MediTracker Footage Analyzer

User Manual

Contents

01. Introduction

The MediTracker Footage Analyzer is a system designed to analyze video footage in medical environments. It automatically detects and tracks medical equipment and logs their movements.

02. System Requirements

- Python 3.10 or higher (recommended)
- CUDA-capable GPU (recommended for faster processing)
- Sufficient storage for video files and temporary data

03. Installation

- Install required dependencies: *pip install -r requirements.txt*
- Ensure you have the necessary credentials for MySQL database and Firebase.

04. Configuration

Before running the analyzer, configure the config.json file. Open it in a text editor and set the following parameters:

- **Database Settings**

  `database_host`: Your MySQL server address

  `database_user`: Database username

  `database_password`: Database password

  `database_name`: Name of the database

- **Firebase Settings**

  `firebase_credentials_path`: Path to your Firebase credentials JSON file

  `firebase_storage_bucket`: Your Firebase storage bucket name

- **Video Processing Settings**

  Detailed explanations of these parameters are provided in Section 9.

  `model_path`: Path to the YOLOv5 model file

  `clear_temp`: Boolean to clear temporary files after processing

  `confidence_threshold`: Minimum confidence level for object detection

  `frame_skip`: Number of frames to skip between processed frames buffer_size: Number of frames to retain for movement tracking

min_frames_for_logging: Minimum frames an object must be tracked before logging

movement_threshold: Minimum pixel movement to consider an object as moving

frame_width: Width to resize video frames

frame_height: Height to resize video frames

local_video_path: Path to the video file for analysis

enable_preview: Boolean to show preview window

05. Preparing Video Footage Files

- Name your video footage files in the format: cam**X**_loc**Y**_Z.mp4

  X: Camera ID
  Y: Location ID
  Z: Footage number/name (optional)

  Example: *cam1_loc2_vid06.mp4*

- Place the video file in the directory specified by local_video_path in *config.json*.

06. Running the Analyzer

- Open a terminal and navigate to the *MediTrackerFootageAnalyzer/* directory.
- Run the main script: *python3 main.py*
- The system will start analyzing the video footage file specified in *config.json*.

07. Monitoring Progress

- If enable_preview is set to **true**, a window will show the video with detection boxes.
- The console will display messages about detected equipment and their movements.

08. Viewing Results

- Detection logs are stored in the MySQL database specified in *config.json*.
- Frames with detected equipment are uploaded to Firebase Storage.

09. Detailed Configuration Parameters

model_path:

Description	Usage	Impact
File path to the YOLOv5 model used for object detection.	Specify the full path to your trained YOLOv5 model file (e.g., "/path/to/best.pt").	Directly affects the types of objects detected and the accuracy of detection.

clear_temp:

Description	Usage	Impact
Determines whether to clear temporary files after processing.	Set to true for automatic cleanup, false to retain files for debugging.	Affects disk usage and ability to review intermediate processing results.

confidence_threshold:

Description	Usage	Impact
Minimum confidence level required for an object detection to be considered valid.	Set a value between 0 and 1 (e.g., 0.4 for 40% confidence).	Lower values increase detections but may include more false positives. Higher values result in fewer but more confident detections.

frame_skip:

Description	Usage	Impact
Number of frames to skip between each processed frame.	Set to 0 to analyze all frames, or a positive integer to skip frames.	Higher values speed up processing but may miss quick movements. Lower values provide more thorough analysis but increase processing time.

buffer_size:

Description	Usage	Impact
Number of recent frame positions to retain for each detected object.	Set a positive integer value (e.g., 30 for 30 frames).	Larger values allow for more accurate movement tracking over longer periods but require more memory.

min_frames_for_logging:

Description	Usage	Impact
Minimum number of frames an object must be tracked before its movement is logged.	Set a positive integer value, typically similar to or less than buffer_size.	Higher values reduce false positives and short-lived detections. Lower values capture more brief object appearances.

movement_threshold:

Description	Usage	Impact
Minimum pixel distance an object must move to be considered as having changed position.	Set a positive integer value representing pixels (e.g., 20 for 20 pixels).	Higher values ignore small movements, reducing false movement detections. Lower values capture subtler movements but may be sensitive to noise.

frame_width and **frame_height**:

Description	Usage	Impact
Dimensions to resize video frames for processing.	Set to 0 to analyze footages at its original resolution, or positive integer values to custom resolution. (e.g., 640 for 640 pixels)	Affects processing speed and memory usage. Smaller dimensions (except 0), increase speed but may reduce detection accuracy for small objects.

local_video_path:

Description	Usage	Impact
File path to the video to be analyzed.	Specify the full path to your video file (e.g., "/path/to/cam1_loc2_vid06.mp4").	Determines which video is processed by the system.

enable_preview:

Description	Usage	Impact
Controls whether to display a preview window showing the video with detection boxes.	Set to true to enable the preview, false to disable.	Enabling preview allows real-time monitoring but may slightly increase processing load.

10. Customizing Detection

To modify detectable objects:

- Update the classes section in config.json with your desired object classes.
- Ensure your YOLOv5 model is trained to detect these classes.
- Update model_path to point to your custom-trained model.

11. Troubleshooting

- Verify all paths in *config.json* are correct and accessible.
- Check database and Firebase credentials.
- If detections are inaccurate, adjust confidence_threshold or use a better-trained model.
- For performance issues, try increasing frame_skip or reducing frame_width and frame_height.

12. Performance Optimization

➢ Frame Processing Optimization

- Frame Skip (frame_skip)

        - **Description**: Number of frames to skip between processed frames.
        - **Optimization**: Increase this value to process fewer frames, significantly reducing processing time.
        - **Trade-off**: Higher values may miss quick movements.
        - **Example**: `"frame_skip": 2` processes every third frame, potentially tripling speed.
        - **Recommendation**: Start with 1 or 2 and increase gradually, monitoring detection quality.
    - Frame Resizing (`frame_width` and `frame_height`)
        - **Description**: Dimensions to resize video frames for processing.
        - **Optimization**: Reduce these values (except 0) to decrease the amount of data processed per frame.
        - **Trade-off**: Lower resolution may affect detection accuracy, especially for small objects.
        - **Example**: `"frame_width": 640, "frame_height": 480` instead of 1920x1080.
        - **Recommendation**: Find the smallest size that maintains acceptable detection accuracy.
    - Confidence Threshold (`confidence_threshold`)
        - **Description**: Minimum confidence level for object detection.
        - **Optimization**: Increase this value to reduce the number of detections processed.
        - **Trade-off**: Higher values may miss valid detections.
        - **Example**: `"confidence_threshold": 0.4` for 40% confidence.
        - **Recommendation**: Adjust based on the model's performance characteristics.

- Movement Tracking Optimization
    - Buffer Size (`buffer_size`)
        - **Description**: Number of frames retained for movement tracking.
        - **Optimization**: Reduce this value to lower memory usage and potentially increase speed.

        - **Trade-off**: Smaller buffers may affect movement direction accuracy.
        - **Example**: `"buffer_size": 20` instead of 30.
        - **Recommendation**: Set to the minimum value that accurately captures typical movement patterns.
    - Minimum Frames for Logging (`min_frames_for_logging`)
        - **Description**: Minimum frames an object must be tracked before logging.
        - **Optimization**: Increase to reduce processing and logging of brief detections.
        - **Trade-off**: May miss logging of quickly passing objects.
        - **Example**: `"min_frames_for_logging": 15`
        - **Recommendation**: Set close to but slightly lower than `buffer_size`.
    - Movement Threshold (`movement_threshold`)
        - **Description**: Minimum pixel movement to consider an object as moving.
        - **Optimization**: Increase to reduce false movement detections and associated processing.
        - **Trade-off**: May miss subtle movements.
        - **Example**: `"movement_threshold": 25`
        - **Recommendation**: Adjust based on video resolution and expected movement patterns.
- Resource Management
    - Clear Temporary Files (`clear_temp`)
        - **Description**: Whether to clear temporary files after processing.
        - **Optimization**: Set to true to manage disk space and potentially improve I/O performance.
        - **Trade-off**: Loses ability to review intermediate results.
        - **Example**: `"clear_temp": true`
        - **Recommendation**: Enable unless debugging is needed.
    - Enable Preview (`enable_preview`)

        - **Description**: Controls display of preview window with detection boxes.
        - **Optimization**: Set to false to reduce GUI-related processing overhead.
        - **Trade-off**: Loses real-time visual feedback.
        - **Example**: `"enable_preview": false`
        - **Recommendation**: Disable for batch processing or on headless systems.

- ➢ Environment-specific Optimization:
    - For GPU environments, focus on maximizing GPU utilization.
    - For CPU-only environments, consider more aggressive frame skipping and resizing.
- ➢ Workload-based Tuning:
    - Adjust parameters based on typical video characteristics (e.g., camera movement, object size, and speed).
