

Internally Funded Project

Gesture Controlled Smart Suitcase

Prithivi Rajan D – 3122 22 5001 099

Rishab S – 3122 22 5001 104

Tarunraj R -3122 22 1002 117

Guided by Dr K R Sarath Chandran

PROBLEM STATEMENT

To create a convenient and adaptable luggage transportation system that prioritizes user experience, safety, and accessibility by employing an intuitive gesture-based control mechanism, addressing current inefficiencies and challenges in conventional methods.

OBJECTIVE

The objective behind addressing the inefficiencies and challenges in conventional luggage transportation methods comes from an interest to enhance user experience, safety, and accessibility. The recognition that existing systems may cause inconvenience and lack adaptability drives the desire to create a solution that leverages intuitive gesture-based control. The aim is to make luggage transportation more user-friendly and accommodating, aligning with the evolving needs and expectations of users in various contexts, from travel to broader applications in sectors such as healthcare, retail, and logistics.

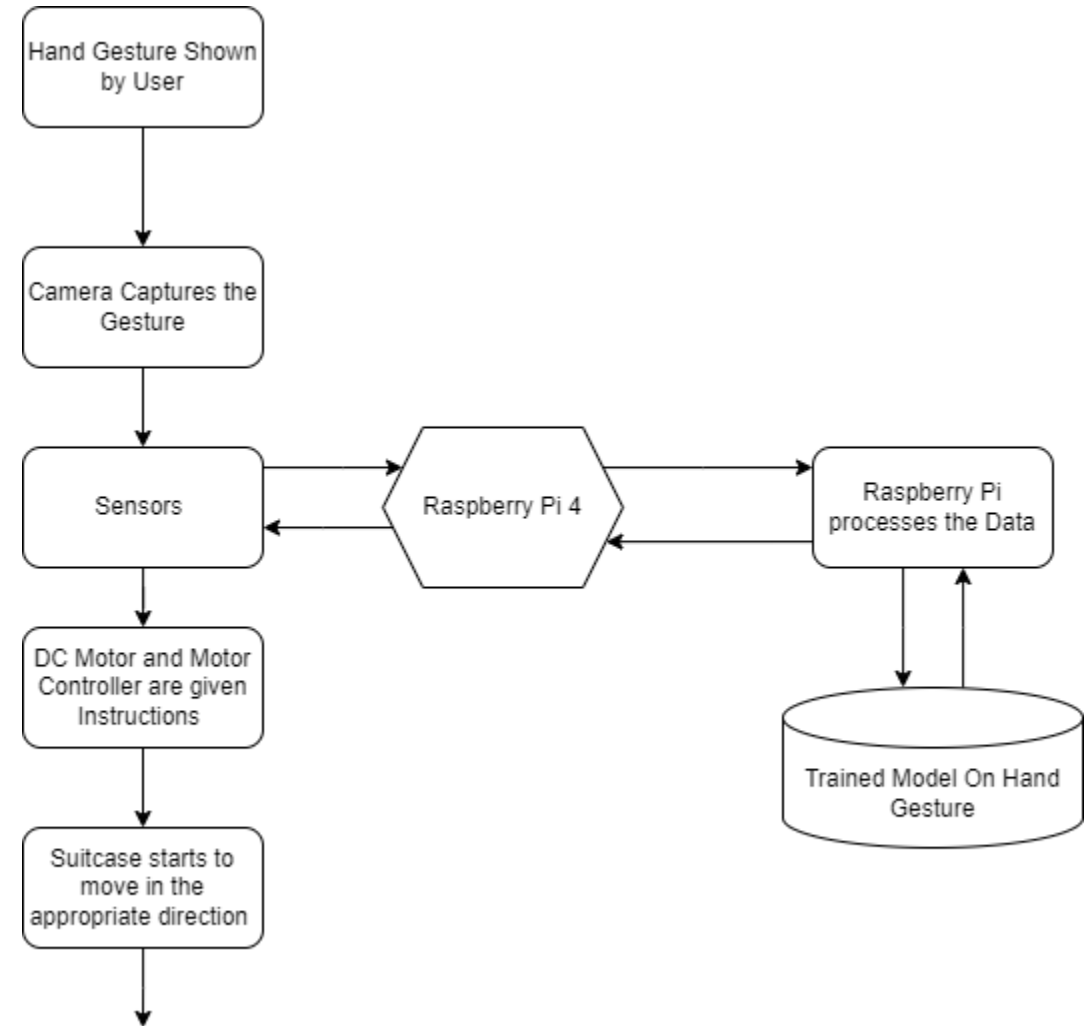
EXISTING SYSTEM

- **Wheeled Suitcases:** Standard suitcases equipped with wheels and a retractable handle, requiring users to pull or push the luggage.
- **Smart Suitcases:** Some modern suitcases come with built-in features like GPS tracking, built-in scales, or USB charging ports, but these usually don't address the physical effort of moving the suitcase.
- **Automated Luggage:** Some companies offer automated or robotic luggage that can follow users autonomously. However, these solutions often involve additional equipment and may not be widely accessible.
- **Porter Services:** Available in some places, porter services involve hiring individuals to assist with luggage handling, but this is not a universal solution.
- Our "Gesture-Controlled Smart Suitcase" project seeks to innovate beyond these existing solutions by introducing a hands-free, gesture-based control system to enhance user convenience and reduce physical strain..

PROPOSED SYSTEM AND INNOVATION

- The proposed solution is the development of a "Gesture-Controlled Smart Suitcase." This innovative luggage transportation system aims to redefine the way people interact with and move their luggage. The system utilizes cutting-edge gesture recognition technology and integrates hardware components, including the Raspberry Pi 4 with 8GB of RAM, a USB camera, Arduino sensors, motor controllers, and four DC motors within the suitcase base.
- The key aspect of the solution is the seamless integration of gesture recognition technology, where users can control the movement and direction of the smart suitcase through natural gestures. The USB camera captures real-time visual input, while Arduino sensors detect and interpret user gestures. These gestures are then translated into precise commands that activate the DC motors embedded in the suitcase, allowing users to guide the suitcase effortlessly without the need for physical lifting or pulling.
- The proposed solution not only aims to enhance the travel experience by reducing the physical strain associated with traditional luggage handling but also envisions broader applications in healthcare, retail, and logistics. The gesture recognition technology employed in the smart suitcase could potentially streamline processes and improve overall efficiency in various industries.

WORKING



MODULES(IMPLEMENTATION)

- **OpenCV (cv2)** handles image manipulation and video capture.
- **ImageDataGenerator** from TensorFlow's Keras preprocesses image data.
- The **ResNet50** model aids in image classification, and components like **Sequential** models and the **Adam optimizer** facilitate deep learning model construction and optimization for efficient computer vision applications.

```
from google.colab import drive
drive.mount('/content/drive/')

Drive already mounted at /content/drive/; to attempt to forcibly remount, call d

[5] import cv2 as cv

[6] from google.colab.patches import cv2_imshow

[7] import tensorflow as tf

[8] from tensorflow.keras import layers, models
    from tensorflow.keras.applications import ResNet50
    from tensorflow.keras.preprocessing.image import ImageDataGenerator

[9] import os
    import pathlib
    from tensorflow import keras
    from tensorflow.keras import layers
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.optimizers import Adam
```

MODULES

- **Converting to Grayscale:**
- **Listing Files:** The code uses `os.listdir()` to retrieve a list of filenames from a specified directory, enabling easy access to the files for processing.
- **Converting to Grayscale:** It iterates through each file in the directory, converts them to grayscale using `cv.cvtColor()`, and saves the resulting images to a new directory. This conversion simplifies image processing tasks by reducing color complexity.

```
import os

print(os.listdir("/content/drive/MyDrive/gestures/test/test"))
# folder path
dir_path = "/content/drive/MyDrive/gestures/test/test/move"

# list to store files
res = []

# Iterate directory
for path in os.listdir(dir_path):
    # check if current path is a file
    if os.path.isfile(os.path.join(dir_path, path)):
        res.append(path)
print(res)

for i in res:
    img = cv.imread("/content/drive/MyDrive/gestures/test/test/move/"+i)
    img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    img = cv.imwrite("/content/drive/MyDrive/newgestures/test/move/"+i, img)
```



MODULES

- **Dataset Loading:**

- Loads image data from a specified directory using TensorFlow's `image_dataset_from_directory()` function. It creates a training dataset (`train_ds`) with images resized to a specified height and width (`img_height` and `img_width`). Additionally, it sets the batch size for training to 32 and configures categorical labels for classification tasks.

```
img_height, img_width = 180, 180
batch_size = 32
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    seed = 123,
    label_mode = "categorical",
    image_size = (img_height, img_width),
    batch_size = 32
)
```

Found 87 files belonging to 4 classes.

```
[ ] val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    test_dir,
    seed = 123,
    label_mode = "categorical",
    image_size = (img_height, img_width),
    batch_size = 32
)
```

Found 1200 files belonging to 4 classes.

MODULES

- **ResNet Learning Model:**
- This code initializes a neural network model (`resnet_model`) using the ResNet50 architecture, pre-trained on the ImageNet dataset. The pre-trained layers are frozen to retain learned features. The model includes additional layers for classification tasks, followed by a summary of the model architecture.

```
resnet_model = Sequential()

pretrained_model = tf.keras.applications.ResNet50(
    include_top = False,
    input_shape = (img_height, img_width, 3),
    pooling = "avg",
    classes = 4,
    weights = "imagenet"
)
for layer in pretrained_model.layers:
    layer.trainable = False

resnet_model.add(pretrained_model)
resnet_model.add(keras.layers.Flatten())
resnet_model.add(keras.layers.Dense(512, activation="relu"))
resnet_model.add(keras.layers.Dense(4, activation="softmax"))

resnet_model.summary()
```

MODULES

- **Model Training:**
 - This code trains the ResNet50 model (`resnet_model`) on the training dataset (`train_ds`) for a 'n' epoch. The training history is stored in the `history` variable. After training, the model is saved to the specified directory (`'/content/drive/My Drive/IFP/model.keras'`) for later use or deployment.

```
epochs = 1  
history = resnet_model.fit(  
    train_ds,  
    epochs = epochs  
)
```

MODULES

- **Image Prediction:**

- This code predicts the class of a single image using the ResNet50 model. It loads and resizes the image, makes predictions, and identifies the predicted class label based on the highest probability.



```
import numpy as np

image = cv.imread("/content/drive/My Drive/stop.jpg")
image_resized = cv.resize(image, (img_height, img_width))
cv2_imshow(image_resized)
image = np.expand_dims(image_resized, axis=0)
print(image.shape)

[ ] pred = resnet_model.predict(image)
pred = list(pred[0])
print(pred)
print(class_names[pred.index(max(pred))])


1/1 [-----] - 0s 148ms/step
[1.3159597e-06, 0.000113904964, 1.1832451e-05, 0.9998729]
Stop
```



(1, 180, 180, 3)

```
image = cv.imread("/content/drive/My Drive/move.jpg")
image_resized = cv.resize(image, (img_height, img_width))
cv2_imshow(image_resized)
image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
image = cv.cvtColor(image, cv.COLOR_GRAY2BGR)
image = np.expand_dims(image_resized, axis=0)
print(image.shape)

pred = resnet_model.predict(image)
pred = list(pred[0])
print(pred)
print(class_names[pred.index(max(pred))])
```




(1, 180, 180, 3)

1/1 [=====] - 0s 136ms/step
[0.06572174, 0.8305525, 0.09374518, 0.009980526]
Move

```
image = cv.imread("/content/drive/My Drive/left.jpg")
image_resized = cv.resize(image, (img_height, img_width))
cv2_imshow(image_resized)
image = np.expand_dims(image_resized, axis=0)
print(image.shape)

pred = resnet_model.predict(image)
pred = list(pred[0])
print(class_names[pred.index(max(pred))])
```



(1, 180, 180, 3)

1/1 [=====] - 0s 167ms/step
Right

```
[ ] image = cv.imread("/content/drive/My Drive/right.jpg")
image_resized = cv.resize(image, (img_height, img_width))
cv2_imshow(image_resized)
image = np.expand_dims(image_resized, axis=0)
print(image.shape)

pred = resnet_model.predict(image)
pred = list(pred[0])
print(pred)
print(class_names[pred.index(max(pred))])
```



(1, 180, 180, 3)

1/1 [=====] - 0s 140ms/step
[0.002098192, 0.004491524, 0.9779555, 0.015454693]
Right

Timeline

Activity/Month	1	2	3	4	5	6	7	8	9	10	11	12
Procuring the components and required materials												
Algorithmic Software Development												
Assembling Hardware												
Compiling Modules												
Tech Integration												
Testing and Enhancing												
Task Completion and Document Publication												

BUDGET DETAILS

<u>S.No</u>	Name	Quantity	Rate	Cost
1.	Raspberry Pi 4, 8GB RAM	1	10000	10000
2.	USB Camera	1	1600	1600
3.	DC Motors	4	250	1000
4.	Motor Controller	2	300	600
5.	Suitcase	1	5500	5500
6.	Base	1	1300	1300
			<u>TOTAL :</u>	20000