Instituto Federal de Educação, Ciência e Tecnologia do Ceará

Engenharia da Computação

Visão Computacional

Prof. Pedro Pedrosa

Relatório N^o 03 - Métodos de Extração de Características e Classificação de Imagens.

André Vieira da Silva

Fortaleza

Abril de 2018

1 Abstract

The digital image processing is one of the branches of scientific computing with more comprehensiveness in applications of the results obtained in surveys.In an application environment, we have the image 'raw', without processing on it, as an ideal information for the various procedures that involve and compose the entire image processing chain.In an application environment we rarely have the image 'raw', without processing on it, as an ideal information for the various procedures which involve and compose the entire image processing chain.Then it is necessary to use appropriate techniques in degree, amount of interactions and versions to suit the images to the appropriate applicability as input parameter of the processes that will make use of these.Esse documento apresenta alguns desses processos essenciais, de forma que, sem eles, o processamento de imagens se tornaria enviável.We will make an analysis of the application, processing and iteration with images in order to expose the main characteristics of each process, highlighting the importance of each one.

2 Resumo

O processamento digital de imagens é um dos ramos da computação científica com mais abrangência em aplicações dos resultados obtidos em pesquisas.Como se sabe, raramente, em um ambiente de aplicação temos a imagem 'crua',sem processamento sobre a mesma,como uma informação ideal para os diversos procedimentos que envolvem e compõem toda a cadeia de processamento de imagens.Então se faz necessário o uso de técnicas adequadas em grau, quantidade de interações e versões para a adequação das imagens à devida aplicabilidade como parâmetro de entrada dos processos que irão fazer uso dessas.Esse documento apresenta parte desses processos básicos mas ainda essenciais de forma que sem eles, o processamento de imagens se tornaria enviável.Faremos uma analise da aplicação,processamento e iteração com imagens afim de expor as características principais de cada processo evidenciando a importância de cada um.

Sumário

3 Introdução

Para fins de aplicação acadêmica ,esse trabalho realizado, procura entender menos superficialmente os métodos básicos no que diz respeito ao processamento digital de imagens.Através da implementação,da simulação dos algoritmos ,da pesquisa e da analise dos resultados e métricas ou ainda heurística e empirismo se pretende alcançar um limiar mais profundo e prático.

4 Apresentação

Este documento pretende ,ainda que brevemente,tratar das operações básicas no processamento digital de imagens expondo através das implementações realizadas os resultados obtidos dos algoritmos escritos assim como as devidas explanações.

5 Descrição de atividades

Entrando diretamente na parte prática do presente relatório serão apresentadas implementações dos algoritmos em linguagem python que implementam as operações básicas usadas no processamento de imagens será ainda apresentada uma breve descrição dos mesmos e posteriormente mostrado os resultados obtidos ainda também com os devidos comentários baseados em pesquisa e fornecendo na medida do possível os equivalentes matemáticos,fórmulas,heurísticas,entre outras formas que direcionam ao conhecimento do objetivo deste trabalho.

Nos debruçaremos sobre dois descritores de imagens expondo e seus resultados e suas aplicações.Posteriormente nos atentaremos a alguns detalhes classificação de imagens.

6 Atributos,Características de uma imagem

Na visão computacional e no processamento de imagens , atributos são informações relevantes à natureza de um problema de uma dada aplicação computacional. Esse conceito é muito empregado nas aplicações que fazem uso dos recursos de Machine Learning e reconhecimento de padrões.O atributos ,características de uma imagem ,no processamento de imagens,são em geral uma coleção considerável de elementos de origem matemática,probabilística,heurística,formas,estruturas enfim muitos elementos e fontes de origem de operações aplicadas a imagem em processamento.Isto torna o conceituação de caracterização de uma imagem muito abrangente sendo necessário técnicas adequadas para a extração de atributos específicos,uma filtragem dos elementos descritores que são realmente relevantes ao domínio da aplicação em desenvolvimento.

Limitando um pouco o nível de abstração ,extrair os atributos ou características ,em visão computacional ,de uma imagem nada mais é que associar à imagem a ser processada,o resultado de um processamento computacional qualquer.

Com todos essas afirmações chegamos a um problema recorrente,o de como representar uma imagem,que dados,que parâmetros que características podem identificar de forma única cada imagem dentro de um conjunto de elementos .O que os métodos de extração de atributos pretendem individualmente ou em conjunto com outros métodos computacionais disponibilizar ao mundo infinito das aplicações ,lembrando que é a natureza do problema que define o leque dos parâmetros da solução.

6.1 Tipos de Descritores

Em geral os descritores de uma imagem podem ser de:

1. Características Internas(fronteira de objetos):
    - Bordas
    - Cantos (Pontos de Interesse)
    - Cumes
    - Formas
2. Características Externas(Blob - conjunto de Pixels de uma região):

- Cor
- Textura

6.2 Métodos de Extração de Atributos de uma Imagem

Faremos a analise de dois extratores de atributos de forma a apresentar seus devidos resultados .Para tal faremos uso de uma base de dados de imagens numérica fornecida pelo laboratório Lapisco - IFCE.

prof. Pedrosa. *Base de Dados Numerica*, A base é composta por um número em cada linha, contendo número de 0 a 9. Cada imagem formada por uma linha do TXT possui dimensões 35x35, sendo cada linha representada em sequencia no TXT, e portanto cada linha do TXT possui 1225 numeros (0 ou 1), e no fim o rótulo a qual representa aquela imagem. Ao formar a imagem, vocês tem que considerar o 0 como preto e 1 como branco.

As imagens foram convertidas para seus devidos formatos com o uso do seguinte programa em linguagem python :

```
import cv2

def retImg(line,numPar):
  _lineImg = []
  _Img = []
  number = line[-2]
  for index in range(len(line)):
    if(line[index].isnumeric()):
      retNumVal(_lineImg, index, line)
      if(index % 70 == 0 and index > 0):
        _Img.append(_lineImg)
        _lineImg = []
  createImage(_Img, int(number),numPar)


def createImage(_Img, number,numPar):
  canvas = np.ones ((len (_Img[0]) - 2, len (_Img[0]) - 2, 1))
  for i in range (len (canvas[0]) - 1):
    for l in range (len (canvas[0]) - 1):
      canvas[i][l] = _Img[i][l]
  if(numberCount[number] < 160):
    cv2.imwrite ("NumberEval/Number_"+str(number)+"_"+str(numPar)+".jpg
      ",canvas)
    numberCount[number]+=1
  else:
    cv2.imwrite ("NumberTreino/Number_"+str(number)+"_"+str(numPar)+".
      jpg",canvas)

  def retNumVal(_lineImg, index, line):
    if (line[index] == '0'):
      _lineImg.append (0)
    else:
      _lineImg.append (255)
```

Figura 1: Implementação Conversão de Imagens em txt para .jpg

6.2.1 Histograma de Gradientes Orientados (Histogram of Oriented Gradients - HOG)

O histograma de gradientes orientados (HOG) é um descritor de características para fins de classificação de objetos. A técnica consiste no calculo e concatenação de diversos histogramas orientados de uma imagem.A ideia do algoritmo consiste em dividir uma imagem em varias janelas de tamanho pre-definido(célula) é feito o cálculo do histograma

para cada célula e consequentemente a concatenação destes,e isso define o descritor HOG.

wikipedia. *Descritor$_H$OG*. URL: `https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients`, O descritor HOG tem algumas vantagens importantes sobre outros descritores. Como opera em células locais, é invariante para transformações geométricas e fotométricas, exceto pela orientação a objetos. Tais mudanças só apareceriam em regiões espaciais maiores. Além disso, como Dalal e Triggs descobriram, amostragem espacial grosseira, amostragem de orientação precisa e forte normalização fotométrica local permitem que o movimento individual do corpo dos pedestres seja ignorado, desde que eles mantenham uma posição mais ou menos ereta. O descritor HOG é, portanto, particularmente adequado para detecção humana em imagens.

6.2.2 Algumas Características do Descritor HOG

Abaixo algumas características do Descritor de atributos HOG:

1. Não precisa de grandes regiões para perceber características relevantes de objetos da imagem.

2. Opera em células locais, sendo invariante para transformações geométricas e fotométricas, a não ser pela pelo posicionamento dos objetos.

3. A amostragem espacial simples ,a amostragem de orientação,a normalização fotométrica local são tão precisas que permitem que o movimento individual de objetos seja ignorado quando da extração do histograma.

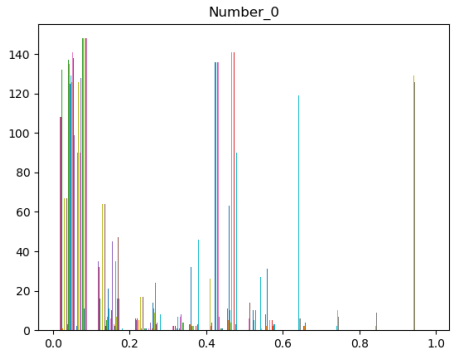
6.2.3 Extraindo Atributos de base de dados com descritor Hog

Para verificar os resultados da aplicação foram usadas a imagens gerados pelo algoritmo 6.2.8 ainda foi feito pra agilizar e simplificar a implementação do algoritmo uso da biblioteca Opencv versão 3.4 bastante conhecida nas áreas de PDI,visão computacional e outras.

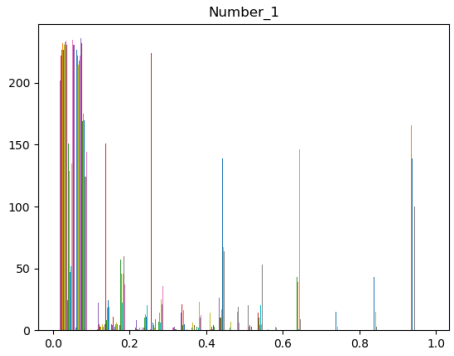
Segue a abaixo o código do algoritmo usado para a abtenção dos resultados da aplicação do descritor HOG sobre a base de dados numérica obtida.

```
import cv2
import os
import numpy as np
from numpy.ma import ids
from matplotlib import pyplot as plt

winSize = (20, 20)
blockSize = (10, 10)
blockStride = (5, 5)
cellSize = (10, 10)
nbins = 9
derivAperture = 1
winSigma = -1.
histogramNormType = 0
L2HysThreshold = 0.2
gammaCorrection = 1
nlevels = 64
signedGradient = True

hog = cv2.HOGDescriptor (winSize, blockSize, blockStride, cellSize,
    nbins, derivAperture, winSigma,
histogramNormType, L2HysThreshold, gammaCorrection, nlevels,
    signedGradient)


def getImageComId(trainId):
  caminhos = [os.path.join('NumberTreino',f)for f in os.listdir('
      NumberTreino')]
  # print(caminhos)
  faces = [],ids = []
  for caminhosImagens in caminhos:
    imagemFace = cv2.cvtColor(cv2.imread(caminhosImagens),cv2.
        COLOR_BGR2GRAY)
      id = int(os.path.split(caminhosImagens)[-1].split('_')[1])
    if(id == trainId):
      ids.append(id)
      faces.append(imagemFace)
  return np.array(ids),faces

def trainById(id):
  ids,faces = getImageComId(id)
  hog_descriptors = []
  for img in faces:
    hog_descriptors.append (hog.compute(img))
  hog_descriptors = np.squeeze (hog_descriptors)
  numtex = "Number_{number}".format(number=id)
  plt.title(numtex)
  plt.hist(hog_descriptors)
  plt.savefig("HogFiles/"+numtex)
  plt.show()
  print("salvando ..."+numtex)
```

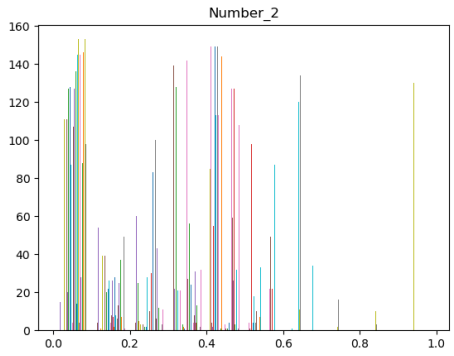
A implementação permite através de uma base de imagens indexadas previamente aplicar o algoritmo de treino dentro de um certo contexto a essa base.Para o caso do algoritmo de extração do descritor HOG o resultado obtidos podem ser melhor visualizados na forma de gráfico já que a observação e analise dos valores gerados no conjunto de treino só tem sentido do ponto de vista computacional.



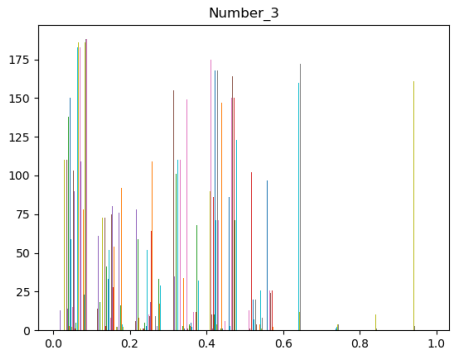
(a) HOG para o treino numero 0



(b) HOG para o treino numero 1



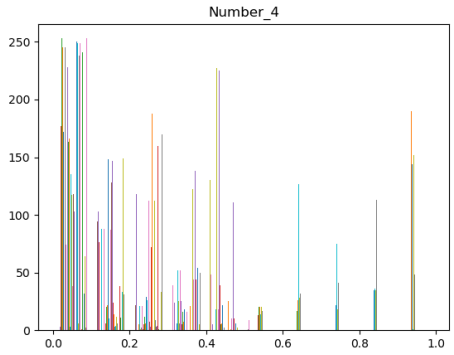
(c) HOG para o treino numero 2



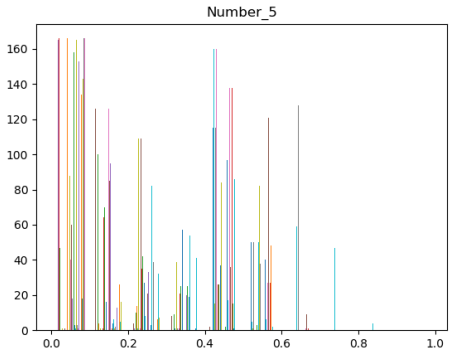
(d) HOG para o treino numero 3

Figura 3: HOGs Resultantes do treino com base numérica

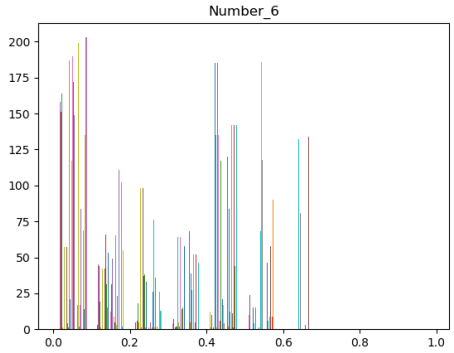
6.2.3



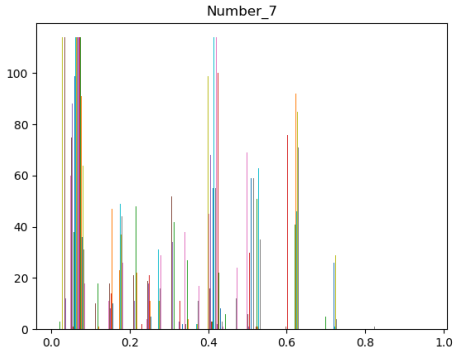
(a) HOG para o treino numero 4



(b) HOG para o treino numero 5



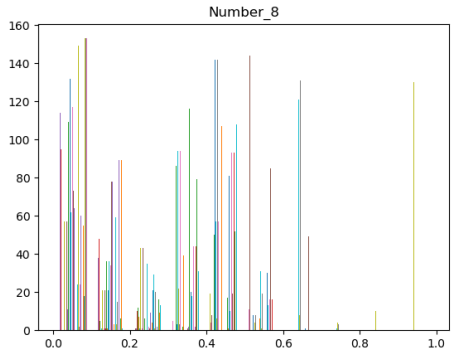
(c) HOG para o treino numero 6



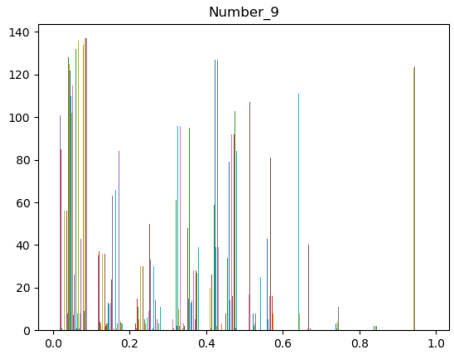
(d) HOG para o treino numero 7

Figura 4: HOGs Resultantes do treino com base numérica

6.2.3



(a) HOG para o treino numero 8



(b) HOG para o treino numero 9

Figura 5: HOGs Resultantes do treino com base numérica

6.2.3

6.2.4 Resultados

Como se pode observar o resultado da aplicação do descritor Hog sobre as imagens foram que para cada número, de acordo com o algoritmo ,o descritor gera um histograma cujo os valores identificam individualmente cada elemento no caso numeros.Assim cada histograma funciona como uma assinatura para cada classe dos elementos da aplicados al treino.Para uma base de classificação os valores desses histogramas poderiam ser aplicados de forma a permitir ao algoritmo de classificação o seu resultado esperado.Os parametros usados para o algoritmo foram estão na implementação do código e mais informações na documentação oficial da biblioteca opencv.Opencv. *HogDocs*. URL: `https://docs.opencv.org/3.4.1/d5/d33/structcv_1_1HOGDescriptor.html`;

6.2.5 Momentos como descritores de Uma Imagem

Os momentos de uma imagem fazem parte da área de processamento digital de em geral fornecem um conjunto de características obtidas matematicamente da imagem de quantidade bastante abrangente.wikipedia. *Momentos invariantes de uma imagem.* URL: `https://pt.wikipedia.org/wiki/Momentos_invariantes_de_uma_imagem`, Eles permitem o cálculo da área de um objeto (conjunto de pixels), centroide de um objeto ou também permite identificar um determinado objeto mesmo que tenha sofrido mudança de tamanho ou mesmo que seja rotacionado. Esta teoria é muito utilizada em reconhecimento de padrões. Geralmente utiliza-se algum tipo de software que extrai os referidos momentos de uma imagem binarizada.

6.2.6 Momentos Invariantes de Hu

Lembrando que um bom descritor de imagem deve ser invariante,ou mais próximo disso, ao tamanho(escala),a rotação e a translação de uma imagem e também deve fornecer atributos viáveis computacionalmente e bem definidos nesse contexto se insere o descritor de atributos de imagens de HU.

wikipedia. *Momentos de Hu.* URL: `https://pt.wikipedia.org/wiki/Momentos_invariantes_de_uma_imagem`, Ming-Kuei Hu, num trabalho publicado em 1962 organizou um conjunto de equações em que os momentos são invariantes em relação à escala, rotação e também translação. As equações a seguir são também conhecidas como equações de Hu ou também Momentos Invariantes.

Essas equações são :

$$I_1 = \eta_{20} + \eta_{02} \qquad \text{(Momento 1)}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \qquad \text{(Momento 2)}$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \qquad \text{(Momento 3)}$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad \text{(Momento 4)}$$

$$I_5 = (\eta_{30}-3\eta_{12})(\eta_{30}+\eta_{12})[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2]+(3\eta_{21}-\eta_{03})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2] \quad \text{(Momento 5)}$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \quad \text{(Momento 6)}$$

$$I_7 = (3\eta_{21}-\eta_{03})(\eta_{30}+\eta_{12})[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2]-(\eta_{30}-3\eta_{12})(\eta_{21}+\eta_{03})[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2] \quad \text{(Momento 7)}$$

Figura 6: As sete Equações/Momentos Invariantes de Hu

6.2.7 Extraindo atributos com extrator de Atributo de HU

Usaremos a imagem abaixo para a extração de atributos com o extrator Hu.Optaremos pela facilidade de implementação usando a biblioteca opencv pretendendo mostrar que ele é um ótimo descritor mais ainda assim se deve atenção a detalhes quando do uso da mesma.

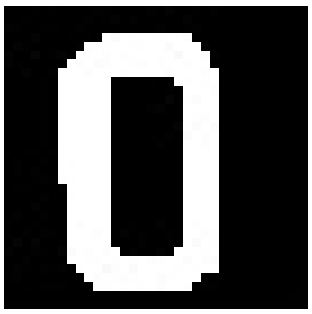


Figura 7:

Figura 8: Numero obtido em Pedrosa, *Base de Dados Numerica*, op. cit.

Momento	Imagem Normal	Imagem Reduzida	Imagem Aumentada
I_1	229.69148863339248	234.53613413766897	230.96256144497872
I_2	0.4663136703336924	0.4555491944604682	0.46681590897585584
I_3	3.8800388228495145e-06	5.083551066761607e-06	3.8906422832269504e-06
I_4	1.082890746399092e-06	1.9566144556675984e-06	1.0636270676888128e-06
I_5	2.4030268e-18	-3.2731953800000004e-17	2.3417062000000002e-18
I_6	2.032439473780481e-09	2.2492021749264772e-09	1.9885333950220835e-09
I_7	-2.2066578800000004e-17	-5.231153470000001e-17	-2.1509792500000004e-17

Tabela 1: Resultados da Aplicação do extrator Hu com escalaridade de 0.5 e 2 vezes o tamanho original da imagem

6.2.8 Resultados

Como é de esperar o descritor Hu apresenta quase os mesmos valores para a variação de escalaridade da imagem contudo ,ainda assim ,alguns valores diferem o que requer uma atenção por parte de quem desenvolve a aplicação.No caso dessa implementação os valores tiveram esse comportamento devido a binarização da imagem e a quantidade de pixels ,que no caso quando foram reduzidos provocaram uma perda da linearidade dos valores do descritor o que se evidencia que para imagens aonde a carga de informação e muito reduzida o algoritmo pode divergir sendo necessário descartar o momentos que levaram

a ocorrência de problemas.Mais detalhes sobre o algoritmo podem ser vistos em *Visual Pattern Recognition by Moment Invariants"*. URL: `http://www.sci.utah.edu/~gerig/CS7960-S2010/handouts/Hu.pdf`, Visual Pattern Recognition by Moment Invariants

Segue abaixo o código utilizado para a geração dos resultados.

```
def trainHu(id,img):
    namefile = "NumberHuResult/Number_"+str(id)
    f = open(namefile+".txt",'w')
    def save(stri,vet):
        lis = ""
        lis += "Numero_{id}\r\n".format(id=id)
        lis += stri

        for i in vet:
            lis += str(round(i,30)*100000) + ","
        lis += "\r\n"
        f.write(lis)

    save("Imagem normal : ",cv2.HuMoments(cv2.moments(img)).flatten())

    reducedimage = cv2.resize (img, None, fx=0.5, fy=0.5)  # reduzida
    save("Imagem Reduzida : ",cv2.HuMoments(cv2.moments(reducedimage)).
        flatten())

    increasedimage = cv2.resize (img, None, fx=2, fy=2)  # aumentado
    save("Imagem Aumentada : ",cv2.HuMoments(cv2.moments(increasedimage)).
        flatten())

    f.close()

def getImageComId():
    caminhos = [os.path.join('NumberHu',f)for f in os.listdir('NumberHu')]
    for caminhosImagens in caminhos:
        imagemFace = cv2.cvtColor(cv2.imread(caminhosImagens),cv2.
            COLOR_BGR2GRAY)
        id = int(os.path.split(caminhosImagens)[-1].split('_')[1])
        trainHu(id,imagemFace)
```

Figura 9: Implementação descritor Hu para figura 7

6.3 Classificação de Objetos em Imagens

Classificação é o processo de reconhecimento de um elemento a um determinado conjunto de forma que os mesmos continuem a compartilhar a mesmas características. Os classificadores podem ser :

- Pixel a Pixel : utilizam a as intensidades de cada pixel para encontrar regiões que compartilham dos mesmos atributos,homogeneidade de área.
- Regiões : conseguem fazer uso de áreas ,conjunto de pixels ,de forma a encontrar similaridades entre as fronteiras ou não desses conjuntos de maneira a identificar uni e separa essas áreas.

O primeiro passo para a classificação de objetos é fase de treinamento.O treinamento poderá ser:

- Supervisionado : Quando existem regiões da imagem em que se dispõe de informações que permitam a identificação de uma classe de interesse.
- Não Supervisionado : Quando se utiliza algoritmos para reconhecer as classes presentes na imagem, o treinamento é dito não-supervisionado.

6.3.1 Classificação de Objetos com o Algoritmo Local Binary Patern - LBP

O algoritmo de LBPH é um algoritmo que já data de alguns anos,muito usado e conhecido no reconhecimento de objetos na área de visão computacional para a classificação.O algoritmo tem a característica de não ser muito sensível a a luminosidade.

Kevin Salton. *Reconhecimento Facial: Como funciona o LBPH*. URL: `https://updatedcode.wordpress.com/2017/11/26/reconhecimento-facial-como-funciona-o-lbph/`, O 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.

6.3.2 Classificando Objetos com O LBPH

Para a implementação da analise do algoritmo de classificação em questão foram usados a base de dados numérica fornecida pelo laboratório Pedrosa, *Base de Dados Numerica*, op. cit., LAPISCO - IFCE com consentimento do professor Pedro Pedrosa que á foi previamente processada e preparada para a aplicação neste trabalho tem uma base muito semelhante a implementação do descritor HOG.

O LBPH em termos de algoritmo precisa possui 4 parâmetros sendo:

1. Raio : Adaptação feita para o algoritmo considerar uma distancia circular para construção do padrão binário,o padrão desse valor é um.
2. Vizinhos : tamanho da vizinhança ao redor do ponto na iteração corrente ,custo computacional é proporcional a esse parâmetro.
3. 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.
4. Grade Y : Igual a Grade X só que na vertical.

Abaixo o algoritmo implementado para o treinamento e classificação dos objetos para a a base de dados,foram considerados para essa etapa 50% das imagens da base para treino e os outros 50% para o reconhecimento ,classificação.

```
import cv2
import  os
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

reconhecedor = cv2.face.LBPHFaceRecognizer_create()
reconhecedor.read("classificadorLbph.yml")
largura,altura = 35,35
def recNumber():
  caminhos = [os.path.join('NumberEval',f)for f in os.listdir('NumberEval
      ')]
  # print(caminhos)
  idCorretoSeq= []
  idsPredictSeq = []
  for caminhosImagens in caminhos:
  imagemNumber = cv2.cvtColor(cv2.imread(caminhosImagens),cv2.
      COLOR_BGR2GRAY)
  idCorreto = (int(os.path.split(caminhosImagens)[-1].split('_')[1]))
  idTest, confianca = reconhecedor.predict (imagemNumber)
  idCorretoSeq.append(idCorreto)
  idsPredictSeq.append(idTest)
  if(idCorreto == idTest):
    print("Acertou!!!",end=' ,')
  else:
    print ("Errou!!!", end=' ,')
  print(confianca)
idCorretoSeq = np.array(idCorretoSeq)
idsPredictSeq = np.array(idsPredictSeq)
print(confusion_matrix(idCorretoSeq,idsPredictSeq))
print ("Accuracy = " + str (accuracy_score (idCorretoSeq, idsPredictSeq))
    )
```

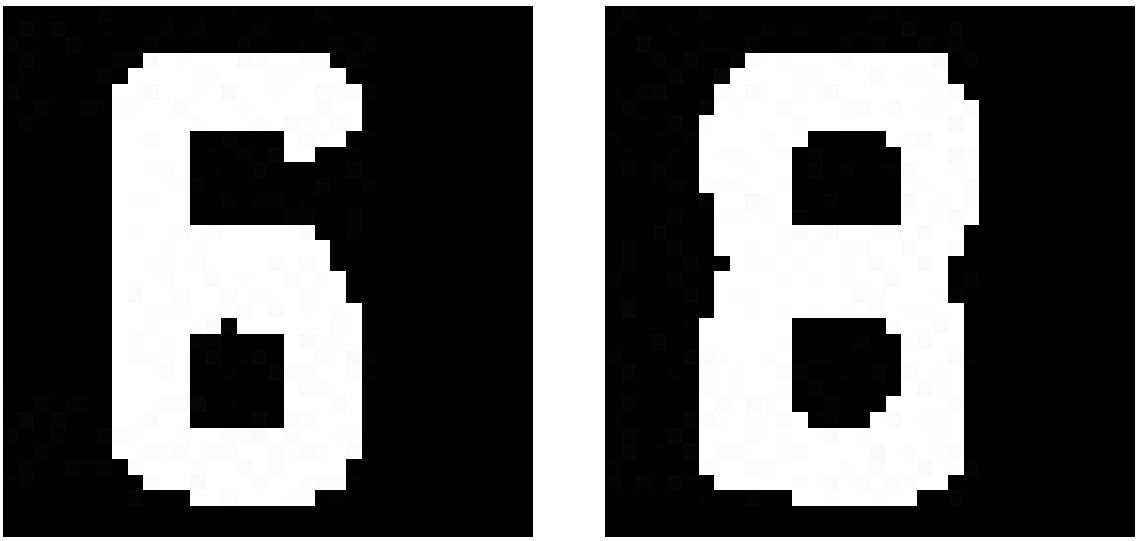
Figura 10: Implementação Classificador LBPH em liguagem python

6.3.3 Resultados

Numero 0	0	1	2	3	4	5	6	7	8	9
0	158	0	0	1	0	0	1	0	0	0
1	0	160	0	0	0	0	0	0	0	0
2	0	0	151	7	0	0	0	2	0	0
3	1	1	0	158		0	0	0	0	0
4	1	2	0	0	158	0	0	0	0	0
5	0	0	0	0	0	160	0	0	0	0
6	3	0	0	0	0	0	134	0	20	3
7	0	0	0	0	0	0	0	157	0	0
8	5	0	0	5	0	0	45	0	99	5
9	4	0	0	6	0	3	7	1	3	136

Tabela 2: Matriz de Confusão do Algoritmo LBPH

Com os resultados acima obtivemos uma acurácia de 0.919375 o que nos mostra que para a base de dados fornecida o algoritmo conseguiu uma taxa de acerto razoável ainda pela matriz de confusão se percebe que a maior ocorrência de erros se deu entre os números 6 e 8 muito provavelmente pela semelhança nas imagens dos conjuntos de pixels nas regiões de mesma posição em ambas as imagens.Basta ver que,visualmente o 8 pode ser visto o como uma complemento do numero 6 e o erro ocorre pois na base de treino há ocorrências da situação da coincidência de pixels já citada acima.



(a) Numero 6 (b) Numero 8

Figura 11: Números 6 e 8 muitos pixels nas mesmas áreas
??

6.3.4 Classificação de Objetos com o Algoritmo EigenFaces

O algoritmo de EigenFaces é um muito utilizado para reconhecimento facial contudo pode ser facilmente adaptado para o reconhecimento de objetos diversos.

Eigenfaces é um algoritmo com forte apelo geométrico analítico.O conceito de um conjunto de autovetores é a base para a execução do algoritmo.Os próprios auto-vetores em conjunto com os seus escalares permite a reconstrução de uma imagem a ser analisada e formam um conjunto base de todas as imagens usadas para construir a matriz de covariância. Isso produz a redução de dimensão, permitindo que o conjunto menor de imagens da base de treino represente as imagens de treinamento originais. A classificação pode ser obtida comparando como as faces são representadas pelo conjunto de bases.

O algoritmo ainda faz uso do Principal Component Analisis(PCA) consiste basicamente na otimização no uso e consideração dos auto-vetores na fase de treino de forma a descartar os auto-vetores que possuem informação menos relevante ou repetitiva para a classificação dos objetos em questão, o que permite em geral mais agilidade ao processo todo de classificação.

Com o algoritmo treinado reconhecimento se pelo uso do algoritmo KNN,tradução para vizinhos mais próximos ,que calcula recorrentemente a proximidade das características de

um elemento a classificar com as características de uma determinada classe,sendo que quando mais características em comum mais o elemento pertence àquela classe segundo a própria definição do algoritmo.Ankush Raut. *k Nearest Neighbors. Explained.* URL: https://medium.com/data-science-group-iitr/k-nearest-neighbors-knn-500f0d17c8f1

```
import cv2
import os
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score


reconhecedor = cv2.face.FisherFaceRecognizer_create()
reconhecedor.read("classificadorEigen.yml")
largura,altura = 35,35

def recNumber():
  caminhos = [os.path.join('NumberEval',f)for f in os.listdir('NumberEval')]
  # print(caminhos)
  idCorretoSeq= []
  idsPredictSeq = []
    for caminhosImagens in caminhos:
    imagemNumber = cv2.cvtColor(cv2.imread(caminhosImagens),cv2.COLOR_BGR2GRAY)
    idCorreto = (int(os.path.split(caminhosImagens)[-1].split('_')[1]))
    idTest, confianca = reconhecedor.predict (imagemNumber)
    idCorretoSeq.append(idCorreto)
    idsPredictSeq.append(idTest)
    if(idCorreto == idTest):
      print("Acertou!!!",end=' ,')
    else:
      print ("Errou!!!", end=' ,')
  print(confianca)
  idCorretoSeq = np.array(idCorretoSeq)
  idsPredictSeq = np.array(idsPredictSeq)
  print(confusion_matrix(idCorretoSeq,idsPredictSeq))
  print ("Accuracy = " + str (accuracy_score (idCorretoSeq, idsPredictSeq)))
```

Figura 12: Implementação Classificador EigenFaces em linguagem python

6.3.5 Resultados

Numero	0	1	2	3	4	5	6	7	8	9
0	160	0	0	0	0	0	0	0	0	0
1	0	160	0	0	0	0	0	0	0	0
2	0	0	160	0	0	0	0	0	0	0
3	0	1	0	159	0	0	0	0	0	0
4	1	0	0	0	160	0	0	0	0	0
5	0	0	0	0	0	160	0	0	0	0
6	0	0	0	0	0	0	160	0	0	0
7	0	0	0	0	0	0	0	160	0	0
8	0	0	0	0	0	0	0	0	159	0
9	0	0	0	0	0	0	0	0	0	160

Tabela 3: Matriz de Confusão para o Classificador EigenFaces

Com os resultados acima obtivemos uma acurácia de 0.99875 o que nos mostra que para a base de dados fornecida o algoritmo conseguiu uma taxa de acerto considerado muito boa o que mostra que o algoritmo de classificação EigenFaces foi muito eficiente no quesito precisão o que é de se esperar já que apesar de nos ser abstrato implementa em seu algoritmo uma otimização PCA,bastante relevante, e um algoritmo especifico de classificação de objetos o KNN.

Referências

Opencv. *HogDocs*. URL: https://docs.opencv.org/3.4.1/d5/d33/structcv_1_1HOGDescriptor.html.

Pedrosa, prof. *Base de Dados Numerica*.

Raut, Ankush. *k Nearest Neighbors. Explained.* URL: https://medium.com/data-science-group-iitr/k-nearest-neighbors-knn-500f0d17c8f1.

Salton, Kevin. *Reconhecimento Facial: Como funciona o LBPH*. URL: https://updatedcode.wordpress.com/2017/11/26/reconhecimento-facial-como-funciona-o-lbph/.

Visual Pattern Recognition by Moment Invariants". URL: http://www.sci.utah.edu/~gerig/CS7960-S2010/handouts/Hu.pdf.

wikipedia. *Descritor$_H$OG*. URL: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients.

— *Momentos de Hu*. URL: https://pt.wikipedia.org/wiki/Momentos_invariantes_de_uma_imagem.

— *Momentos invariantes de uma imagem*. URL: https://pt.wikipedia.org/wiki/Momentos_invariantes_de_uma_imagem.