FIAP

Projeto Final de Avaliação Substitutiva

Tecnologia de Processamento de Imagens

Allan Almeida **RM: 330864**

OpenCV

OpenCV (Open Source Computer Vision Library). Originalmente, desenvolvida pela Intel, em 2000, é uma biblioteca multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de visão computacional, bastando seguir o modelo de licença BSD Intel. O OpenCV possui módulos de processamento de imagens e Video I/O, estrutura de dados, álgebra Linear, GUI (Interface Gráfica do Usuário) básica com sistema de janelas independentes, Controle de mouse e teclado, além de mais de 350 algoritmos de Visão computacional como: Filtros de imagem, calibração de câmera, reconhecimento de faces(Eingenfaces, Fisherfaces e Local Binary Patterns Histograms), análise estrutural e outros.

EigenFaces

O método Eigenface baseia-se em linearmente projetar o espaço de imagens em um espaço de características com dimensões reduzidas obtido fazendo uso da análise de componentes principais (PCA), também conhecido como método Karhunen-Loeve. Entretanto, produz direções de projeção que maximiza a dispersão dos pontos no gráfico em todas as classes, isto é, em todas as imagens faciais mantém as variações indesejadas causadas pela iluminação e expressão facial. Este método baseia-se em autovetores e autovalores de uma matriz simétrica, que é a matriz de covariância. Os passos principais para a modelagem utilizando este método, PCA, são:

1. Dada uma coleção de m imagens de treinamento identificadas, ou seja, tendo uma base de imagens com cada imagem de tamanho matricial de n × o , com alguma identificação, cria-se uma matriz Xij , onde j=1,2,...,m é a quantidade de imagens de treinamento, e i é o tamanho das imagens em formato de vetor, isto é, i = n × o , fazer:

a. Computar a imagem média

$$M = \sum_{r=1}^{j} X_{ir}, i = 1,2,...,(n \times o)$$

b. Centralizar os vetores das imagens subtraindo cada um dos vetores pela média dos vetores encontrados.

$$\overline{X} = X - M$$

c. Calcular a matriz de covariância

$$\Lambda = M * M^T$$

d. Computar os k autovetores, vk, da matriz de covariância correspondente aos k maiores autovalores, k. Como a matriz de covariância é real e simétrica, todos os autovalores e autovetores serão também reais e simétricos. Além disso, se essa matriz é de ordem i, então existirá i autovetores associados à i autovalores. Os autovetores são, de certa forma, imagens, que são agrupadas em uma matriz W com k colunas.

$$W_{ik} = \{v_1, v_2, ..., v_k\}$$
$$\lambda_k = \{\lambda_1, \lambda_2, ..., \lambda_n\}$$

e. Projetar cada uma das imagens de treinamento no autoespaço k-dimensional criando um vetor de tamanho reduzido para cada uma das imagens, facilitando a comparação entre os vetores. A projeção é realizada multiplicando cada um dos vetores imagens pelo autoespaço.

$$\hat{X} = W \cdot \overline{X}$$

Os maiores autovalores da matriz de covariância tendem não ser fixo. Após ter realizado esses primeiros cálculos no banco de faces, realiza-se o reconhecimento:

2. Dada uma imagem de teste Y, projetá-la no autoespaço, após tê-la centralizada também

com aquele mesmo vetor de médias, assim como as de treinamento.

$$\overline{Y} = Y - M$$
$$\hat{Y} = W \cdot \overline{Y}$$

3. Classificá-la com as imagens de treinamento projetadas, fazendo uso de um classificador definido ou, às vezes, pode-se combinar dois ou mais classificadores.

Pontos Possitivos EigenFace.

Eigenface fornece uma maneira fácil e barata de perceber o reconhecimento facial em que:
Seu processo de treinamento é completamente automático e fácil de codificar. Eigenface reduz adequadamente a complexidade estatística na representação da imagem de face. Uma vez que as eigenfaces de um banco de dados são calculadas, o reconhecimento facial pode ser alcançado em tempo real.
Eigenface pode manipular grandes bancos de dados.

Pontos negativos EigenFace.

As deficiências do método de eigenface também são óbvias:
Muito sensível à iluminação, escala e tradução; requer um ambiente altamente controlado. Eigenface tem dificuldade em capturar mudanças de expressão. Os eigenfaces mais significativos são principalmente sobre a codificação de iluminação e não fornecem informações úteis sobre o rosto real. Para lidar com a distração de iluminação na prática, o método de eigenface geralmente descarta as três primeiras eigenfaces do conjunto de dados. Como a iluminação geralmente é a causa por trás das maiores variações nas imagens faciais, as três primeiras eigenfaces capturarão principalmente as informações das trocas de iluminação tridimensionais, que têm pouca contribuição para o reconhecimento facial. Ao descartar esses três eigenfaces, haverá uma quantidade decente de aumento na precisão do reconhecimento de faces.

FisherFace

O discriminante linear de Fisher (FLD), também conhecido com análise de discriminantes linear (LDA), foi desenvolvido por R. A. Fisher na década de 1930, porém, apenas recentemente tem sido utilizado para o reconhecimento de objetos. É um método específico à classe, pois, ele trabalha com o uso de “rótulos”, isto é, uma vez identificado os rostos dizendo qual face pertence a qual pessoa, os mesmos são agrupados por pessoa, e cada agrupamento desses é conhecido como classe. O método tenta modelar a dispersão dos pontos visando maior confiabilidade para a classificação. O LDA busca otimizar a melhor linha em uma superfície que separa satisfatoriamente as classes. Inicia-se o algoritmo obtendo as matrizes de dispersão entre classes, interclasse, e dentro das classes, intraclasse. A projeção é feita maximizando a dispersão interclasse e minimizando a intraclasse, formulado pela razão entre as determinantes de ambas as matrizes, com isso diferindo do PCA, que maximiza o espalhamento, dispersão, dos padrões no espaço de características, independente da classe em que esses pertencem. As duas medidas citadas, matematicamente são definidas como:

1. matriz de dispersão intraclasses, within class:

$$S_w = \sum_{j=1}^{c} \sum_{i=1}^{|T_j|} \left(x_i^j - \mu_j\right) \cdot \left(x_i^j - \mu_j\right)^T$$

, em que x_i^j é o i-ésimo exemplo da classe j, j é a média da classe j, c é o número de classes, e $|T_j|$ o número de exemplos na classe j;

2. matriz de dispersão interclasses, between class:

$$S_b = \sum_{j=1}^{c} (\mu_j - \mu) \cdot (\mu_j - \mu)^T$$

,em que μ representa a média de todas as classes. A maximização da medida inter-classes e a minimização da intra-classes são obtidas ao maximizar a taxa $\frac{\det(S_b)}{\det(S_x)}$. O espaço de projeção é então encontrado resolvendo a equação $S_bW = \lambda S_wW$, onde W é a matriz com autovetores generalizados associados com λ , que é a matriz diagonal com autovalores. Essas matrizes estão limitadas à ordem c-1, em que c é i número de classes, limitação devido à comparação ser realizada entre duas classes diferentes.

Para identificar uma imagem de teste funciona da mesma forma que o Eigenface. A imagem de teste é projetada e comparada com cada uma das faces de treinamento também projetadas, identificando-a com a de treinamento que mais se aproxima. A comparação, de novo, é feita utilizando um classificador específico ou a combinação de dois ou mais.

Pontos Positivos.

Não é tão sensível a variação de iluminação, o processamento demanda menos recurso.

Local Binary Pattern Histograms(LBPH)

Local Binary Pattern (LBP) é um operador de textura simples, porém eficiente, que rotula os pixels de uma imagem ao limitar a vizinhança de cada pixel e considera o resultado como um número binário.
Foi descrito pela primeira vez em 1994 (LBP) e, desde então, foi considerado um recurso poderoso para a classificação de textura. Ainda, quando o LBP é combinado com os histograms of oriented gradients (HOG), ele melhora o desempenho da detecção consideravelmente em alguns conjuntos de dados.
Usando o LBP combinado com histogramas, podemos representar as imagens do rosto como um vetor de dados simples.
Como o LBP é um descritor visual, ele também pode ser usado para tarefas de reconhecimento facial, como pode ser visto na seguinte explicação.

Etapas do algoritimo:

1. Parâmetros: o LBPH usa 4 parâmetros:

Raio: o raio é usado para construir o padrão binário circular e representa o raio ao redor do pixel central. Geralmente, é definido como 1.

Vizinhos: o número de pontos de amostra para construir o padrão binário circular local. Tenha em mente que: quanto mais pontos de amostra você incluir, maior será o custo computacional. Geralmente é definido como 8.

Grade X: o número de células na direção horizontal. Quanto mais células mais fina é a grade e maior é a dimensionalidade do vetor de características resultante. Geralmente é definido como 8.

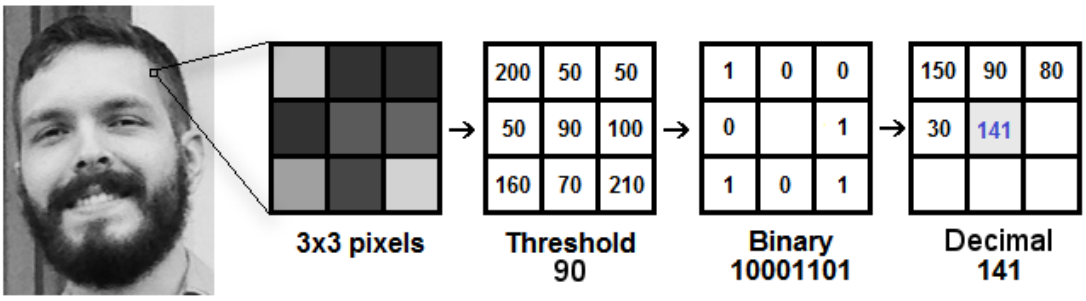
Grade Y: o número de células na direção vertical. Quanto mais células, mais fina é a grade e maior é a dimensionalidade do vetor de características resultante. Geralmente é definido como 8.
Não se preocupe com os parâmetros no momento, você os entenderá depois de ler os próximos passos.

2. Treinando o Algoritmo: Primeiro, precisamos treinar o algoritmo. Para fazer isso, precisamos usar um conjunto de dados com as imagens faciais das pessoas que queremos reconhecer. Nós também precisamos definir um ID (pode

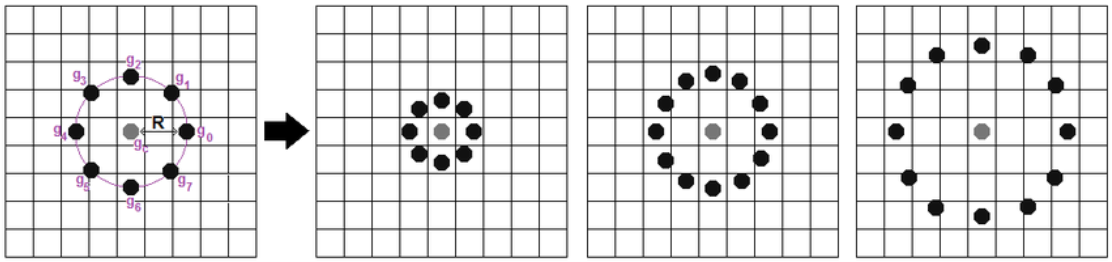
ser um número ou o nome da pessoa) para cada imagem, então o algoritmo usará essas informações para reconhecer uma imagem de entrada e dar-lhe uma saída. Imagens da mesma pessoa devem ter o mesmo ID. Com o conjunto de treinamento já construído, vejamos os passos computacionais do LBPH.

3. Aplicando a operação LBP: O primeiro passo computacional do LBPH é criar uma imagem intermediária que descreva melhor a imagem original, destacando as características faciais. Para fazer isso, o algoritmo usa um conceito de janela deslizante, com base nos parâmetros raio e vizinhos.

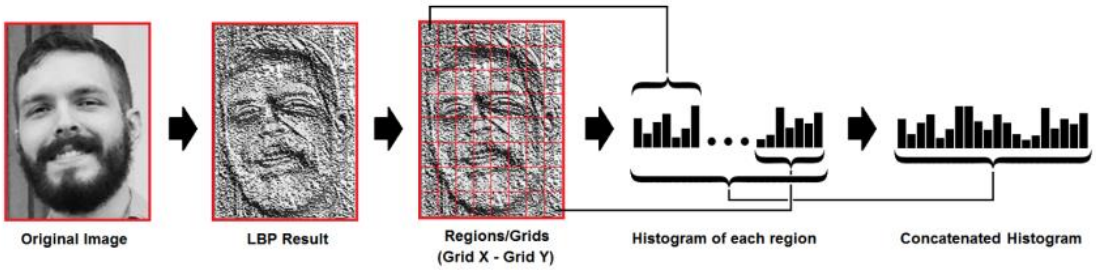
A imagem abaixo mostra esse procedimento:



Com base na imagem acima, vamos dividir em várias pequenas etapas para que possamos entender isso facilmente:
Suponha que tenhamos uma imagem facial em escala de cinza.
Podemos obter parte desta imagem como uma janela de 3×3 pixels.
Ele também pode ser representado como uma matriz 3×3 contendo a intensidade de cada pixel (0 ~ 255).
Então, precisamos tomar o valor central da matriz para ser usado como limiar.
Esse valor será usado para definir os novos valores dos 8 vizinhos.
Para cada vizinho do valor central (limiar), estabelecemos um novo valor binário.
Definimos 1 para valores iguais ou superiores ao limiar e 0 para valores inferiores ao limiar.
Agora, a matriz conterá apenas valores binários (ignorando o valor central).
Precisamos concatenar cada valor binário de cada posição da matriz linha por linha para um novo valor binário (por exemplo, 10001101). Nota: alguns autores usam outras abordagens para concatenar os valores binários (por exemplo, no sentido horário), mas o resultado final será o mesmo.
Então, convertemos esse valor binário para um valor decimal e colocamos ele na posição central da matriz, que é realmente um pixel da nova imagem.
No final deste procedimento (chamado LBP), temos uma nova imagem que representa melhor as características da imagem original.
Nota: O procedimento LBP foi expandido para usar um número diferente de raio e vizinhos, é chamado de Circular LBP.



Isso pode ser feito usando a interpolação bilinear. Se algum ponto de dados estiverem entre os pixels, ele usa os valores dos 4 pixels mais próximos (2×2) para estimar o valor do novo ponto de dados.
4. Extraindo os histogramas: agora, usando a imagem gerada no último passo, podemos usar os parâmetros Grade X e Grade Y para dividir a imagem em múltiplas grades, como pode ser visto na imagem a seguir:



Com base na imagem acima, podemos extrair o histograma de cada região da seguinte maneira:
Como temos uma imagem em escala de cinza, cada histograma (de cada grade) conterá apenas 256 posições (0 ~ 255) que representam as ocorrências de cada intensidade de pixel.
Então, precisamos concatenar cada histograma para criar um histograma novo e maior. Supondo que tenhamos redes 8×8, teremos 8x8x256 = 16.384 posições no histograma final. O histograma final representa as características da imagem original da imagem.
O algoritmo LBPH é praticamente isso.
5. Realizando o reconhecimento facial: nesta etapa, o algoritmo já está treinado. Cada histograma criado é usado para representar cada imagem do conjunto de dados de treinamento. Assim, dada uma imagem de entrada, nós executamos as etapas novamente para esta nova imagem e criamos um histograma que representa a imagem.
Então, para encontrar a imagem que corresponde à imagem de entrada, precisamos comparar dois histogramas e devolver a imagem com o histograma mais próximo.
Podemos usar várias abordagens para comparar os histogramas (calcular a distância entre dois histogramas), por exemplo: distância euclidiana, qui-quadrado, valor absoluto, etc. Neste exemplo, podemos usar a distância euclidiana (que é bastante conhecida) baseada na seguinte fórmula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

Portanto, a saída do algoritmo é o ID da imagem com base no histograma mais próximo. O algoritmo também deve retornar a distância calculada, que pode ser usada como medida de “confiança”. Nota: não se deixe enganar com o nome da “confiança”, pois as confianças mais baixas são melhores porque significa que a distância entre os dois histogramas é mais próxima.
Podemos usar um limite e a “confiança” para estimar automaticamente se o algoritmo reconheceu corretamente a imagem. Podemos assumir que a pessoa foi reconhecida com sucesso se a confiança for menor do que um limiar definido.

Pontos Positivos

Variação da luminosidade não afeta o processo de reconhecimento

Conclusão

Eigenface tenta maximizar a variação e não é supervisionado, se baseia em PCA seu tempo de execução é mais demorado que os outros. Então, com PCA eles geralmente recebem um modelo decente do rosto e tendo em vista que que a variação na luminosidade afeta diretamente a eficácia no processo de reconhecimento realizado por este algoritmo. O método Fisherface é um aprimoramento do método Eigenface que utiliza a Análise Linear Discriminante de Fisher (FLDA ou LDA) para a redução de dimensionalidade. O LDA maximiza a proporção de dispersão entre classes para a dispersão dentro da classe, portanto, funciona melhor do que o PCA para fins de discriminação e é especialmente útil quando se há grandes variações de iluminação em relação ao eigenface.

O LBPH por utilizar uma matriz binária para representar as imagens tem uma maior eficácia quando utilizado os seus parâmetros padrões e a variação da luminosidade não afeta o processo de reconhecimento assim como afeta os demais algoritmos, porque ao modificar a variação da luz no ambiente todos os elementos da matriz serão alterados na mesma proporção o que não afetará no resultado final do reconhecimento.

Referencias

JAIN, A. K., DUIN, R. P. W., MAO, J. “Statistical pattern recognition: a review.” IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1):4-37.

SPERANDIO, D., MENDES, J. T., SILVA, L. H. M., “Cálculo Numérico: características matemáticas e computacionais dos métodos numéricos”. Editora Pearson. São Paulo: 2003.

TURK, M. A; PENTLAND, A. P. Face recognition using eigenfaces, In Proc. of the IEEE Computer Society Conferece. 1991.

AHONEN, TIMO, Abdenour Hadid, and Matti Pietikäinen. “Face recognition with local binary patterns.” Computer vision-eccv 2004 (2004): 469–481.