**INSTRUCTIONS**

This project implements an intelligent sliding door system using a Raspberry Pi 5, a camera, and an AI object detection model (YOLOv8n). The system detects individuals approaching the door and opens it when someone is detected from the front. It also provides visual feedback through LEDs and an active buzzer.

To implement the Yolov8-based algorithm, follow these steps:

**Step 1: Data Collection**

**Collect Data:**

* Record video footage of people approaching and passing through doors, capturing various movements and body directions.
* A diverse collection of images of individuals and groups of people in different environments.
* Include diverse scenarios in terms of angles, lighting conditions, and subjects to create a robust dataset.

**Annotate Data:**

* Use annotation tools like CVAT or Roboflow to label frames with bounding boxes and direction labels (e.g., front, side, back).

**Step 2: Model Training with YOLOv8**

**Prepare the Dataset:**

* Format your annotated data into a format compatible with YOLOv8 (bounding boxes and labels in YOLO format).

**Training YOLOv8:**

* Use Ultralytics YOLOv8 for training. Fine-tune a pre-trained YOLOv8 model on your custom dataset.
* Ensure the training data includes diverse examples to enhance model generalization.
* Ensure the training is done on a notebook with GPU runtime.

!pip install ultralytics

from ultralytics import YOLO

# Path to the configuration file

config\_path = os.path.join(root\_path, "config.yaml")

# Load a model

model = YOLO("yolov8n.pt")

# Use the model

results = model.train(data=config\_path, epochs=150, workers=2)  # train the model

**Step 3: Direction Recognition Model**

**Feature Extraction:**

* Use the bounding boxes and keypoints detected by YOLOv8 to determine the direction of movement.
* Analyze the trajectory of detected objects over successive frames.

**Direction Classification:**

* Implement a simple rule-based algorithm or train a small neural network to classify the movement direction based on bounding box coordinates and their changes over time.

**Download the trained YOLOv8n model:**

* You can obtain the best.pt in the ‘weights’ folder of the trained Yolov8n model using custom dataset.

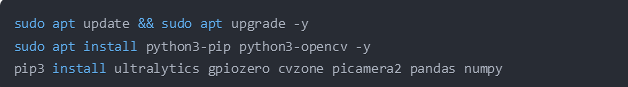
**Create a class file:**

* Create a text file named coco.txt and add the class names (e.g., back, front, side, etc.) corresponding to the trained model.

**Step 4: Deployment on Raspberry Pi 5**

**Set Up the Raspberry Pi:**

* Ensure the Raspberry Pi 5 is set up with the necessary libraries: OpenCV, Ultralytics YOLOv8, and any other dependencies.
* Install necessary packages:



Connect the Pi Camera to the Raspberry Pi.

Connect the LEDs and the buzzer to the GPIO pins on the Raspberry Pi according to the following mapping:

* Green LED: GPIO 17 (Door Open indicator)
* Red LED: GPIO 27 (Door Closed indicator)
* Yellow LED: GPIO 22 (Detection indicator)
* Blue LED: GPIO 23 (Standby indicator)
* Buzzer: GPIO 24 (Door action indication)

Ensure the Raspberry Pi has proper power supply and is connected to a monitor.

**Real-time Inference:**

* Capture video frames using the Raspberry Pi camera.
* Run the YOLOv8 model on each frame to detect humans and predict their movement direction.

**Step 5: System Integration and Testing**

**Integrate with Door Control System:**

* Develop an interface to send control signals to the door mechanism based on detected movements. This is implemented using the visual feedback from the LEDs.

**Test and Optimize:**

* Test the entire system to ensure it works reliably under various conditions.
* Optimize for speed and accuracy, ensuring it runs smoothly on the Raspberry Pi 5.

**Tools and Libraries**

* **Pose Estimation and Object Detection:** YOLOv8 (Ultralytics)
* **Real-time Video Processing:** OpenCV
* **Data Annotation:** CVAT, Roboflow

**Considerations**

* **Hardware:** Ensure the Raspberry Pi 5 has sufficient power and cooling for running the models in real-time.
* **Edge Computing:** Optimize the YOLOv8 model using desired optimizer (Kingma, 2015).

**Notes**

* The system is designed to open the door when a person is detected approaching from the front and close it when no person is detected, side view detected or when a person is detected moving away.
* Ensure the Raspberry Pi is properly configured for camera support (enable camera interface in raspi-config).
* Modify GPIO pin assignments in the code if using different pin configurations.

**Troubleshooting**

* If the camera is not detected, ensure it is properly connected and enabled.
* Check the GPIO pin connections if the LEDs or buzzer are not responding as expected.
* Review logs in the terminal for any errors related to model prediction or frame capture.

**License**

This project is Open Licensed