Tic-Tac-Toe Game with YOLO

Hazem Essam

September 17, 2024

Contents

1 Introduction

This document outlines the development and implementation of a real-time Tic-Tac-Toe game using YOLO for gesture recognition. The game allows players to make moves by performing specific hand gestures, which are detected and processed by the YOLO model.

2 Requirements

- Develop a real-time Tic-Tac-Toe game using computer vision.
- Use YOLO for detecting hand gestures: "X" and "O".
- Divide the screen into a 3x3 grid for gameplay.
- Collect and annotate a dataset of hand gestures.
- Document the project using LaTeX.

3 Dataset Gathering

- Collected and annotated dataset with gestures "X" and "O".
- Added gestures like "Like" and "Dislike" for enhanced interaction.
- Dataset split: 85% training, 15% validation.
- Model chosen: YOLOv8s.

4 Methodology

- Implemented YOLOv8s for gesture detection.
- Developed game logic to handle gesture-based inputs.
- Integrated YOLO model with real-time video capture using OpenCV.

5 Code Implementation

5.1 Key Functions

Listing 1: Drawing the Tic-Tac-Toe grid

```
def draw_grid(frame):
    width = frame.shape[1]
    height = frame.shape[0]
    cv.line(frame, (int(width / 3), 0), (int(width / 3), height), (200, 255, 100),
    thickness=3)
    cv.line(frame, (int((2 * width) / 3), 0), (int((2 * width) / 3), height), (200,
    255, 100), thickness=3)
    cv.line(frame, (0, int(height / 3)), (width, int(height / 3)), (200, 255, 100),
    thickness=3)
    cv.line(frame, (0, int((2 * height) / 3)), (width, int((2 * height) / 3)), (200,
    255, 100), thickness=3)
```

Listing 2: Function to draw "X" on the grid

```
def draw_x(frame, center_x, center_y, player):
    size = 70
    for row in pin:
        for box in row:
```

```
            if (center_x <= (box[0] + (box_width / 2))) and (center_x >= (box[0] - (
    box_width / 2))):
                if (center_y <= (box[1] + (box_height / 2))) and (center_y >= (box[1]
    - (box_height / 2))) and player:
                    if box[2] != 2:
                        box[2] = 1
            if box[2] == 1:
                top_left = (box[0] - size // 2, box[1] - size // 2)
                bottom_right = (box[0] + size // 2, box[1] + size // 2)
                top_right = (box[0] + size // 2, box[1] - size // 2)
                bottom_left = (box[0] - size // 2, box[1] + size // 2)
                cv.line(frame, top_left, bottom_right, (0, 200, 255), thickness=4)
                cv.line(frame, top_right, bottom_left, (0, 200, 255), thickness=4)
```

Listing 3: Main Loop of the Game

```
cap = cv.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv.flip(frame, 1)
    draw_grid(frame)
    results = model(frame)
    for result in results:
        for detection in result.boxes:
            cls = detection.cls[0]
            xyxy = detection.xyxy[0]
            conf = detection.conf[0]
            center_x = int((xyxy[0] + xyxy[2]) / 2)
            center_y = int((xyxy[1] + xyxy[3]) / 2)
            if conf > 0.5:
                if cls == 0:
                    draw_o(frame, center_x, center_y, x_player1)
                elif cls == 1:
                    draw_x(frame, center_x, center_y, x_player1)
    get_turn()
    cv.imshow('TicTacToe', frame)
    key = cv.waitKey(1)
    if key == ord('q'):
        break
cap.release()
cv.destroyAllWindows()
```

6 Challenges and Solutions

6.1 Gesture Recognition

- **Issue**: The model initially struggled to accurately recognize the "Like" gesture.
- **Solution**: Added more training samples and improved data quality by including images with varied lighting and hand positions.

6.2 Integration Problems

- **Issue**: The script processed only the first frame due to debugging settings.
- **Solution**: Implemented a 2-second counter to correctly pass the frame to the YOLO model and updated the game logic to handle predictions accurately.

7 Results

7.1 Performance

- Improved gesture recognition accuracy with the addition of new training data.
- Stable integration of the YOLO model with real-time game updates.

7.2 Gameplay

- The game now successfully recognizes and responds to player gestures, alternating between players and updating the game grid accordingly.

8 Conclusion

The project successfully integrates YOLO object detection with a real-time Tic-Tac-Toe game, providing a seamless and interactive experience. The system accurately detects gestures and updates the game state, demonstrating the potential for real-time computer vision applications.

9 References

- YOLOv8 Documentation
- OpenCV Documentation
- PyAutoGUI Documentation