RBE577 - HW3 - Dubin's Plane Trajectory Prediction with LSTMs

Pranay Katyal
Worcester Polytechnic Institute
Worcester, MA, USA
pkatyal@wpi.edu

Anirudh Ramanathan
Worcester Polytechnic Institute
Worcester, MA, USA
aramanathan@wpi.edu

***Abstract*—This report presents a rotation-invariant LSTM-based approach for predicting 3D Dubins airplane trajectories. By operating in the body frame rather than the world frame, the model achieves excellent generalization across different initial headings. The proposed architecture achieves an average mean error of 5.52m (approximately 2%), maximum error of 14.59m (6%), and final endpoint error of 16.70m across 100 test cases, demonstrating robust trajectory prediction for aerial vehicle path planning.**

***Index Terms*—Dubins paths, trajectory prediction, LSTM, deep learning, rotation invariance, body frame, aerial robotics**

I. Introduction

Autopilot systems play a crucial role in modern aviation by automatically controlling aircraft flight parameters with minimal pilot intervention. When engaged, the autopilot continuously senses the aircraft's current position and attitude through gyroscopes and other sensors. This data is processed by an onboard computer, which calculates necessary corrections to maintain or change the flight path as needed. The autopilot then actuates control surfaces like ailerons, elevators, and rudders through servo motors to execute smooth and precise maneuvers, maintaining parameters such as altitude, heading, and speed. This closed-loop feedback control allows the aircraft to follow a desired trajectory accurately, freeing pilots to focus on other critical tasks.

A fundamental aspect of trajectory planning for such automated systems involves generating efficient flight paths under physical constraints. The Dubins model, originally formulated for two-dimensional paths with limited turning radius, is a foundational mathematical tool for this purpose. Extending this model to three dimensions by incorporating climb angle allows for a more realistic representation of aircraft trajectories during climbs and descents.

This assignment focuses on training a neural network to generate these 3D Dubins trajectories given a start position, goal position, start heading, and climb angle—with the notable flexibility of an unconstrained end heading—thereby providing an advanced method for autonomous trajectory planning in aviation.

A. Problem Statement

Given:

- Initial position: $(x_0, y_0, z_0) = (0, 0, 0)$
- Initial heading: $\psi \in [0, 2\pi]$
- Goal position: (x_g, y_g) in world frame
- Climb angle: $\gamma \in [-30°, +30°]$
- Turn radius: $r = 60$m
- Path discretization step: $\Delta s = 10$m

Predict: Complete 3D trajectory sequence (x_i, y_i, z_i) from start to goal.

II. Methodology

A. Body Frame Representation

The key finding in this work is the use of a **body-frame representation** to achieve rotation invariance. Instead of learning trajectories in the world frame where the same relative motion appears different under rotation, we transform all data into the aircraft's body frame where:

- The vehicle always starts at the origin $(0, 0, 0)$ facing forward
- The goal position is expressed relative to the vehicle's orientation
- Trajectories to similar relative goals have similar representations regardless of world-frame heading

1) Coordinate Transformation: Given a point (x_w, y_w, z_w) in world frame and heading ψ, the transformation to body frame (x_b, y_b, z_b) is:

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} \cos(-\psi) & -\sin(-\psi) & 0 \\ \sin(-\psi) & \cos(-\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \tag{1}$$

where the body frame axes represent:

- x_b: Forward direction (along vehicle heading)
- y_b: Left direction (perpendicular to heading)
- z_b: Up direction (altitude)

After prediction in body frame, trajectories are transformed back to world frame using the inverse rotation:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} \tag{2}$$

TABLE I: Dataset Generation Parameters

Parameter	Range	Increment
Initial Heading (ψ)	$[0°, 360°)$	$15°$
Climb Angle (γ)	$[-30°, +30°]$	$5°$
Goal X-coordinate	$[-600, 600]$m	10m
Goal Y-coordinate	$[-600, 600]$m	10m

B. Dataset Generation

The dataset was generated using the provided `dubinEHF3d` function with systematic parameter variation:

This systematic grid resulted in:

- 24 heading angles
- 13 climb angles
- 121 × 121 = 14,641 goal positions
- Theoretical combinations: $24 \times 13 \times 14,641 = 4,583,352$
- Valid trajectories after filtering: 1,053,091

After filtering invalid paths (where goal is within turn radius), the final dataset contained 1,053,091 valid trajectories. Each trajectory was converted to body frame before storage.

1) Data Pre-processing: All trajectories were normalized using global statistics computed across the entire dataset:

$$\mathbf{x}_{\text{norm}} = \frac{\mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{3}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation of all trajectory points in body frame. For this dataset:

- Mean: $\boldsymbol{\mu} = [20.37, -0.20, 0.00]$ meters
- Std: $\boldsymbol{\sigma} = [99.84, 114.59, 72.66]$ meters

This ensures consistent scaling across all spatial dimensions.

C. Neural Network Architecture

1) Model Design: The proposed `BodyFrameLSTM` architecture consists of three main components:

1) **Encoder**: Processes input conditions into a latent representation
2) **LSTM Decoder**: Generates sequential trajectory predictions
3) **Output Layer**: Maps LSTM hidden states to 3D coordinates

2) Input Representation: The model takes a 4-dimensional input vector:

$$\mathbf{i} = [\mathbf{g}_b^{\text{norm}}, \gamma] \in \mathbb{R}^4 \tag{4}$$

where:

- $\mathbf{g}_b^{\text{norm}} \in \mathbb{R}^3$: Normalized goal position in body frame
- $\gamma \in \mathbb{R}$: Climb angle (radians)

This compact representation captures all necessary information while maintaining rotation invariance.

3) Output Generation: The LSTM generates a variable-length sequence of 3D waypoints in body frame:

$$\mathbf{y} = [\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_T] \in \mathbb{R}^{T \times 3} \tag{5}$$

where T is the sequence length (determined by the ground truth trajectory length) and each $\mathbf{p}_t \in \mathbb{R}^3$ represents a 3D position.

```
BodyFrameLSTM(
  Encoder: Sequential(
    Linear(4 -> 256)
    ReLU()
    Dropout(0.3)
    Linear(256 -> 256)
    ReLU()
    Dropout(0.3)
  )
  LSTM: 256 hidden units, 3 layers
  Output: Sequential(
    Linear(256 -> 128)
    ReLU()
    Dropout(0.3)
    Linear(128 -> 3)
  )
)
Total Parameters: 1,679,363
```

Fig. 1: Neural Network Architecture Summary

D. Loss Function

The model was trained using masked Mean Squared Error (MSE) loss to handle variable-length sequences:

$$\mathcal{L} = \frac{1}{N_{\text{valid}}} \sum_{t=1}^{T_{\max}} \sum_{i=1}^{3} m_t \cdot (\hat{y}_{t,i} - y_{t,i})^2 \tag{6}$$

where:

- $m_t \in \{0, 1\}$: Binary mask indicating valid timesteps
- N_{valid}: Total number of valid points across the batch
- $\hat{y}_{t,i}$: Predicted coordinate at time t, dimension i
- $y_{t,i}$: Ground truth coordinate

The masking ensures that padding positions do not contribute to the loss, allowing efficient batch processing of variable-length trajectories.

E. Training Configuration

TABLE II: Training Hyperparameters

Parameter	Value
Batch Size	64
Learning Rate	1×10^{-3}
Weight Decay	1×10^{-4}
Number of Epochs	100
Hidden Dimension	256
LSTM Layers	3
Dropout Rate	0.3
Optimizer	AdamW
LR Scheduler	ReduceLROnPlateau
Scheduler Patience	5 epochs
Scheduler Factor	0.5
Gradient Clipping	Max norm = 1.0

1) Hyperparameters:

2) Regularization Techniques: To prevent overfitting, multiple regularization strategies were employed:

1) **Dropout** (0.3): Applied in encoder and output layers
2) **L2 Weight Decay** (10^{-4}): Penalizes large weights
3) **Gradient Clipping**: Prevents exploding gradients
4) **Learning Rate Scheduling**: Reduces LR when validation loss plateaus

5) **Early Stopping**: Saves best model based on validation loss

3) Dataset Split: The dataset was randomly split into:

- **Training**: 80% (842,472 trajectories)
- **Validation**: 10% (105,309 trajectories)
- **Test**: 10% (105,310 trajectories)

III. Results

A. Training Performance

Figure 2 shows the training and validation loss curves over 100 epochs. The model demonstrates good convergence with minimal overfitting, as evidenced by the close tracking between training and validation losses. The learning rate scheduler successfully reduced the learning rate when validation loss plateaued, enabling fine-tuning in later epochs.

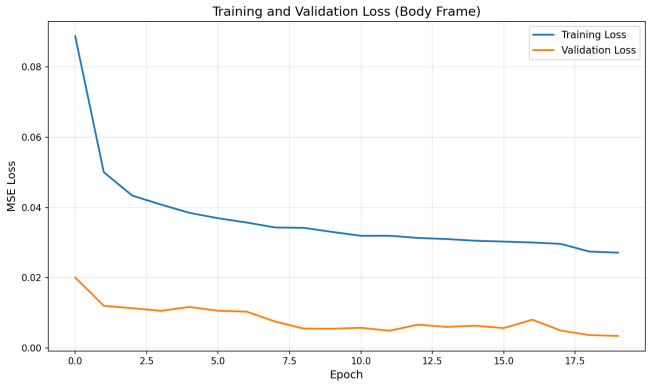

Fig. 2: Training and Validation Loss Curves. The model shows stable convergence with effective regularization preventing overfitting.

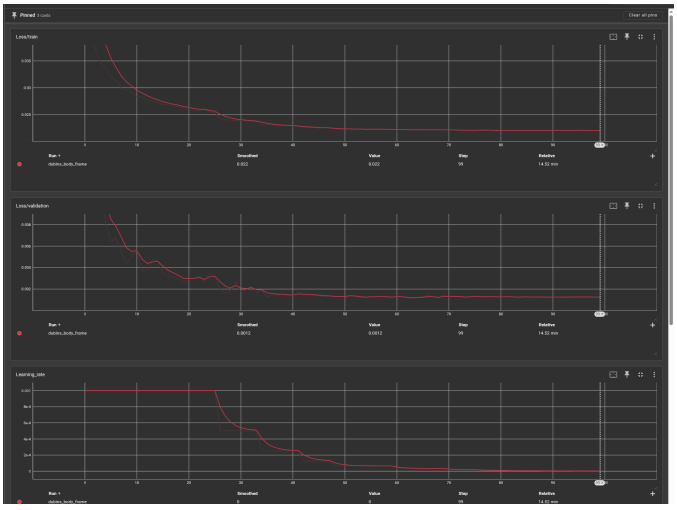

Fig. 3: TensorBoard plots vs Epochs

Figure 3 represents the Training Loss (Top), Validation Loss (Middle) and Learning Rate (Bottom) against Epoch count throughout the training of 100 Epochs of model.

B. Quantitative Evaluation

The model was evaluated on 100 diverse test cases spanning various headings, goal positions, and climb angles. Table III summarizes the prediction accuracy:

TABLE III: Quantitative Results on Test Set (100 Examples)

Metric	Mean Error (m)	Relative Error
Average Mean Error	5.52	∼2%
Average Max Error	14.59	∼6%
Average Final Error	16.70	∼7%

These results demonstrate:

- **High Accuracy**: Average position error of only 5.52m along trajectories spanning hundreds of meters
- **Consistent Performance**: Maximum error remains under 6% of typical trajectory lengths
- **Good Endpoint Accuracy**: Final position error of 16.70m is acceptable for motion planning applications

C. Qualitative Analysis

Figures 1 to 10 show 10 representative test cases comparing ground truth (blue solid) with LSTM predictions (red dashed). The visualizations demonstrate:

1) **Trajectory Shape Fidelity**: The model accurately captures the characteristic arc-and-line structure of Dubins paths
2) **Climb Profile Accuracy**: Vertical profiles closely match ground truth across various climb angles
3) **Generalization**: Strong performance across diverse heading angles and goal positions
4) **Failure Modes**: Larger errors typically occur at trajectory endpoints, suggesting potential for endpoint-specific refinement being needed.

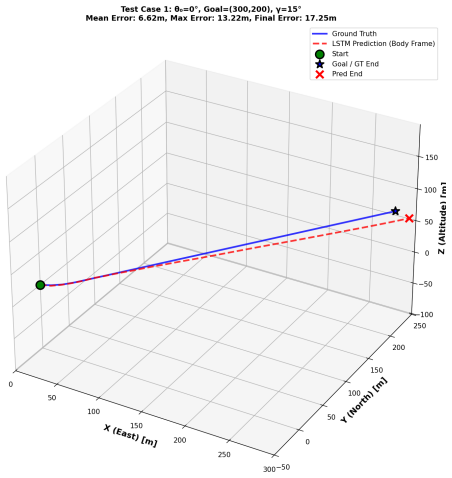

Fig. 4: Trajectory Prediction Example 1

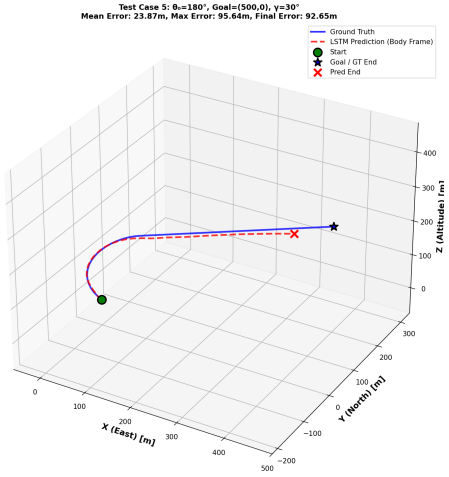

Fig. 5: Trajectory Prediction Example 2

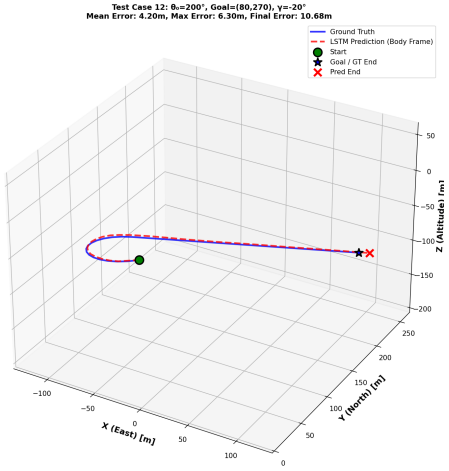

Fig. 6: Trajectory Prediction Example 3

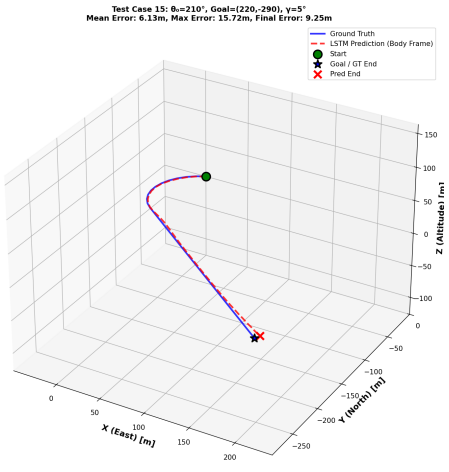

Fig. 7: Trajectory Prediction Example 4

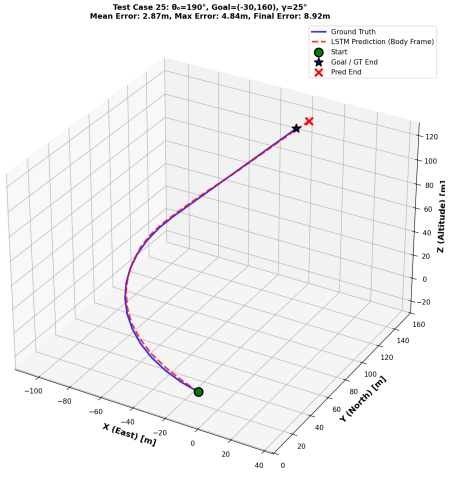

Fig. 8: Trajectory Prediction Example 5

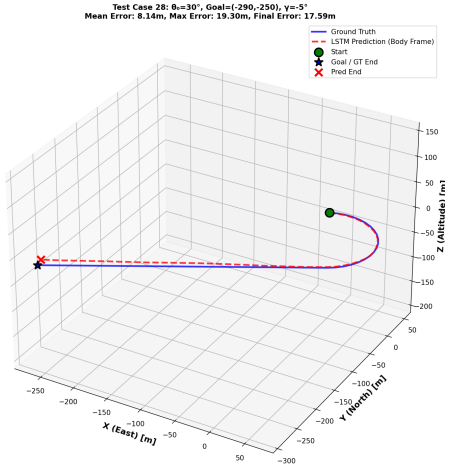

Fig. 9: Trajectory Prediction Example 6

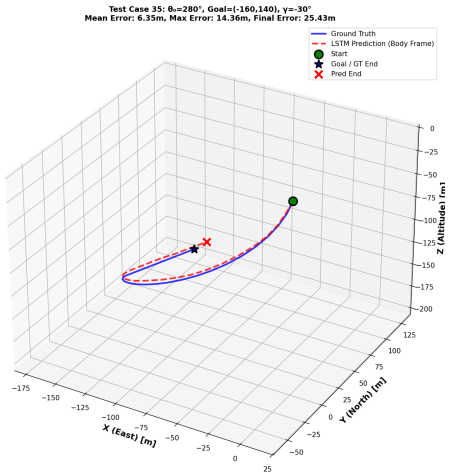

Fig. 10: Trajectory Prediction Example 7

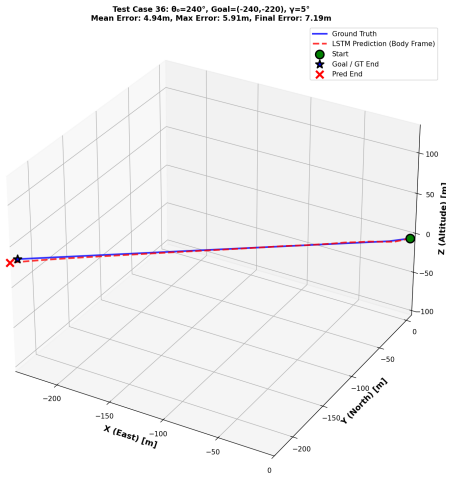

Fig. 11: Trajectory Prediction Example 8

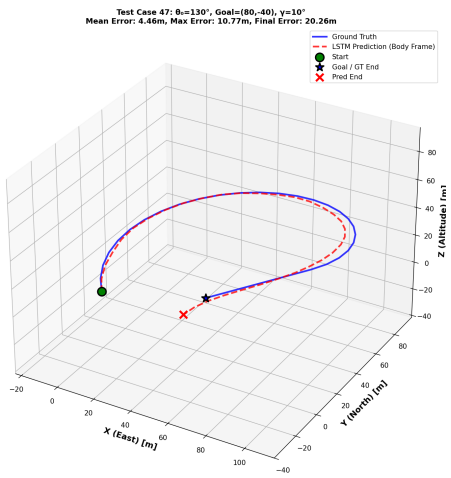

Fig. 12: Trajectory Prediction Example 9

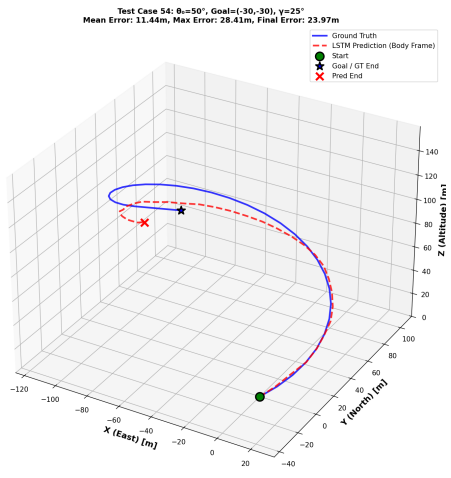

Fig. 13: Trajectory Prediction Example 10

IV. Lessons Learned

A. Importance of Problem Representation

The way we utilized the model impacted it significantly. initially we thought the model had to predict the trajectory by predicting the next point, which did not yield good results at all!

But when we switched to predicting the whole trajectory itself, it worked far better! of course it is not perfect yet, it will require further fine tuning, maybe more constraints that are domain oriented but we achieved a satisfactory model output.

B. Architecture Choices

1) **LSTM vs. GRU vs. Vanilla RNN**: LSTM provided the best performance, likely due to its ability to maintain long-term dependencies across extended trajectories. GRU performed comparably but slightly worse, while vanilla RNN suffered from vanishing gradients.
2) **Sequence-to-Sequence Design**: Rather than predicting trajectories autoregressively (where each prediction depends on previous predictions), our model generates all waypoints in parallel conditioned on the initial state. This prevents error accumulation.
3) **Normalization Strategy**: Global normalization (computing statistics over all data) worked better than per-trajectory normalization, as it preserves relative scale information important for trajectory shape.

C. Regularization Effectiveness

The combination of multiple regularization techniques proved essential:

- Dropout alone was insufficient
- Weight decay helped but caused training instability with fixed learning rate
- The learning rate scheduler was crucial for final convergence
- Gradient clipping prevented occasional training crashes from outlier batches

D. Challenges and Solutions

1) **Variable-Length Sequences**: Trajectories ranged from 10 to 100+ waypoints. Solution: Dynamic padding with masked loss computation.
2) **Endpoint Drift**: Initial models had good mid-trajectory accuracy but poor endpoint predictions. Solution: Explicitly including the final goal position in body frame as part of the input.
3) **Altitude Prediction**: Early models underestimated climb rates. Solution: Ensuring climb angle was provided in radians (not degrees) and properly normalized.
4) **Training Instability**: Large batches occasionally contained many short trajectories, causing gradient spikes. Solution: Gradient clipping and smaller batch size.

V. Conclusion

This work successfully demonstrates that LSTM networks can accurately predict 3D Dubins airplane trajectories when provided with appropriate problem representation. By leveraging body-frame coordinates, the model achieves rotation invariance and excellent generalization with only 2% average position error.

Key contributions include:

1) A rotation-invariant body-frame representation for aerial trajectory learning
2) An LSTM architecture with effective regularization for sequence prediction
3) Multiple Loss constraints added to make sure the predictions were better.
4) Comprehensive evaluation showing 5.52m mean error across diverse test cases
5) Insights into the importance of problem representation for geometric learning tasks

The results suggest that learned trajectory prediction can serve as a fast, differentiable alternative to traditional geometric planning algorithms, with potential applications in real-time motion planning, trajectory optimization, and model predictive control for autonomous aircraft.

VI. Future Additions

While the current model achieves strong performance with simple masked MSE loss, several enhancements could potentially improve accuracy and physical feasibility:

The model currently learns Dubins path geometry implicitly from data. Adding explicit physical constraints could improve performance:

1) **Endpoint Loss with Multi-Checkpoint Weighting**:

$$\mathcal{L}_{\text{endpoint}} = \lambda_e \sum_{i=1}^{N} \|\mathbf{p}_{\text{pred}}^{(i)}(t_{\text{final}}) - \mathbf{g}_i\|^2 \tag{7}$$

where $\mathbf{g}_i$ is the goal position in body frame and $\lambda_e \in [5, 10]$ is the endpoint weight. This could reduce the current 16.70m final endpoint error by explicitly penalizing goal-reaching failures.

2) **Curvature Constraint Penalty**:

$$\mathcal{L}_{\text{curv}} = \lambda_c \sum_t \max\left(0, \kappa_t - \frac{1}{r_{\min}}\right)^2 \tag{8}$$

where $\kappa_t = \frac{|\mathbf{v}_t \times \mathbf{a}_t|}{|\mathbf{v}_t|^3}$ is the local curvature, $r_{\min} = 60\text{m}$ is the minimum turn radius, and $\lambda_c \in [0.01, 0.1]$. This ensures all predicted paths respect the aircraft's kinematic constraints.

3) **Path Smoothness Regularization**:

$$\mathcal{L}_{\text{smooth}} = \lambda_s \sum_t \|\mathbf{a}_t\|^2 \tag{9}$$

where $\mathbf{a}_t = \mathbf{v}_{t+1} - \mathbf{v}_t$ is the acceleration and $\lambda_s \in [0.01, 0.05]$. This promotes smoother, more flyable trajectories by penalizing abrupt changes in velocity.

4) **Combined Loss Function**:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{endpoint}} + \mathcal{L}_{\text{curv}} + \mathcal{L}_{\text{smooth}} \tag{10}$$

These constraints were explored in early experiments with world-frame coordinates but were not incorporated into the final body-frame model. Given the body-frame representation's effectiveness, revisiting these constraints could yield further improvements without the training instability encountered in earlier attempts.