Software Plan for Human Obstacle Detection and Tracking

1. Overview

Our purpose with this project is to design and develop a human obstacle detection and tracking system for Acme Robotics using the YOLO object detection algorithm. The goal is to enable real-time human detection and tracking for future integration into robotic systems, primarily for autonomous navigation. Initially, testing will be conducted virtually using a laptop camera, but the system is designed with robotics applications in mind.

Project Objectives:

- Develop a real-time human obstacle detection and tracking system.
- Ensure the system can be easily integrated into autonomous robotic platforms for navigation and safety in dynamic environments.
- Test the system using a laptop camera in a virtual setup.

Scope:

- **Covered**: Human detection and tracking in real-time using YOLO, processing camera input, and providing obstacle coordinates for navigation applications.
- **Not Covered**: Integration with specific robotic hardware or ROS; limited to laptop camera testing and software-only components.

Assumptions:

- Required hardware (laptop with camera) and software libraries will be available.
- Future integration will target robotic systems with standard RGB cameras for visual input.
- No additional sensors (e.g., LiDAR) will be used for this phase.

Constraints:

- Performance is bound by the processing power of the laptop used for testing.
- No physical robot will be tested in this phase; only virtual simulations are conducted.
- Strict adherence to deadlines for testing, development, and final integration deliverables.

Deliverables:

- Human detection and tracking software using C++ with YOLO (via Darknet).
- Documentation of system design, usage, and setup.
- Testing results from virtual environments with various test scenarios.

Evolution and Range: This plan covers initial development and testing phases. Given the complexity of integrating detection into a full robotic control system, further iterations of development will be necessary after this phase for integration and testing in actual robotic environments. This plan will be revised upon completion of virtual testing.

Summaries:

- Process: Real-time human detection using YOLO, data fed to tracking algorithms, and human movement prediction for path planning.
- Organization: Kashif (Driver), Abhishek (Navigator), Piyush (Design Keeper).
- Management: Agile-based process with pair programming and frequent reviews.

---

2. Reference Materials

- **YOLO Object Detection**: Darknet framework documentation.
- **Image Processing**: OpenCV documentation for handling camera inputs.
- **Tracking Algorithms**: Resources on filters and object tracking in dynamic environments.
- **Related Projects**: Research papers on human obstacle detection for mobile robots.

---

3. Definitions and Acronyms

- **YOLO**: You Only Look Once, an object detection algorithm.
- **Darknet**: An open-source neural network framework written in C and C++.
- **RGB**: Red-Green-Blue, a color model used for digital imaging.

---

4. Process

- **Initial Development**:
  - Implementation of YOLO using Darknet in C++.
  - Camera input handled via OpenCV for real-time video processing.
  - Human detection coordinates passed to the filter for tracking and movement prediction.
- **Testing Plan**:
  - Virtual testing with a laptop camera under various lighting conditions and environments.
  - Test accuracy of detection and tracking for different movement patterns.
- **Quality Assurance**:
  - Unit testing for YOLO detection accuracy and tracking stability.
  - Code inspections and continuous integration to ensure modularity and maintainability.

---

5. Organization

- **Kashif Ansari (Driver)**: Leads implementation of YOLO and real-time processing in C++.
- **Abhishek Avhad (Navigator)**: Reviews code, suggests improvements, and assists with testing and debugging.

- **Piyush Goenka (Design Keeper)**: Ensures the overall system architecture is aligned with the project's objectives and maintains consistency in design decisions.

---

6. Technologies

- **Programming Language**: C++ for YOLO implementation and processing.
- **Framework**: Darknet (C++-based framework for YOLO), OpenCV (for camera input and image processing).
- **Version Control**: Git for version control.
- **Hardware**: Standard laptop with a built-in camera.
- **Build System**: CMake for managing project builds.

---

7. Management

- **Monitoring**: Progress will be tracked through Agile sprints, with regular updates and task completion reports. Git will be used for version control, and automated tests will track detection accuracy and tracking performance.
- **Metrics**: Key performance indicators (KPIs) will include detection accuracy, tracking precision, and real-time performance (frames per second).
- **Risk Management**:
    - **Detection Accuracy in Challenging Conditions**: Test under varied lighting and occlusion scenarios.
    - **Performance**: Test on multiple hardware configurations and use hardware acceleration (e.g., GPU) if needed.

---

8. Cost

- **Estimated Cost**:
  The project will utilize open-source libraries and standard hardware (laptop). Costs are expected to remain low, covering minimal equipment upgrades or software licenses, if necessary. A contingency budget is reserved for potential hardware acceleration (GPU) needs in the future.
