Group 14

Human Obstacle Detector and Tracker Using YOLO

Component Description

We propose to design and develop a human obstacle detector and tracker using the **YOLO** (You Only Look Once) object detection algorithm. While the system will be initially tested virtually using a laptop camera, it will be developed with future robotics applications in mind, such as integration into mobile robots for autonomous navigation in public environments (e.g., shopping malls, airports, warehouses). The system's primary function is to detect and track humans in real-time, enabling a robot to avoid collisions, track movement, and adjust its behavior for safe navigation around people.

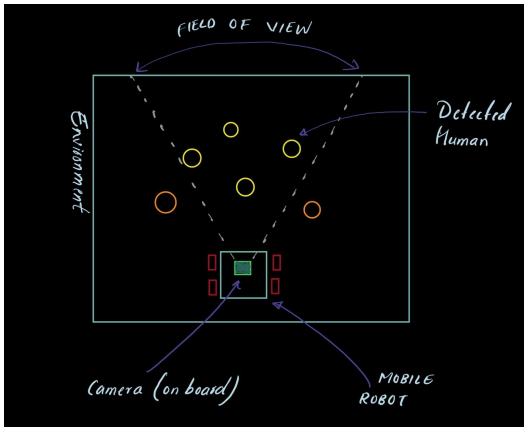
Importance and Usage

The human obstacle detector and tracker is a crucial safety component in robotic systems that operate in environments with human presence. It ensures that the robot can:

- Detect human presence reliably and maintain a safe distance.
- Adjust its path dynamically to avoid collisions.
- Track the movement of people to maintain safe proximity or follow certain individuals.

Interface with Other Components

- **Camera Input**: The YOLO-based detector will receive video input from laptop cameras (for testing purposes) or RGB cameras (for robotics applications).
- **Robot Navigation stack (Expecting future Robotic Application)**: The human detection and tracking data would be used as input to the navigation stack of the robot to make the robot move intelligently around humans.



Assumptions

- The system will operate in an indoor environment.
- RGB cameras will provide visual data input (using the laptop camera for testing).
- In future use cases, the robot will have an existing control system capable of adjusting its path based on obstacle information.

Design and Development Process

1. **Requirement Gathering**: Identify specific requirements such as detection range, environment complexity, and real-time processing constraints.
2. **Design**: Architect the system to integrate YOLO with camera input and control modules (for robotics), keeping modularity for future robotics integration.
3. **Development**:
    - Implement the YOLO-based human detector.
    - Create the necessary interfaces to integrate with a robot's control system (future step).
4. **Testing**: Perform testing using the laptop camera in various virtual environments to ensure reliable detection and tracking.

5. **Quality Assurance**: Implement continuous integration with unit tests focusing on detection accuracy and system responsiveness.

Technologies and Tools

- **Programming Language**: C++ for implementing YOLO and real-time processing.
- **Framework**: Darknet (a C++-based framework for YOLO implementation), OpenCV (for camera input and image processing).
- **Build System**: CMake for project management.
- **Libraries**: Darknet (for YOLO), OpenCV.
- **Testing Tools**: Laptop camera (for virtual testing), Git for version control.

Algorithm Overview

- YOLO will be used for real-time human detection, processing each camera frame to identify people.
- Detected human locations will be represented as obstacle coordinates relative to the camera or robot.

Risks/Unknowns and Mitigation

- **Technical Risk**: YOLO's detection accuracy may decrease in challenging conditions such as low-light environments or with occlusions.
  Mitigation: Consider testing in well-lit environments and eventually adding support for other sensors (e.g., LiDAR) for sensor fusion.
- **Real-time Performance**: YOLO may face performance bottlenecks when processing real-time video feeds, especially on non-GPU hardware.
  Mitigation: Use a GPU-accelerated laptop for testing and consider further optimization for robotics applications.
- **Unknown Environments**: In future robotic applications, the system may encounter unexpected obstacles (furniture, walls).
  Mitigation: Test in diverse environments and eventually integrate other obstacle detection systems alongside YOLO.

Final Deliverables to Acme

- YOLO-based human detection and tracking system tested using a laptop camera, designed for future integration into a robot's navigation system.
- Documentation detailing the design, setup, and usage instructions.
- Test results demonstrating performance in different environments.
- Codebase with a modular design that allows further integration and improvements.

Team and Pair Programming

- **Team Members**:
  - Kashif Ansari (Driver)
  - Abhishek Avhad (Navigator)
  - Piyush Goenka (Design Keeper)

- **Pair Programming**: We will adopt a driver-navigator model where one person codes (driver) while the other reviews and suggests improvements (navigator). Each session will be recorded in version control (Git), with brief notes on the tasks completed.