Autonomous Image Analysis and Threat Assessment System Requirements

1. General Requirements

1.1. Performance Requirements

1.1.1. Real-Time Processing: The system must be capable of processing videos in real-time, enabling immediate object identification.

1.1.2. High Speed: Leveraging GPU usage, the system should achieve high processing speeds with minimal latency.

1.1.3. Scalability: The system should efficiently handle videos of varying resolutions and lengths.

1.2. Reliability Requirements

1.2.1. Continuous Tracking: Objects must be tracked continuously using the Deep SORT algorithm.

1.2.2. Data Consistency: Information related to identified objects should be accurately and consistently recorded.

1.3. Usability Requirements

1.3.1. User-Friendly Interface: Provide an intuitive GUI that facilitates easy interaction for users.

1.3.2. Comprehensive Documentation: Offer detailed user manuals and documentation to enable effective system usage.

---

2. Graphical User Interface (GUI) Requirements

2.1. General Interface Structure

2.1.1. Dual Window Layout: The system should consist of two separate windows—one for video display and another for control buttons.

2.1.2. Button Placement: Buttons should be arranged in a user-friendly layout for easy access and interaction.

2.2. Video Window

2.2.1. Video Loading: Video files are opened using OpenCV.

2.2.2. Video Control Buttons

2.2.2.1. Select Video Button: Allows the user to choose a video file to load into the system.

2.2.2.2. Exit Button: Closes the application and terminates all processes.

2.2.3. Playback Controls Buttons

2.2.3.1. Play Button: Starts video playback from the current position, activating object detection and tracking in real-time.

2.2.3.2. Pause Button: Pauses the video at the current frame, keeping all detected objects and tracking data visible.

2.2.3.3. Rewind Button: Plays the video in reverse from the current frame, maintaining object detection and tracking functions.

2.2.4. Object Selection: Draw bounding boxes around detected objects, allowing users to select objects by clicking on these boxes.

2.2.5. Information Frames: Display information such as object type, friend/enemy status, and threat level above the bounding boxes of selected objects.

2.3. Control Window

2.3.1. Classification Buttons

2.3.1.1. Select Object Button: Allows the user to select a single object in the video for further processing such as classification or tracking.

2.3.1.2. Select Region Button: Allows the user to select multiple objects by drawing a specific region on the video frame.

2.3.1.3. Friend Button: Classifies the selected object as a "friend" and turns its bounding box to green.

2.3.1.4. Enemy Button: Classifies the selected object as a "enemy" and turns its bounding box to red.

2.3.1.5. Track Button: Starts real-time tracking of the selected object in the video.

2.3.1.6. Stop Tracking Button: Stops tracking of the selected object and removes all tracking indicators.

2.3.1.7. Threat Assessment Button: Automatically assesses the threat level of the selected object according to its classification.

---

3. Object Detection Requirements

3.1. Model Training

3.1.1. YOLOv8 Utilization: The system must train the YOLOv8 model using appropriate datasets.

3.1.2. Dataset Management: Carefully select and diversify datasets to enhance the model's overall performance.

3.1.3. Model Optimization: Ensure the model operates quickly and efficiently by utilizing GPU acceleration.

3.2. Real-Time Detection

3.2.1. Object Identification: Upon execution, the system should identify and classify each object using pre-trained information.

3.2.2. Multiple Object Detection: Ability to identify and classify multiple objects simultaneously.

3.2.3. Detection Accuracy: Ensure high accuracy in object classification and identification.

---

4. Object Tracking Requirements

4.1. Tracking Algorithm

4.1.1. Deep SORT Integration: Integrate the Deep SORT algorithm to track objects identified by YOLOv8.

4.1.2. ID Preservation: Ensure that objects exiting and re-entering the frame retain the same ID.

4.2. Performance

4.2.1. Continuous Tracking: Maintain ongoing tracking of objects with minimal loss.

4.2.2. Rapid Updates: Quickly update object positions to reflect real-time movement.

---

5. Object Identification Requirements

5.1. Data Recording and Management

5.1.1. Object Information Recording: Record details of each identified object (ID, type, classification status, etc.) within the system.

5.1.2. Data Access: When objects exit and re-enter the frame, match them with previously recorded information to ensure continuity.

5.2. Performance

5.2.1. Fast Matching: Object identification processes should occur swiftly.

5.2.2. Accuracy: Ensure objects are correctly re-identified based on stored information.

6. Threat Assessment Requirements

6.1. Threat Level Determination

6.1.1. Default Levels: The threat level of detected objects should initially be determined automatically.

6.1.2. Dynamic Adjustment: For objects classified as "Friendly" or "Adversary" by the user, the threat level should be updated dynamically (such as Adversary: High, Friendly: Low).

6.2. Information Visualization

6.2.1. Frame Colors

6.2.1.1. Friend Objects: Green bounding boxes.

6.2.1.2. Enemy Objects: Red bounding boxes.

6.2.2. Information Labels: Display object type, classification status, and threat level above the bounding boxes in a readable format.

7. Video Processing and Control Requirements

7.1. Video Loading and Management

7.1.1. OpenCV Integration: Videos should be able to run using OpenCV.

7.1.2. User-Controlled Playback: Video and all associated functions should commence upon the user pressing the "Play" button.

7.2. Playback Controls

7.2.1. Play: Starts video playback and activates all functions.

7.2.2. Pause: Stops video playback and temporarily halts all functions.

7.2.3. Rewind: Plays the video in reverse while running all functions accordingly.

7.3. Video Pause and Resume

7.3.1. Resuming Paused Video: When the "Play" button is pressed on a paused video, playback should resume with all functionalities intact.

8. Object Selection and Classification Requirements

8.1. Object Selection

8.1.1. Bounding Boxes: Draw bounding boxes around detected objects.

8.1.2. Click-to-Select: Allow users to select objects by clicking on their bounding boxes.

8.2. Classification Operations

8.2.1. Friend Classification

8.2.1.1. Button: Pressing the "Friend" button classifies the selected object as a friend.

8.2.1.2. Visual Feedback: Bounding box turns green upon classification.

8.2.1.3. Threat Level: Set to low.

8.2.2. Enemy Classification

8.2.2.1. Button: Pressing the "Enemy" button classifies the selected object as an enemy.

8.2.2.2. Visual Feedback: Bounding box turns red upon classification.

8.2.2.3. Threat Level: Set to high.

8.2.3. Track/Stop Tracking

8.2.3.1. Track Button: Starts tracking the selected object, displaying its threat level and type.

8.2.3.2. Stop Tracking Button: Stops tracking the selected object and removes its information frame.

8.3. Classification Status Recording

8.3.1. Data Recording: After user classification, store the object's status and information within the system.

8.3.2. Status Update: When an object re-enters the frame, update its status based on previously stored information.

---

9. Technical Requirements

9.1. Hardware Requirements

9.1.1. GPU Support: The system must be compatible with high-performance GPUs and support GPU-accelerated processing.

9.1.2. Memory and Storage: Ensure sufficient RAM and storage capacity.

9.2. Software Requirements

9.2.1. Programming Language: Python, a performance-oriented language, is used.

9.2.2. Libraries and Frameworks:

9.2.2.1. YOLOv8: For object detection.

9.2.2.2. Deep SORT: For object tracking.

9.2.2.3. OpenCV: For video processing and GUI development.

9.2.2.4. GUI Framework: Use frameworks like PyQt, Tkinter, or similar for GUI development.

---

10. Testing and Validation Requirements

10.1. Functional Testing

10.1.1. Object Detection Accuracy: Verify that the YOLOv8 model correctly identifies objects.

10.1.2. Tracking Performance: Confirm that the Deep SORT algorithm tracks objects continuously without interruption.

10.1.3. GUI Functionality: Test that all buttons operate correctly and user interactions behave as expected.

10.2. Performance Testing

10.2.1. Processing Speed: Measure whether the system meets the defined performance criteria.

10.2.2. Stress Testing: Assess system performance with high-resolution and long-duration videos.

10.3. Usability Testing

10.3.1. User Experience: Evaluate whether users can easily navigate and use the system’s interface.

10.3.2. Error Handling: Test how the system manages errors and provides feedback to users during failure scenarios.