API Documentation for Multi-Object Tracking with Threat Analysis System

Abstract

This document provides a formal specification of the REST API endpoints for the Multi-Object Tracking with Threat Analysis System, a Flask-based application designed for video analysis, object tracking, and threat assessment. It details endpoint functionality, request and response formats, and usage examples.

General Information

Base URL

All endpoints are accessed relative to the base URL: `http://127.0.0.1:5000` (or the deployed URL when using tools like ngrok for remote access).

Authentication

The current implementation does not require authentication. Future iterations may incorporate token-based authentication for enhanced security.

API Endpoints

1. Upload Page

- **Endpoint**: `/upload`
- **Method**: `GET`, `POST`
- **Description**: Renders the upload interface for video files or processes uploaded video files.
- **Request (GET)**:
  - No parameters required.
- **Response (GET)**:
  - **Status**: 200 OK
  - **Content-Type**: `text/html`
  - **Body**: HTML page (`upload.html`) for video upload.
- **Request (POST)**:
  - **Form Data**:
    * `file`: Video file (e.g., `.mp4`, `.avi`, `.mov`)

  - **Content-Type**: `multipart/form-data`

- **Response (POST)**:
  - **Status**: 302 Found
  - **Location**: Redirects to `/play/<filename>` (e.g., `/play/video.mp4`)
  - **Error**:
    * **Status**: 400 Bad Request
    * **Body**: `{"error": "Invalid file type"}` (if file type is not allowed)

- **Example**:
  - **GET Request**:

```
GET /upload HTTP/1.1
Host: 127.0.0.1:5000
```

  - **POST Request (using curl)**:

```
curl -X POST -F "file=@video.mp4" http://127.0.0.1:5000/upload
```

2. Play Video and Analyze Frames

- **Endpoint**: `/play/<filename>`
- **Method**: `GET`
- **Description**: Renders a page to play the uploaded video, analyze frames, and initiate object tracking.
- **Parameters**:
  - `filename` (path): Name of the uploaded video file (e.g., `video.mp4`).
- **Response**:
  - **Status**: 200 OK
  - **Content-Type**: `text/html`
  - **Body**: HTML page (`play.html`) with video player and analysis controls.
- **Example**:

```
GET /play/video.mp4 HTTP/1.1
Host: 127.0.0.1:5000
```

3. Analyze Frame

- **Endpoint**: /analyze$_{f}rame/ < filename >$ **Method** : POST
- **Description**: Analyzes a specific frame of the video at the given timestamp and returns detected objects.
- **Request**:
  - **Form Data**:
    * timestamp: Timestamp in seconds (e.g., 10.5)
  - **Content-Type**: application/x-www-form-urlencoded
- **Response**:
  - **Status**: 200 OK
  - **Content-Type**: application/json
  - **Body**:

```
{
  "frame_time": 10.5,
  "detections": [
    {"id": 1, "label": "person", "bbox": [100, 150, 200, 300]},
    {"id": 2, "label": "car", "bbox": [300, 400, 450, 500]}
  ]
}
```

  - **Error**:
    * **Status**: 400 Bad Request
    * **Body**: {"error": "Invalid timestamp"}
- **Example**:

```
POST /analyze_frame/video.mp4 HTTP/1.1
Host: 127.0.0.1:5000
Content-Type: application/x-www-form-urlencoded

timestamp=10.5
```

4. Start Tracking

- **Endpoint**: /start$_{t}racking/ < filename >$ **Method** : POST
- **Description**: Initiates tracking for a specific object in the video starting from the given timestamp.
- **Request**:
  - **Form Data**:

    * `timestamp`: Timestamp in seconds (e.g., 10.5)
  * `object_id` : $IDofthe object to track (e.g., 1)$
  - **Content-Type**: `application/x-www-form-urlencoded`

- **Response**:
  - **Status**: 200 OK
  - **Content-Type**: `application/json`
  - **Body**: `{"status": "Tracking started", "track_id": 1}` **Error** :
  - – **Status**: 400 Bad Request
    - **Body**: `{"error": "Invalid input"}`

Example:

```
POST /start_tracking/video.mp4 HTTP/1.1
Host: 127.0.0.1:5000
Content-Type: application/x-www-form-urlencoded

timestamp=10.5&object_id=1
```

5. Stop Tracking

- **Endpoint**: `/stop_tracking/` $< filename >$ **Method** : `POST`
- **Description**: Terminates tracking for a specific object and generates a tracking summary.
- **Request**:
  - **Form Data**:
  * `track_id` : $IDofthetrackingsessiontostop (e.g., 1)$
  - **Content-Type**: `application/x-www-form-urlencoded`

- **Response**:
  - **Status**: 200 OK
  - **Content-Type**: `application/json`
  - **Body**: `{"status": "Tracking stopped", "summary": "Object 1 moved from (100,150) to (200,300)"}`
  - **Error**:
    * **Status**: 400 Bad Request
  * **Body**: `{"error": "Invalid track_id"}`

- **Example**:

```
POST /stop_tracking/video.mp4 HTTP/1.1
Host: 127.0.0.1:5000
Content-Type: application/x-www-form-urlencoded

track_id=1
```

6. Generate Threat Report

- **Endpoint**: $/\texttt{threat}_{r}eport/ < filename >$ **`Method`** : `POST`
- **Description**: Generates a threat analysis report based on the tracking summary using an LLM and BERT classifier.
- **Request**:
  - **Form Data**:
  * $\texttt{track}_{i}d : IDofthetrackingsession(e.g., 1)$
  - **Content-Type**: `application/x-www-form-urlencoded`
- **Response**:
  - **Status**: 200 OK
  - **Content-Type**: `application/json`
  - **Body**:

```
{
  "track_id": 1,
  "summary": "Object 1 moved quickly in a garage.",
  "refined_summary": "Refined summary: Object 1 moved quickly in a garage.",
  "threat_analysis": {
    "violence": "Low",
    "genocide": "Minimal",
    "hate_speech": "None",
    "risk_score": 0.2
  }
}
```

  - **Error**:
    * **Status**: 400 Bad Request
  * **Body**: {"error": "Invalid $\texttt{track}_{i}d$"}
- **Example**:

```
POST /threat_report/video.mp4 HTTP/1.1
Host: 127.0.0.1:5000
Content-Type: application/x-www-form-urlencoded

track_id=1
```

Conclusion

This API provides a robust interface for interacting with the Multi-Object Tracking with Threat Analysis System. The endpoints support video processing, object tracking, and threat assessment, with potential for expansion in future releases.