



Universidade Federal da Bahia

Instituto de Matemática

Departamento de Ciência da Computação

Um Sistema para Detecção de Faces Humanas

Autor: Eric Jardim <eric@im.ufba.br>

Orientador: Sandro S. Andrade <sandros@ufba.br>

Oriente

*“Se oriente, rapaz
Pela constelação do Cruzeiro do Sul
Se oriente, rapaz
Pela constatação de que a aranha
Vive do que tece
Vê se não se esquece
Pela simples razão de que tudo merece
Consideração
Considere, rapaz
A possibilidade de ir pro Japão
Num cargueiro do Lloyd lavando o porão
Pela curiosidade de ver
Onde o sol se esconde
Vê se compreende
Pela simples razão de que tudo depende
De determinação
Determine, rapaz
Onde vai ser seu curso de pós-graduação
Se oriente, rapaz
Pela rotação da Terra em torno do Sol
Sorridente, rapaz
Pela continuidade do sonho de Adão”*

Gilberto Gil

Dedicatória

*ao meu filho, que é a entidade mais importante da minha vida,
a minha esposa, por ter tido dedicação e paciência todos esses anos,
e ao Shakyamuni Buda por nos iluminar todos os dias.*

Sumário

Oriente	
Dedicatória	
1 Introdução	p. 6
1.1 O Problema da Detecção de Faces Humanas	p. 7
1.2 Computação Visual e Visão Computacional	p. 7
1.3 Aplicações da Detecção de Faces Humanas	p. 8
2 Trabalhos Correlatos	p. 10
2.1 Classificação das Abordagens	p. 10
2.2 Abordagens Baseadas nas Características Facias	p. 10
2.2.1 Análise de Baixo Nível	p. 11
2.2.2 Análise de Características	p. 12
2.2.3 Modelos Ativos de Formas	p. 13
2.3 Abordagens Baseadas na Imagem	p. 13
2.3.1 Abordagens Estatísticas	p. 14
2.3.2 Redes Neurais	p. 15
2.3.3 Subespaços Lineares	p. 16
3 Fundamentos	p. 17
3.1 <i>Principal Components Analysis</i>	p. 17
3.1.1 O Espaço Amostral	p. 17
3.1.2 Evento Amostral	p. 18

3.1.3	Variância e Covariância	p. 18
3.1.4	Matriz de Covariância	p. 19
3.1.5	Componentes Principais	p. 21
3.2	A Distância de Mahalanobis	p. 23
3.2.1	Definição	p. 24
3.2.2	Interpretação Geométrica	p. 25
3.2.3	Distância Normalizada de Mahalanobis	p. 26
3.3	k -médias	p. 27
3.3.1	Agrupamento em Classes	p. 27
3.3.2	Função de Distorção	p. 28
3.3.3	O Algoritmo de k -médias	p. 28
3.4	<i>Multi-Layer Perceptron</i>	p. 29
3.4.1	Classificação <i>a posteriori</i>	p. 29
3.4.2	A topologia de uma rede MLP	p. 29
3.4.3	MLP no Reconhecimento de Padrões Visuais	p. 30
4	O Sistema de Detecção de Faces	p. 32
4.1	Visão Geral	p. 32
4.2	Fase de Treinamento	p. 32
4.2.1	Conjunto de Treinamento	p. 34
4.2.2	Pré-Processamento	p. 34
4.2.2.1	Correção da Iluminação	p. 34
4.2.2.2	Normalização do Histograma	p. 37
4.2.2.3	Equalização do Histograma	p. 37
4.2.3	Agrupamento do Conjunto de Treinamento	p. 37
4.2.4	Cálculo das Distâncias Relativas	p. 39
4.2.4.1	A primeira componente de distância \mathcal{D}_1	p. 39

4.2.4.2	A segunda componente de distância \mathcal{D}_2	p. 40
4.2.5	Treinamento da rede MLP	p. 40
4.3	Fase de Classificação	p. 40
4.3.1	Extração de Janelas	p. 42
5	Implementação	p. 43
5.1	Modelo de Implementação	p. 43
5.1.1	image	p. 44
5.1.2	newmat	p. 44
5.1.3	cluster	p. 44
5.1.4	ann	p. 44
5.1.5	detector	p. 44
5.2	Linguagem de Programação e Compilador	p. 45
5.3	Bibliotecas	p. 45
5.4	Hospedagem do Projeto	p. 45
6	Resultados	p. 46
7	Conclusão	p. 48
7.1	Trabalhos Futuros	p. 48
	Referências	p. 49

1 *Introdução*

A capacidade de ver é um atributo peculiar a apenas algumas espécies de animais, permitindo que estes tenham uma melhor percepção do ambiente a sua volta. Tal sentido capacita os seres de modo que possam reagir melhor a certos eventos, ao contrário de seres que não a possuem. O ser humano é um deste seres e sem a visão, certamente teria mais dificuldades em realizar algumas tarefas como ler ou identificar objetos.

Não é surpreendente que com a evolução tecnológica e a crescente automatização de processos, o ser humano procure formas de dotar um aparelho eletrônico como um computador de tal capacidade, para que este possa na ausência de um homem, realizar tarefas que só este poderia. Tal façanha é um problema clássico da inteligência artificial e um sonho da ficção científica.

Apesar de ser uma tarefa extremamente simples para um ser humano comum, a detecção e o reconhecimento de uma face ou qualquer outro objeto está longe de ser algo trivial para uma máquina. Mesmo possuindo um sensor visual como uma câmera de vídeo ou um digitalizador de fotos, o motivo da dificuldade está na complexidade dos fatores visuais envolvidos no cenário tais como tamanho, posicionamento, rotação, oclusão e iluminação dos objetos a serem detectados ^[1].

Entretanto, vários trabalhos e ramos de pesquisa ^[1] procuram meios para contornar este problema, alguns com resultados bastante satisfatórios. Apesar de ser um problema interessante e com prováveis aplicações, o reconhecimento de faces é um problema distinto, mas intimamente ligado a detecção. Para reconhecermos uma face, é necessário antes detectar em que lugar do cenário se encontra a face. Neste projeto iremos apenas abordar a detecção de faces.

1.1 O Problema da Detecção de Faces Humanas

O problema de detecção pode ser posto da seguinte forma: dada uma imagem, a cores ou em tons de cinza, bidimensional digitalizada, verificar a existência de faces humanas, dispostas frontalmente, em alguma sub-região da imagem, e em caso positivo, identificar a região em termos de posição relativa e tamanho na imagem. A figura 1 dá uma boa idéia de como funciona um detector de faces com uma imagem digital como entrada.

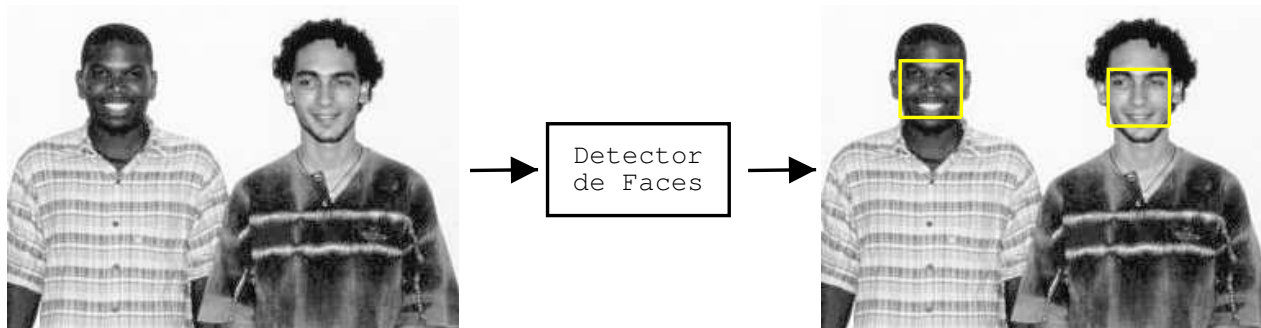


Figura 1: exemplo de como funciona um Detector de Faces

1.2 Computação Visual e Visão Computacional

A *Computação Visual* é uma linha de pesquisa da Computação muito interessante e com uma diversidade de aplicações. Esta área abrange a geração, o processamento e a análise de dados visuais com o auxílio do computador. Basicamente, a Computação Visual pode ser dividida em três grandes sub-áreas: a *Computação Gráfica*, *Processamento de Imagens* e *Visão Computacional* [2].

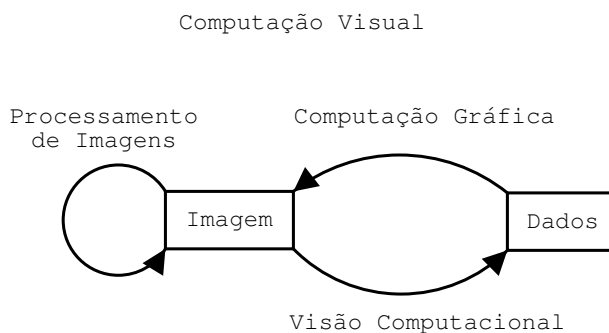


Figura 2: os domínios de cada área da Computação Visual

Como pode ser visto na figura 2, as sub-áreas da Computação Visual atuam em diferentes domínios. O *Processamento de Imagens* atua no domínio da imagem, geralmente transformando uma imagem em outra derivada da primeira. A *Computação Gráfica* é bastante conhecida pelo público leigo, devido aos efeitos especiais no cinema geralmente em filmes de ficção. A Computação Gráfica tenta, a partir de um modelo geométrico, reproduzir uma imagem ou uma sequência de animação com um nível de realismo desejado. Já a *Visão Computacional* não tão conhecida, mas nem por isso menos interessante, procura, a partir de uma imagem, extrair informações do domínio visual para algum modelo de dados, seja ele topológico, geométrico ou físico [2].

O problema de detecção de faces a partir de uma imagem se enquadra como um problema da *Visão Computacional* (ver figura 3), já que tem por objetivo extrair alguma informação do domínio visual. Em particular a detecção de faces é um tópico ligado ao *reconhecimento de padrões* e *aprendizado de máquina*, relacionando-se assim, com *Inteligência Artificial* e *Robótica*.

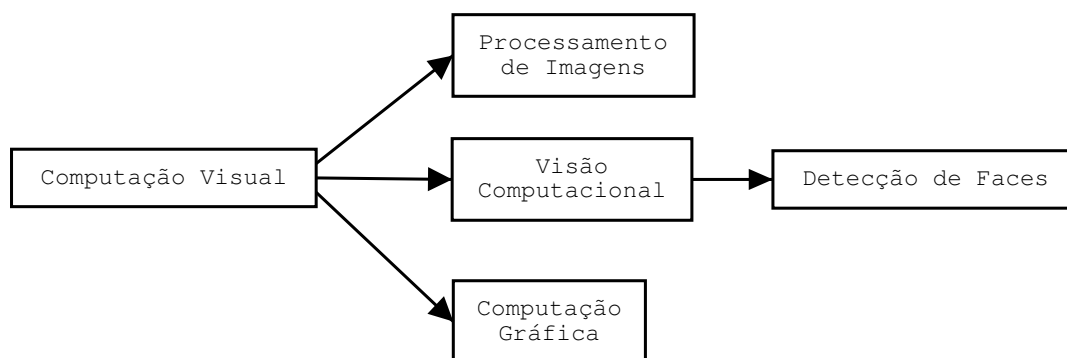


Figura 3: detecção de Faces é um problema da Visão Computacional

1.3 Aplicações da Detecção de Faces Humanas

A detecção de faces é um ramo com grandes possibilidades de aplicação, inclusive quando unida a outras técnicas. A princípio, a aplicação motivadora deste projeto seria um detetor de face para complementar um sistema de reconhecimento de faces como em Andrade^[3], localizando uma face em uma imagem numa primeira etapa, para então ser reconhecida. Entretanto, pode-se identificar algumas possíveis aplicações:

- Incrementar consultas baseadas a conteúdo em um banco de imagens, possibilitando

consultas mais poderosas, inclusive na internet ^[1];

- Auto-ajuste de *webcams* em video conferências ^[1];
- Potencializar um sistema de segurança, identificando a presença de pessoas próximas ao local^[4];
- Auxílio na contagem ou monitoramento de pessoas que passam em frente a uma câmera^[4];
- Pode ser utilizado para auto-rotacionar fotos tiradas de câmeras digitais.

Independente destas aplicações diretas, a detecção de faces se enquadra como um tipo de reconhecimento de padrões em uma imagem. Logo, as técnicas utilizadas neste problema, podem ajudar também no reconhecimento ou detecção de qualquer outro padrão visual, sendo assim válida a exploração de novas abordagens.

2 *Trabalhos Correlatos*

Neste capítulo serão apresentadas algumas técnicas existentes para detecção de faces, eventualmente evidenciando seus métodos, vantagens e desvantagens.

2.1 *Classificação das Abordagens*

Segundo um detalhado levantamento feito em ^[1], as abordagens de detecção de faces então existentes podem ser classificadas em dois grandes grupos: nas abordagens *baseadas nas características faciais* e nas abordagens *baseadas na imagem*, como pode ser visto no diagrama da figura 4.

A principal diferença entre estas duas categorias, está no enfoque dado às características faciais. Nas abordagens baseadas em características faciais, são utilizadas técnicas que exigem explicitamente o conhecimento prévio sobre a face, como a posição dos olhos, boca, narinas e orelhas ou como a cor da pele. Já nas técnicas baseadas na imagem, o problema é em geral tratado como um problema de reconhecimento de padrões, onde tenta-se localizar a face em alguma sub-região da imagem. Neste caso, também é necessário informações prévias, mas normalmente utilizando um conjunto de treinamento para “alimentar” o detector, tornando a solução mais genérica e flexível.

2.2 *Abordagens Baseadas nas Características Facias*

As primeiras técnicas para detecção de face utilizavam métodos baseados nas características facias. Entretanto, além de serem métodos bastante complexos por envolver uma série de fatores antropométricos, tais técnicas possuem fortes restrições quanto a disposição da face e o cenário, oferecendo uma baixa flexibilidade.

Novamente, pode-se dividir esta abordagem em três categorias: na Análise de Baixo

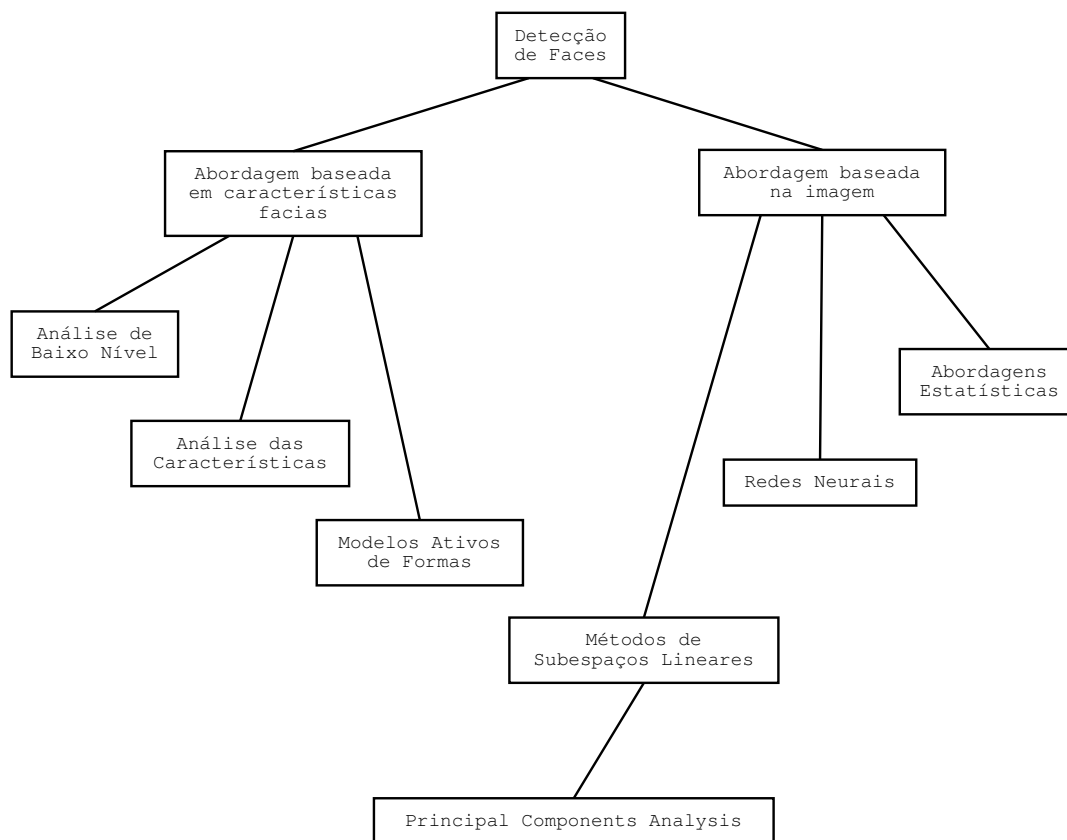


Figura 4: abordagens existentes para detecção de faces.

Nível, a Análise de Características e os Modelos Ativos de Formas.

2.2.1 Análise de Baixo Nível

O principal objetivo das técnicas que envolvem análise de baixo nível é separar a imagem de entrada em regiões prováveis e improváveis em conter uma face. Para isso, tais técnicas utilizam diferentes fatores para realizar a separação. Dentre elas, é possível evidenciar algumas^[1]:

- **Bordas:** Utilizam métodos para detecção de contornos nas imagens. Em seguida procuram por contornos parecidos com o do rosto ou dos olhos, bocas e narizes. Alguns métodos utilizam proporções, a exemplo da *razão áurea*^[1].
- **Tons de Cinza:** Estes métodos procuram por padrões faciais escuros como sobrancelhas, pupilas, narinas e lábios, ou por padrões mais claros como a testa e bochechas.
- **Cores:** A pele humana, apesar de possuir uma certa variação de tonalidade entre

alguns indivíduos, possui pouca variação na crominância, sendo possível descartar certas regiões da imagem com cores improváveis. Entretanto tais métodos enfrentam algumas dificuldades com problemas de iluminação.

- **Movimento:** Nos casos em que a detecção é feita a partir de uma entrada de vídeo, se o cenário for relativamente estático, é muito simples detectar a presença de objetos móveis. Além disto, para identificar a face nas regiões candidatas, também é necessário detectar contornos ou utilizar medidas como o fluxo óptico como visto em [5] apud [1].
- **Medidas Generalizadas:** O cérebro humano, antes de reconhecer algum padrão visual, realiza processamento *pre-atento*¹ das informações visuais para então realizar atividades de reconhecimento visual de maior nitidês^[6]. Com isso foram propostos operadores para realizar processamento pre-atento em sistemas de computação visual. Um deles é o operador de simetria que calcula uma medida de simetria em cada pixel de uma imagem pela contribuição dos pixels adjacentes^[1].

2.2.2 Análise de Características

A utilização pura dos métodos de análise de baixo nível pode levar a detecções incorretas devido a ambiguidades^[1]. Para complementar este tipo de solução, podem ser utilizados conhecimentos geométricos de maior nível das características faciais, que podem ser divididos na Busca por Características e na Análise de Constelações.

- **Busca por Características:** A busca por características é justamente procurar pelas partes da face como olhos, boca, orelhas e nariz. Normalmente depois de localizadas cada uma destas, são realizadas medidas relativas entre as partes para garantir a presença de um rosto. Das características procuradas, a mais comum é o par de olhos^[1].
- **Análise de Constelações:** Uma *constelação* de características faciais é a forma como estas estão dispostas espacialmente. Para dar maior flexibilidade a detecção, é criado um modelo estatístico da disposição das características contornando variações na translação, rotação e escala da face até um certo ponto.

¹Apesar do olho humano poder fixar o olhar sobre um objeto e exergá-lo por completo, somente uma parte da imagem, justamente a que é projetada na *fóvea*, possui alta definição. Com isso o olhar atento se dá sobre a fóvea, e o processo de reconhecimento é precedido pela seleção de quais direções o olho deve focar, assim como na leitura, onde o olhar caminha de uma palavra a outra. Esta decisão é um processamento *pre-atento*^[6]

2.2.3 Modelos Ativos de Formas

Os *modelos ativos de formas* são técnicas de alto nível na análise das características faciais. Um modelo ativo, em geral, quando posicionado perto de uma característica facial, ele interage e se deforma aos poucos até tomar a sua forma. Basicamente existem três tipos de modelos ativos: *Snakes*, Modelos Deformáveis e *Point Distributed Models*

- ***Snakes*:** *Snakes* são modelos ativos do contorno facial. Quando um *snake* é iniciado perto de uma face, ele tende a se deformar até tomar a forma do rosto. A convergência do snake desde o início até a face depende de uma medida de *energia* que representa o quanto o modelo difere de uma face. De fato são consideradas as energias *interna* e *externa* ao snake, de modo que o equilíbrio ocorre quando a energia total é minimizada. Entretanto este método é suscetível a mínimos locais.
- **Modelos Deformáveis:** Os *modelos deformáveis* são modelos ativos das partes faciais como olhos, nariz e boca. O modelo é criado utilizando parâmetros de deformação, que são variados até minimizar a energia. Os tipos mais comuns de modelos deformáveis são o modelo de par de olhos e o modelo de boca.
- ***Point Distributed Models (PDM)*:** Um PDM pode ser considerado como um *snake* melhorado. Diferente do anterior, o contorno do PDM é formado por um conjunto discreto de pontos significativos. Então, a partir de um conjunto de treinamento de faces, a variação destes pontos é verificada e parametrizada. Em seguida é utilizado o método PCA² para realizar uma redução dimensional nos parâmetros.

2.3 Abordagens Baseadas na Imagem

Uma das desvantagens dos métodos baseados nas características faciais é o fato de que dependem de um modelo que tenta representar tais características. Com isso, estes métodos estão sujeitos a imperfeições nestes modelos, além de possuir uma baixa flexibilidade. Recentemente, surgiram novos métodos de busca facial em que a face é tratada como um padrão visual qualquer.

Geralmente, as abordagens baseadas na imagem utilizam métodos de aprendizado a partir de um conjunto de exemplos, conhecido como *conjunto de treinamento*, com o

²O método de PCA, ou *Principal Components Analysis* será abordado no próximo capítulo

objetivo de classificar a imagem como “face” ou “não face”. Além disto, normalmente são feitas buscas exaustivas em posição e tamanho, a depender da necessidade da problema.

Podemos dividir as abordagens baseadas na imagem em três categorias: *Abordagens Estatísticas*, *Redes Neurais* e *Métodos de Subespaços Lineares*.

2.3.1 Abordagens Estatísticas

As abordagens estatísticas vêm se mostrando bastante promissoras na área de reconhecimento de padrões. Algumas propostas utilizam Teoria da Informação, *Support Vector Machine* e a Regra de decisão de Bayes, também conhecida como *Maximum a posteriori*^[1]. Sobre esta última abordagem, vale a pena ressaltar dois trabalhos interessantes e com bons resultados, ambos publicados por Kanade e Schneiderman^[7, 8].

Em ambos os métodos, a estratégia principal é representar as imagens do conjunto de treinamento por coeficientes mais simples para serem utilizados no cálculo de uma probabilidade *a posteriori*. Para realizar a detecção, basta utilizar a probabilidade

$$P(\text{face} \mid \text{imagem})$$

ou seja, dada uma “*imagem*”, qual a probabilidade de ser uma “face”. Entretanto, é utilizada a regra de decisão de Bayes, na forma de teste da razão

$$\frac{P(\text{imagem} \mid \text{face})}{P(\text{imagem} \mid \overline{\text{face}})} > \frac{P(\overline{\text{face}})}{P(\text{face})} = \lambda$$

onde a expressão do lado esquerdo da inequação são as probabilidades *a priori*, e λ pode ser considerado como a sensibilidade do detector e pode ser obtido experimentalmente. A diferença entre os trabalhos está na simplificação do modelo para estimar as probabilidades *a priori*.

No primeiro trabalho, as imagens do conjunto de treinamento são divididas em regiões menores que por simplificação não possuem dependência estatística. Também são realizadas várias simplificações no modelo, além de projeção das imagens num espaço de dimensão menor (PCA), codificação esparsa dos vetores e quantização em um número finito de padrões. Depois de realizadas todas as simplificações, dada uma imagem, basta contar a ocorrência de cada padrão no conjunto de treinamento e estimar as probabilidades *a priori*^[7].

Já no segundo trabalho foi utilizada a transformada de *wavelet*. Uma vantagem de utilizar *wavelets* é fato de não produzir redundância, além de separar a imagem em faixas localizadas em orientação, frequência e escala. Novamente, temos simplificações no modelo e quantização dos coeficientes de *wavelet*. Apesar dos resultados deste trabalho terem sido inferiores no caso de faces frontalmente dispostas, teve um bom resultado com faces de perfil^[8].

2.3.2 Redes Neurais

A utilização direta da imagem como entrada de uma *Rede Neural* é um método de detecção interessante. Nos últimos anos, a utilização de redes neurais no reconhecimento de padrões cresceu bastante, além do surgimento de novos de modelos mais complexos e robustos^[1].

Um trabalho com bons resultados foi proposto por Rowley et al.^[9, 10], onde a rede neural recebe imagens em tons de cinza, realçadas em termos de iluminação e equalização do histograma. Um algoritmo de busca exaustiva é realizado em várias posições tamanhos. Além disso, o sistema trata problemas como a sobreposição de múltiplas detecções realizando operações lógicas como AND e OR.

Uma característica forte do sistema é o seu esquema de treinamento que foi adaptado de ^[4]. Numa primeira etapa o sistema é alimentado com um conjunto de treinamento inicial de faces aumentado com imagens refletidas ou levemente rotacionadas. Então, depois de uma bateria de testes, os falsos positivos são automaticamente incorporados no conjunto de treinamento, melhorando a qualidade do detector e evitando a seleção manual de contra-exemplos.

Posteriormente, em Rowley et al.^[11], é proposto um melhoramento do detector, sendo agora capaz de identificar faces rotacionadas no plano. Apesar de possuir uma taxa de detecção em torno dos 80%, apresentou baixos números de falsos positivos além de abrir novos rumos no reconhecimento de padrões independente a rotações^[1].

2.3.3 Subespaços Lineares

Em geral, os métodos dos Subespaços Lineares tentam representar as imagens de faces humanas como um subespaço menor do espaço das imagens utilizando técnicas estatística de análise multivariada como PCA, *Linear Discriminant Analysis (LDA)* e *Factor Analysis (FA)* ^[1].

Apesar de ser um sistema de reconhecimento de faces, Turk e Pentland^[12] propõem uma medida de erro chamada *Distance from Face-Space (DFFS)*, que representa o quanto uma imagem projetada no espaço das faces distingue em módulo da original e pode ser utilizada como uma indicação da presença de uma face. Uma detector semelhante é proposto em Moghaddam et al.^[13, 14], mas utilizando PCA nas características como olhos, boca e nariz.

Um método melhorado utilizando PCA e Redes Neurais do tipo *Multi-Layer Perceptron (MLP)* é mostrado em Sung e Poggio^[4]. Assumindo que as faces não se dispõem perfeitamente em um elipsóide multi-dimensional, é proposto dividir o conjunto de treinamento em conjuntos menores que possuem uma melhor representação com PCA em detrimento do conjunto todo. Além disto, como já foi previamente comentado, o sistema utiliza um interessante sistema de realimentação de falsos positivos, e pre-processamento tanto nas imagens de exemplo do conjunto de treinamento quanto nas que estão prestes a ser submetidas a detecção.

Além disso, o método utiliza métricas não euclidianas para realizar medidas entre padrões e os centróides dos agrupamentos de faces, combinadas com métricas euclidianas para medidas em relação aos agrupamentos de “não face”. Tais medidas são passadas como entrada de uma rede MLP, que tem por objetivo classificar o padrão como “face” ou “não face”. Neste trabalho será adotado o método proposto por Sung e Poggio^[4], com algumas simplificações.

3 Fundamentos

Antes de um aprofundamento nos detalhes de projeto e implementação do detector de faces proposto neste texto, será mostrada primeiro uma simples introdução dos modelos e técnicas que serão utilizados para construí-lo.

3.1 *Principal Components Analysis*

*Principal Components Analysis*¹ (PCA) é uma técnica estatística da área de análise multivariada ^[15] amplamente utilizada, que tem por objetivo detectar, em um conjunto de amostras multidimensionais os subespaços lineares mais significativos. Encontrando as direções principais, projetam-se as amostras num espaço de dimensão menor, desprezando as direções de menor significância, realizando assim uma redução de dimensionalidade.

3.1.1 O Espaço Amostral

Considere que as amostras estão imersas num espaço vetorial real de dimensão fixa n , e podemos representar uma amostra $\Gamma \in \mathbb{R}^n$ por um vetor

$$\Gamma = (x_1, x_2, \dots, x_n)$$

onde cada coordenada $x_i \in \mathbb{R}$, ou seja, cada x_i pertence a um subespaço do \mathbb{R}^n de dimensão 1. Neste caso, tais subespaços são os espaços gerados pelos vetores canônicos

$$\begin{aligned} e_1 &= (1, 0, \dots, 0) \\ e_2 &= (0, 1, \dots, 0) \\ &\vdots \\ e_n &= (0, 0, \dots, 1) \end{aligned}$$

¹A técnica PCA também é conhecida como Transformada de Karhunen-Loève

formando uma base para representar todo o espaço. De fato, qualquer conjunto de n vetores linearmente independentes do \mathbb{R}^n é suficiente para formar uma base.

3.1.2 Evento Amostral

Para a detecção de padrões, o objetivo de se ter um modelo estatístico é a de identificar um subconjunto do espaço amostral \mathbb{R}^n que representa a presença do padrão procurado. Tal subconjunto é chamado geralmente de *evento amostral* e podemos denominar um evento E como a presença do padrão que procuramos num ponto amostral.

Em um problema de reconhecimento de padrões deseja-se saber, dado um vetor qualquer, se existe a presença ou não de um padrão. Em muitos casos, os elementos do espaço amostral que contém o padrão formam conjunto de densidade infinita, e na prática é impossível representar todos os possíveis elementos. Como esta abordagem é baseada a exemplos, tem-se a princípio um subconjunto amostral $\Omega \subset E$ para representar o evento. Neste caso o conjunto das amostras Ω pode ser chamado de *Training Set*, ou seja, *Conjunto de Treinamento*.

Como se sabe *a priori* a forma geométrica nem as propriedades de E pode-se estimar a presença do padrão a partir das informações obtidas de Ω . Pode-se obter um método de classificação onde, dada uma amostra $\Gamma \in \mathbb{R}^n$, qual a probabilidade que ela pertença a E ou seja

$$P\{E \mid \Gamma\} > \theta$$

onde θ é um limiar que determina a sensibilidade do classificador. Um tipo conhecido de detector estatístico são os *Classificadores Bayesianos*, que tentam a partir de modelos de geração de documentos, estimar qual classe geradora é a mais provável^[16]. Entretanto, neste projeto serão utilizadas redes neurais para realizar a classificação, como será visto mais adiante.

3.1.3 Variância e Covariância

A covariância é uma medida clássica da estatística que reflete a relação de interdependência entre duas dimensões de uma amostra multi-dimensional. O conceito de covariância é uma extensão do conceito tradicional de variância.

Dado um conjunto amostral discreto não vazio $\Omega = \{w_1, w_2, \dots, w_m\}$ e seja X uma variável aleatória que associa cada amostra w_i a um número real X_i , podemos dizer que a variância de X é dada por

$$\text{var}(X) = \frac{1}{m} \sum_{i=1}^m (X_i - \bar{X})^2 \quad (3.1)$$

onde \bar{X} é a média dos X_i , ou seja

$$\bar{X} = \frac{1}{m} \sum_{i=1}^m X_i$$

Da mesma forma, dado um conjunto amostral Ω , mas agora com pelo menos duas variáveis aleatórias associada a cada amostra, digamos X e Y , é possível então definir a covariância entre X e Y como

$$\text{cov}(X, Y) = \frac{1}{m} \sum_{i=1}^m (X_i - \bar{X})(Y_i - \bar{Y}) \quad (3.2)$$

e utilizando as equações 3.1 e 3.2, não é difícil de mostrar que

$$\text{cov}(X, Y) = \text{cov}(Y, X)$$

e

$$\text{cov}(X, X) = \text{var}(X)$$

3.1.4 Matriz de Covariância

Voltando ao caso em que cada amostra é representada por um vetor aleatório $\Gamma \in \mathbb{R}^n$, podemos escrever Γ como

$$\Gamma = (x_1, x_2, \dots, x_n)$$

Sabemos que o conjunto de treinamento Ω é um subconjunto do \mathbb{R}^n e suponha que existem M amostras Γ_i com $1 \leq i \leq M$. Suponha também que este conjunto de treinamento é uma boa representação do evento E , logo o comportamento das amostras reflete uma boa estimativa do comportamento de todo o evento. Obviamente, a qualidade da detecção depende da escolha de boas amostras para o conjunto de treinamento.

Então calculando a média das amostras, é possível definir Ψ , também denominado de

vetor médio, como sendo

$$\Psi = \frac{1}{M} \sum_{i=1}^M \Gamma_i = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$$

e considerando novas amostras Φ_i como sendo

$$\Phi_i = \Gamma_i - \Psi$$

Logo, se temos um vetor Γ qualquer do conjunto de treinamento, obtemos um vetor novo Φ e reescrevendo em termos de novas coordenadas como

$$\Phi = (y_1, y_2, \dots, y_n) = (x_1 - \bar{x}_1, x_2 - \bar{x}_2, \dots, x_n - \bar{x}_n) = \Gamma - \Psi$$

Obviamente o conjunto Ω_Ψ formado pelos Φ_i é o mesmo conjunto de treinamento Ω , transladado pelo vetor $-\Psi$, ou seja, para recuperar qualquer vetor no novo espaço, basta somar o vetor Ψ e tem-se novamente um vetor em Ω . Realizar esta translação garante que o conjunto Ω_Ψ tem média zero, ou seja, que a média de cada coordenada y_j de cada vetor amostral Φ_i é zero. Então, para calcular a covariância² entre quaisquer duas dimensões y_j e y_k basta calcular

$$\text{cov}(y_j, y_k) = \frac{1}{M-1} \sum_{i=1}^M \Phi_{i,j} \Phi_{i,k}$$

onde $\Phi_{i,j}$ é a j -ésima coordenada do vetor Φ_i , e $\Phi_{i,k}$ é a k -ésima coordenada do vetor Φ_i .

Com isto, considerando A como a matriz dos vetores Φ_i dispostos em coluna

$$A = (\Phi_1, \Phi_2, \dots, \Phi_M) = \begin{pmatrix} \Phi_{1,1} & \Phi_{2,1} & \dots & \Phi_{M,1} \\ \Phi_{1,2} & \Phi_{2,2} & \dots & \Phi_{M,2} \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_{1,n} & \Phi_{2,n} & \dots & \Phi_{M,n} \end{pmatrix}$$

²De fato há uma leve diferença entre o fator multiplicativo $\frac{1}{M-1}$ e o fator $\frac{1}{M}$ definido anteriormente na fórmula de covariância. Isto ocorre porque possuímos apenas uma amostra do evento E , logo o primeiro fator dá uma estimativa melhor do valor real de covariância. Este valor também é chamado de *covariância amostral*^[17].

pode-se definir a matriz $C \in \mathbb{R}^{n \times n}$ como

$$\begin{aligned}
C &= \frac{1}{M-1} AA^T \\
&= \frac{1}{M-1} \begin{pmatrix} \Phi_{1,1} & \Phi_{2,1} & \cdots & \Phi_{M,1} \\ \Phi_{1,2} & \Phi_{2,2} & \cdots & \Phi_{M,2} \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_{1,n} & \Phi_{2,n} & \cdots & \Phi_{M,n} \end{pmatrix} \begin{pmatrix} \Phi_{1,1} & \Phi_{1,2} & \cdots & \Phi_{1,n} \\ \Phi_{2,1} & \Phi_{2,2} & \cdots & \Phi_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_{M,1} & \Phi_{M,2} & \cdots & \Phi_{M,n} \end{pmatrix} \\
&= \frac{1}{M-1} \begin{pmatrix} \sum_{i=1}^M \Phi_{i,1} \Phi_{i,1} & \sum_{i=1}^M \Phi_{i,1} \Phi_{i,2} & \cdots & \sum_{i=1}^M \Phi_{i,1} \Phi_{i,n} \\ \sum_{i=1}^M \Phi_{i,2} \Phi_{i,1} & \sum_{i=1}^M \Phi_{i,2} \Phi_{i,2} & \cdots & \sum_{i=1}^M \Phi_{i,2} \Phi_{i,n} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^M \Phi_{i,n} \Phi_{i,1} & \sum_{i=1}^M \Phi_{i,n} \Phi_{i,2} & \cdots & \sum_{i=1}^M \Phi_{i,n} \Phi_{i,n} \end{pmatrix} \\
&= \begin{pmatrix} \text{cov}(y_1, y_1) & \text{cov}(y_1, y_2) & \cdots & \text{cov}(y_1, y_n) \\ \text{cov}(y_2, y_1) & \text{cov}(y_2, y_2) & \cdots & \text{cov}(y_2, y_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(y_n, y_1) & \text{cov}(y_n, y_2) & \cdots & \text{cov}(y_n, y_n) \end{pmatrix}
\end{aligned}$$

e desenvolvendo termo a termo, é visível que cada elemento $c_{j,k}$ da matriz C é a covariância entre as dimensões y_j e y_k . Tal matriz é chamada de *matriz de covariância* de Ω no ponto Ψ ^[15].

É interessante notar que como $\text{cov}(y_j, y_k) = \text{cov}(y_k, y_j)$, a matriz C tem a propriedade de ser uma matriz simétrica, ou seja, $C = C^T$. Isto será importante para garantir a existência das componentes principais.

3.1.5 Componentes Principais

O ponto chave do método PCA é encontrar as direções que melhor representam a variação entre as dimensões, também conhecidas como *Componentes Principais*. Então, a partir de um certo número de amostras suficientemente representativas, pode-se estimar as direções de maior variação dos dados assim como mostra o exemplo da figura 5.

Para encontrar as componentes principais, deve-se primeiro extrair os autovetores da matriz de covariância C . Os autovetores de C são todos vetores v que satisfazem a equação

$$Cv = \lambda v \tag{3.3}$$

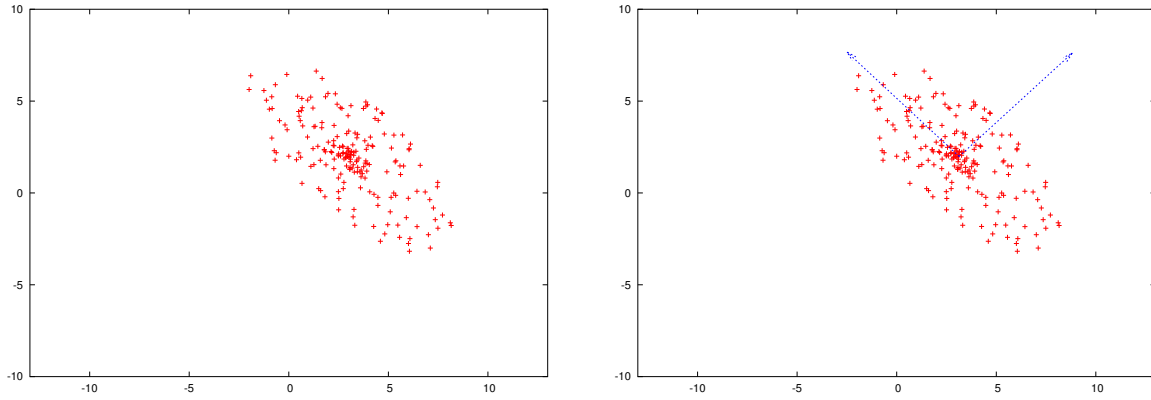


Figura 5: exemplo em duas dimensões da utilização do método PCA para encontrar as direções principais de variação, a partir de amostras estatísticas.

onde λ é o autovalor associado a v . Os autovalores satisfazem a equação

$$\det(C - I\lambda) = 0 \quad (3.4)$$

Como C é uma matriz real simétrica, isto garante que C possui n autovetores e n autovalores associados λ_i ^[18], onde

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$$

Cada autovalor λ_i é uma solução da equação 3.4 e associado a ele existe um autovetor unitário u_k , satisfazendo a equação, 3.3. Os autovetores $\{u_1, u_2, \dots, u_n\}$ são ortogonais, formando uma base ortonormal do \mathbb{R}^n .

Para realizar a redução de dimensionalidade das amostras Γ_i para um subespaço de dimensão $m < n$, serão utilizados os autovetores u_k que possuem os maiores autovalores associados. Tais vetores serão as componentes principais. Para realizar isto, basta calcular as projeções dos vetores Φ_i no subespaço gerado por $\{u_1, u_2, \dots, u_m\}$, ou seja, dado um vetor amostral Γ qualquer temos que

$$\omega_k = \langle u_k, \Gamma - \Psi \rangle = \langle u_k, \Phi \rangle$$

os coeficientes ω_k são as coordenadas do um novo vetor $\vec{\omega}$

$$\vec{\omega} = (\omega_1, \omega_2, \dots, \omega_m)$$

em um novo espaço reduzido de apenas m dimensões.

Como foram desprezados alguns autovetores, quando um vetor projetado é novamente reescrito no espaço amostral de dimensão n , através de uma transformação inversa, há uma “perda de informação” devido ao desprezo das componentes menos significativas^[13]. Felizmente, é possível demonstrar que esta perda é mínima. Para recuperar o vetor reconstruído Φ_f , basta realizar uma combinação linear

$$\Phi_f = \sum_{k=i}^m \omega_k u_k$$

Assim, para estimar o erro gerado pela perda de informação com a redução de dimensionalidade, basta calcular o comprimento do vetor $\Phi - \Phi_f$

$$\varepsilon^2 = \|\Phi - \Phi_f\|^2$$

Certamente, se o erro de cada amostra do conjunto de treinamento é mínima, então isto é um bom sinal de que a redução dimensional foi bem aplicada ao problema.

3.2 A Distância de Mahalanobis

Com a noção das componentes principais, sabe-se que a variação dos dados em cada uma das componentes é proporcional ao autovetor λ_i associado a direção u_i . Então, dado um ponto amostral qualquer, a *Distância de Mahalanobis* será utilizada para indicar se tal ponto está “perto” ou “longe” do conjunto de treinamento.

Nos casos em que o PCA é bem utilizado, nota-se que entre as direções principais, algumas direções possuem maior significância que outras. Isto é refletido diretamente na “forma” que o conjunto de treinamento toma. Por isso, para criar um método de classificação para determinar se um ponto do espaço amostral está de acordo com os dados do conjunto de treinamento, deve-se que utilizar uma métrica que leve em consideração a forma do conjunto.

A métrica euclidiana apesar de ser bastante natural e intuitiva pode acabar cometendo certas “injustiças” se for utilizada como método de classificação, a partir de um limiar baseado na distância. O exemplo na figura 6 exemplifica bem que pontos equidistantes ao centróide segundo a métrica euclidiana, possuem visivelmente distâncias relativas diferentes.

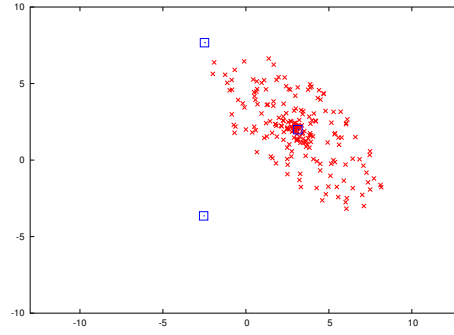


Figura 6: pontos que têm a mesma distância euclidiana em relação ao centróide podem estar a uma distância relativa diferente do conjunto de treinamento.

3.2.1 Definição

Pode-se definir o quadrado da *Distância de Mahalanobis* entre uma amostra Γ e o centróide Ψ como sendo ^[15, 19]

$$\mathcal{M}^2(\Gamma) = \|\Gamma - \Psi\|_{\mathcal{M}}^2 = (\Gamma - \Psi)^T C^{-1} (\Gamma - \Psi)$$

onde C^{-1} é a inversa da matriz de covariância. Note bem que se $C = I$, a distância de Mahalanobis se iguala a distância euclideana.

$$(\Gamma - \Psi)^T (\Gamma - \Psi) = \langle (\Gamma - \Psi), (\Gamma - \Psi) \rangle = \|\Gamma - \Psi\|^2$$

Considerando ν como a matriz dos autovetores u_k dispostos em coluna, pode-se decompor a matriz de covariância ^[20] como

$$\Lambda = \nu^T C \nu$$

onde $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ é a matriz diagonal dos autovalores. Como ν é uma matriz mudança de base formada por vetores ortogonais, sua inversa $\nu^{-1} = \nu^T$ porque $\nu \nu^T = \nu^T \nu = I$, pelo fato de ν ser um operador real ortogonal^[20]. Além disso, sabendo que

$$\Lambda^{-1} = \text{diag}\left(\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \dots, \frac{1}{\lambda_n}\right) \quad (3.5)$$

conclui-se que

$$\begin{aligned}
 \Lambda &= \nu^T C \nu \\
 I &= \nu^T C \nu \Lambda^{-1} \\
 \nu &= C \nu \Lambda^{-1} \\
 C^{-1} \nu &= \nu \Lambda^{-1} \\
 C^{-1} &= \nu \Lambda^{-1} \nu^T
 \end{aligned} \tag{3.6}$$

Logo, considerando novamente o vetor transladado $\Phi = \Gamma - \Psi$ e utilizando 3.5 e 3.6 pode-se reescrever o quadrado da distância de Mahalanobis como

$$\begin{aligned}
 \mathcal{M}^2(\Gamma) &= \Phi^T C^{-1} \Phi \\
 &= \Phi^T \nu \Lambda^{-1} \nu^T \Phi \\
 &= \varphi^T \Lambda^{-1} \varphi \\
 &= \sum_{i=1}^n \frac{z_i^2}{\lambda_i}
 \end{aligned} \tag{3.7}$$

onde $\varphi = (z_1, z_2, \dots, z_n) = \nu^T \Phi$, ou seja o vetor Φ escrito nas coordenadas da base dos autovetores.

3.2.2 Interpretação Geométrica

Para ter uma noção geométrica do significado da distância de Mahalanobis, é necessário analisar os vetores com coordenadas da base ν . Então, fixando uma distância r e verificando qual o lugar geométrico da equação

$$\mathcal{M}(\Gamma) = \sqrt{\sum_{i=1}^n \frac{z_i^2}{\lambda_i}} = r \tag{3.8}$$

podemos ter uma noção de como pontos equidistântes, segundo esta nova distância, estão dispostos.

Para o caso de $n = 2$ vemos que

$$\frac{z_1^2}{\lambda_1} + \frac{z_2^2}{\lambda_2} = r^2 \tag{3.9}$$

$$\frac{z_1^2}{r^2 \lambda_1} + \frac{z_2^2}{r^2 \lambda_2} = 1 \tag{3.10}$$

que nada mais é que uma equação de uma elipse nas coordenadas z_1 e z_2 na base ν . Logo, estendendo para dimensões superiores vemos, especialmente no caso $n = 3$, que o lugar geométrico de 3.8 é um elipsóide n -dimensional. A figura 7 dá uma noção dos pontos equidistântes ao centróide segundo a distância de Mahalanobis.

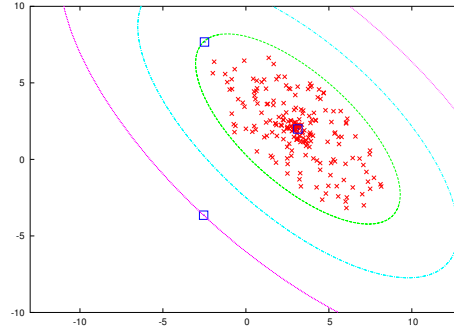


Figura 7: noção de distância ao conjunto de treinamento utilizando elipses

Isto também pressupõe que o “formato” de um conjunto amostral onde se deseja utilizar a distância de Mahalanobis, deva se assemelhar a um elipsóide n -dimensional. Obviamente, tais distribuições são casos particulares e justamente pelo fato de não se saber *a priori* como se comporta a distribuição das faces, esta será representada como uma união de elipsóides n -dimensionais. Mais adiante será visto como realizar a divisão do conjunto de treinamento em classes menores.

3.2.3 Distância Normalizada de Mahalanobis

Como será adotada uma abordagem de *agrupamento* no conjunto de treinamento, é preciso estabelecer como será realizada a divisão em classes menores. De fato isto será abordado mais adiante, mas antes é preciso ter uma noção da *Distância Normalizada de Mahalanobis*.

A noção de distância de Mahalanobis está intimamente ligada com a distribuição Gaussiana n -dimensional ^[19] de um conjunto amostral Ω

$$P\{\gamma \mid \Omega\} = \frac{\exp\left(-\frac{1}{2}(\gamma - \bar{\gamma})^T \Sigma^{-1}(\gamma - \bar{\gamma})\right)}{\sqrt{(2\pi)^n |\Sigma|}}$$

onde Σ é matriz de covariância de Ω no ponto $\bar{\gamma}$.

Como $P\{\gamma \mid \Omega\}$ assume valores no intervalo $[0, 1]$, $-\log P\{\gamma \mid \Omega\}$ assume valores entre

$[0, +\infty]$, de tal forma que

$$\begin{aligned} P\{\gamma \mid \Omega\} \rightarrow 0 &\Rightarrow -\log P\{\gamma \mid \Omega\} \rightarrow +\infty \\ P\{\gamma \mid \Omega\} \rightarrow 1 &\Rightarrow -\log P\{\gamma \mid \Omega\} \rightarrow 0 \end{aligned}$$

e com isso, pode-se interpretar $-\log P\{\gamma \mid \Omega\}$ como uma espécie de distância de γ em relação a Ω . Definimos então a *Distância Normalizada de Mahalanobis*

$$\mathcal{M}_n(\Gamma) = \frac{1}{2} (n \log 2\pi + \log |C| + \mathcal{M}^2(\Gamma))$$

como sendo $-\log P\{\Phi \mid \Omega\}$, neste caso Ω é de fato o conjunto de treinamento, ou como será visto mais adiante, uma parte dele.

Como sugerido em ^[4], pode-se utilizar a distância Mahalanobis para realizar medidas de proximidade em algoritmos de agrupamentos convencionais, para tentar estimar grupos de amostras que possuem melhor representação Gaussiana em detrimento de todo o conjunto de treinamento Ω . O motivo da normalização é para garantir a estabilidade do algoritmo de k -Médias adaptativo que será utilizado.

3.3 k -médias

O algoritmo de k -médias, também conhecido como k -means, é uma técnica de agrupamento de vetores bastante utilizada, devido a sua simplicidade e eficiência^[21]. Em geral, as técnicas de agrupamento são úteis em áreas tais como compressão de dados, quantização de vetores, reconhecimento e classificação de padrões.

3.3.1 Agrupamento em Classes

Dado $\Omega = \{x_1, x_2, \dots, x_M\}$ um conjunto com M vetores do \mathbb{R}^n , e seja k um inteiro positivo, pode-se dividir Ω em k classes $w = \{w_1, w_2, \dots, w_k\}$ de tal forma que a para cada classe w_j existe um *centro* c_j e para todo x_i

$$x_i \in w_j \iff (x_i - c_j)^2 \leq (x_i - c_h)^2 \quad \forall h \neq j$$

ou seja, x_i pertence a classe w_j que possui o centro c_j mais próximo a x_i . Logo w é determinado pelos centros c_j .

3.3.2 Função de Distorção

Dado $\Omega = \{x_1, x_2, \dots, x_M\}$ e k , como escolher os centros c_j de tal forma que w minimize a *função de distorção*^[22]

$$D(w) = \sum_{i=1}^M L(x_i, w) = \sum_{j=1}^k \sum_{x_i \in w_j} \|x_i - c_j\|^2$$

Em outras palavras, considerando Ω e k , quais são os centros c_j que intuitivamente, representam melhor o conjunto Ω com apenas k elementos. Em particular, no caso $k = 1$ a solução deste problema é o vetor médio.

3.3.3 O Algoritmo de k -médias

De fato existem várias versões otimizadas do algoritmo de k -médias^[22], mas em geral, a idéia do algoritmo pode ser expressa de uma forma mais direta e intuitiva, como mostrado em^[23].

As entradas do algoritmo de k -médias são: o conjunto Ω que pretende ser dividido classes, o número de classes k e k centros iniciais, que podem ser escolhidos aleatoriamente, de preferência que estejam distantes entre si e ainda assim próximos dos pontos do conjunto, para garantir uma convergência rápida. Além disso, o critério de parada depende de um número máximo de iterações ou um erro mínimo ϵ entre os valores de $D(w)$. Dados os valores de entrada, o algoritmo é o seguinte:

1. Atribui x_i à classe w_j com centro c_j mais próximo.
2. Calcula novos centros c_j com sendo o vetor médio de cada classe w_j .
3. Considerando esta como sendo a l -ésima passada do algoritmo, calcula a função de distorção $D_l(w)$.
4. Se um número predeterminado de passos foi atingido, ou se $|D_l(w) - D_{l-1}(w)| < \epsilon$ termina o algoritmo com as classes w_j como saída, senão volta para 1.

Se for iniciado com um conjunto qualquer de centros, o algoritmo converge eventualmente para um mínimo local^[21]. Na figura 8 pode ser visto um exemplo da utilização do algoritmo.

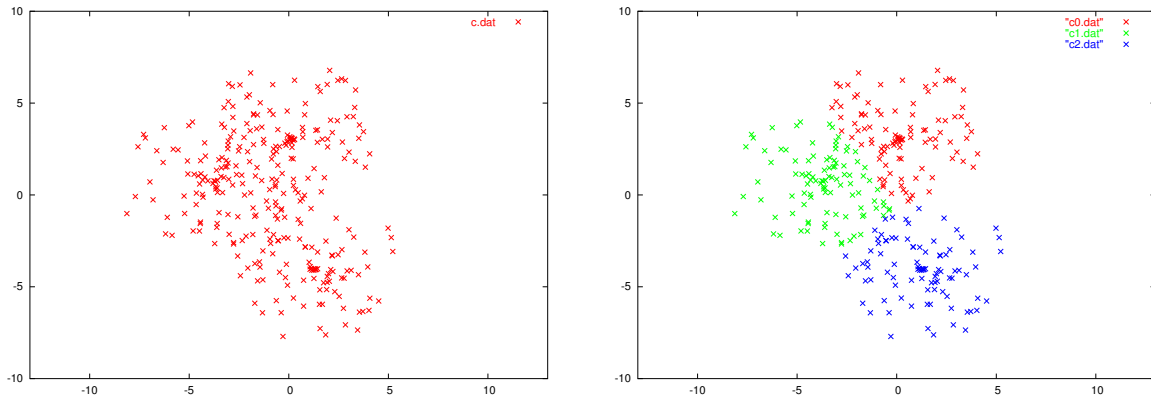


Figura 8: um exemplo de agrupamento feito utilizando o algoritmo de k -means com $k = 3$.

3.4 Multi-Layer Perceptron

O *Multi-Layer Perceptron* (MLP) é um tipo comum de rede neural muito utilizada em problemas de reconhecimento e classificação de padrões estatísticos. Uma propriedade importante das redes MLP, é o fato dela aproximar bem o resultado obtido em uma classificação *a posteriori* utilizando a regra de decisão de Bayes^[24].

3.4.1 Classificação *a posteriori*

Em um problema de classificação num espaço amostral dividido em n classes previamente estabelecidas $\{w_1, w_2, \dots, w_n\}$, onde se conhece apenas alguns elementos de cada classe, que formam o conjunto de treinamento Ω , deseja-se estimar a probabilidade *a posteriori* de cada classe w_j

$$P(w_j | \gamma) \quad (3.11)$$

que representa a probabilidade do ponto amostral γ pertencer a classe w_j .

Como se sabe, é possível estimar 3.11 a partir das probabilidades *a priori* $P(\gamma | w_j)$, como nos modelos de classificação bayesiana. Além disso, se as probabilidades *a priori* forem boas, a classificação é ótima. As redes MLP aproximam o resultado de uma classificação bayesiana, além de capturar algumas propriedades não-lineares^[25].

3.4.2 A topologia de uma rede MLP

Em geral, a topologia de um MLP é dividida em *camadas* ordenadas de neurônios, sendo elas a *camada de entrada*, as *camadas escondidas* e a *camada de saída*. As conexões

só ocorrem entre os neurônios de camadas adjacentes. As camadas de entrada e saída são adaptadas para receber e indentificar respectivamente, um vetor característico e a classe a que pertence. A figura 9 mostra um exemplo de MLP com duas camadas escondidas.

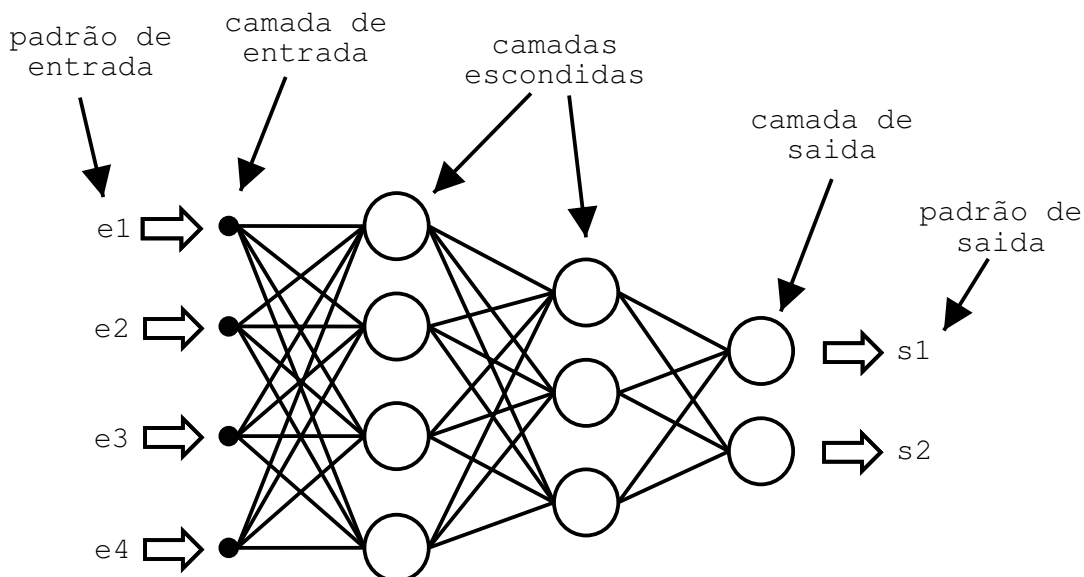


Figura 9: um exemplo da topologia de uma rede MLP

3.4.3 MLP no Reconhecimento de Padrões Visuais

No caso de reconhecimento de padrões visuais, é suficiente dividir o espaço amostral em duas classes: a classe em o padrão está presente, e a classe em que não está. Adicionalmente, a imagem de entrada deve ser antes traduzida para a forma de vetor característico, a depender da implementação. Um exemplo da utilização de uma simples rede MLP para reconhecer um losango no centro do plano pode ser visto na figura 10

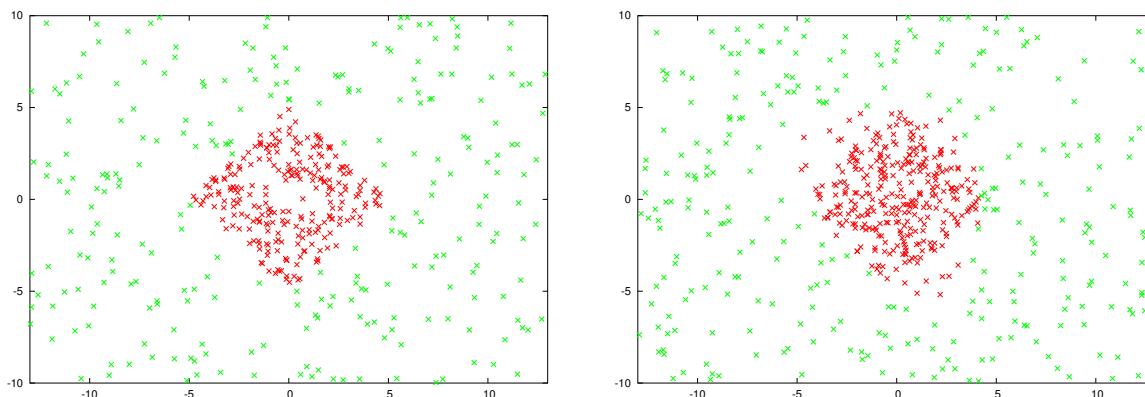


Figura 10: (esq) o conjunto de treinamento com amostras aleatórias de dentro e fora do losango e (dir) um conjunto de testes submetido a classificação pela rede MLP

Uma fase fundamental no ciclo de vida da rede neural é a *fase de treinamento*. A qualidade do reconhecimento depende profundamente tanto da estrutura quanto a escolha do conjunto de treinamento. Na literatura é comentado que é importante ter um número equivalente de representantes de cada classe no conjunto de treinamento, para evitar problemas como *parcialidade*^[26].

4 *O Sistema de Detecção de Faces*

4.1 Visão Geral

O sistema de detecção de faces descrito neste trabalho foi baseado no sistema proposto por Sung e Poggio^[4] que utiliza uma rede neural MLP para classificar imagens em tons de cinza como sendo “face” ou “não face”. A rede neural recebe como entrada distâncias relativas a agrupamentos gerados a partir das amostras do conjunto de treinamento.

O sistema pode ser dividido em dois momentos: a Fase de Treinamento, onde são realizadas as preparações necessárias a detecção como o agrupamento das amostras e o treinamento da rede neural e a Fase de Classificação, que contempla a busca em posição e tamanho, por janelas¹ que serão submetidas a classificação pela rede neural, além própria classificação e rotulação das imagens.

4.2 Fase de Treinamento

A fase de treinamento contempla todas as atividades necessárias para se obter a rede neural treinada e apta a classificar uma janela de 19×19 *pixels*, que será reorganizada lexicograficamente em um vetor de intensidades com 361 dimensões. Contudo, a rede neural não classifica diretamente uma imagem a partir de suas intensidades em tons-de-cinza, mas a partir de distâncias relativas aos agrupamentos de amostras. Os agrupamentos são gerados a partir de um conjunto sinteticamente aumentado a partir do conjunto de treinamento, utilizando-se um algoritmo de agrupamento derivado do *k*-médias, que utiliza, adaptativamente, a distância de Mahalanobis de cada agrupamento.

¹As janelas correspondem às sub-imagens das imagens de entrada do sistema

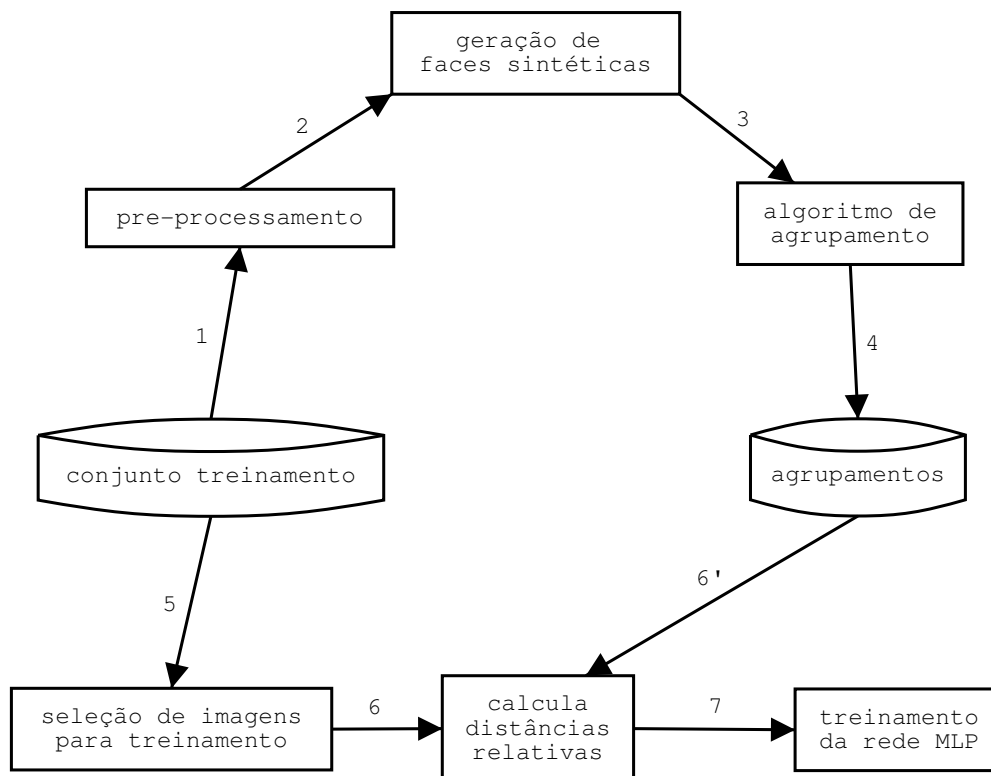


Figura 11: diagrama da fase de treinamento

Como mostrado no diagrama da figura 11, a fase de treinamento pode ser dividida nas seguintes etapas:

1. As imagens do conjunto de treinamento são submetidas a técnicas de pré-processamento, como a correção da iluminação, normalização e equalização do histograma, para diminuir aspectos de iluminação ou digitalização que possam dificultar a detecção.
2. Para melhorar a representação da distribuição do espaço das faces no espaço das imagens, são geradas sinteticamente algumas imagens adicionais a partir das imagens provindas do conjunto de treinamento, utilizando reflexão horizontal.
3. O conjunto aumentado de faces é submetido a um algoritmo de agrupamento, dividindo o conjunto num certo número de agrupamento de faces. Adicionalmente são obtidos o centróide do agrupamento e a matriz de covariância amostral em volta do centróide.
4. Os agrupamentos e suas informações adicionais, são armazenados em disco para posterior utilização no treinamento da rede neural, na fase de classificação e num possível reagrupamento com na realimentação de falsos positivos.

5. Numa segunda etapa, algumas imagens do conjunto de treinamento são selecionadas para participar do treinamento da rede neural, sendo que metade delas devem pertencer a classe das faces, e a outra metade não. Além disso, são implicitamente aplicadas as etapas 1 e 2.
6. Utilizando as imagens selecionadas em forma de vetor e as informações obtidas no agrupamento, são calculadas as distâncias relativas entre cada vetor e cada centróide dos agrupamentos, como será descrito precisamente mais adiante.
7. Utilizando as distâncias relativas, forma-se um vetor característico de cada imagem, para servir de entrada na rede neural. Então a rede neural é treinada até atingir uma faixa mínima de erro.

Os demais tópicos desta seção explicam mais detalhadamente algumas etapas e elementos da fase de treinamento.

4.2.1 Conjunto de Treinamento

As imagens do conjunto de treinamento foram obtidas no sítio do *Center for Biological and Computational Learning*^[27] do MIT e o pacote disponível para *download* consiste em imagens com 19×19 *pixels* e 256 intensidades de tons-de-cinza no formato *Portable Gray Map* (PGM), sendo que existem 2.901 imagens de faces e 28.121 imagens de “não faces”. Nem todas as imagens fornecidas foram utilizadas no conjunto.

4.2.2 Pré-Processamento

O pré-processamento é uma etapa em comum nas fases de treinamento e classificação com o intuito de diminuir efeitos negativos de iluminação na imagem tanto nas amostras do conjunto de treinamento, quanto nas janelas obtidas nas imagens a serem detectadas. A seguir, são exibidas as técnicas utilizadas.

4.2.2.1 Correção da Iluminação

Um dos problemas que dificulta a detecção das faces é a imprevisibilidade da iluminação. Uma mesma face, exposta a diversos ângulos de iluminação, pode ter diferentes representações no espaço das faces. Para diminuir o efeito de iluminações muito angulosas, pode-se utilizar uma técnica simples, encontrando o plano que melhor se ajusta a

superfície gerada pelos pontos das intensidades.

Supondo que a imagem a ser corrigida possui N *pixels*, o i -ésimo *pixel* da imagem possui coordenadas (x_i, y_i) e intensidade z_i e pode ser associado ao vetor (x_i, y_i, z_i) . Já um plano pode ser parametrizado por uma função bilinear $g(x, y)$

$$g(x, y) = ax + by + c \quad (4.1)$$

de tal forma que temos a superfície parametrizada

$$(x, y) \mapsto (x, y, g(x, y))$$

Utilizando o método dos mínimos quadrados, deseja-se encontrar a função g que minimiza o erro quadrático

$$\epsilon = \sum_{i=1}^N [g(x_i, y_i) - z_i]^2 \quad (4.2)$$

ou seja, deseja-se encontrar os coeficientes a, b e c que minimizam

$$\epsilon(a, b, c) = \sum_{i=1}^N [ax_i + by_i + c - z_i]^2 \quad (4.3)$$

e desenvolvendo a expressão 4.3

$$\begin{aligned} \epsilon(a, b, c) &= \sum_{i=1}^N [ax_i + by_i + c - z_i]^2 = \\ &= \sum_{i=1}^N [a^2x_i^2 + b^2y_i^2 + c^2 + 2(abx_iy_i + acx_i + bcy_i - ax_iz_i - by_iz_i - cz_i) + z_i^2] = \\ &= a^2 \sum_{i=1}^N x_i^2 + b^2 \sum_{i=1}^N y_i^2 + Nc^2 + \sum_{i=1}^N z_i^2 + \\ &\quad 2(ab \sum_{i=1}^N x_iy_i + ac \sum_{i=1}^N x_i + bc \sum_{i=1}^N y_i - a \sum_{i=1}^N x_iz_i - b \sum_{i=1}^N y_iz_i - c \sum_{i=1}^N z_i) \end{aligned}$$

é possível verificar, substituindo cada coeficiente convenientemente, que o erro ϵ é uma expressão quadrática da forma

$$\epsilon(a, b, c) = Aa^2 + Bb^2 + Cc^2 + 2Dab + 2Eac + 2Fbc - 2Ga - 2Hb - 2Ic + J$$

Logo, o ponto que minimiza ϵ , satisfaz a equação

$$\nabla\epsilon(a, b, c) = \left(\frac{\partial\epsilon}{\partial a}, \frac{\partial\epsilon}{\partial b}, \frac{\partial\epsilon}{\partial c} \right) = \vec{0}$$

ou seja, deve satisfazer o sistema matricial

$$Mv = b$$

$$\begin{pmatrix} A & D & E \\ D & B & F \\ E & F & C \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} G \\ H \\ I \end{pmatrix}$$

que pode ser facilmente solucionado encontrando a matriz inversa de M

$$v = M^{-1}b$$

De posse de a , b e c , obtem-se $g(x, y)$ que é utilizado para realizar a correção de iluminação, de tal forma que a nova intensidade z_i do i -ésimo *pixel* é

$$z_i = (1 - \lambda)g(x_i, y_i) + \lambda(\hat{z} - z_i)$$

onde $0 \leq \lambda \leq 1$ é a sensibilidade da correção e \hat{z} é o intensidade máxima do *pixel*, neste caso $\hat{z} = 255$ e $\lambda = 0.5$. Alguns exemplos da correção de iluminação podem ser vistos na figura 12.

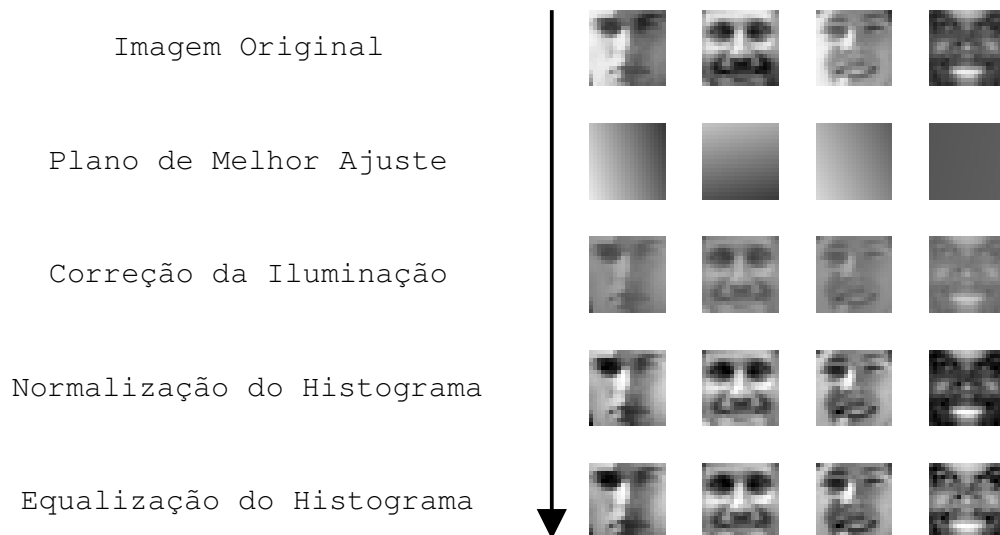


Figura 12: exemplos da correção de iluminação, normalização e equalização do histograma das intensidades

4.2.2.2 Normalização do Histograma

A normalização do histograma é uma tentativa de melhorar o contraste da imagem, ajustando as intensidades de tal forma que a faixa entre as intensidades mínima e máxima seja total.

Para realizar a normalização, deve-se primeiro encontrar as intensidades mínima e máxima, z_{min} e z_{max} e aplicar esta fórmula para cada pixel

$$z_i = \frac{\bar{z}}{z_{max} - z_{min}} z_i$$

Alguns exemplos da normalização podem ser vistos na figura 12.

4.2.2.3 Equalização do Histograma

A equalização do histograma é uma forma de melhorar a distribuição das intensidades, deslocando a intensidade média z_{med} para \bar{z} , que neste caso é 127. A aplicação da equalização ajuda a reduzir diferenças nas tonalidades da pele.

$$z_i = \begin{cases} \frac{\bar{z}}{z_{med}} z_i & \text{se } z_i \leq z_{med} \\ \frac{\bar{z}}{\hat{z} - z_{med}} (m - z_i) + \bar{z} & \text{se } z_i > z_{med} \end{cases}$$

Alguns exemplos da equalização podem ser vistos na figura 12.

4.2.3 Agrupamento do Conjunto de Treinamento

O objetivo de agrupar os vetores do conjunto de treinamento em classes menores é uma tentativa de capturar melhor a forma de como as faces estão distribuídas no espaço. Para tal, será utilizado um algoritmo de k -médias modificado que tenta, além de agrupar as imagens em proximidade, estimar a matriz de covariância de cada classe. São parâmetros de entrada do algoritmo as imagens em forma de vetor, e o número de classes k em que se deseja dividir o conjunto. Mais adiante são apresentados resultados com valores fixos de k .

O diagrama da figura 13 mostra os passos do algoritmo, que são os seguintes:

1. Como os vetores do conjunto de treinamento estão limitados por um paralelepípedo

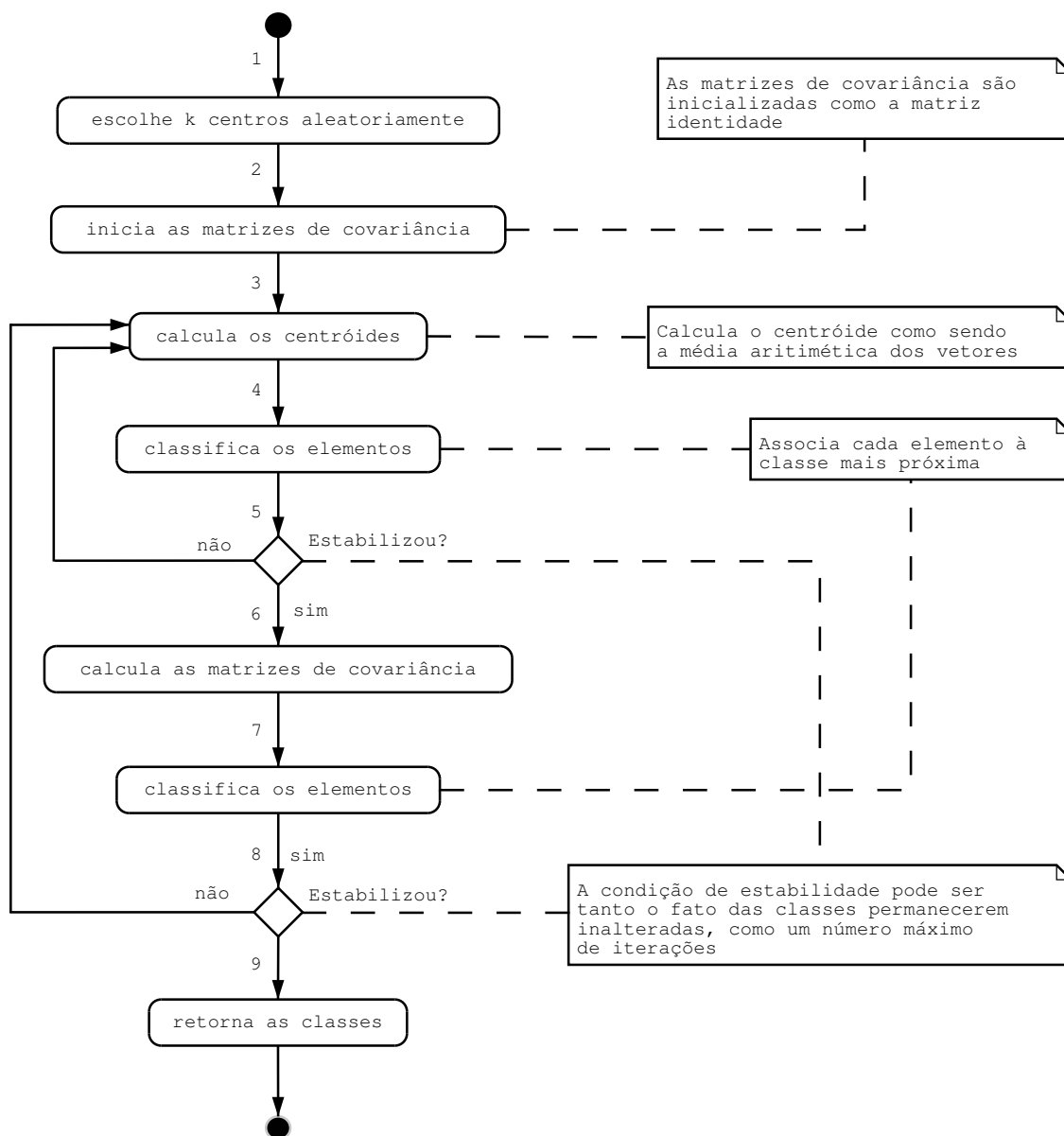


Figura 13: algoritmo de k -médias adaptado

n -dimensional, escolhem-se k centros aleatoriamente, e utiliza-se a distância euclidiana para classificar os primeiros agrupamentos.

2. Inicia as matrizes de covariância de cada classe como a matriz identidade, significando que as primeiras medidas serão aproximadamente euclidianas.
3. Recalcula-se o centróide de cada classe como sendo o vetor médio de todos os elementos desta.
4. Utilizando os centróides e as matrizes de covariâncias atuais, reclassifica todos os elementos, atribuindo cada elemento à classe mais próxima, segundo a distância normalizada de Mahalanobis.

5. Se todas as classes não se modificaram na última classificação, ou se um número máximo de iterações foi atingido, segue para a próxima etapa, senão retora para a etapa 3.
6. Recalculam-se as matrizes de covariância a partir dos atuais elementos de cada classe.
7. Utilizando os centróides e as matrizes de covariâncias atuais, reclassifica todos os elementos, atribuindo cada elemento à classe mais próxima, segundo a distância normalizada de Mahalanobis.
8. Se todas as classes não se modificaram na última classificação, ou se um número máximo de iterações foi atingido, segue para a próxima etapa, senão retora para a etapa 3.
9. Retorna os k centros e matrizes de covariância.

4.2.4 Cálculo das Distâncias Relativas

As distâncias relativas a cada agrupamento são medidas fundamentais para a detecção, pois são as entradas da rede neural e a qualidade das medidas implica na eficiência da detecção. De fato, esta medida é composta por duas componentes. A primeira é a distância normalizada Mahalanobis e a outra distância é a norma euclídeana da diferença entre um vetor e a reconstrução de sua projeção no espaço gerado pelas $m < n$ componentes principais. Para cada padrão de entrada, ambas componentes da distância são calculadas em relação a cada agrupamento e são submetidas a classificação na forma de um vetor característico.

4.2.4.1 A primeira componente de distância \mathcal{D}_1

A componente \mathcal{D}_1 , é a distância relativa que um vetor qualquer tem em relação ao centróide de uma classe, ponderada pela distribuição do subespaço gerado pelas direções mais significativas. Dado um vetor Γ qualquer no \mathbb{R}^n a componente da distância \mathcal{D}_1 é dada pela fórmula

$$\mathcal{D}_1(\Gamma) = \log |\Sigma| + (\Gamma - \Psi)^T \Sigma^{-1} (\Gamma - \Psi)$$

onde Σ é a matriz de covariância do agrupamento.

A expressão difere levemente da distância de normalizada de Mahalanobis definida

no capítulo anterior, devido ao fato de que todos os agrupamentos possuem termos em comum, sendo redundante considerá-los.

4.2.4.2 A segunda componente de distância \mathcal{D}_2

A segunda componente da distância \mathcal{D}_2 mede o quanto o vetor Γ “sai” do espaço das faces gerado pela classe e pode ser calculado com a seguinte fórmula

$$\mathcal{D}_2(\Gamma) = \|\Gamma - \Gamma_p\| = \|(I_n - E_m E_m^T)(\Gamma - \Psi)\|$$

onde E_m é uma matriz de m colunas, onde a i -ésima coluna corresponde ao autovetor u_i com autovalor associado λ_i em ordem decrescente, ou seja $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_m$, e Γ_p é a projeção de Γ no espaço das faces, reconstruído a partir de uma combinação linear.

4.2.5 Treinamento da rede MLP

Assumindo que o conjunto de treinamento foi dividido em k agrupamentos, sejam eles de imagens de faces ou não, então, cada imagem do conjunto de treinamento representada por um vetor Γ , está associada a uma distância da forma $(\mathcal{D}_1, \mathcal{D}_2)_j$ em relação a j -ésima classe. Logo, a rede neural deverá ter $2k$ neurônios de entrada, um para cada componente de distância de cada classe.

Quanto ao tamanho da camada interna, este número deve ser proporcional ao número de classes, ou seja, αk , que pode ser determinado α experimentalmente e será mostrado mais adiante. Esta rede neural difere da sugerida em ^[4], entretanto neste mesmo trabalho, foi determinado experimentalmente que uma topologia complexa de rede não é um fator crucial para este tipo de problema. Uma restrição do treinamento é o número de amostras de “faces” e “não faces”, que devem ser o mesmo para evitar parcialidade.

4.3 Fase de Classificação

A fase de classificação é onde ocorre a detecção propriamente dita utilizando a rede MLP. Passada a fase de treinamento, a rede neural está pronta para classificar janelas. Contudo, dada uma imagem de $N \times M$ *pixels*, algumas etapas devem ser cumpridas para realizar a detecção. O diagrama da figura 14 ilustra tais passos que são abaixo descritos:

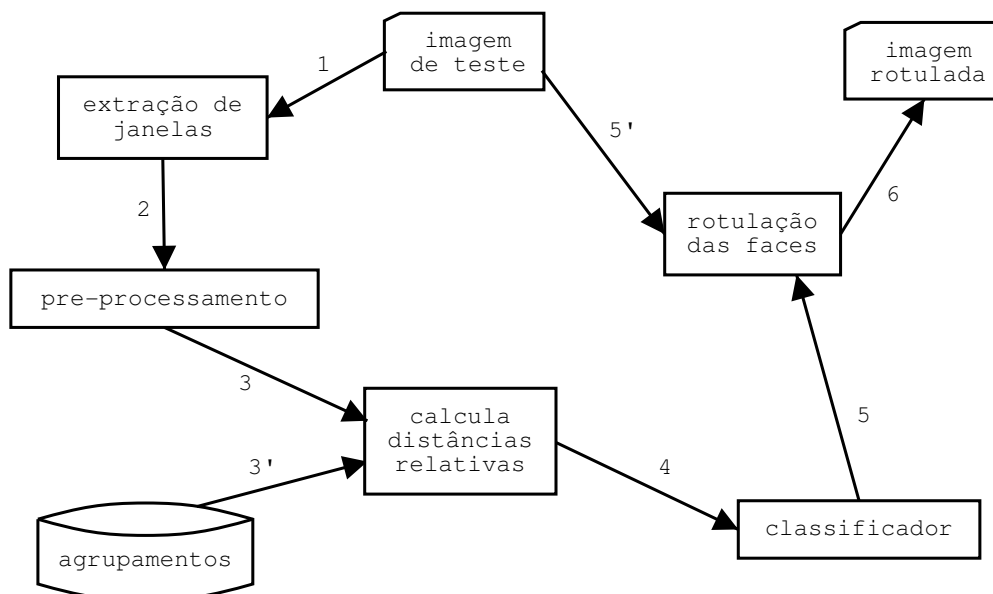


Figura 14: diagrama da fase de classificação

1. A foto é varrida por janelas de 19×19 *pixels* que são submetidas à próxima etapa, ao terminar a varredura, reduz-se a imagem de entrada em uma certa proporção, e repete-se a varredura. A busca termina quando não for mais possível que uma janela seja contida na imagem, ou seja, se largura ou a altura forem menor que 19.
2. A etapa de pre-processamento é exatamente a mesma descrita na fase de treinamento, só que desta vez é aplicada às janelas que são submetidas a classificação.
3. O processo de cálculo de distância não difere do descrito na seção anterior, com exceção que as imagens são as janelas, ao invés de amostras do conjunto de treinamento.
4. As distâncias obtidas na etapa anterior serão as entradas da rede neural, que irá classificar a janela como sendo “face” ou “não face”. A rede neural é a mesma que foi treinada na fase de treinamento.
5. A rede neural passa as posições e tamanhos das supostas faces detectadas para o módulo de rotulção.
6. Por fim é obtida uma nova imagem, rotulada com retângulos em torno das supostas faces, gerada a partir da imagem original, e das posições e tamanhos das janelas que foram classificadas como “face”.

A seguir, serão detalhadas algumas etapas da fase de classificação.

4.3.1 Extração de Janelas

O algoritmo de extração de janelas é uma busca exaustiva por janelas possivelmente sobrepostas, de uma imagem de entrada, como ilustrado na figura 15. As possíveis janelas são todas as sub-imagens de 19×19 *pixels* que couberem na imagem.

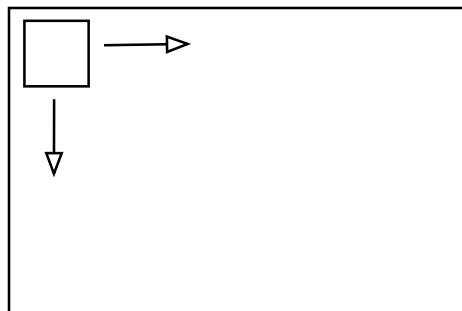


Figura 15: a varredura com janelas de 19×19 *pixels* na imagem $N \times M$ com $N \geq 19$ e $M \geq 19$

Quando toda a janela é percorrida, é aplicado um fator de redução $0 < \rho < 1$ nas dimensões da imagem gerando uma imagem menor (ver figura 16), através de subamostragem. Novamente é realizada outra varredura semelhante. Ao

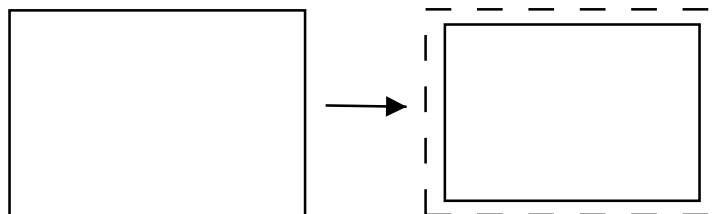


Figura 16: uma imagem de entrada reduzida pelo fator ρ

O deslocamento em *pixels* utilizados para a varredura, e o fator de redução ρ são parâmetros do algoritmo de extração de imagens e devem ser adequados a aplicação.

5 Implementação

5.1 Modelo de Implementação

O modelo de implementação do projeto do detector de faces pode ser vislumbrado no diagrama de classe da figura 17. As classes podem ser agrupadas por funcionalidade em cinco pacotes, sendo eles **image**, **newmat**, **cluster**, **ann** e **detector**.

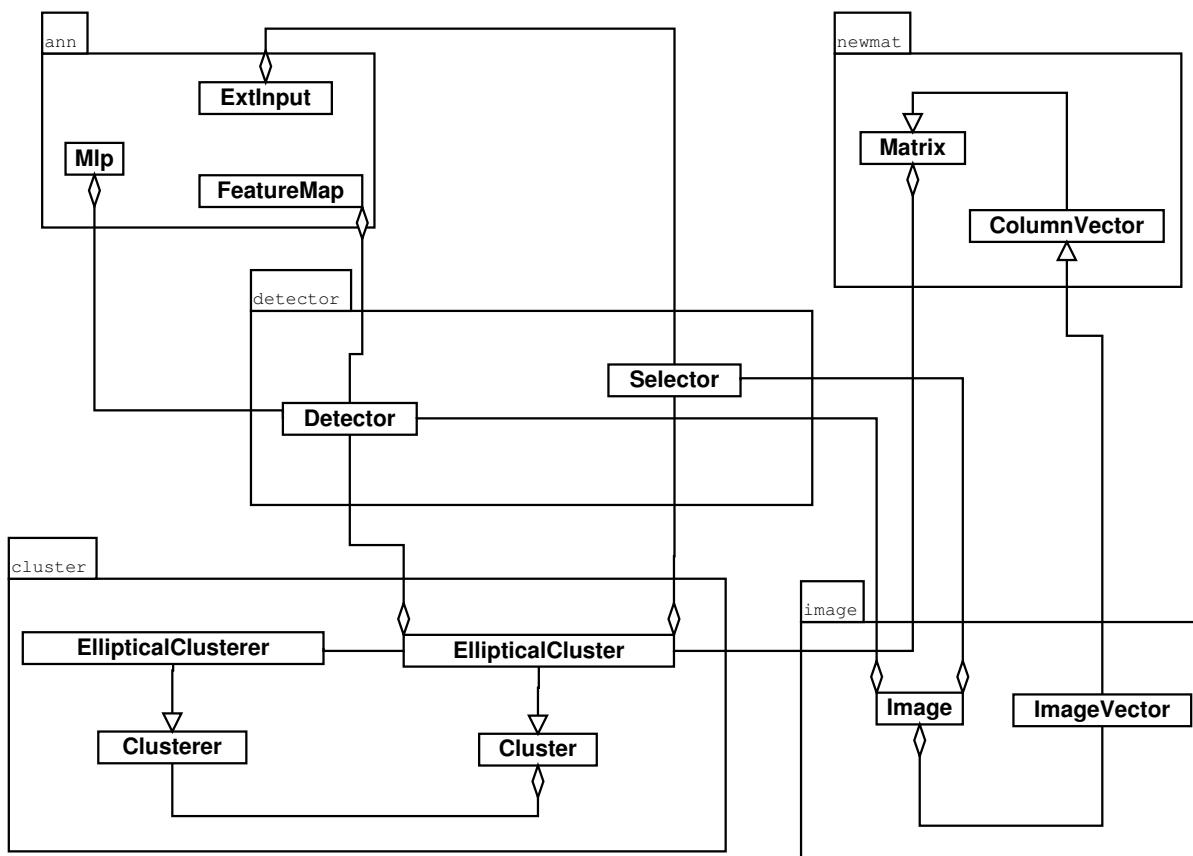


Figura 17: diagrama de classes

5.1.1 image

O pacote **image** contém a classe **Image** que encapsula as funcionalidades relacionadas a manipulação de imagens como leitura e escrita em arquivos de imagens, modificação e acesso aos *pixels* da imagem além de realizar as operações de pré-processamento descritas no capítulo anterior. A classe **ImageVector** é responsável pela correspondência entre uma imagem e um vetor, sendo possível obter instâncias de vetores com coordenadas relacionadas com os *pixels* de uma imagem.

5.1.2 newmat

O pacote **newmat** agrupa as classes de manipulação de vetores e matrizes, **Matrix** e **ColumnVector**, sendo possível realizar operações entre matrizes como soma e produto, além de inverter, transpor e calcular o determinante. Também é possível extrair os autovetores e seus respectivos autovalores de uma matriz simétrica. Estas classes foram desenvolvidas por terceiros, e incorporadas neste sistema.

5.1.3 cluster

As classes **EllipticalCluster** e **EllipticalClusterer** são derivadas, respectivamente das classes abstratas **Cluster** e **Clusterer** que encapsulam agrupamentos e agrupadores. A classe de **EllipticalCluster** e **EllipticalClusterer** são implementações concretas do agrupamento de k -médias elíptico utilizando a distância normalizada de Mahalanobis, sendo responsáveis pela persistência dos agrupamentos e cálculo das distâncias relativas.

5.1.4 ann

O pacote **ann** também foi desenvolvido por terceiros, e as classes **Mlp**, **ExtInput** e **FeatureMap** encapsulam respectivamente as rede MLP, os vetores característicos de entrada da rede e o conjunto de treinamento.

5.1.5 detector

O pacote **detector** é o centro do projeto e possui a classe **Detector**, que é responsável por todo o processo de treinamento e classificação da rede MLP. Já a classe **Selector** é responsável pela seleção e conversão de conteúdo para alimentação do detector.

5.2 Linguagem de Programação e Compilador

Todo código do detector de faces foi desenvolvido utilizando a linguagem C++, sendo a princípio desenvolvido para a plataforma GNU/Linux. Entretanto, o projeto deve ser facilmente portado para outras plataformas visto que o compilador utilizado foi o **gcc** que é portátil a vários ambientes e todas as bibliotecas utilizadas são multi-plataforma.

5.3 Bibliotecas

Com exceção das bibliotecas padrão do C/C++ que são automaticamente ligadas durante a compilação, foram utilizadas duas bibliotecas de terceiros para implementação. As bibliotecas utilizadas foram a

- **newmat**:¹ é uma excelente biblioteca de manipulação de matrizes, com uma série de recursos e uma boa documentação. Neste projeto, foi utilizada a versão estável *newmat10*.
- **libann**:² é uma biblioteca de criação, treinamento, persistência e utilização de redes neurais dos tipos MLP, Kohonen, Boltzmann e Hopfield. Apesar de funcionar bem, possui uma limitada documentação. A versão utilizada foi a *libann1.4*.

Ambas as bibliotecas, assim como o projeto do detector de faces, estão sob a licença de uso GNU *General Public License* (GPL)³.

5.4 Hospedagem do Projeto

Todo a documentação e códigos produzidos serão disponibilizados sob a licença GNU GPL no site <http://coisa.im.ufba.br/~ericjardim/> e futuramente será hospedado no www.sourceforge.net

¹a biblioteca newmat e sua documentação podem ser encontrados no sítio <http://www.robertnz.net/>

²a biblioteca <http://www.nongnu.org/libann/>

³para mais detalhes acessar o site www.gnu.org

6 *Resultados*

Para se realizar os testes com o sistema, k foi escolhido empiricamente como sendo igual a 4. O conjunto de treinamento consistia em 2247 imagens de faces e 2247 imagens de “não faces”. As imagens foram submetidas a etapa de pré-processamento, como descrito anteriormente, e foram sinteticamente aumentadas usando o espelhamento horizontal. Em seguida as imagens foram submetidas ao agrupamento utilizando o algoritmo de k -médias elíptico.

Para o treinamento da rede, foram utilizadas as mesmas imagens do conjunto de treinamento pré-processadas, mas sem o espelhamento. Após o treinamento, a rede MLP foi capaz de classificar o próprio conjunto de treinamento com 96,4% de acerto como mostra a tabela abaixo

	Faces	Não-Faces
acertos	2114	2219
erros	133	28

Esta mesma rede MLP foi utilizada para realizar detecção de faces em algumas fotos. O resultado pode ser visto nas figuras abaixo.

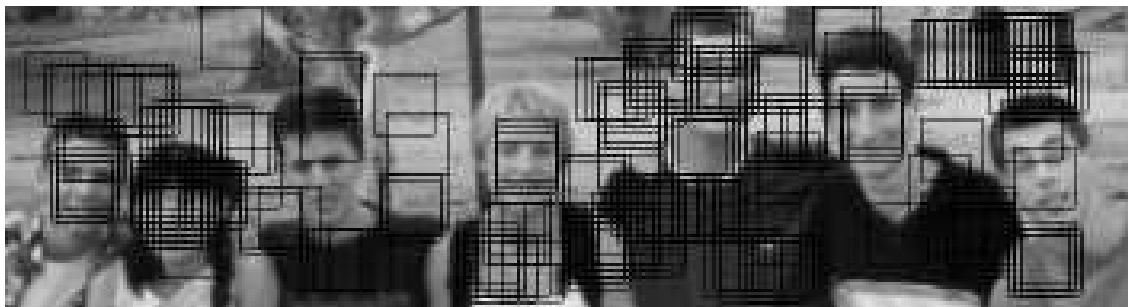


Figura 18:

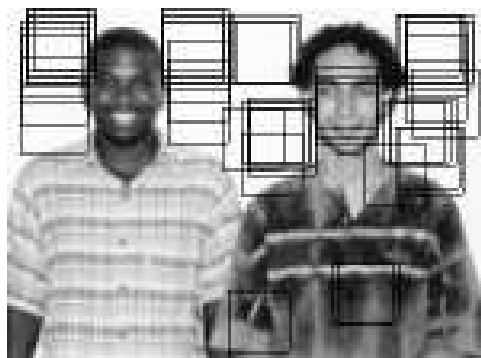


Figura 19:



Figura 20:

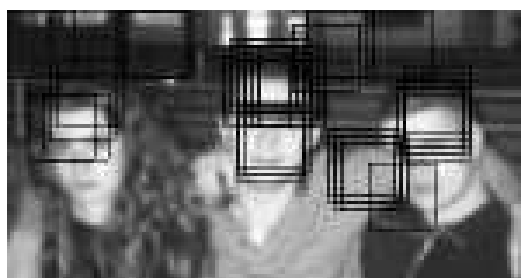


Figura 21:

7 Conclusão

Apesar dos testes com as imagens não terem sido visivelmente bem sucedidos, o método de cálculo de distâncias relativas aos agrupamentos se mostrou um excelente redutor de dimensionalidade, simplificando bastante o trabalho para a rede neural.

Como pode ser visto em quase todas as figuras, algumas janelas lisas foram classificadas como face, o que indica que ou treinamento da rede não foi bom, ou o conjunto de treinamento foi pobre ao tentar representar imagens de “não face”, o que é mais provável.

7.1 Trabalhos Futuros

- Para melhorar a qualidade da detecção pela rede neural e diminuir o número de falsos positivos, assim como sugerido em ^[4] e ^[11], pode-se utilizar as imagens erroneamente classificadas como face para realimentar conjunto de treinamento como contra-exemplos. Este processo deve ser repetido de forma a refinar a percepção da rede.
- Além disso, uma outra abordagem é a de criar agrupamentos negativos^[4], ou seja, agrupamentos de imagens de “não faces”, de forma melhorar a definição do contorno e as concavidades adjacentes ao espaço das faces.
- Como o sistema realiza busca exaustiva pela face em janelas independentes, uma sugestão é aproveitar o paralelismo do problema em ambientes distribuídos e potencializar o sistema.
- Tentar automatizar os processos de treinamento, classificação e realimentação de falsos-positivos ao máximo, além de permitir mais flexibilidade na escolha do número de agrupamentos, das distâncias a serem utilizadas, nos parâmetros

Referências

- 1 HJELMAS, E. Face detection: A survey. *Computer Vision and Image Understanding*, v. 83, p. 236–274, 2001.
- 2 VELHO, J. G. e L. *Computação Gráfica: Imagem*. Segunda. Rio de Janeiro: Instituto de Matemática Pura e Aplicada, 2002. (Série Computação e Matemática).
- 3 ANDRADE, S. S. Reconhecimento de faces com pca e redes neurais. *Programa de Pós-graduação em Mecatrônica: Universidade Federal da Bahia*, Salvador, 2003.
- 4 SUNG, K. K.; POGGIO, T. Example-based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 1, p. 39–51, 1998. Disponível em: <citeseer.nj.nec.com/sung95example.html>.
- 5 LEE, J. S. K. C. H.; PARK, K. H. Automatic human face location in a complex background. *Pattern Recognition*, v. 29, p. 1877–1889, 1996.
- 6 EXEL, S.; PESSOA, L. Attentive visual recognition. 1998.
- 7 SCHNEIDERMAN, H.; KANADE, T. Probabilistic modeling of local appearance and spatial relationships for object recognition. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*. Santa Barbara, CA: [s.n.], 1998. p. 45–51.
- 8 SCHNEIDERMAN, H.; KANADE, T. *A statistical approach to 3D object detection applied to faces and cars*. 2000. Disponível em: <citeseer.nj.nec.com/schneiderman00statistical.html>.
- 9 ROWLEY, H. A.; BALUJA, S.; KANADE, T. Human face detection in visual scenes. In: TOURETZKY, D. S.; MOZER, M. C.; HASSELMO, M. E. (Ed.). *Advances in Neural Information Processing Systems*. The MIT Press, 1996. v. 8, p. 875–881. Disponível em: <citeseer.nj.nec.com/rowley95human.html>.
- 10 ROWLEY, H. A.; BALUJA, S.; KANADE, T. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 20, n. 1, p. 23–38, 1998. Disponível em: <citeseer.nj.nec.com/rowley96neural.html>.
- 11 ROWLEY, H.; BALUJA, S.; KANADE, T. *Rotation Invariant Neural Network-Based Face Detection*. 1998. Disponível em: <citeseer.nj.nec.com/rowley98rotation.html>.
- 12 TURK, M.; PENTLAND, A. A. *Eigenfaces for Recognition*. 1991. 71–86 p.
- 13 PENTLAND, A.; MOGHADDAM, B.; STARNER, T. View-based and modular eigenspaces for face recognition. In: *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'94)*. Seattle, WA: [s.n.], 1994. Disponível em: <citeseer.nj.nec.com/pentland94viewbased.html>.

- 14 MOGHADDAM, B.; PENTLAND, A. Face recognition using view-based and modular eigenspaces. In: *Automatic Systems for the Identification and Inspection of Humans, SPIE'94*. [s.n.], 1994. v. 2257. Disponível em: <citeseer.nj.nec.com/moghaddam94face.html>.
- 15 MARDIA, K. V.; KENT, J. T. *Multivariate Analysis*. [S.l.]: Academic Press, 1979.
- 16 MCCALLUM, A.; NIGAM, K. *A comparison of event models for Naive Bayes text classification*. 1998. Disponível em: <citeseer.nj.nec.com/mccallum98comparison.html>.
- 17 MEYER, P. L. *Probabilidade: aplicações e estatísticas*. [S.l.]: Livros Técnicos e Científicos Editora S.A., 1975.
- 18 TANAKA, T.; YAMASHITA, Y. *Vectorembdedded Karhunen-Loeve transform and its application in orientation adaptive coding of images*. 2000. Disponível em: <citeseer.nj.nec.com/tanaka00vectorembdedded.html>.
- 19 LI, X.; KING, I. *Gaussian Mixture Distance for Information Retrieval*. 1999. Disponível em: <citeseer.nj.nec.com/article/li99gaussian.html>.
- 20 LIMA, E. L. *Álgebra Linear*. Segunda. Rio de Janeiro: Instituto de Matemática Pura e Aplicada, 1996. (Coleção Matemática Universitária).
- 21 KANUNGO, T. et al. The analysis of a simple k -means clustering algorithm. In: *Symposium on Computational Geometry*. [s.n.], 2000. p. 100–109. Disponível em: <citeseer.nj.nec.com/kanungo00analysis.html>.
- 22 BOTTOU, L.; BENGIO, Y. Convergence properties of the K -means algorithms. In: TESAURO, G.; TOURETZKY, D.; LEEN, T. (Ed.). *Advances in Neural Information Processing Systems*. The MIT Press, 1995. v. 7, p. 585–592. Disponível em: <citeseer.nj.nec.com/bottou95convergence.html>.
- 23 ALSABTI; RANKA; SINGH. An efficient parallel algorithm for high dimensional similarity join. In: *IPPS: 11th International Parallel Processing Symposium*. IEEE Computer Society Press, 1998. Disponível em: <citeseer.nj.nec.com/alsabti98efficient.html>.
- 24 HAMPSHIRE, J. B. I.; PERLMUTTER, B. A. Equivalence proofs for multilayer perceptron classifiers and the Bayesian discriminant function. In: *Proceedings of the 1990 Connectionist Models Summer School, 1990*. D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton, eds. Morgan Kaufmann, San Mateo, CA. [s.n.], 1990. Disponível em: <citeseer.nj.nec.com/hampshire90equivalence.html>.
- 25 PESSOA, L. F. *Multilayer Perceptrons versus Hidden Markov Models: Comparisons and Applications to Image Analysis and Visual Pattern Recognition*. Disponível em: <citeseer.nj.nec.com/pessoa95multilayer.html>.
- 26 LAWRENCE, S. et al. Neural network classification and prior class probabilities. In: ORR, G.; Müller, K.-R.; CARUANA, R. (Ed.). *Tricks of the Trade*. [S.l.]: Springer Verlag, 1998, (Lecture Notes in Computer Science State-of-the-Art Surveys). p. 299–314.
- 27 CBCL FACE DATABASE 1. Center for Biological and Computational Learning - MIT. Disponível em: <<http://www.ai.mit.edu/projects/cbcl/software-datasets/>>.