Vehicle Tracking Using Background Subtraction and RNN - Detailed Report

Table of Contents

---

1. Introduction

Object tracking is a critical task in computer vision, with numerous applications such as autonomous driving, surveillance systems, and sports analysis. This project aims to demonstrate a robust object tracking system using a combination of background subtraction for object detection and a Recurrent Neural Network (RNN), specifically an LSTM (Long Short-Term Memory) model, for predicting object trajectories over time.

The main goal is to detect vehicles in a video, track their movement, and predict their future positions even when they are partially or completely occluded.

---

2. Project Overview

This project integrates the strengths of traditional computer vision techniques (background subtraction) with machine learning (LSTM) to perform object tracking. The key components include:

- Background Subtraction: Used to detect moving objects in the video.
- LSTM-based RNN Model: Trained to predict future object positions using historical data.

- Object Tracking Data Preparation: Extracts positional information from detected objects to feed the RNN model.

---

3. System Architecture

3.1. Background Subtraction

The first step involves detecting moving objects using OpenCV's BackgroundSubtractorMOG2, which performs background subtraction to differentiate between the background and moving foreground objects. The detected objects are then filtered based on their size and contours to ensure that only significant objects (e.g., vehicles) are tracked.

3.2. RNN-based Object Tracking

After detecting the objects in each frame, their positions are recorded as sequences of coordinates (x, y) and dimensions (width, height). This data is then used to train an LSTM model, which is a type of RNN particularly effective at learning temporal sequences.

The trained LSTM model is used to predict the next positions of the objects, allowing for continuous tracking even in cases of partial occlusion.

---

4. Implementation Details

4.1. Background Subtraction using OpenCV

- Library: OpenCV
- Algorithm: BackgroundSubtractorMOG2
- Steps:
    1. Read the video file frame by frame.
    2. Apply background subtraction using `cv2.createBackgroundSubtractorMOG2()`.
    3. Filter out shadows by applying a binary threshold.
    4. Use `cv2.findContours()` to detect object contours.
    5. Extract bounding boxes for each detected object and filter based on size.

4.2. Object Tracking with LSTM-based RNN

- Library: TensorFlow
- Model: LSTM Network
- Steps:
    1. Prepare sequences of past object positions and sizes for training.
    2. Train an LSTM model with input features: (x, y, width, height).
    3. Use the trained model to predict future positions for tracking.

4.3. Data Preparation

- Extract object positions and sizes for every frame.
- Group the extracted data into sequences for model training.

- Split the dataset into training and testing sets.

4.4. Tracking Predictions

- After training the model, use it to predict object movements in real-time.
- Draw bounding boxes on frames based on predicted coordinates.

---

5. Workflow

The complete workflow of the project is as follows:

1. Background Subtraction:
    - Process the input video to detect moving objects using background subtraction.
    - Extract contours and filter out small, insignificant objects.
2. Prepare Data for RNN:
    - Record object positions and sizes over time.
    - Group this data into sequences suitable for RNN training.
3. Train the LSTM Model:
    - Use the recorded sequences to train the LSTM model.
    - Validate the model to ensure accuracy.
4. Object Tracking:
    - Use the trained model to predict the next positions of objects.
    - Draw predicted bounding boxes on the video frames.
5. Output:
    - Save the output video with the tracked objects highlighted.

---

6. Experimental Results

6.1. Tracking Visualizations

The output video shows bounding boxes around detected vehicles, with predictions from the RNN model. The tracking remains consistent, even during partial occlusions, showcasing the model's ability to predict and track vehicle movement.

6.2. Performance Metrics

- Accuracy: The model demonstrates high accuracy in tracking objects across frames.
- Robustness: Effective in handling partial occlusions.
- Speed: Suitable for real-time tracking in moderate video resolutions.

---

7. Potential Improvements

While the current approach performs well in controlled environments, several enhancements can be considered:

- Handling Occlusions: Improving the model to handle full occlusions more effectively using techniques like Kalman filters.
- Advanced Detection Models: Integrating state-of-the-art object detection models (e.g., YOLO, SSD) for better initial bounding box predictions.
- Model Optimization: Fine-tuning the LSTM model to handle higher frame rates for real-time applications.
- Multiple Object Interactions: Addressing scenarios where multiple objects interact closely.

---

8. Conclusion

This project demonstrates a powerful combination of background subtraction and RNN-based prediction for vehicle tracking. The integration of traditional computer vision techniques with machine learning allows for robust and continuous tracking, even in challenging scenarios involving occlusions.

The approach is not limited to vehicle tracking and can be extended to other applications like pedestrian detection, sports analytics, or surveillance systems.

---

9. Appendix

9.1. Requirements

- Python 3.x
- OpenCV (`pip install opencv-python`)
- TensorFlow (`pip install tensorflow`)
- NumPy (`pip install numpy`)

9.2. How to Run the Project

1. Ensure all required libraries are installed.
2. Place your input video file (`vehicles.mp4`) in the project directory.
3. Run the script:

   ```
   python vehicle_tracking.py
   ```

4. The output video with tracked vehicles will be saved in the project directory.
5. Review the output video to observe the tracked objects.

This report covers the core concepts, implementation details, and results of the vehicle tracking project. Feel free to modify the code to suit your specific use case or explore further improvements to enhance performance.

4o