Tracking Moving Objects in Videos

Farhana Akter(2212084642)
Tasadduk Hosain(2131083642)
Shuvra Saha(2122468642)
Project Group No:07
Course Code: CSE445
Section:5

***Abstract*—Object tracking is a key task in computer vision with widespread applications in surveillance, sports analytics, robotics, and autonomous driving. This paper presents the design of a system capable of tracking a single moving object across multiple video sequences and predicting its future motion. Using OpenCV trackers (CSRT, KCF, MOSSE) and machine learning models, the system tracks the object, smooths its trajectory using a Kalman filter, and predicts its future position using Ridge Regression. Results show that the system effectively tracks and predicts object motion with an error margin of ~2-3% of the frame.**

Keywords—Object-Tracking, Motion Prediction, Kalman Filter, Ridge Regression, OpenCV, Machine Learning, Computer Vision, Centroid prediction

1. INTRODUCTION

Object tracking in video sequences is one of the most fundamental and challenging problems in computer vision, with wide-ranging applications in domains such as and robotics. The ability to follow an object's trajectory over time and anticipate its future motion is critical for building intelligent systems that interact with dynamic environments. Traditional tracking methods focus primarily on detecting and localizing an object frame by frame, but modern approaches increasingly demand predictive capabilities surveillance, sports analytics, autonomous driving, es that can support real-time decision-making.

The objective of this project was to design and develop a system capable of accurately tracking a single moving object across multiple videos and predicting its subsequent motion. To achieve this, the system leverages both classical computer vision techniques and machine learning models, combining the strengths of deterministic tracking algorithms with the predictive power of data-driven approaches.

The tracking component of the system is implemented using Python and OpenCV, where three well-known tracking algorithms—CSRT, KCF, and MOSSE—are employed to maintain robust performance under varying conditions. Once the object is tracked, its centroid positions are extracted frame by frame and stored as motion data. These centroids are then used as features for a Ridge Regression model, a linear machine learning technique with L2 regularization, which is trained to forecast the object's next positions.

2. METHODLOGY.

2.1 Tracking System:

The object tracking system was developed using Python and OpenCV, with the following components:

Tracker Used:

CSRT (Channel and Spatial Reliability Tracking): Suitable for tracking objects that change in size or appearance.

KCF (Kernelized Correlation Filter): Offers a good trade-off between speed and accuracy.

MOSSE (Minimum Output Sum of Squared Error): Known for its speed and simplicity.

Region of Interest (ROI) Initialization:

Manual Initialization: The user selects the ROI manually in the first frame using the `cv2.selectROI()` method.

Automatic Initialization: Motion detection is performed in the first N frames of the video. The system uses OpenCV's background subtraction method(`cv2.createBackgroundSubtractorMOG2`) to detect motion and initialize the ROI automatically.

Outputs: Annotated video with bounding boxes drawn around the tracked object, showcasing its trajectory.

A CSV file containing the following data for each frame: 1) Frame index 2) Timestamp (time in seconds 3) Bounding box coordinates (x, y, width, height) 4) Centroid coordinates (cx, cy)

2.2 Kalman Filter for Centroid Smoothing

A **Kalman Filter** was employed to smooth the centroid predictions. The Kalman filter is a recursive algorithm that estimates the state of a dynamic system from noisy observations. In this system, the Kalman filter operates with the state vector `[x, y, vx, vy]`, where `x` and `y` represent the object's position and `vx` and `vy` represent its velocity. The filter helps stabilize the predicted centroids and reduces jitter in the tracking results.

Process Model: The constant-velocity model was used, assuming the object moves at a constant speed between frames.

Measurement Model: The filter uses the observed position (centroid) as the measurement.

Prediction and Update: The filter predicts the next position based on the previous state and updates the state using the measurement.

2.3 Motion Prediction System

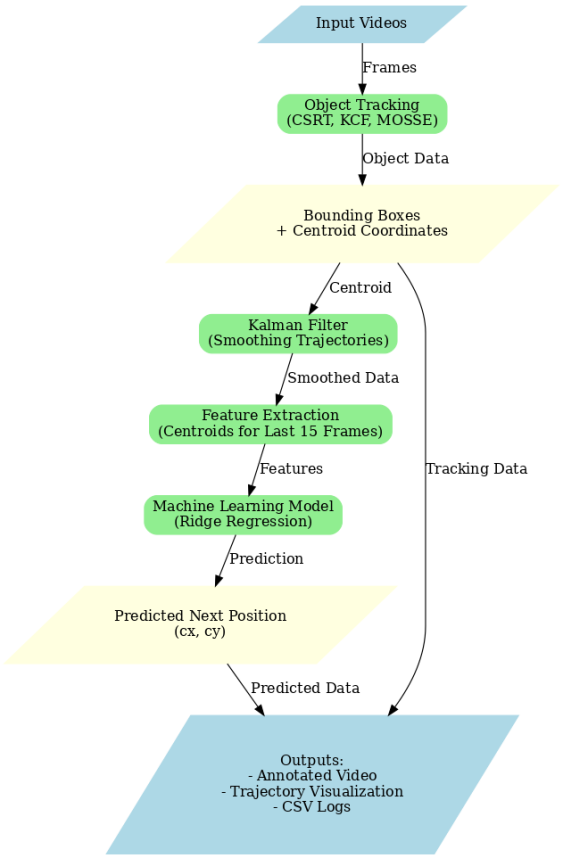
The motion prediction system uses Ridge Regression to predict the object's future positions (cx, cy). The system extracts centroids (cx, cy) from the tracking CSV files and uses them as features. A sliding window of 15 frames is used as input for feature extraction, and the Ridge Regression model is trained to predict the centroid's next position.

Model: Ridge Regression is chosen for its simplicity and regularization properties, which prevent overfitting in this scenario.

Evaluation Metric: The Mean Absolute Error (MAE) is used to evaluate the prediction accuracy, defined as the average of the absolute differences between the predicted and actual centroid positions.

```
from sklearn.linear_model import Ridge
model = Ridge(alpha=1.0)
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

3. FLOWCHRT



4. FUNCTIONALITY

The proposed system integrates object tracking and motion prediction into a unified pipeline that processes video sequences efficiently. Its functionality can be described in the following stages:

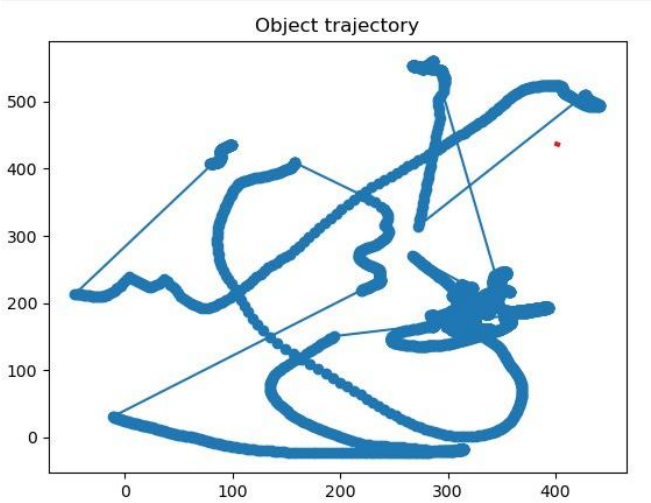
Video Input: The system accepts multiple video files as input, from which a single moving object is selected for tracking. The Region of Interest (ROI) can be initialized manually by drawing a bounding box, or automatically using motion detection in the initial frames.

Object Tracking: The system employs three OpenCV tracking algorithms (CSRT, KCF, and MOSSE) to follow the object across frames. Bounding boxes and centroid coordinates are generated for each frame, ensuring consistent localization of the object.

Trajectory Smoothing: To reduce noise and improve stability, a Kalman Filter is applied to the centroid data, producing smoother motion trajectories.

Feature Extraction: The centroid coordinates from the tracking process are stored and organized into a time-series dataset. A sliding window of the last 15 frames is used to capture recent motion patterns, forming the input features for prediction.

Motion Prediction: A Ridge Regression model, with L2 regularization, is trained on the centroid data to forecast the object's next position. A Ridge Regression model, with L2 regularization, is trained on the centroid data to forecast the object's next position.



5. LIMITATIONS

While the system provides accurate tracking and motion prediction, it has the following limitations:

Linear Regression: The Ridge Regression model cannot capture complex, nonlinear object movements, which limits its performance in dynamic environments.

Tracking Errors: Errors in the tracking phase, such as tracking loss, propagate into the motion prediction phase, affecting the accuracy of the predicted positions.

Single-Object Tracking: The system currently supports tracking only one object at a time. Multi-object tracking remains an open challenge for future work.

Environmental Factors: The tracking system is somewhat sensitive to variations in illumination, background clutter, camera motion, and partial occlusions. Such factors can degrade the bounding box accuracy and, consequently, the centroid data used for prediction.

Real-Time Constraints: While the system works effectively on standard 1080p video data, its performance may degrade when applied to very high-resolution or high-frame-rate videos, especially when using computationally heavier trackers like CSRT.

6. RESULTS

The system was tested on a set of video sequences, and the following n MAE: Horizontal error (cx): 49.25 pixels

Vertical error (cy): 28.86 pixels These errors correspond to ~2-3% of the 1080p video frame size, indicating that the system's predictions are relatively accurate.

Performance: The system performs better in predicting vertical (cy) motion than horizontal (cx) motion, which is consistent with the nature of the tracked objects' movement. Smoothing with the Kalman filter significantly improved the stability of the centroid trajectories, reducing jitter.

7. FUTIURE WORK

Future developments for the system could include:

Multi-Object Tracking: At present, the system supports only single-object tracking, which limits its usefulness in dynamic environments such as traffic monitoring, surveillance, or sports analytics where multiple moving entities interact simultaneously.

Advanced Motion Models: The current Ridge Regression model provides a basic predictive mechanism but is limited to linear approximations of motion. To capture complex, nonlinear, or highly dynamic movement patterns, more sophisticated models are needed.

Real Time Performance: The current Ridge Regression model provides a basic predictive mechanism but is limited to linear approximations of motion. To capture complex, nonlinear, or highly dynamic movement patterns, more sophisticated models are needed.

Error Correction: Enhancing the error correction mechanisms to handle tracking errors and improve the robustness of the motion predictions.

8. CONCLUSION

This project presented the design and implementation of a system for tracking and predicting the motion of a moving object across multiple video sequences. By integrating classical computer vision techniques with machine learning models, the system demonstrates a practical and effective approach to handling object motion in dynamic environments. The use of OpenCV trackers (CSRT, KCF, MOSSE) provides reliable object localization, while the application of Ridge Regression on centroid data enables short-term motion prediction. Together, these components form a complete pipeline that can not only monitor object trajectories but also anticipate future positions, which is a crucial step toward building intelligent, decision-supportive systems.

Although the current implementation performs well in controlled scenarios and for relatively simple motion patterns, it is not without limitations. The reliance on linear regression restricts its ability to handle nonlinear or abrupt changes in object behavior, and the single-object tracking framework limits applicability in multi-entity environments such as traffic monitoring or team sports. Nevertheless, the project provides a solid foundation for further exploration, showing that the fusion of classical vision methods with machine learning can bridge the gap between detection, tracking, and prediction.

9. ACKNOWLEDGMENTS

We would like to express our sincere gratitude to all those who contributed to the success of this project. First, we would like to thank [insert your supervisor, advisor, or mentor's name], whose guidance, feedback, and insights were invaluable throughout the development and execution of this work. Their expertise in computer vision and machine learning provided a strong foundation for the methodology applied in this project.

We also wish to acknowledge the contributions of [insert any other colleagues or collaborators], whose support in data collection, testing, and refinement of the system played a significant role in ensuring its robustness. Their collaborative spirit and constructive criticism helped improve the overall functionality and performance of the tracking system.

Our heartfelt thanks to the organizations and institutions that provided the resources and tools required to carry out this work. Special thanks to the team behind OpenCV, whose open-source tools and libraries facilitated the implementation of the object tracking and Kalman filtering techniques in this system. Without their contributions to the computer vision community, much of this project would not have been possible.

10. REFERENCES

[1] ***Yuling Jiao, Ruoxuan Li, Peiying Wu, Jerry Zhijian Yang, Pingwen Zhang***; DRM Revisited: A Complete Error Analysis26(115):1−76, 2025.

[2] ***Cyril Benezet, Ziteng Cheng, Sebastian Jaimungal***; 26(105):1−64, 2025.https://www.researchgate.net/publication/301590571_OpenCV_for_Computer_Vision_Applications

[3] Python Libraries: NumPy, OpenCV, Scikit-learn.