Gesture Play

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE & ENGINEERING

Submitted to:
Prof. Kanwaldeep Kaur

Submitted By:

NAME: Karan Srivastava

UID: 18BCS1130

NAME: Divyansh Sachan

UID: 18BCS2265

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Chandigarh University, Gharuan

January 2021

PROJECT DESIGN

The project is divided into the following modules:

i. Preprocessing

ii. gesture registration

iii. Gesture extraction

iv. gesture classification

Description about all these processes are given below-

1. Preprocessing :

Preprocessing is a common name for operations with images at the lowest level of abstraction bothinput and output are intensity images. Most preprocessing steps that are implemented are –

a. Reduce the noise

b. Convert The Image To Binary/Grayscale.

c. Pixel Brightness Transformation.

d. Geometric Transformation.

2. Gesture Registration:

Registration is a computer technology being used in a variety of applications that identifies gestures in digital images. In this face registration step, faces are first located in the image usingsome set of landmark points called “localization” or “detection”. These detected faces are then geometrically normalized to match some template image in a process called “registration”.

3. Gesture Extraction:

Extraction is an important step in face recognition and is defined as the process of locating specific regions, points, landmarks, or curves/contours in a given 2-D image or a 3D rangeimage. In this feature extraction step, a numerical feature vector is generated from the resulting registered image.

4 .Emotion Classification :

In the third step, of classification, the algorithm attempts to classify the given gestures portraying one of the seven basic gestures.

DESIGN

DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

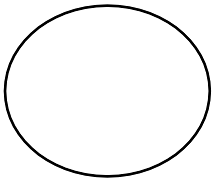
Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top- down approach to Systems Design.

Symbols and Notations Used in DFDs

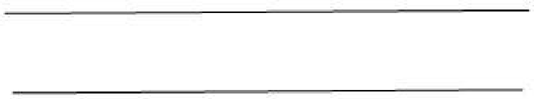
Using any convention's DFD rules or guidelines, the symbols depict the four components of data flow diagrams –

External entity: an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

Process: any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules.



Data store: files or repositories that hold information for later use, such as a database table or a membership form.



Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name, like “Billing details.”

DFD levels and layers

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

DFD Level 0 is also called a Context Diagram. It‟s a basic overview of the whole system or process being analyzed or modeled. It‟s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its subprocesses.

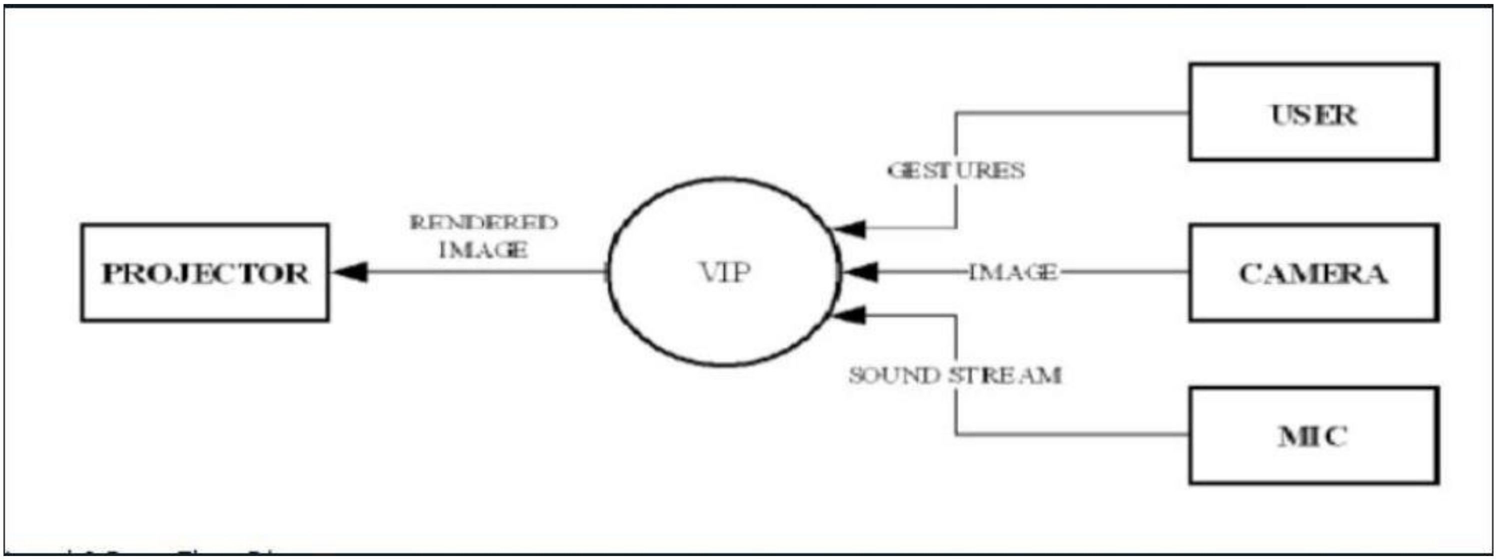
DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary

level of detail about the system's functioning.

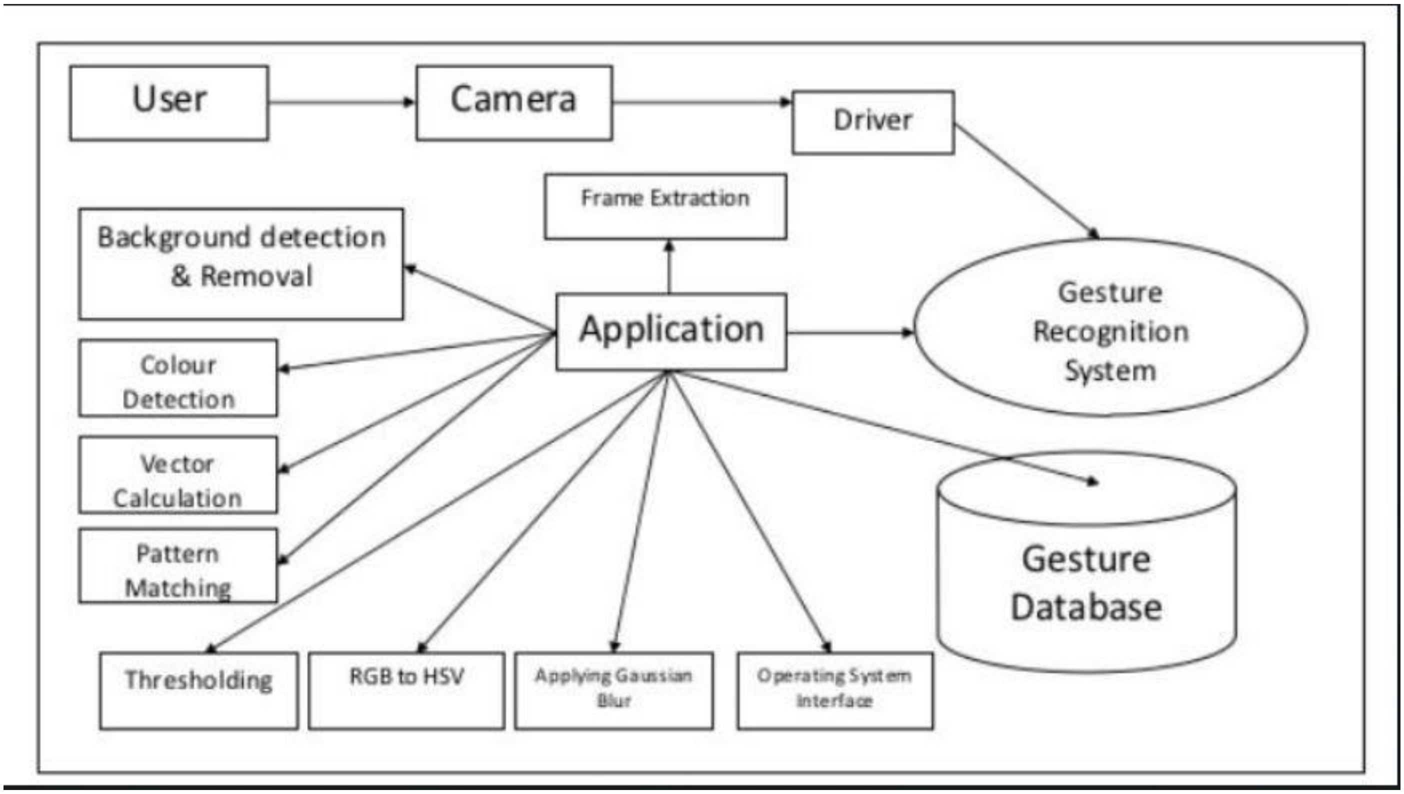
Progression to Levels 3, 4 and beyond is possible, but going beyond Level 3 is uncommon. Doing so can create complexity that makes it difficult to communicate, compare or model effectively.

Using DFD layers, the cascading levels can be nested directly in the diagram, providing a cleaner look with easy access to the deeper dive.

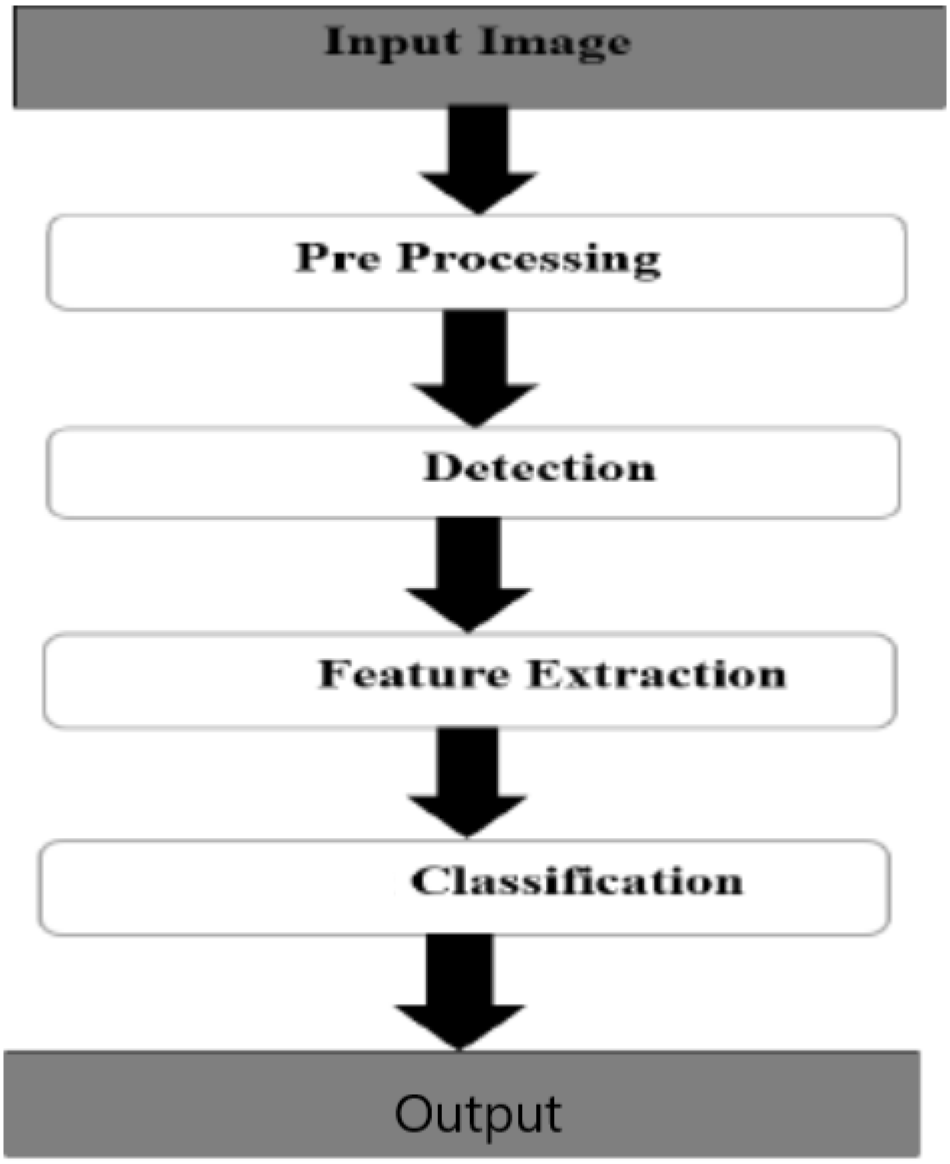
Level 0


USER
GESTURES
RENDERED
IMAGE
PROJECTOR
VIP
IMAGE
CAMERA
SOUND STREAM
MIC

Level 1


User
Camera
Driver
Frame Extraction
Background detection & Removal
Application
Gesture Recognition System
Colour Detection
Vector Calculation
Pattern Matching
Gesture Database
Thresholding
RGB to HSV
Applying Gaussian Blur
Operating System Interface

IMPLEMENTATION:



CODE:

```python
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense,Dropout,Activation,Flatten,BatchNormalization
from keras.layers import Conv2D,MaxPooling2D
import os

num_classes = 5
img_rows,img_cols = 48,48
batch_size = 32

train_data_dir= r'C:\Users\drsre\Desktop\Facial\train'
validation_data_dir= r'C:\Users\drsre\Desktop\Facial\validation'

train_datagen = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=30,
                    shear_range=0.3,
                    zoom_range=0.3,
                    width_shift_range=0.4,
                    height_shift_range=0.4,
                    horizontal_flip=True,
                    fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
                    train_data_dir,
                    color_mode='grayscale',
                    target_size=(img_rows,img_cols),
                    batch_size=batch_size,
                    class_mode='categorical',
                    shuffle=True)

validation_generator = validation_datagen.flow_from_directory(
                            validation_data_dir,
                            color_mode='grayscale',
                            target_size=(img_rows,img_cols),
```

```python
print(model.summary())

from keras.optimizers import RMSprop,SGD,Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint(r'C:\Users\drsre\Desktop\Facial\Emotion_little_vgg.h5',
                             monitor='val_loss',
                             mode='min',
                             save_best_only=True,
                             verbose=1)

earlystop = EarlyStopping(monitor='val_loss',
                          min_delta=0,
                          patience=3,
                          verbose=1,
                          restore_best_weights=True
                          )

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,
                              patience=3,
                              verbose=1,
                              min_delta=0.0001)

callbacks = [earlystop,checkpoint,reduce_lr]

model.compile(loss='categorical_crossentropy',
              optimizer = Adam(lr=0.001),
              metrics=['accuracy'])

nb_train_samples = 24176
nb_validation_samples = 3006
epochs=25

history=model.fit_generator(
                train_generator,
```

BIBLIOGRAPHY

1. https://medium.com/datadriveninvestor/real-time-facial-expression-recognition-f860dacfeb6a
2. https://www.rcciit.org/students_projects/projects/it/2018/GR8.pdf
3. https://machinelearningmastery.com/what-is-deep-learning/
4. https://sightcorp.com/knowledge-base/facial-expression-recognition/#:~:text=Facial%20expression%20recognition%20or%20computer,important%20role%20in%20interpersonal%20relations.
5. http://www.ijmer.com/papers/Vol2_Issue4/A02414491453.pdf
6. https://www.frontiersin.org/articles/10.3389/fpsyg.2017.02175/full
7. https://en.wikipedia.org/wiki/Python_(programming_language)