Nome: Emanoel Bruno, Rogénes Reis

RELATÓRIO TRABALHO FINAL

INTRODUÇÃO

O reconhecimento facial é um problema já conhecido pela área da Visão Computacional, por isso esse trabalho tem como foco desenvolver um algoritmo de reconhecimento facial e analisar os já existentes. Todo algoritmo de reconhecimento facial precisa de no mínimo quatro etapas detecção da face, reunião de informação, comparação de dados e reconhecimento facial.

A etapa de detecção da face é essência pois sem ele não é possível localizar as faces presentes na imagem, inviabilizando as outras etapas. Já a reunião de dados é a etapa onde o programa reuni informações para poder diferenciar as pessoas. Para diferencia-las na etapa de comparação de dados são feitas comparações entre a face a ser identificada e o dataset que o software possui. A última etapa é agora determinar se é a pessoa ou não, o que é feito na etapa de reconhecimento facial.

DETECÇÃO DE FACE

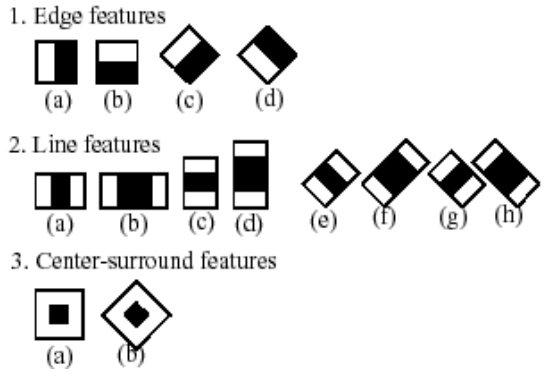
Nessa etapa é utilizado um programa para reconhecer se a imagem possui ou não possui uma face. Programas que fazem isso são conhecidos como classificadores, eles são treinados por dezenas de milhares de imagens para aprender a classificar uma nova imagem corretamente. Por não ser o foco do trabalho desenvolver um classificador optamos por utilizar um já treinado do Opencv. O Opencv possui sois deles que são:

- *Haar Classifier*
- *LBP Classifier*

Para utilizá-los é necessário importar um arquivo que contém as configurações do classificador, as quais são encontradas dentro do Opencv.

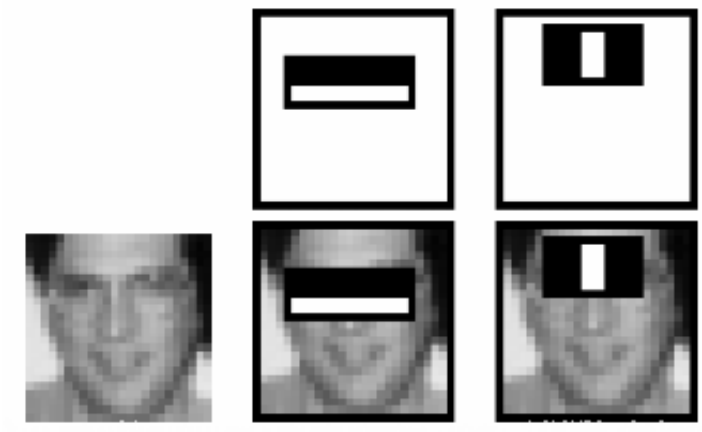
HAAR CLASSIFIER

O *Haar Classifier* é uma aproximação baseada em *machine learning*, ou seja, recebe diversas imagens para treina-lo e extrai informações delas que são utilizadas para poder conseguir diferenciar uma imagem positiva (com face) de uma negativa (sem face). Abaixo está uma imagem que mostra alguns dos conteúdos extraídos das imagens:



Fonte: docs.opencv.org

Cada uma das janelas acima é utilizada para calcular um valor obtido por meio da subtração dos pixels da parte branca da janela e a soma dos presentes na parte preta da janela. Por exemplo na imagem abaixo vemos o cálculo de dois desses valores um relacionado a propriedade que a região dos olhos é muitas vezes mais escura do que a área do nariz e bochechas. Já o segundo depende da propriedade que os olhos são mais escuros do que a parte superior do nariz.

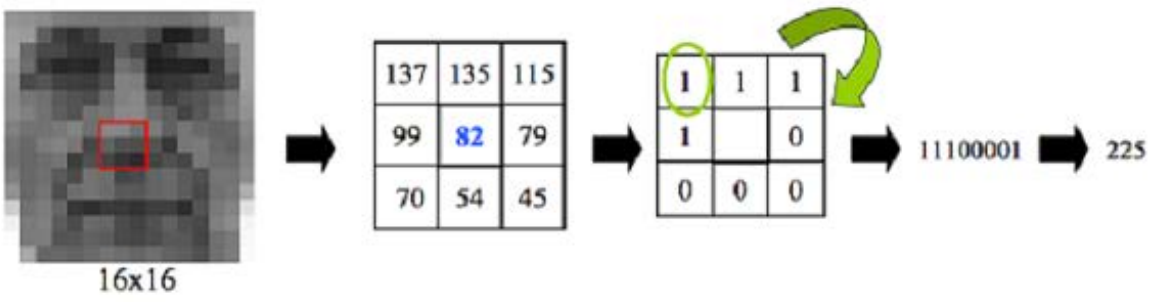


Fonte: docs.opencv.org

LBP CASCADE CLASSIFIER

LBP ou Local Bynary Patterns também precisa ser treinado para conseguir identificar as imagens. Esse algoritmo busca identificar as imagens extraindo componentes para formar um vetor de componentes que diferenciam uma imagem positiva de uma negativa.

O algoritmo divide as imagens em blocos e percorre cada um dos blocos em janelas 3x3 e compara o valor do pixel central com o de cada vizinho na janela. Se o pixel vizinho é maior ou igual ao do centro, ele define seu valor como 1 senão define como 0. Então, ele lê os valores de pixel atualizados na janela em uma ordem no sentido horário e forma um número binário, que ele converte em um número decimal e esse número decimal é o novo valor do pixel central. Repete-se o processo para cada pixel em um bloco. Abaixo está uma imagem que mostra o processo descrito:



Por último ele converte os valores de cada bloco em um histograma e utiliza ele para formar o vetor de componentes que identifica a imagem.

VANTAGENS E DESVANTAGENS

Abaixo está uma tabela que fala sobre as vantagens e desvantagens de cada método:

Algoritmo	Vantagens	Desvantagens
Haar	• Alta acurácia de detecção • Baixa taxa de falsos positivos	• Computacionalmente mais caro e complexo • Maior tempo de treino • Menos preciso em caras negras • Limitações em condições de iluminação difíceis • Menos robusto a oclusão
LBP	• Computacionalmente mais barato e simples • Menor tempo de treinamento • Robusto a mudanças de iluminação • Robusto a oclusão	• Menor acurácia • Alta taxa de falsos positivos

No fim nós optamos por utilizar o classificador de Haar, pois ele possui uma acurácia maior e possui menor taxa de falsos positivos e ele já estava treinado.

REUNIÃO DE DADOS

Essa é a etapa onde se treina o algoritmo para conseguir diferenciar as pessoas. Para isso é necessário utilizar um *dataset* para isso. Nós optamos por utilizar o Yale Facedatabase A (mais conhecido como Yalefaces) que é um conjunto de dados mais apropriado para experimentos iniciais, porque o problema de reconhecimento é mais difícil. O banco de dados é composto por 15 pessoas (14 homens, 1 mulher), cada uma com 11 imagens em tons de cinza de tamanho de pixel. Há mudanças nas condições de luz (luz central, luz esquerda, luz direita), expressões faciais (feliz, normal, triste, sonolento, surpresa, piscadela) e óculos (óculos, sem óculos).

Para pode usar as imagens presentes no dataset armazenamos as imagens de cada pessoa em pastas diferentes as quais lemos e classificamos com labels diferentes para cada pessoa e enviamos para o algoritmo para treina-lo.

Infelizmente não conseguimos implementar um algoritmo de reconhecimento facial eficiente ou nem mesmo utilizável, ele estava na etapa de reunião de dados, mas a coleta de dados não foi bem-sucedida. Então mudamos o foco do trabalho para uma análise dos algoritmos já existentes e com um foco especial em explica-los e compará-los entre si. Mas continuamos a usar o Yalefaces para treiná-los e compará-los. Os algoritmos a ser analisados são: Eigenfaces,

Fisherfaces, Local Binary Patterns Histograms (LBPH). Os algoritmos citados unem as etapas de reunião de dados e comparação de dados em um só, que é treinar o reconhecedor.

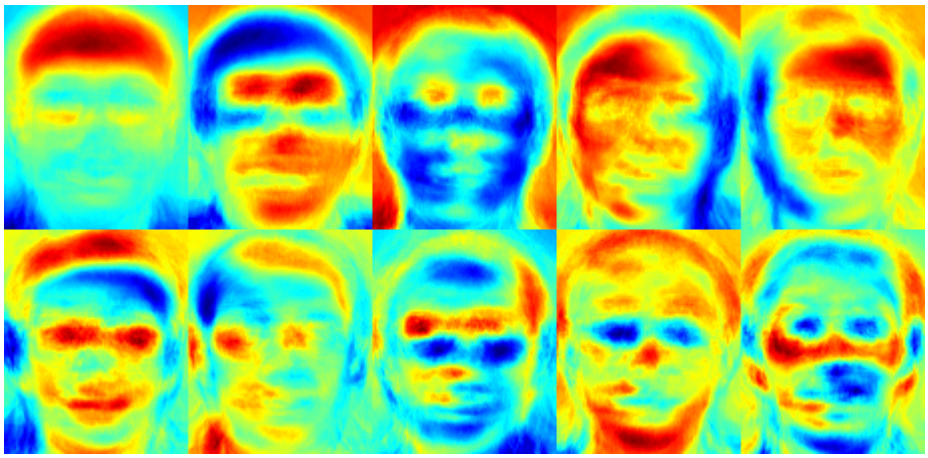
TREINAR O RECONHECEDOR

Todos os algoritmos citados recebem o mesmo tipo de entrada como parâmetro para treinar o algoritmo de reconhecimento que é as faces de pessoas já conhecidas e labels que identificam elas. A diferença está no que eles fazem com tais faces.

EIGENFACES

Este algoritmo considera que nem todas as partes de uma face são igualmente úteis para o reconhecimento facial. Ele se concentra nas áreas de máxima mudança. Por exemplo, dos olhos ao nariz há uma mudança significativa, e o mesmo se aplica do nariz à boca.

O Eigenfaces compara os rostos observando as áreas de maior mudança, pois ao capturar a variação máxima entre os rostos diferencia-los se torna uma tarefa mais fácil. Basicamente ele olha para todas as imagens de treinamento de todas as pessoas como um todo e tenta extrair os componentes que são relevantes e úteis e descarta o resto. Esses recursos importantes são chamados de componentes principais. Abaixo está uma imagem mostrando a variação extraída de uma lista de faces.



Fonte: docs.opencv.org

O reconhecedor Eigenfaces treina a si mesmo extraindo componentes principais e mantendo um registro de quais pertencem a qual pessoa e sempre que tiver que reconhecer uma nova imagem ele segue os seguintes passos:

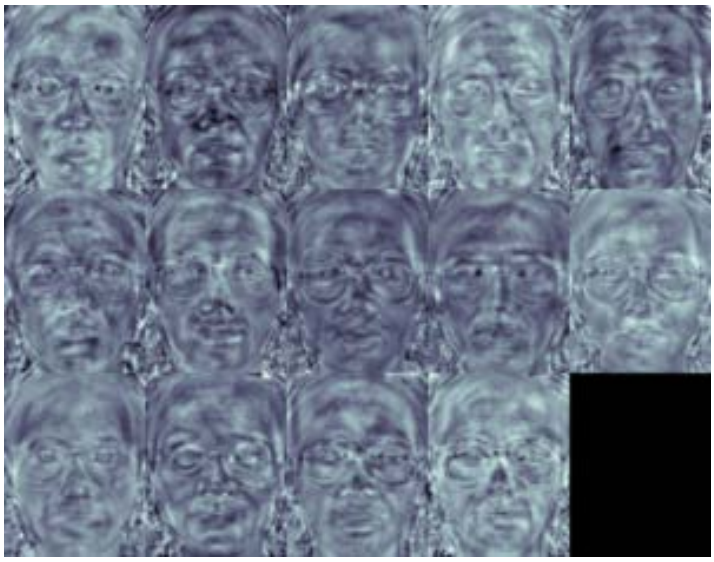
- Extrai os componentes principais da nova imagem.
- Compara esses recursos com a lista de elementos armazenados durante o treinamento.
- Encontra aqueles com a melhor correspondência.
- Devolve o marcador associado ao melhor componente de correspondência.

Vale apena ressaltar que o algoritmo Eigenfaces também considera a iluminação como uma característica importante. Por conseguinte, as luzes e sombras também são captadas pelo mesmo.

FISHERFACES

O Fisherfaces é uma versão melhorada do último algoritmo. Como o Eigenfaces examina todos os rostos de treinamento de todas as pessoas de uma só vez e encontra componentes principais de todos eles combinados, então uma imagem de uma pessoa com condições de iluminação muito diferentes afetaria todas as outras. Pois, como o Eigenfaces a variação da iluminação muito relevante para o reconhecimento facial, ele pode descartar as características dos rostos das outras pessoas, considerando-as menos úteis. No final, acaba que a variância que o Eigenfaces extraiu representa apenas características faciais de um indivíduo.

Para solucionar esse problema o Fisherfaces extrai recursos úteis das faces de cada pessoa separadamente, em vez de extraí-los de todas as faces combinadas. Dessa forma, mesmo que uma pessoa tenha altas mudanças de iluminação, ela não afetará o processo de extração de recursos de outras pessoas. Abaixo está uma imagem dos principais componentes usando o algoritmo Fisherfaces.

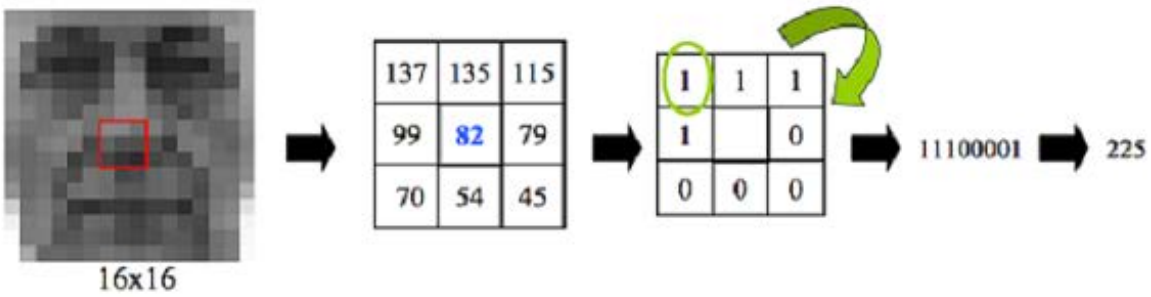

Fonte: docs.opencv.org

LOCAL BINARY PATTERNS HISTOGRAMS (LBPH)

Como vimos nos outros algoritmos, considerar a iluminação pode afetar muito os resultados pois no mundo real não possuímos condições perfeitas de iluminação. Por isso a ideia do LBPH é não olhar para imagem como um todo mas sim ir percorrendo ela com uma pequena janela e olhando os pixels da vizinhança. Processo o qual é muito semelhante ao LBP.

Assim como no LBP ele percorre a imagem toda com uma janela de 3x3 e em cada local compara o valor do pixel central com o de cada vizinho na janela. Se o pixel vizinho é maior ou igual ao do centro, ele define seu valor como 1 senão define como 0. Então, ele lê os valores de pixel atualizados na janela em uma ordem no sentido horário e forma um número binário, que ele converte em um número decimal e esse número decimal é o novo valor do pixel central. Abaixo está

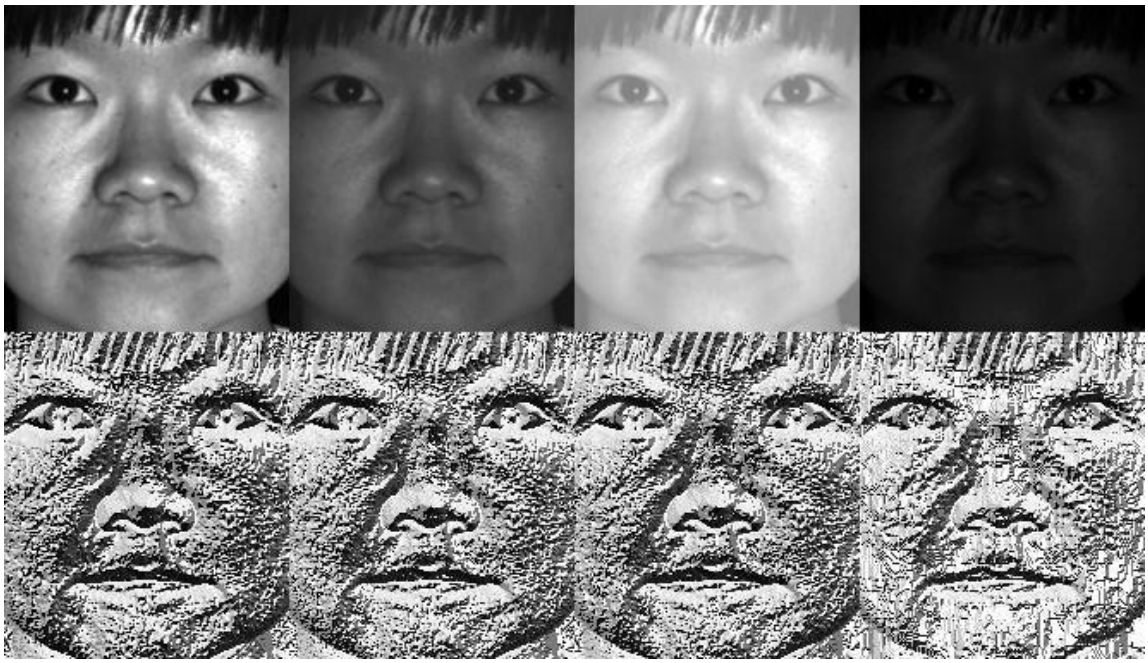
uma imagem que mostra o processo descrito:



Depois ele constrói um histograma de todos os valores decimais e guarda esse histograma relacionado a pessoa da imagem. No fim tem se um histograma para cada imagem de cada pessoa no dataset e quando deseja-se reconhecer uma pessoa o programa segue os seguintes passos:

- O reconhecedor gera um histograma para essa nova imagem.
- Compara esse histograma com os histogramas que já possui.
- Encontra a melhor correspondência e retorna o marcador de pessoa associado à melhor correspondência.

Abaixo está uma imagem que demonstra como esse método não é afetado pela iluminação:



Fonte: docs.opencv.org

ANÁLISE

Para verificarmos essas diferenças na prática nós implementamos um programa usando o dataset Yalefaces para testar a performance dos algoritmos. A partir dos testes obtivemos as seguintes tabelas:

TABELAS

TABELA DE IMAGENS X PESSOAS IDENTIFICADAS

	Eigen	Fisher	LBPH	Resultado Esperado
1	s1	s1	s1	s1
2	s5	s2	s2	s2
3	s3	s3	s3	s3
4	s1	s3	s4	s4
5	s1	s6	s2	s5
6	s6	s6	s6	s6
7	s6	s7	s7	s7
8	s2	s5	s8	s8
9	s2	s2	s9	s9
10	s6	s5	s10	s10
11	s11	s11	s11	s11
12	s1	s3	s5	s12
13	s6	s7	s13	s13
14	s6	s14	s14	s14
15	s3	s3	s15	s15

Para validarmos o funcionamento do algoritmo nos utilizamos 15 imagens diferentes, uma de cada pessoa no dataset. Como elas não vieram com nome demos os nomes de s1 até s15. Sabendo disso podemos ver pela tabela acima que o algoritmo que encontrou mais pessoas corretamente foi o LBPH como as amostrar continham imagens de teste continham variações na iluminação, na feição e na aparência (óculos).

Tais dados também podem ser constatados pela tabela abaixo:

TABELA DADOS ESTATÍSTICOS

	Acurácia	Erro	Falsos Positivos	Acertos
Eigen	0,27	0,73	11	4
Fisher	0,47	0,53	8	7
LBPH	0,87	0,13	2	13

Outra coisa que pode ser vista é que o Fisher apresentou uma performance melhor que o Eigen.