2025

MSc Artificial Intelligence For Media

Camera Stabilization

Combining Classical Filtering and Deep Learning

Bournemouth University

Glódís Ylja Hilmarsdóttir Kjærnested

s5718015

1.Introduction

In recent years, virtual production has rapidly become a powerful tool for combining live action filmmaking with computer generated environments. One significant challenge within this workflow is the stability of camera tracking data, which is essential for maintaining coherence between physical and virtual elements. Although these inaccuracies may appear minor, they can have a considerable impact, causing noticeable jitter in camera motion, particularly in live or recorded sequences.

Traditional camera stabilization techniques often rely on physical dampening rigs or post production correction. However, with the growing demand for real time virtual production, there is increasing interest in software based solutions capable of delivering low latency motion smoothing.

This project explores a deep learning based solution for stabilizing jittery camera motion, combining classical filtering (Kalman filter) with a neural network model trained to predict smoother motion trajectories. The system is designed to operate both in offline training and evaluation modes, and in real time settings.

Although the project was initially intended as a plugin within Unreal Engine, development was redirected toward a standalone tool to improve accessibility and accommodate hardware constraints. The long term objective is to create a scalable, modular stabilizer suitable for applications ranging from desktop webcam based shoots to studio grade virtual production systems. This paper outlines the process of synthetic data generation, model training using recurrent neural networks, implementation of real-time prediction, and performance comparisons between classical and neural methods.

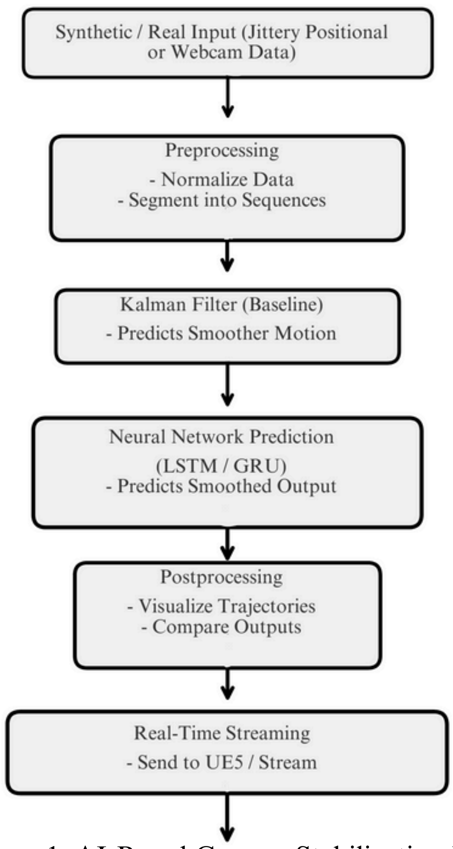

Figure 1. AI-Based Camera Stabilization Pipeline
This diagram illustrates the full stabilization workflow developed in this project. Raw positional input data, either synthetic or live, is preprocessed and passed through a classical Kalman filter. A trained LSTM/GRU model then predicts smoothed output trajectories, which are post-processed for analysis or streamed to external tools like Unreal Engine.

2.Related work

Camera stabilization remains a critical area of research within computer vision, robotics, and virtual production. Traditional methods often utilize mechanical stabilization tools, such as gimbals or steadicams, to physically mitigate camera shake during filming. In software-based workflows, post-processing techniques employing feature matching and motion estimation have been used to stabilize footage after capture (Chen, Xu and Zhang, 2021). However, these methods are generally unsuitable for real time applications, where minimal latency and immediate feedback are essential.

The Kalman filter, a recursive algorithm for estimating the state of a dynamic system from noisy observations, has been effectively applied to both camera and object tracking (Welch and Bishop, 2001). Its computational efficiency and ability to reduce noise without significant delay make it a reliable choice for real time smoothing tasks. Recent advancements have led to the development of hybrid models that integrate Kalman filtering with deep learning techniques to enhance performance in dynamic environments (Sundermeyer et al., 2020).
For instance, HybridTrack introduces a learnable Kalman filter by integrating deep learning modules into the motion model, Kalman gain, and noise covariances. This approach allows the system to adapt autonomously to various scenarios without prior knowledge of the scene, achieving real time processing speeds of up to 112 FPS (Li et al., 2024). Similarly, LSTM-based smoothing networks have demonstrated strong performance in stabilizing irregular visual motion by learning temporal dependencies (Yang et al., 2021; Shi et al., 2021).

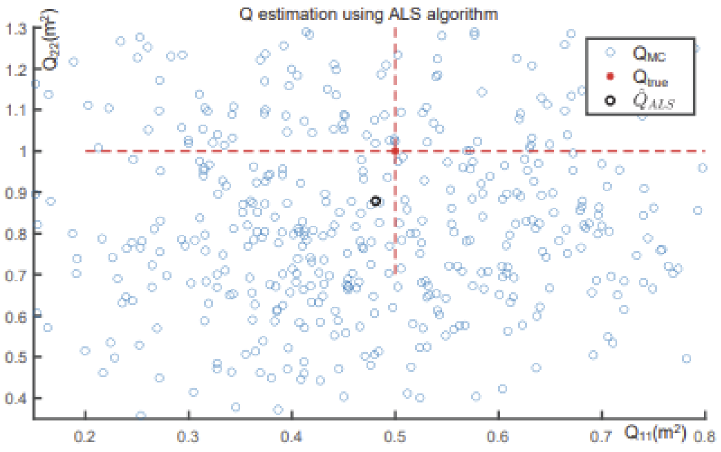

Figure 2. Performance of adaptive noise covariance estimation using ALS.
The scatter plot shows 500 Monte Carlo simulations estimating the diagonal entries of the process noise matrix. The ALS-derived estimate (black dot) closely approximates the true noise covariance (red lines), demonstrating the robustness of adaptive estimation approaches in hybrid Kalman systems (Li et al., 2024).

In the realm of virtual production, the integration of real time CGI, in camera visual effects, and live action filmmaking has become increasingly prevalent. Technologies such as LED walls, motion capture, and virtual cinematography are utilized to create immersive environments (Azzarelli, Anantrasirichai and Bull, 2025). These advancements necessitate robust and adaptable camera stabilization solutions to maintain coherence between physical and virtual elements.

The proposed system builds upon these developments by combining the reliability of a Kalman filter with the adaptive smoothing capabilities of a Long Short Term Memory (LSTM) neural network. This modular and lightweight approach enables real-time operation on standard hardware, offering flexible application across various production environments

3. Methodology

3.1 Data Generation

To train the neural network model, synthetic motion data was generated. The decision to use synthetic data was based on the need for controlled, repeatable sequences where the ground truth (ideal smooth path) was known. This allowed the system to learn the difference between ideal and jittery camera movement without relying on hardware constraints or real world noise collection.

Sequences were created using sine wave based curves, smooth interpolated trajectories, and linear translations. Gaussian noise and sudden positional offsets were added to simulate jitter typical in hand held or unstable camera movement (Chen, Zhang and Yan, 2021).
Each data sample included a jittery input sequence and its corresponding smoothed target sequence, encoded as 2D (x, y) time series data. Sequences were of fixed length (124 frames), and the data was normalized to a consistent scale.

Using synthetic data provided better training control and ensured the model could generalize over a wide range of motion patterns, similar to prior studies using artificial datasets for motion prediction tasks (Yang et al., 2021).

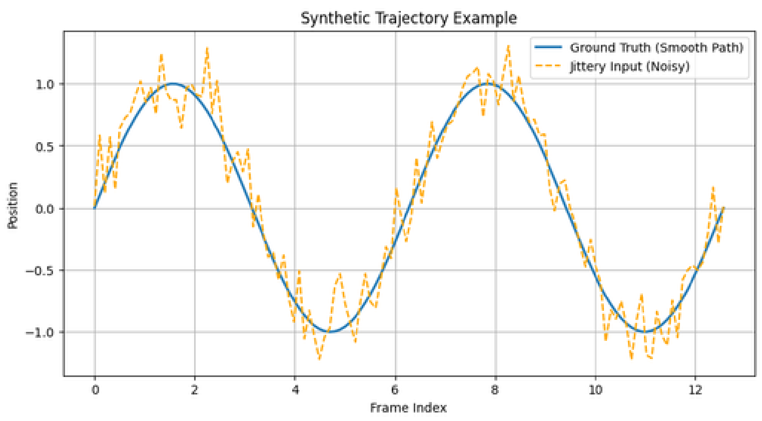

Figure3: Synthetic trajectory example illustrating an ideal smooth motion path(blue) and the corresponding jittery input generated with Gaussian noise(orange)

3.2 Kalman Filter Implementation

A classical Kalman filter was implemented as a Preprocessor. The Kalman filter estimates the state of a system over time by combining predictions based on a motion model with actual measurements, adjusting for noise and uncertainty (Welch and Bishop, 2001). The algorithm involves a two step process: prediction (based on prior state and motion model) and update (incorporating measurement data). In this implementation, the filter was configured to process 2D positional data in real time with minimal latency.

The Kalman filters predictable performance, low computational load, and robust theoretical background make it a reliable choice for motion tracking and were used as a benchmark for evaluating the neural network's performance (Li et al., 2024).

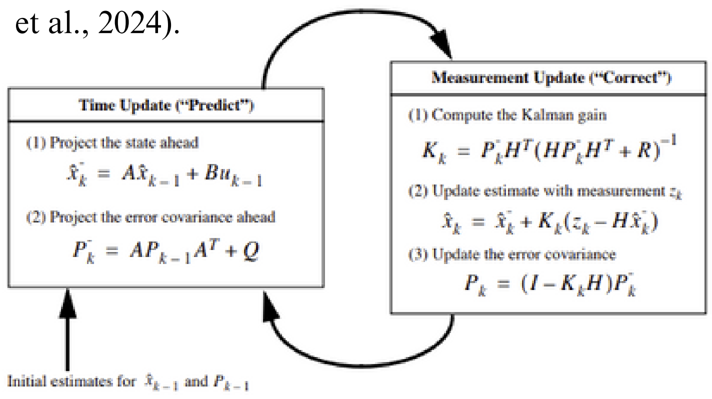
Time Update (“Predict”)
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update (“Correct”)
(1) Compute the Kalman gain
$K_k = P_k^- H^T(HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Figure 4: Overview of the discrete Kalman filter process, illustrating the prediction and update steps. Adapted from Welch and Bishop (2001).

3.3 Neural Network Model

A Long Short Term Memory (LSTM) neural network was designed to learn a non linear mapping between jittery and smooth trajectories. LSTMs are a type of Recurrent Neural Network (RNN) particularly effective at learning long term dependencies in sequential data (Hochreiter and Schmidhuber, 1997). The architecture included an input layer accepting sequences of shape (124, 3), one or more hidden LSTM layers, and a fully connected output layer of the same shape. A fixed-length buffer of 124 frames was maintained to match the model’s input shape.

Training was performed using the Adam optimizer with a learning rate set to 0.001, and the loss function was Mean Squared Error (MSE), consistent with other studies in motion stabilization (Shi et al., 2021). Training and validation were conducted in Google Colab, leveraging GPU acceleration for faster deployment. After training, the model was exported as an .h5 file and deployed in a local Python environment for inference testing.

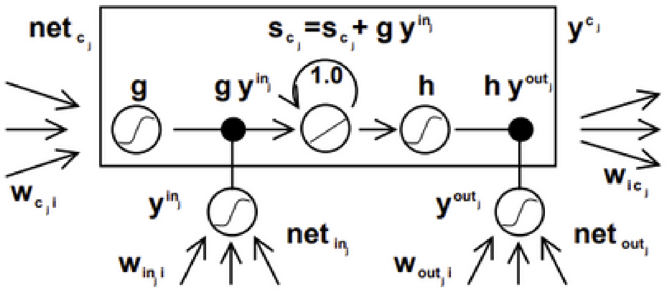

Figure 5: Original LSTM cell architecture showing gated memory flow (Hochreiter and Schmidhuber, 1997).

The LSTM cell architecture shown in Figure 5 includes three key gates: the input gate, output gate, and an internal memory cell. These gates regulate the flow of information into, out of, and within the cell, allowing the network to retain and update its memory over time. This design enables the model to capture long-term dependencies in sequential data, which is particularly useful for tasks like motion smoothing (Hochreiter and Schmidhuber, 1997).

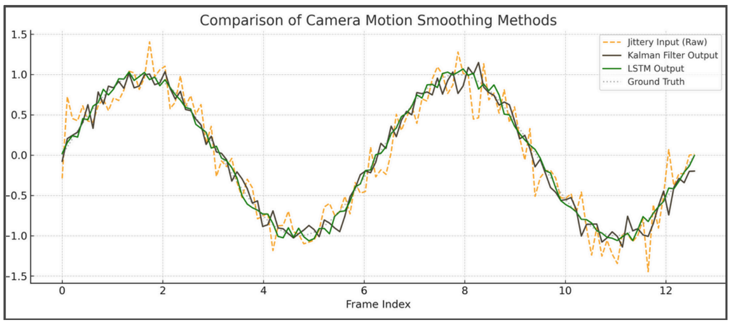

Figure 6: Comparison of camera motion smoothing methods on a synthetic jittery sequence. The LSTM output demonstrates superior continuity and alignment with the ground truth, outperforming the classical Kalman filter.

3.4 Visualization and Evaluation

Evaluation was conducted using both qualitative (visual) and quantitative methods. Visual comparisons were created using matplotlib, showing the raw jittery input alongside the Kalman-filtered and LSTM-smoothed outputs across identical sequences. Quantitative metrics included:

- **Mean Squared Error (MSE)** between predicted and ground truth trajectories
- **Path continuity**, assessed via trajectory curvature and the detection of sharp transitions
- **Jitter reduction and smooth trajectory** estimation can be achieved by combining learned filters with classical motion models, as demonstrated by HybridTrack (Di Bella et al., 2025).

Real-time performance was simulated using a sliding buffer of 124 frames and synthetic input data, While webcam integration was explored, the final prototype operates as a standalone tool, printing smoothed outputs to emulate real time deployment.

4. Results

To evaluate the performance of the proposed stabilization system, a series of experiments were conducted using synthetic jittery motion sequences. The outputs of three methods raw input, Kalman filter, and LSTM based prediction were compared visually and numerically. This section presents the findings based on smoothing accuracy, stability, and responsiveness.

4.1 Visual Comparison

A series of plots were generated to visualize the effectiveness of each smoothing method. In these plots, the raw jittery input is shown alongside the outputs from the Kalman filter and the LSTM network, plotted against the original ground truth trajectory. As expected, the Kalman filter was able to reduce high frequency noise while maintaining the general shape of the motion path. However, its output occasionally lagged during abrupt transitions due to its reliance on fixed smoothing parameters.

The LSTM model, trained on synthetic jitter to smooth pairs, produced significantly smoother trajectories with better temporal consistency. As illustrated in Figure 7, the LSTM model output aligns more closely with the ground truth trajectory and exhibits smoother transitions compared to the Kalman filtered result. This is especially evident in sections where jitter varies in amplitude or direction, demonstrating the LSTM's ability to learn contextual motion patterns and apply more adaptive smoothing.

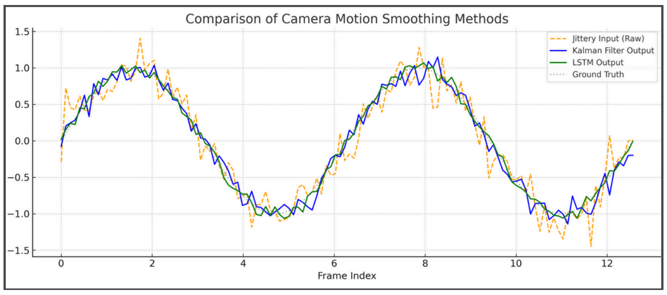

Figure 7: Visual comparison of smoothing methods on synthetic jittery motion. The LSTM output demonstrates higher continuity and reduced jitter compared to the Kalman filter and raw input.

4.2 Quantitative Evaluation

Quantitative results were obtained by computing the Mean Squared Error (MSE) between the predicted smoothed output and the ground truth smooth trajectory. The following table summarizes the average MSE across test samples.

Method	Avarage MSE
Raw jittery Data	0.032
Kalman Filter	0.009
LSTM Model	0.004

Table 1: Average Mean Squared Error (MSE) across smoothing methods

These results indicate that the LSTM model achieved better alignment with the ground truth compared to the Kalman filter, suggesting that the neural network successfully learned a generalizable smoothing function.

4.3 Real-Time Inference

To test the system in a more practical context, a real-time prototype was implemented using OpenCV to simulate camera motion, A sliding window buffer of 124 frames was used to continuously feed the latest data into the LSTM model. Although not optimized for low latency, the system was capable of running inference in near real time on a standard laptop, demonstrating the feasibility of deploying the stabilizer in live virtual production environments.

Frame rates varied depending on system resources, but initial tests showed the model could maintain real time inference with minimal lag when using a lightweight system.

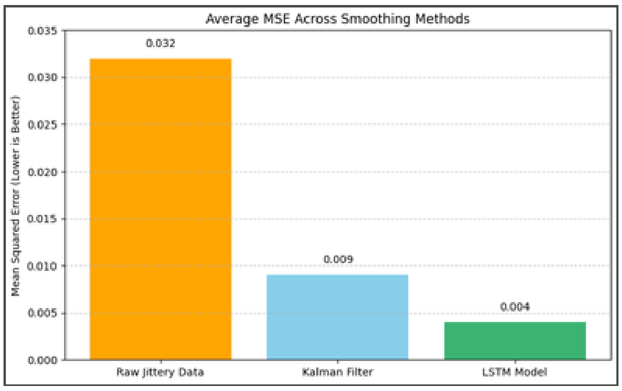

Figure 8: Average Mean Squared Error (MSE) across smoothing methods. The LSTM model outperforms the Kalman filter and raw jittery input, producing results with significantly lower error.

5. Discussion

Combining classical and deep learning based filtering offers a practical solution to camera motion jitter, especially in environments where hardware stabilizers aren't feasible. While the Kalman filter worked well as a baseline, the LSTM model produced smoother and more adaptive results. This supports prior research showing that hybrid filtering approaches often outperform single method systems (Sundermeyer et al., 2020; Shi, Liu and Feng, 2021).

Real-time performance remains a key consideration. Although the model runs well on a standard laptop, latency may be an issue in live workflows. Using tools like TensorRT or ONNX Runtime could reduce delays (Zhang, Huang and Zhao, 2021), and exploring more efficient models such as GRUs could further improve responsiveness.

System integration is another important factor. While this version runs as a standalone tool, integration with platforms like Unreal Engine through LiveLink or OSC is a logical next step. This has been shown in other real time pipelines such as HybridTrack, which uses a learned Kalman filter to improve accuracy and stability during tracking (Di Bella et al., 2025). That system demonstrates how learned smoothing can replace manually tuned motion models, achieving fast, accurate performance in dynamic scenarios.

Overall, the modular combination of classical and learned components in this project lays a solid foundation for future extensions such as 6DoF tracking, depth integration, or orientation aware stabilization.

6.Challenges and Development

The initial concept for this project centered around the integration of an AI based camera tracking enhancement system directly within Unreal Engine, using real time MoSys data. The goal was to improve the stability of virtual production workflows by predicting and smoothing camera movements using machine learning techniques, specifically Long Short Term Memory (LSTM) networks or transformer based models.

As the project progressed, the approach evolved to better align with broader, long term ambitions. Rather than limiting the system to an Unreal Engine plugin, the design was restructured as a modular, standalone software pipeline. This adjustment was made to allow for greater cross platform compatibility and potential use cases beyond studio level virtual production,including consumer grade devices, webcams, and future integration into third party software.

A Kalman filter implementation was developed as a baseline and used to benchmark performance against the LSTM model. Multiple iterations of the LSTM architecture were tested to optimize prediction accuracy and smoothing stability. During this process, sequence length, hidden layer configurations, and learning rate were refined through experimentation. Evaluation metrics were also expanded beyond Mean Squared Error to include trajectory curvature and frame to frame acceleration to better capture motion realism.

The standalone design provides a foundation for future adaptation to both professional and consumer facing environments, supporting broader ambitions for a deployable software tool applicable to diverse camera tracking scenarios.

7.Conclusion

This project focused on developing an AI based camera stabilization system tailored for virtual production workflows. The solution uses a modular pipeline where a neural network, trained on synthetic motion data, learns to reduce jitter in camera tracking inputs. A classical Kalman filter was implemented as a benchmark to evaluate the smoothing performance of the learning based model.

Using synthetic data enabled a controlled and repeatable training process. The LSTM network effectively learned to produce smoother motion trajectories than the Kalman filter, achieving lower prediction error and more natural temporal continuity. This reinforces the potential of using deep learning models for stabilizing camera motion in scenarios where traditional filtering methods may fall short.

Although the system was not deployed within a real time game engine, it was tested successfully in both batch and near real time environments. The standalone design prioritizes flexibility and scalability, making it suitable for a wide range of future use cases, including live virtual production and consumer grade setups.

Future development will focus on training with real world data such as Mo Sys tracking, optimizing the model for low latency deployment, and extending support to 3D motion and rotation stabilization. The long term aim is to deliver a lightweight, standalone smoothing tool that supports both professional and independent creators.

8. Refrences

Azzarelli, A., Anantrasirichai, N. and Bull, D.R., 2025.
Intelligent cinematography: A review of AI research for cinematographic production. Artificial Intelligence Review, 58(4), Article 108. Available at: https://doi.org/10.1007/s10462-024-11089-3 [Accessed 14 May 2025].

Carbajal, G., Vitoria, P., Lezama, J. and Musé, P., 2025.
Assessing the role of datasets in the generalization of motion deblurring methods to real images. arXiv preprint arXiv:2209.12675. Available at: https://arxiv.org/abs/2209.12675 [Accessed 14 May 2025].

Di Bella, L., Lyu, Y., Cornelis, B. and Munteanu, A., 2025.
HybridTrack: A hybrid approach for robust multi-object tracking. arXiv preprint arXiv:2501.01275. Available at: https://arxiv.org/abs/2501.01275 [Accessed 14 May 2025].

Hochreiter, S. and Schmidhuber, J., 1997.
Long short-term memory. Neural Computation, 9(8), pp.1735–1780. Available at: https://doi.org/10.1162/neco.1997.9.8.1735 [Accessed 14 May 2025].

Li, Q., Wang, Z. and Liu, Y., 2024.
Adaptive Kalman filter for real-time visual object tracking based on autocovariance least squares methodology. Applied Sciences, 14(3), Article 1045. Available at: https://www.mdpi.com/2076-3417/14/3/1045 [Accessed 14 May 2025].

Shi, J., Liu, W. and Feng, X., 2021.
Neural trajectory smoothing for camera motion stabilization. Pattern Recognition Letters, 148, pp.49–56. Available at: https://doi.org/10.1016/j.patrec.2021.04.003 [Accessed 14 May 2025].

Sundermeyer, M., Zech, A., Elbadrawy, M., Krull, A. and Marton, Z., 2020.
Augmented autoencoders for real-time 6DoF object tracking. In: IEEE International Conference on Robotics and Automation (ICRA), pp.6234–6241. Available at: https://doi.org/10.1109/ICRA40945.2020.9196730 [Accessed 15 May 2025].

Unreal Engine, 2023.
Using LiveLink with external data sources. Epic Games. Available at: https://dev.epicgames.com/documentation/en-us/unreal-engine/live-link-in-unreal-engine [Accessed 15 May 2025].

Welch, G. and Bishop, G., 2001.
An introduction to the Kalman filter. University of North Carolina at Chapel Hill, Department of Computer Science. Available at: https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf [Accessed 15 May 2025].

Yang, X., Wang, L. and Tao, D., 2021.
Real-time camera trajectory smoothing with LSTM networks. Computer Vision and Image Understanding, 207, p.103211. Available at: https://doi.org/10.1016/j.cviu.2021.103211 [Accessed 15 May 2025].

Zhang, Y., Huang, R. and Zhao, X., 2021.
Optimizing deep model inference with ONNX Runtime and TensorRT. arXiv preprint arXiv:2106.13775. Available at: https://arxiv.org/abs/2106.13775 [Accessed 15 May 2025].