Projeto em Python – Detecção de Cores usando Pandas e OpenCV

POR DATAFLAIR TEAM· ATUALIZADO · 7 DE JANEIRO DE 2020

Projeto Python sobre detecção de cores

O projeto de hoje será emocionante e divertido de construir. Trabalharemos com cores e você aprenderá sobre muitos conceitos ao longo deste projeto. A detecção de cores é necessária para reconhecer objetos, ele também é usado como uma ferramenta em vários aplicativos de edição e desenho de imagens.

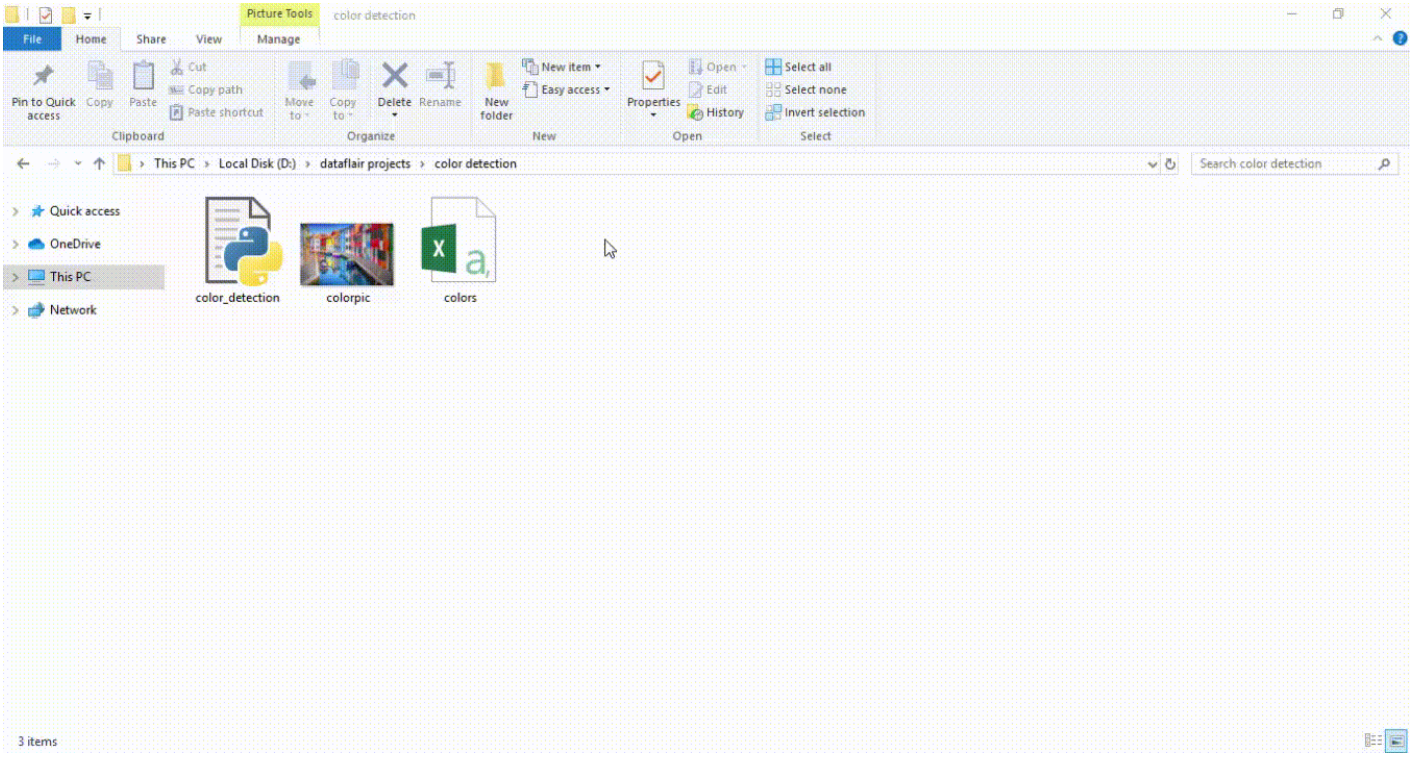
Este é o 10º projeto da série de 20 projetos Python do DataFlair. Sugiro que marque os projetos anteriores:

1. Projeto Python de detecção de fake news
2. Projeto Python de detecção de doença de Parkinson
3. Projeto Python de detecção de cores
4. Projeto Python de reconhecimento de emoção de fala
5. Projeto Python de Classificação do Câncer de Mama
6. Projeto Python de detecção de idade e gênero
7. Projeto Python de reconhecimento de dígitos manuscrito
8. Projeto Chatbot Python
9. Projeto Python de detecção de sonolência do driver
10. Projeto Python de reconhecimento de sinais de trânsito
11. Projeto Python gerador de legendas de imagem

Keeping you updated with latest technology trends, Join DataFlair on Telegram

O que é detecção de cores?

A detecção de cores é o processo de detecção do nome de qualquer cor. Simples, não é? Bem, para os humanos esta é uma tarefa extremamente fácil, mas para computadores, não é simples. Olhos humanos e cérebros trabalham juntos para traduzir luz em cor. Receptores de luz que estão presentes em nossos olhos transmitem o sinal para o cérebro. Nosso cérebro então reconhece a cor. Desde a infância, mapeamos certas luzes com seus nomes de cores. Usaremos a mesma estratégia para detectar nomes de cores.



Sobre o Projeto Python

Neste projeto Python de detecção de cores, vamos construir um aplicativo através do qual você pode obter automaticamente o nome da cor clicando neles. Então, para isso, teremos um arquivo de dados que contém o nome da cor e seus valores. Então vamos calcular a distância de cada cor e encontrar a mais curta.

O conjunto de dados

As cores são compostas de 3 cores primárias; vermelho, verde e azul. Em computadores, definimos cada valor de cor dentro de uma faixa de 0 a 255. Então, de quantas maneiras podemos definir uma cor? A resposta é 256 * 256 * 256 = 16.581.375. Existem aproximadamente 16,5 milhões de maneiras diferentes de representar uma cor. Em nosso conjunto de dados, precisamos mapear os valores de cada cor com seus nomes correspondentes. Mas não se preocupe, não precisamos mapear todos os valores. Usaremos um conjunto de dados que contém valores RGB com seus nomes correspondentes. O arquivo CSV para o nosso conjunto de dados foi retirado deste link:

Conjunto de dados de cores

O arquivo colors.csv inclui 865 nomes de cores, juntamente com seus valores RGB e hex.

Pré-requisitos

Antes de começar com este projeto Python com código fonte, você deve estar familiarizado com a biblioteca de visão computacional do Python que é ***OpenCV*** e ***Pandas***.

OpenCV, Pandas e numpy são os pacotes Python necessários para este projeto em Python. Para instalá-los, basta executar este comando pip em seu terminal:

```
pip instalar opencv-python numpy pandas
```

Etapas para construir um projeto em Python – Detecção de cores

Aqui estão as etapas para construir um aplicativo em Python que pode detectar cores:

1. Baixe e desfaça o arquivo zip

Arquivo zip de detecção de cores

A pasta do projeto contém 3 arquivos:

- **Color_detection.py**– principal fonte do nosso projeto.
- **Colorpic.jpg**– imagem de amostra para experimentação.
- **Colors.csv**– um arquivo que contém nosso conjunto de dados.

2. Tirar uma imagem do usuário

Estamos usando a biblioteca argparse para criar um analisador de argumentos. Podemos dar diretamente um caminho de imagem a partir do prompt de comando:

```
importação argparse

ap = argparse. ArgumentParser()
p.add_argument('-i',  '--image',required=True, help="Image Path")
args =  vars(ap.parse_args ())
img_path = args['imagem']
#Reading imagem com opencv
img = cv2. imread(img_path)
```

3. Em seguida, lemos o arquivo CSV com pandas

A biblioteca pandas é muito útil quando precisamos executar várias operações em arquivos de dados como CSV. **pd.read_csv()**lê o arquivo CSV e o carrega no DataFrame pandas. Nós designamos cada coluna com um nome para fácil acesso.

```
#Reading arquivo csv com pandas e dando nomes para cada coluna
índice=["cor","color_name","hex","R","G","B"]
csv = pd. read_csv('colors.csv',names=index, header=None)
```

4. Defina um evento de retorno de chamada do mouse em uma janela

Primeiro, criamos uma janela na qual a imagem de entrada será exibida. Em seguida, definimos uma função de retorno de chamada que será chamada quando um evento do mouse acontecer.

```
cv2. chamadoJanela('imagem' )
cv2. setMouseCallback('imagem'draw_function )
```

Com essas linhas, nomeamos nossa janela como 'imagem' e definimos uma função de retorno de chamada que chamará **de draw_function()**sempre que ocorrer um evento do mouse.

Quer rever os conceitos python?

Confira 270+ Tutoriais pythone melhore seu básico

5. Crie o draw_function

Ele vai calcular os valores rgb do pixel que nós clicamos duas vezes. Os parâmetros de função têm o nome do evento, (x,y) coordenadas da posição do mouse, etc. Na função, verificamos se o evento é duplo-clicado, então calculamos e definimos os valores r,g,b junto com as posições x,y do mouse.

```
def  draw_function (evento, x,y, bandeiras, param):
 se evento == cv2. EVENT_LBUTTONDBLCLK:
 global b,g,r,xpos,ypos, clicado
 clicado = True
        xpos = x
        ypos = y
b,g,r = img[y,x]
b =  int(b)
g =  int(g)
r =  int(r)
```

6. Calcule a distância para obter o nome da cor

Temos os valores r,g e b. Agora, precisamos de outra função que nos devolva o nome de cor dos valores RGB. Para obter o nome da cor, calculamos uma distância(d) que nos diz o quão perto estamos da cor e escolhemos a que tem distância mínima.

Nossa distância é calculada por esta fórmula:

d = abs (Vermelho – ithRedColor) + (Verde – ithGreenColor) + (Azul – ithBlueColor)

```
def  getColorName (R,G,B):
mínimo =  10000
para i no  intervalo(len(csv):
d =  abdominais(R-int  (csv.loc(i,"R"))  +  abdômen(G-  int(csv.loc(i,"G")) +
abdômen(B-  int(csv.loc[i,"B" ]))
        se(d<=mínimo):
 mínimo = d
cname = csv.loc[i,"color_name"]
 retornar cname
```

7. Exibir imagem na janela

Sempre que ocorrer um evento de duplo clique, ele atualizará o nome da cor e os valores de RGB na janela.

Usando a função **cv2.imshow(),**desenhamos a imagem na janela. Quando o usuário clica duas vezes na janela, desenhamos um retângulo e onome de cor para desenhar texto na janela usando funções **cv2.retângulo**e **cv2.putText().**

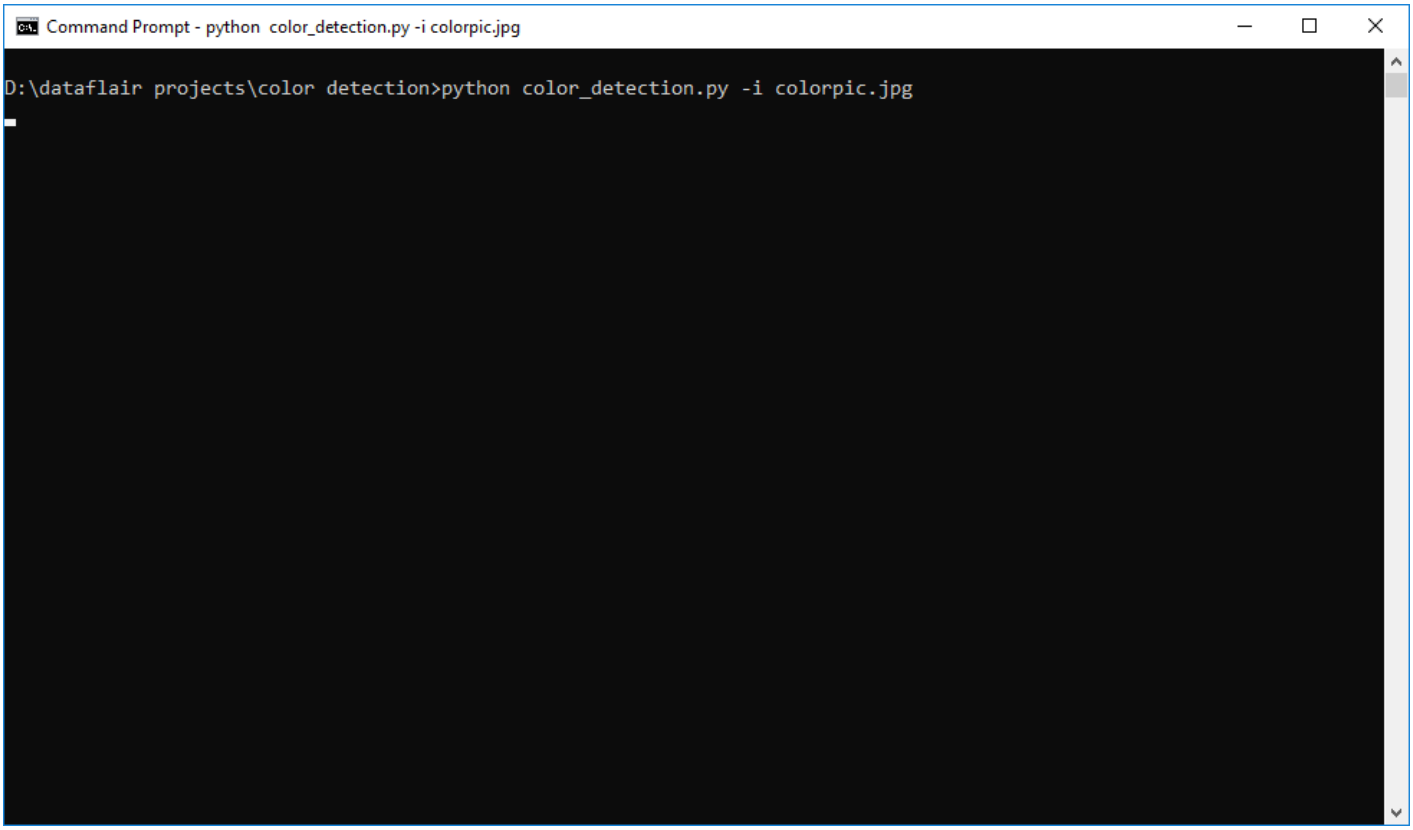
```
(1):
cv2. imshow("imagem",img )
    se  (clicado):
        #cv2.retângulo (imagem, ponto de partida, ponto final, cor, espessura)
-1 espessura preenche totalmente o retângulo
cv2. retângulo(img,(20,20),  (750,60),  (b,g,r), -1 )

        #Creating seqüência de texto para exibir (nome de cor e valores RGB)
text =  getColorName(r,g,b)  +  ' R='+  str(r)  +  ' G='+  str(g)  +  ' B='+
str(b) )

        #cv2.putText (img,text,start,font(0-7), fontScale, cor, espessura,
lineType, (fundo opcionalLeft bool) )
cv2. putText(img, texto,(50,50),2,0,8,(255,255,255),2,cv2. LINE_AA)
  #For very light colours we will display text in black colour
        if(r+g+b>=600):
            cv2.putText(img, text,(50,50),2,0.8,(0,0,0),2,cv2.LINE_AA)

```

```
            clicked=False

        #Break the loop when user hits 'esc' key
        if cv2.waitKey(20) & 0xFF ==27:
            break

cv2.destroyAllWindows()
```

8. Run Python File

The beginner Python project is now complete, you can run the Python file from the command prompt. Make sure to give an image path using '-i' argument. If the image is in another directory, then you need to give full path of the image:

```
python color_detection.py -i <add your image path here>
```



Screenshots:

color_detection.py - D:\dataflair projects\color detection\color_detection.py (3.6.0)

File Edit Format Run Options Window Help

```
import cv2
import numpy as np
import pandas as pd
import argparse

#Creating argument parser to take image path from command line
ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', required=True, help="Image Path")
args = vars(ap.parse_args())
img_path = args['image']

#Reading the image with opencv
img = cv2.imread(img_path)

#declaring global variables (are used later on)
clicked = False
r = g = b = xpos = ypos = 0

#Reading csv file with pandas and giving names to each column
index=["color","color_name","hex","R","G","B"]
csv = pd.read_csv('colors.csv', names=index, header=None)
```

color_detection.py - D:\dataflair projects\color detection\color_detection.py (3.6.0)

File Edit Format Run Options Window Help

```
#function to calculate minimum distance from all colors and get the most matching color
def getColorName(R,G,B):
    minimum = 10000
    for i in range(len(csv)):
        d = abs(R- int(csv.loc[i,"R"])) + abs(G- int(csv.loc[i,"G"]))+ abs(B- int(csv.loc[i,"B"]))
        if(d<=minimum):
            minimum = d
            cname = csv.loc[i,"color_name"]
    return cname

#function to get x,y coordinates of mouse double click
def draw_function(event, x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b,g,r,xpos,ypos, clicked
        clicked = True
        xpos = x
        ypos = y
        b,g,r = img[y,x]
        b = int(b)
        g = int(g)
        r = int(r)

cv2.namedWindow('image')
cv2.setMouseCallback('image',draw_function)
```

```
while(1):

    cv2.imshow("image",img)
    if (clicked):

        #cv2.rectangle(image, startpoint, endpoint, color, thickness)-1 fills entire rectangle
        cv2.rectangle(img,(20,20), (750,60), (b,g,r), -1)

        #Creating text string to display( Color name and RGB values )
        text = getColorName(r,g,b) + ' R='+ str(r) +  ' G='+ str(g) +  ' B='+ str(b)

        #cv2.putText(img,text,start,font(0-7),fontScale,color,thickness,lineType )
        cv2.putText(img, text,(50,50),2,0.8,(255,255,255),2,cv2.LINE_AA)

        #For very light colours we will display text in black colour
        if(r+g+b>=600):
            cv2.putText(img, text,(50,50),2,0.8,(0,0,0),2,cv2.LINE_AA)

        clicked=False

    #Break the loop when user hits 'esc' key
    if cv2.waitKey(20) & 0xFF ==27:
        break

cv2.destroyAllWindows()
```

Ln: 59 Col: 0

Output:

Double click on the window to know the name of the pixel color

Resumo

Neste projeto Python com código fonte, aprendemos sobre cores e como podemos extrair valores de Cores RGB e o nome de cor de um pixel. Aprendemos como lidar com eventos como clicar duas vezes na janela e vimos como ler arquivos CSV com pandas e realizar operações com dados. Isso é usado em inúmeros aplicativos de edição e desenho de imagens.

Quer se preparar para sua entrevista com python?

Pratique 150+ Python Interview Questions& get hired as Python expert

Espero que você tenha gostado de construir este projeto e continue visitando para projetos mais legais.