IT & Robotic Lab

Author : Hasan Emre AYDIN, Alperen ISIK, Ya-Sin MAFTOUH

Cursus : EENG4

UE : IT & Robotic Lab

Version : 0.1

Date : 02/10/2023

Human-Robot RPS Project

Agenda

- Project Objective and Technologies
- Robot's Operating Principle
- Arduino Code Overview
- Python Interface Features
- Python Code Key Functions
- YOLO Model and Image Processing
- Game Flow and Modes
- Code Logic Details
- Project Completion
- Challenges and Solutions

Introduction

Project Objective and Technologies

Project Objective

Enable a rock-paper-scissors game between a human and a robot, fostering entertaining human-machine interaction.

Technologies Used

Utilized Python for developing the game's interface and controlling the robot, ensuring seamless interaction.

Arduino Integration

Arduino controlled servo motors to execute the robot's hand movements based on detected gestures.

Gesture Detection with YOLO

Employed the YOLO model for real-time hand gesture detection, allowing the robot to respond accurately to user inputs.

Robot's Operating Principle

The robot employs a camera to capture real-time video of the user, utilizing the YOLO model for gesture detection. Once a gesture is identified, the robot translates this input into specific movement commands, such as closing fingers for rock, keeping fingers open for paper, or specific finger positions for scissors. These commands are then sent to the Arduino, which controls the servo motors to replicate the corresponding gesture accurately.

Code

Arduino Code Overview

01

Servo Motor Settings

Servo motors require specific minimum and maximum values to function correctly, which were calibrated individually to ensure accurate movement.

02

Main Commands

Key functions include makeRock(), which closes all fingers for rock, makeScissors(), which keeps the middle and index fingers open for scissors, and makePaper(), which opens all fingers to represent paper.

03

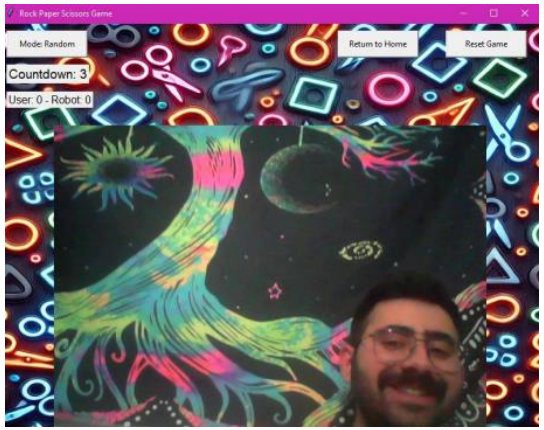
Communication Protocol

The Arduino receives commands for 'rock,' 'sci,' or 'paper' via the serial port, interpreting these inputs to perform the corresponding motor movements.

Python Interface Features

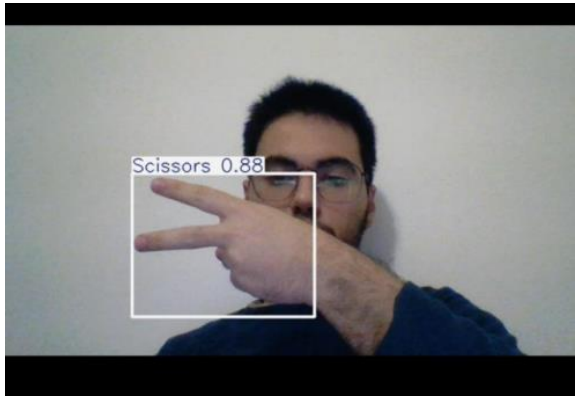
Main Menu

The main menu offers options to start the game or exit the application, providing a user-friendly entry point.



Game Screen

The game screen shows the choices made by both the user and the robot, including a scoreboard to track scores and outcomes.



Camera Gesture Detection

A dedicated area displays the camera feed, which actively detects user gestures for rock, paper, or scissors using the YOLO model.

Functionality

Python Code Key Functions

01 **Game Modes**

The game features two modes: Random Mode, where the robot makes random choices, and Robot Wins Mode, where the robot selects moves to win against the user.

02 **User Detection**

Utilizes the YOLO model to accurately detect user hand gestures for 'Rock,' 'Paper,' and 'Scissors' during gameplay.

03 **Result Calculation**

Compares the detected gestures of the user and the robot to determine the winner, displaying results such as 'Draw,' 'User Wins,' or 'Robot Wins.'

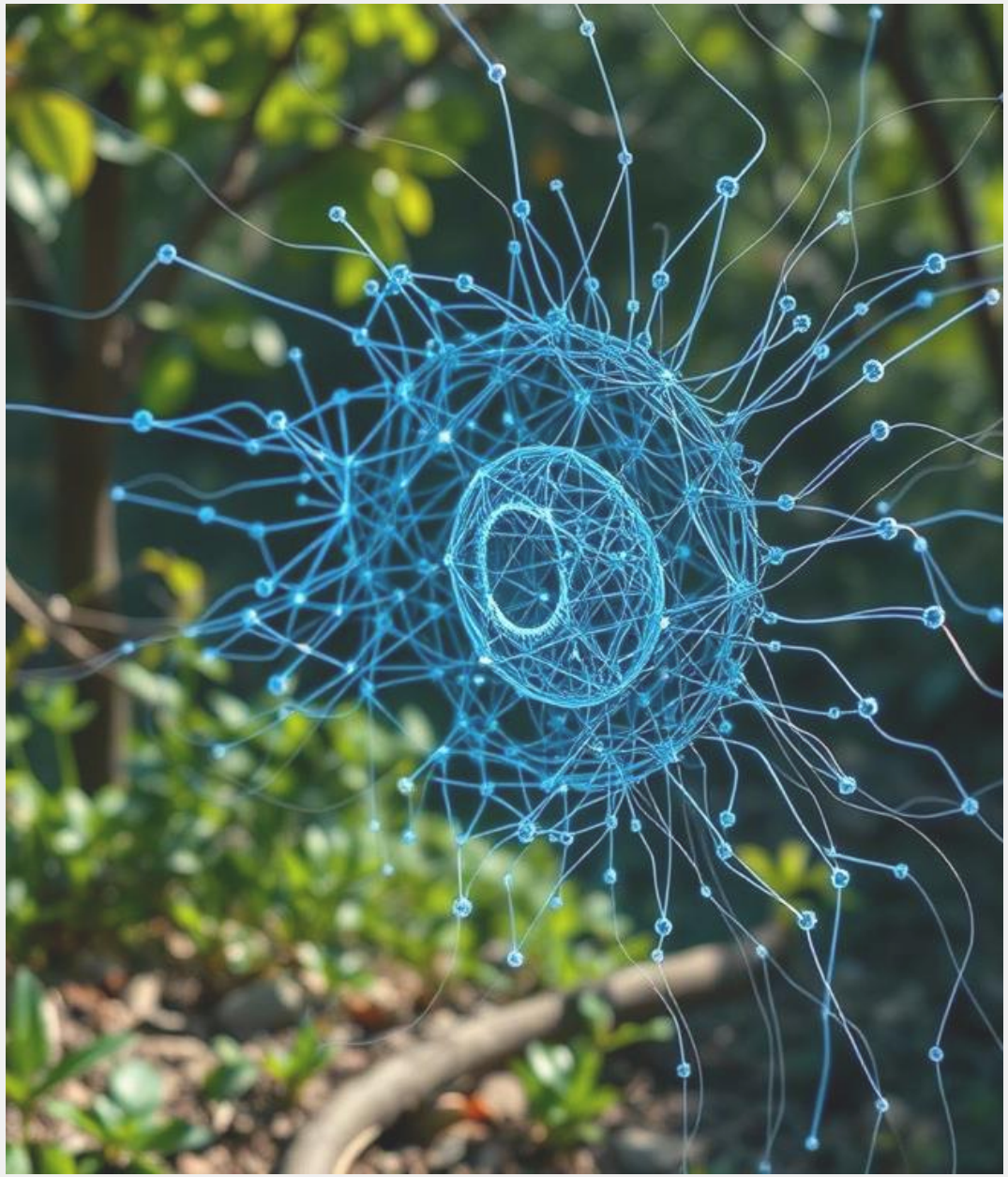
04 **Robot Control**

Commands for robot gestures are sent to the Arduino through the serial port, facilitating real-time interaction based on user input.

AI, Deep Learning, and YOLO Overview

Key Concepts

- Artificial Intelligence (AI) refers to the simulation of human intelligence processes by machines, particularly computer systems. AI can perform tasks such as reasoning, learning, and problem-solving.
- Deep Learning is a subset of machine learning that utilizes neural networks with three or more layers to analyze various factors of data. It enables the recognition of patterns and complex structures in large datasets.
- The YOLO (You Only Look Once) model is a real-time object detection system that identifies and classifies objects in images or videos in a single pass, allowing for efficient processing and high accuracy.

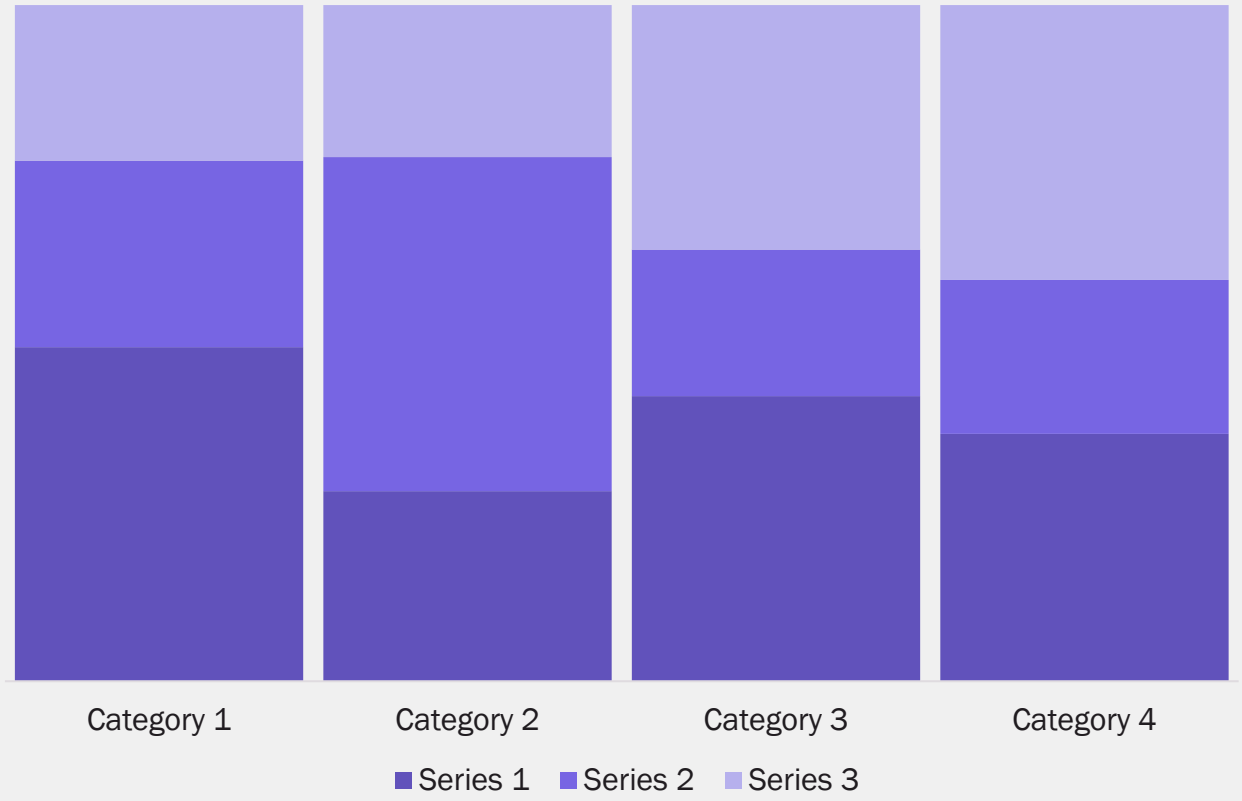


Dataset from Roboflow

Overview of Data Split

- Data was labeled using Roboflow, ensuring high-quality annotations for accurate training.
- The dataset was split into three segments: 88% for training, 8% for validation, and 4% for testing.

Data Labeling and Distribution



Model Training in Google Colab

01

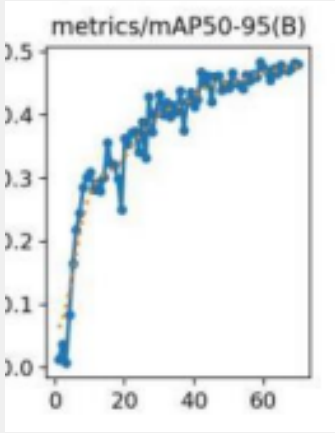
Model Training Process

- The YOLO model is trained using labeled data in Google Colab.
- Colab provides a cloud-based environment with GPU support for accelerated training.
- Data preprocessing steps include normalization and augmentation to enhance performance.

02

Impact of Epoch Count

- Increasing the epoch count allows the model to learn more from the training data.
- Higher epochs can lead to better feature extraction and improved accuracy.
- However, too many epochs may cause overfitting, monitoring validation loss is essential.



Functionality

Role of the YOLO Model

Gesture Recognition Process

- The YOLO model processes video frames in real-time to identify hand gestures.
- It employs a convolutional neural network to detect and classify gestures.
- Bounding boxes are drawn around recognized gestures to provide visual feedback.

Results Interpretation

- Detected gestures trigger specific actions or commands based on user input.
- Results are displayed on the screen, indicating the recognized gesture.
- The system continually updates as new gestures are detected.

Game Mechanics

Game Flow and Modes

Countdown

The game begins with the robot counting down from 3 to signal the start of the round, preparing the user to make their gesture.

Countdown timer displayed on screen

Gesture Detection

The robot uses the YOLO model to detect the user's hand gesture as soon as it is made, identifying 'Rock', 'Paper', or 'Scissors'.

Detected gesture relayed to the robot

Robot Move

After detecting the user's gesture, the robot randomly selects its own gesture in Random Mode or chooses the optimal move in Robot Wins Mode, executing it with servo motors.

Robot's gesture displayed and executed

Result Calculation

The game logic compares the user's gesture to the robot's gesture to determine the outcome, updating the scoreboard with the results.

Results displayed: 'Draw', 'User Wins', or 'Robot Wins' and updated scores

Code Logic Details

01 **Camera Functionality**

The camera is accessed using the cv2.VideoCapture method, which captures video frames for processing.

02 **Gesture Comparison**

Captured frames are analyzed to compare user gestures against predefined gestures, determining outcomes like 'User Wins' or 'Draw'.

03 **Arduino Communication**

Commands indicating user gestures such as 'rock', 'paper', or 'scissors' are sent to the Arduino via the serial port for robot actions.

Overview

The project successfully implemented an engaging rock-paper-scissors game between a human and a robot.

All targeted functionalities were achieved, allowing for seamless interaction and entertainment. The robot accurately recognized user gestures and responded appropriately, creating a lively experience. This project demonstrated the capabilities of human-robot interaction and showcased effective communication between software and hardware components.

Challenges and Solutions

01 Servo Motor Settings Issues

Movement limits of the servo motors were not properly calibrated, causing inconsistent actions during gameplay.

02 YOLO Model Misinterpretation

The YOLO model struggled with accurately recognizing hand gestures, leading to incorrect game outcomes.

03 Serial Port Delays

Delays in communication between the Python script and Arduino hindered timely responses from the robot.

04 Implemented Solutions

Specific min/max values were set for servo motors, the YOLO model was retrained with additional data, and optimized timeout values were established to enhance communication.

Thanks for your attention