**Ing PROJECT 4 FINAL REPORT**

**Transforming Ideas to Innovation I&II**

Date April 28, 2023

Graphical user interface, website

Description automatically generated

Celebrity Guessing Game with *Haar Cascade* and *OpenCV* Face Recognition Algorithms

Team #16

Ju Na Lee

# INTRODUCTION

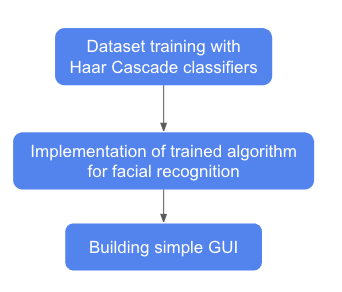
Image processing involves the gathering of information from media files, and is implemented in the fields of AI, ML, and deep learning. Additionally, face detection and recognition are branches of image processing that identifies and authenticates individuals from their facial features. For this project, an online OpenCV tutorial from *freecodecamp.org* was used as a reference. The first algorithm used for face detection is Haar Cascade Classifiers, which identifies faces in real-time image (or a video) through edge or line detection features using pixel intensities. The second algorithm was Local Binary Patterns (LBPs), and was used for facial recognition, which is built into OpenCV’s library. LBPs compare neighboring pixels in binary code and create a histogram for LBP features.

This topic was chosen because I am interested in pursuing research in image processing as well as implementing object detection in autonomous vehicles. More importantly, I am also in Electric Vehicle Club, where object detection is a critical aspect to autonomous vehicle driving.

The **deliverable** is a *Celebrity Guessing Game* combined OpenCV for face detection and recognition as well as the CustomTkinter library for building a GUI. The program can be divided into three components. The first part involves training the dataset using a Haar Cascade face detection algorithm. The second part involves implementing the algorithm in addition to OpenCV’s LBP (local binary patterns)

Diagram

Description automatically generated



*Fig. 1 Functional breakdown flowchart (left), GUI breakdown (right)*

# Overview of inputs and outputs

When the program runs, the user will be displayed with a game introduction page, prompting them to press the “Start Game” button. Then, a random image is picked from the image dataset (using the random image generation function) where the user is prompted to make a guess. Hence, the **input** is the user’s guess of who the celebrity might be. The **output** validates the user’s input and displays whether they are correct with green (correct) or red (incorrect) colored text. A text widget will also print out the algorithm’s confidence, as well as identifying the detected face by surrounding the object with a green rectangle. The algorithm’s prediction is also validated against the actual answer by looking at the file path.

# user-defined function descriptions

|  |
| --- |
| **Name of Function:** create\_train  **Return:** void  **Parameters:** none  **Description:** Takes in an image path, and turns them into grayscale. Finds region of interest. |
| **Name of Function:** RIG  **Return:**   1. path2 – randomly selected file path 2. ind – celebrity name from file path   **Parameters:** subjects – list of celebrities to choose from  **Description:** Randomly picks an image using the random library and choice function. |
| **Name of Function:** GUI  **Return:** None  **Parameters:** celebrities – list of celebrities to choose from  **Description:** Graphic user interface programmed with *tkinter*, uses nested functions for various functionalities. |
| **Name of Function:** haar\_analysis  **Return:**   1. name – recognized face with haar cascade algorithm 2. confidence – calculated confidence level with haar cascade algorithm   **Parameters:** img\_path – image file path  **Description:** Performs face recognition and finds region of interest to recognize celebrity and calculate confidence level. |
| **Name of Function:** on\_submit  **Return:** img\_path – image file path  **Parameters:**   1. entry – user-input entry widget in root frame 2. img\_path – image file path 3. celeb – celebrity folder from image file path 4. img\_label – image label widget 5. instr – instructions text widget   **Description:** Upon pressing enter button, removes widget on root frame and checks for input. Displays outputs and validates user input with algorithm classifier, and draws a green rectangle around identified face. |
| **Name of Function:** check\_special\_char  **Return:** boolean  **Parameters:** string – user-input entry string  **Description:** Checks if user input contains any special characters. |
| **Name of Function:** check\_special\_char  **Return:** boolean  **Parameters:** string – user-input entry string  **Description:** Checks if user input contains any special characters. |
| **Name of Function:** user\_input  **Return:** none  **Parameters:**   1. img\_path – image file path 2. celeb - celebrity folder from image file path 3. img\_label – image label widget 4. instr - instructions text widget   **Description:** Removes user-input entry widget and upon pressing enter button, calls on\_submit. |
| **Name of Function:** disp\_img  **Return:** none  **Parameters:**   1. button – initial “Start Game” button 2. description – initial game description text widget   **Description:** Displays randomly picked image on root frame. |
| **Name of Function:** main  **Return:** none  **Parameters:** none  **Description:** Calls all functions together to build celebrity guessing game. |

# User manual

This program is a simple “Celebrity Guessing Game”. An image is shown to the player, and they must guess who it is.

*Libraries*

1. os – to access files and directories in local device
2. cv2 – openCV for image processing
3. numpy – to convert image pixels into arrays
4. random – to randomly generate an image path
5. customtkinter – to create GUI
6. PIL (Image, ImageTk) – to read in images onto root frame
7. tkinter – for image labelling

*Data Input*

This program was trained 1800+ images of celebrity face of 18 Hollywood celebrities. The dataset was taken from Kaggle (see references section below).

The celebrities contained in this dataset include:

* Angelina Jolie
* Brad Pitt
* Denzel Washington
* Hugh Jackman
* Jennifer Lawrence
* Johnny Depp
* Kate Winslet
* Leonardo DiCaprio
* Megan Fox
* Natalie Portman
* Nicole Kidman
* Robert Downey Jr.
* Sandra Bullock
* Scarlett Johansson
* Tom Cruise
* Tom Hanks
* Will Smith

*Sample Input/Outputs*

**Input:** Angelina Jolie (or angelina jolie)

**Output:** You are correct, the algorithm detected Angelina Jolie with a confidence of 0.82. The actual answer is Angelina Jolie. The algorithm was correct.

A picture containing text, person

Description automatically generatedGraphical user interface, text, application

Description automatically generated

*Root Frame*: displays game description and “Start Game” button (left)

*Image Frame:* shows randomly celebrity and prompts user for a guess (right)

A picture containing text, person

Description automatically generated

*Sample Output:* case where user input is wrong, but algorithm is correct

# appendix

**File 1:** celebrity\_guessing\_game.py

import os

import numpy as np

import cv2 as cv

#Global Variables

DIR = r'/Users/junaleeiscool/Desktop/Project4/dataset'

celebrities = []

for i in os.listdir(DIR):

celebrities.append(i)

features = [] #image array of faces

labels = [] #who's face in index format

haar\_cascade = cv.CascadeClassifier('haar\_face.xml') #face classifier

#Takes in an image path and turns them grayscale, finds region of interest

def create\_train():

for cel in celebrities:

path = os.path.join(DIR, cel)

label = celebrities.index(cel)

img\_path = []

for img in os.listdir(path):

img\_path.append(os.path.join(path, img))

for i in range(len(img\_path)):

if img\_path[i].endswith('.DS\_Store'):

del img\_path[i]

for i in img\_path:

img\_array = cv.imread(i)

gray = cv.cvtColor(img\_array, cv.COLOR\_BGR2GRAY) #turns grayscale

#haar\_cascade classifier

faces\_rect = haar\_cascade.detectMultiScale(gray, scaleFactor = 1.1, minNeighbors = 6)

for (x, y, w, h) in faces\_rect:

faces\_roi = gray[y: y+h, x: x+w] #crop image

features.append(faces\_roi)

labels.append(label)

create\_train()

print("---------- Training complete ----------")

#Training the recognizer with features & labels list

face\_recognizer = cv.face.LBPHFaceRecognizer\_create() #uses local binary patterns (LBPs)

features = np.array(features, dtype = 'object')

labels = np.array(labels)

face\_recognizer.train(features, labels)

face\_recognizer.save('celebrity\_game\_trained.yml') #saves as a yaml source file

# 

**File 2:** project4\_final\_lee4078.py

import os

import cv2 as cv

import numpy as np

import random as r

import customtkinter as ctk

from PIL import Image, ImageTk

from tkinter import Label

# Global variables

DIR = r'/Users/junaleeiscool/Desktop/Project4/dataset'

haar\_cascade = cv.CascadeClassifier('haar\_face.xml')

# Random image picker

def RIG(subjects):

# First random picks celebrity

ind = r.choice(subjects)

path1 = os.path.join(DIR, ind)

files = os.listdir(path1)

# Deletes DS Store files (no directory)

for i in range(len(files)):

if files[i].endswith('.DS\_Store'):

del files[i]

# Second random picks image

img = r.choice(files)

path2 = os.path.join(path1, img)

return(path2, ind)

# GUI page

def GUI(celebrities):

root = ctk.CTk()

root.geometry("700x700")

main\_title = root.title("Guess the Celebrity")

title\_label = ctk.CTkLabel(root, text = "Guess Who?", font = ctk.CTkFont(size = 30, weight = "bold"))

title\_label.pack(pady = 20)

# Game description

description = ctk.CTkLabel(root, text = "This is a celebrity guessing game. \n"

"The image dataset was taken from Kaggle by Vishesh Thakur: \n"

"kaggle.com/datasets/vishesh1412/celebrity-face-image-dataset \n\n"

"Haar cascade was used to train 1800+ images for face recognition. \n"

" \n"

"github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade\_frontalface\_default.xml\n"

"Customtkinter library was used to code the GUI. \n\n"

"Learning objectives: OpenCV, face recognition, image processing, Tkinter GUI. \n\n"

"ENGR 130 Project 4 \n"

"Author: Ju Na Lee (April 2023)", font = ctk.CTkFont(size = 15)) #EXPAND THE DESCRIPTION

description.pack(pady = 20)

#------ Part 2: Implementing trained algorithm & finding ROI

def haar\_analysis(img\_path):

haar\_cascade = cv.CascadeClassifier('haar\_face.xml') # Classifier

face\_recognizer = cv.face.LBPHFaceRecognizer\_create() # Uses local binary patterns (LBPs)

# Reading the trained Haar cascade

face\_recognizer.read('celebrity\_game\_trained.yml')

img = cv.imread(img\_path)

gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

faces\_rect = haar\_cascade.detectMultiScale(gray, 1.1, 4)

confidence = 0 # Initializing confidence value

for (x, y, w, h) in faces\_rect:

faces\_roi = gray[y: y + h, x: x + w]

faces\_roi = np.array(faces\_roi, dtype=np.uint8)

# Who the algorithm thinks it is

index, conf = face\_recognizer.predict(faces\_roi)

name = celebrities[index]

# Calculating confidence level

confidence = 1.0 - float(w \* h) / float(img.shape[0] \* img.shape[1])

# Draws rectangle over detected face

cv.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), thickness = 1)

b, g, r = cv.split(img)

img = cv.merge((r, g, b))

im = Image.fromarray(img)

resize = im.resize((450, 450))

imgtk = ImageTk.PhotoImage(image = resize)

label = Label(root, image = imgtk)

label.image = imgtk

label.pack(pady = 20)

return(name, confidence)

def on\_submit(entry, img\_path, celeb, img\_label, instr):

entry.pack\_forget()

instr.pack\_forget()

img\_label.pack\_forget()

input = entry.get()

def check\_special\_char(string):

for c in string:

if not (c.isalpha() or c.isdigit() or c == ' '):

return True

return False

name, confidence = haar\_analysis(img\_path)

# Checks if user input has special characters

if check\_special\_char(input):

result = ctk.CTkLabel(root, text = "ERROR: input contains special characters.\n"

"Restart the program.", font = ctk.CTkFont(size = 20), text\_color = "#D30000")

result.pack(pady = 5)

else:

if (input == celeb) or (input.lower() == celeb.lower()):

result2 = (f"You are correct, the algorithm detected {name}\n"

f"with a confidence of {confidence: .2f}.\n"

f"The actual answer is {celeb}.")

result = ctk.CTkLabel(root, text = result2, font = ctk.CTkFont(size = 15), text\_color = "#74B72E")

result.pack(pady = 3)

# Checks if algorithm is correct or incorrect

if celeb == name:

result = ctk.CTkLabel(root, text = "\nThe algorithm was correct.", font = ctk.CTkFont(size = 15), text\_color = "#74B72E")

result.pack(pady = 3)

else:

result = ctk.CTkLabel(root, text = "\nThe algorithm was incorrect.", font = ctk.CTkFont(size = 15), text\_color = "#D30000")

result.pack(pady = 3)

else:

result2 = (f"You are incorrect, the algorithm detected {name}\n"

f"with a confidence of {confidence: .2f}.\n"

f"The actual answer is {celeb}.")

result = ctk.CTkLabel(root, text = result2, font = ctk.CTkFont(size = 15), text\_color = "#D30000")

result.pack(pady = 3)

# Checks if algorithm is correct or incorrect

if celeb == name:

result = ctk.CTkLabel(root, text = "\nThe algorithm was correct.", font = ctk.CTkFont(size = 15), text\_color = "#74B72E")

result.pack(pady = 3)

else:

result = ctk.CTkLabel(root, text = "\nThe algorithm was incorrect.", font = ctk.CTkFont(size = 15), text\_color = "#D30000")

result.pack(pady = 3)

def user\_input(img\_path, celeb, img\_label, instr):

entry = ctk.CTkEntry(root, placeholder\_text = "Make a guess...", width = 250)

entry.pack(pady = 15)

entry.bind("<Return>", lambda event: on\_submit(entry, img\_path, celeb, img\_label, instr)) #ENTER EVENT!!!

def disp\_img(button, description):

button.pack\_forget()

description.pack\_forget()

img\_path, celeb = RIG(celebrities) # "celeb" is the real answer from file directory

img = cv.imread(img\_path)

b, g, r = cv.split(img)

img = cv.merge((r, g, b))

im = Image.fromarray(img)

resize = im.resize((450, 450))

imgtk = ImageTk.PhotoImage(image = resize)

img\_label = Label(root, image = imgtk)

img\_label.image = imgtk

img\_label.pack(pady = 20)

instr = ctk.CTkLabel(root, text = "Press Enter key when done.\n"

"Examples of accepted input:\n"

"(e.g. Tom Cruise or tom cruise)",

font = ctk.CTkFont(size = 15))

instr.pack(pady = 5)

user\_input(img\_path, celeb, img\_label, instr)

button = ctk.CTkButton(root, width = 300, height = 40, text = "Start Game", font = ctk.CTkFont(size = 15), command = lambda: disp\_img(button, description))

button.pack(pady = 20)

counter = 1

root.mainloop()

# End screen

#------ Main Function

def main():

#------ Part 1: Creating trained Haar cascade classifier

# List of celebrities

celebrities = []

for i in os.listdir(DIR):

celebrities.append(i)

#------ Part 3: GUI design

GUI(celebrities)

if \_\_name\_\_ == '\_\_main\_\_':

main()

# REFERENCES

Dsouza, Jason. (2021) Open CV Course – Full Tutorial with Python.

*Freecodecamp.org*. Retrieved from <https://www.youtube.com/watch?v=oXlwWbU8l2o&ab_channel=freeCodeCamp.org>

Thakur, Vishesh. (2022) Celebrity Face Image Dataset. *Kaggle.*com Retrieved from <https://www.kaggle.com/datasets/vishesh1412/celebrity-face-image-dataset>