto de dados usado para avaliar os classificadores é composto de um total de 2790 imagens, a partir de 15 pessoas, cada uma variando os ângulos *pitch* e *yaw* entre $-90°$ e $90°$. Para cada pessoa existem duas séries de 93 imagens, onde cada uma das 93 imagens representa uma pose, e cada pose é considerada uma classe. Com isso, para cada pose existem 30 imagens, sendo usadas 15 para treino e 15 para teste em todos os cenários. A quantidade de imagens de treino e teste foi definida através do código de navegação de arquivos fornecido com a base de dados, que facilita a implementação utilizando uma série para treino e a outra para teste. As imagens já estão rotuladas em termos de seus ângulos. Os sujeitos nas imagens são diversos, variando em itens como usar ou não óculos e no seu tom de pele, permitindo uma maior abrangência de resultados.

Para cada tipo de experimento foram feitas 5 execuções com cada um dos classificadores, com conjuntos distintos de classes em cada uma das execuções, e cada execução sendo composta de treino e teste. Em todos os cenários é considerado acerto quando a classe estimada é a correta, ou seja, ambos os ângulos estão corretos. Para cada execução foram salvas as matrizes de confusão de todos os classificadores e também foi coletado o tempo de execução de cada classificador (treino e teste), além das taxas de acerto para cada execução. A taxa de acertos média das execuções é computada e exibida graficamente.

B. Resultados

Para a exibição de alguns resultados são utilizadas matrizes de confusão, que contêm informações sobre as poses reais e estimadas, e os desempenhos podem ser avaliados a partir destas matrizes [33]. Em todas as matrizes apresentadas as colunas de informação representam as poses reais e as linhas representam as poses estimadas. No primeiro cenário são apresentadas as matrizes de confusão para testes com uma quantidade de classes igual a 2. As Tabelas I, II e III apresentam as matrizes de confusão para o KPCA, o LDA e o SVC, respectivamente, em duas das cinco execuções que aconteceram em cada teste, nos casos em que os pares de pose eram (30, 56) e (19, 42). É possível observar que mesmo com uma quantidade pequena de classes existem casos onde nem todas as estimativas são corretas (caso de todos os classificadores nas poses 30 e 56), mas ainda assim a taxa de acertos é alta. Neste cenário de duas classes as taxas médias de acerto foram de $93,8\%$ para o KPCA, $94,6\%$ para o LDA e $95,2\%$ para o SVC. No segundo cenário foram feitos testes com quantidade de classes igual a 15. As matrizes de confusão não são exibidas por questões de espaço disponível, porém as taxas de acerto médias foram de $73,6\%$ para o KPCA, $78,6\%$ para o LDA e $80,6\%$ para o SVC.

Tabela I: Matrizes de confusão do KPCA com quantidade de classes 2

		Pose real			Pose real	
		30	**56**		**19**	**42**
Pose estimada	**30**	13	1	**19**	15	0
	56	2	14	**42**	0	15

Fonte: o autor

Tabela II: Matrizes de confusão do LDA com quantidade de classes 2

		Pose real			Pose real	
		30	**56**		**19**	**42**
Pose estimada	**30**	13	0	**19**	15	0
	56	2	15	**42**	0	15

Fonte: o autor

Tabela III: Matrizes de confusão do SVC com quantidade de classes 2

		Pose real			Pose real	
		30	**56**		**19**	**42**
Pose estimada	**30**	14	0	**19**	15	1
	56	1	15	**42**	0	14

Fonte: o autor

Na Figura 6 é mostrada a evolução das taxas médias de acertos à medida que a quantidade de classes aumenta. Foram feitos testes até uma quantidade de classes igual a 23. É possível observar a tendência de queda das taxas médias de acertos com o aumento da quantidade de classes para todos os classificadores, com destaque para o KPCA que tem uma queda levemente mais acentuada que os demais classificadores. Este resultado é condizente pois apesar de o KPCA ser ótimo para redução de dimensionalidade ele não é tão bom para uma análise discriminante de classes [9].

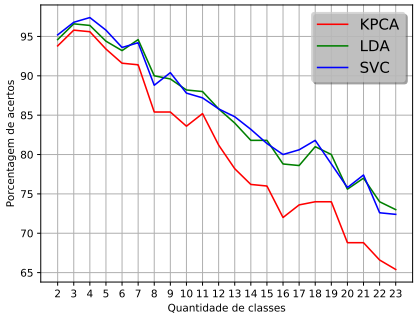

Figura 6: Taxas médias de acerto por quantidade de classes (maior é melhor). Fonte: o autor

Na Figura 7 é mostrada a evolução dos tempos médios de execução à medida que a quantidade de classes aumenta. Novamente foram feitos testes até a quantidade de classes igual a 23. Neste cenário o KPCA e o LDA apresentam um crescimento aproximadamente linear, enquanto o SVC apresenta um crescimento que tende a uma exponencial. Estes resultados servem como estimativa do custo computacional associado a cada um dos classificadores, mostrando que apesar de o SVC possuir uma taxa de acertos ligeiramente melhor que os demais, seu custo computacional pode tornar questionável o seu uso nessa aplicação.

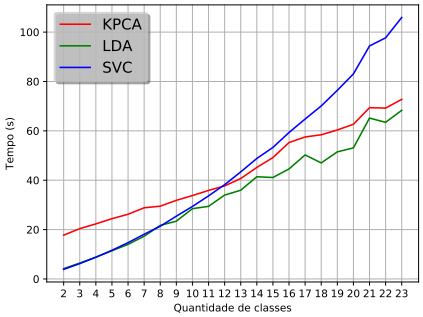

Figura 7: Tempos médios de execução por quantidade de classes (menor é melhor). Fonte: o autor

Dois casos particulares de experimentos foram realizados. Na Figura 8 é mostrado o primeiro deles em que todas as classes do conjunto de dados foram utilizadas. Aqui as taxas de acerto são significativamente mais baixas, onde a quantidade de erros é maior que a de acertos. A taxa de acertos é de 34% para o KPCA, 33% para o LDA e 43% para o SVC. Como comentado anteriormente o SVC obteve uma taxa de acertos maior, porém continua sendo uma estimativa ruim. Para fins de comparação, na implementação de Wu e Trivedi [9] para o KPCA a taxa de acertos foi de 42% e para o LDA foi de 40%, porém os testes foram feitos com 86 classes e 7788 imagens no total, contra 93 classes e 2790 imagens no conjunto de dados utilizado [31]. O SVC, que não foi utilizado na implementação

citada, possui uma taxa de acertos maior, mesmo com a quantidade de imagens inferior no conjunto de dados.

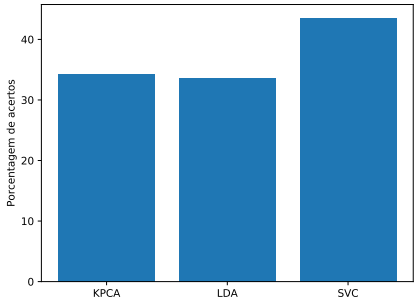

Figura 8: Taxa de acertos para quantidade de classes = 93 (maior é melhor). Fonte: o autor

Um teste com o mesmo conjunto de dados não foi possível pois a abordagem de comparação utilizou um conjunto de dados próprio. Apesar disso, é possível observar que nenhum dos classificadores sozinho é suficiente para obter um nível razoável de precisão, o que é compreensível visto que, no caso do KPCA e do LDA, a análise em subespaço é bastante sensível a ruídos, como o fundo das imagens, o desalinhamento das poses e oclusões causadas por cabelos e óculos. Para o caso do SVC, com base no gráfico mostrado na Figura 7, pode-se afirmar que o custo computacional elevado compromete a eficiência do classificador quando a quantidade de classes é bastante alta.

O segundo caso particular é exibido na Figura 9. Ela exibe a taxa de acertos usando KPCA novamente para o caso onde as 93 classes foram utilizadas, porém, dessa vez, sendo alterada a quantidade de autovetores. É possível observar que a taxa de acertos cresce com o aumento da quantidade de autovetores, entretanto, a partir de 7 autovetores o crescimento passa a ser mínimo (a taxa de acertos fica estagnada próximo de 34%), com o máximo sendo em 10 autovetores. Esse resultado justifica a escolha dessa quantidade de autovetores para realizar os demais testes.

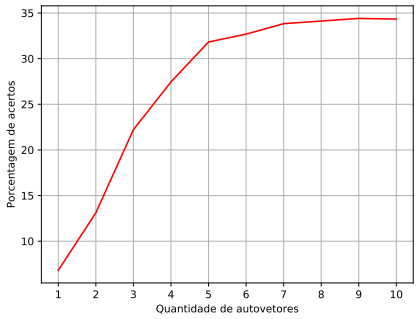

Figura 9: Taxa de acertos do KPCA com quantidades diferentes de autovetores (maior é melhor). Fonte: o autor

V. Conclusões

Neste trabalho foram apresentadas as comparações de desempenho dos classificadores *Kernel Principal Component Analysis*, *Linear Discriminant Analysis* e *Support Vector Classification* para aplicação em estimação de posicionamento de cabeça onde as poses estão discretizadas no conjunto de dados. Foi utilizada a Transformada *Wavelet* de Gabor para extração de características das imagens de entrada dos classificadores. Para o caso do KPCA foram criados vetores de características protótipos que representassem cada uma das classes. O experimento foi realizado com um conjunto de imagens disponível publicamente na internet. A implementação foi feita utilizando softwares livres e o código fonte foi disponibilizado em repositório online.

Com base nos resultados foi possível concluir que, apesar de terem taxas de acerto razoáveis para quantidades pequenas, nenhum dos classificadores sozinho conseguiu um desempenho satisfatório na aplicação em questão quando a quantidade de classes é grande, o que se aproxima do caso real. Testes também mostraram a influência da quantidade de autovetores da projeção KPCA na taxa de acertos, onde uma maior quantidade de autovetores implica em uma porcentagem maior de acertos, porém, limitada a um valor máximo que foi atingido com 10 autovetores. Foi possível concluir também que o KPCA obteve o pior resultado em praticamente todas as análises, enquanto o SVC obteve taxas de acertos maiores que os demais.

Dentre as possibilidades futuras tem-se utilizar as análises de subespaço em conjunto, para verificar se o uso em cascata das projeções pode melhorar o desempenho, utilizando implementações como KPCA+LDA em sequência. Outra opção é realizar uma segunda etapa para refinamento, pois é possível utilizar o resultado desta primeira estimação como entrada para uma segunda etapa, que pode ser a criação de grafos de grupo como adaptação do proposto por Wiskott [34] para reconhecimento facial. Pode-se também avaliar a necessidade ou não do ângulo *roll*, para verificar se a implementação deste é requisito para uma estimação de pose satisfatória com base no contexto objetivado no trabalho, e também estimar a pose de maneira contínua, e não quantizados em poses como apresentado aqui.

Referências

[1] X. Wang, K. Liu, and X. Qian, "A survey on gaze estimation," in *2015 International Conference on Intelligent Systems and Knowledge Engineering*. IEEE, 2015, pp. 260–267.

[2] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[3] (2018) Heading, pitch and roll angles - 3dengine.org. [Online]. Available: http://3dengine.org/Heading,_pitch_and_roll_angles

[4] Q. Ji, "3d face pose estimation and tracking from a monocular camera," *Pattern Recognition*, no. 20, pp. 499–511, 2002.

[5] J. Tu, Y. Fu, and T. S. Huang, "Locating nose-tips and estimating head poses in images by tensorposes," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, vol. 19, no. 1, pp. 90–102, 2009.

[6] M. Miyama and Y. Matsuda, "Integrated face detection, tracking, and pose estimation," in *2012 IEEE 11th International Conference on Signal Processing*, vol. 2, Oct 2012, pp. 1056–1059.

[7] L. Chen, L. Zhang, Y. Hu, M. Li, and H. Zhang, "Head pose estimation using fisher manifold learning," in *2003 IEEE International SOI Conference. Proceedings (Cat. No.03CH37443)*, Oct 2003, pp. 203–207.

[8] Y. Fu and T. S. Huang, "Graph embedded analysis for head pose estimation," in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, April 2006, pp. 6 pp.–8.

[9] J. Wu and M. M. Trivedi, "A two-stage head pose estimation framework and evaluation," *Pattern Recognition*, no. 41, pp. 1138–1158, 2008.

[10] B. Maclennan, "Gabor representations of spatiotemporal visual images," University of Tennessee, Tech. Rep., 11 1994.

[11] R. C. Gonzalez and R. C. Woods, *Processamento Digital de Imagens*, 3rd ed., P. P. Hall, Ed. Pearson Education do Brasil, 2010.

[12] T. Malathi and M. K. Bhuyan, "Performance analysis of gabor wavelet for extracting most informative and efficient features," *Multimed Tools Appl*, no. 76, pp. 8449–8469, 2016.

[13] M. Günther, "Statistical gabor graph based techniques for the detection, recognition, classification, and visualization of human faces," Ph.D. dissertation, Fakultät für Informatik und Automatisierung, Technischen Universität Ilmenau, Ilmenau, Deutschland, 2011. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.725.3165&rep=rep1&type=pdf

[14] A. Anjos, L. E. Shafey, R. Wallace, M. Günther, C. McCool, and S. Marcel, "Bob: a free signal processing and machine learning toolbox for researchers," in *20th ACM Conference on Multimedia Systems (ACMMM), Nara, Japan*, Oct. 2012. [Online]. Available: https://publications.idiap.ch/downloads/papers/2012/Anjos_Bob_ACMMM12.pdf

[15] B. Zhang, S. Shan, X. Chen, and W. Gao, "Histogram of gabor phase patterns (hgpp): A novel object representation approach for face recognition," *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 57–68, Jan 2007.

[16] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, Jul 1989.

[17] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37–52, 1987, proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0169743987800849

[18] B. Boser, I. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifier," in *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, vol. 5, 08 1996.

[19] B. Schölkopf, A. Smola, and K.-R. Müller, *Kernel principal component analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 583–588. [Online]. Available: https://doi.org/10.1007/BFb0020217

[20] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *Proceedings of the Twenty-first International Conference on Machine Learning*, ser. ICML '04. New York, NY, USA: ACM, 2004, pp. 47–. [Online]. Available: http://doi.acm.org/10.1145/1015330.1015417

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] A. Tharwat, T. Gaber, A. Ibrahim, and A. E. Hassanien, "Linear discriminant analysis: A detailed tutorial," in *Ai Communications*, vol. 30, 05 2017, pp. 169–190,.

[23] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed. Springer-Verlag New York, 2009.

[24] S. R. Gunn, *Support Vector Machines for Classification and Regression*, May 1998.

[25] A. Nazemi and M. Dehghan, "A neural network method for solving support vector classification problems," *Neurocomputing*, vol. 152, pp. 369 – 376, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231214014398

[26] S. van der Walt, J. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. Warner, N. Yager, E. Gouillart, T. Yu, and t. scikit-image contributors, "scikit-image: Image processing in python," *PeerJ*, vol. 2, 07 2014.

[27] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011. [Online]. Available: http://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37

[28] M. Günther, D. Haufe, and R. P. Würtz, "Face recognition with disparity corrected Gabor phase differences," in *Artificial Neural Networks and Machine Learning*, ser. Lecture Notes in Computer Science, A. E. P. Villa, W. Duch, P. Érdi, F. Masulli, and G. Palm, Eds., vol. 7552. Springer, Sep. 2012, pp. 411–418.

[29] O. team. (2017) Opencv library. [Online]. Available: https://opencv.org/

[30] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[31] N. Gourier, D. Hall, and J. L. Crowley, "Estimating face orientation from robust detection of salient facial features," in *Proceedings of Pointing 2004*, ICPR, Ed., 2004.

[32] L. Shen and L. Bai, "Gabor feature based face recognition using kernel methods," in *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, May 2004, pp. 170–176.

[33] R. Kohavi and F. Provost, "Special issue on applications of machine learning and the knowledge discovery process," *Machine Learning*, vol. 30, pp. 271–274, 1998.

[34] L. Wiskott, N. Krüger, N. Kuiger, and C. von der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, July 1997.