MacV Object Tracker Assessment – Report – Sai Krishi S

Objective:

A real-time object tracking system that uses YOLOv8 and Streamlit to provide an interactive dashboard for video analysis and object tracking. The system processes videos locally on your hardware and provides both HTML reports and an interactive web interface.

Overview:

The Object Tracker is a powerful tool for real-time object detection and tracking in video streams. It provides two interfaces:

1. A standalone application for direct video processing with HTML report generation

2. A web-based dashboard using Streamlit for interactive analysis and real-time processing

Features:

- Real-time object detection and tracking using YOLOv8

- Multiple object tracking with unique ID assignment

- Object trail visualization

- Time tracking for each detected object

- Two output interfaces:

  1. **Interactive Web Dashboard (Streamlit):**

     - Real-time video processing

     - Object tracking statistics

     - Active objects over time visualization

     - Detailed object tracking times

     - Class-wise object counts

     - Progress tracking

     - Configurable parameters

2. **HTML Report:**

   - Embedded video playback

   - Comprehensive statistics

   - Object tracking summary

   - Class-wise object counts

   - Easy sharing and viewing

- Local hardware processing (CPU/GPU)

- Configurable confidence threshold

- Support for multiple video formats (MP4, AVI, MOV)

Tech Stack:

Core Technologies:

- Python 3.8+

- OpenCV (cv2)

- YOLOv8 (Ultralytics)

- Streamlit

- PyTorch

Additional Libraries:

- NumPy

- Pandas

- Plotly Express

- Jinja2 (for HTML report generation)

Implementation Details

Object Tracking System:

The system uses YOLOv8's tracking capabilities with the following components:

1. **Model Configuration:**

- Uses YOLOv8s.pt as the base model
- Configurable confidence threshold (default: 0.5)
- IOU threshold: 0.4
- BoTSORT tracker for robust object tracking

2. **Tracking Features:**

- Unique ID assignment for each detected object
- Centroid-based trail visualization
- Time tracking based on video frame timestamps
- Color-coded bounding boxes for different objects

3. **Performance Optimizations:**

- CPU-based processing for broader compatibility
- Efficient frame processing with OpenCV
- Optimized trail visualization (30-point history)

Web Dashboard (Streamlit)

The Streamlit interface provides:

1. **Video Processing:**

- File upload support
- Real-time processing status
- Progress tracking
- Local hardware utilization

2. **Statistics Dashboard:**

- Active objects over time graph
- Processing metrics (FPS, total frames)
- Object tracking duration statistics
- Class-wise object counts

3. **Interactive Features:**

- Adjustable confidence threshold
- Model size selection
- Real-time video playback
- Processing status monitoring

HTML Report Generation

The system generates comprehensive HTML reports that include:

1. **Video Playback:**

- Embedded processed video
- Browser-compatible format
- Easy sharing capabilities

2. **Statistics Summary:**

- Total objects tracked
- Unique objects count
- Active objects at completion
- Processing FPS
- Class-wise object distribution

3. **Report Features:**

- Clean, professional layout
- Responsive design
- Easy to share and view
- No additional software required

Installation

1. **Clone the repository:**

git clone https://github.com/SaiKrishi25/Object_Tracker

cd macv-object-tracker

2. **Install dependencies:**

pip install -r requirements.txt

Usage

Web Dashboard

streamlit run app.py

- Upload your video through the web interface
- Adjust parameters as needed
- View real-time processing and results
- Download processed video and statistics

Standalone Application

python main.py

- Process videos directly
- Generate HTML reports
- View real-time tracking
- Save processed videos

Configuration

- **Model Selection:**
  - Available sizes: nano, small
  - Default: small (yolov8s.pt)
- **Tracking Parameters:**
  - Confidence threshold: 0.0 - 1.0 (default: 0.5)
  - IOU threshold: 0.4
  - Trail length: 30 frames

Performance Considerations

Processing speed depends on:

- Video resolution
- System hardware
- Selected model size
- Number of objects in frame

Average performance:

- CPU: 6-8 FPS
- GPU: 20-30 FPS (when available)

Local Processing Benefits:

- No data upload required
- Faster processing
- Privacy maintained
- No internet dependency

Future Improvements

1. **Performance Enhancements:**

   - GPU acceleration support
   - Multi-threading for frame processing
   - Batch processing optimization

2. **Feature Additions:**

   - Object classification statistics
   - Motion path analysis
   - Zone-based tracking
   - Custom model support

3. **UI Improvements:**

   - Real-time parameter adjustment
   - Advanced visualization options
   - Export capabilities for analysis data
   - Enhanced HTML report customization