Appendix A. Extra Algorithm Details

A.1. Algorithm: Executing Recurrent Policy

Algorithm 1 Executing Recurrent Policy

Input: recurrent policy parameters λ
Output: trajectory τ
Acquire vehicle's initial state x_1, initialize vehicle's belief state b_0, dummy control u_0 and trajectory $\tau = \{\}$
for $t = 1, 2, ...T$
  Update belief state and parameters with recurrent policy: $b_t, \theta_t = \text{RNN}(x_t, u_{t-1}, b_{t-1}; \lambda)$
  Calculate vehicle's control with MPC controller: $u_t = \text{MPC}(x_t, \theta_t)$
  Execute u_t on the autonomous driving system and observe reward r_t and new state x_{t+1}
  Incorporate experience into the trajectory: $\tau \leftarrow \tau \cup \{u_{t-1}, b_{t-1}, x_t, r_t\}$
end for

A.2. Algorithm: Training Recurrent Policy

Algorithm 2 Training Recurrent Policy

Input: recurrent policy parameters λ_0, initial value function parameters ϕ_0
Output: final recurrent policy parameters λ_K
for $k = 0, 1, 2, ...K$
Initialize the trajectories set $D_k = \{\}$
for $i = 0, 1, 2, ...N$
Execute Algorithm 1 to collect a trajectory $\tau_i = (u_0, b_0, x_1, r_1, ...)$ with horizon T
Calculate the episode reward $R_t = \sum_{j=t}^{T} \gamma^{j-t} r_j$ for $t = 1, ...T$ and add them into the trajectory τ_i
Enlarge the trajectories set $D_k \leftarrow D_k \cup \{\tau_i\}$
end for
Update recurrent policy parameters λ by minimizing the following objective:

$$\lambda_{k+1} = \arg\min_{\lambda} \frac{1}{|D_k|T} \sum_{\tau_i \in D_k} \sum_{t=1}^{T} -Q(x_t, \theta_t; \phi_k) + \alpha[x_t + f(x_t, u_t; \theta_t)\Delta t - x_{t+1}]^2$$

where $b_t, \theta_t = \text{RNN}(x_t, u_{t-1}, b_{t-1}; \lambda)$

Update value function parameters ϕ by fitting the episode reward:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|D_k|T} \sum_{\tau_i \in D_k} \sum_{t=1}^{T} [R_t - Q(x_t, \theta_t; \phi)]^2$$

end for

Appendix B. Extra Experimental Setups

B.1. Vehicle Dynamics Model

For the formulation of f_1 and f_2, certain symmetric properties with respect to the velocity angle β and the steering control w must be satisfied so that it is aligned with the physical vehicle system:

$$f_1(v, \beta, w, y, z; \theta_1) = f_1(v, -\beta, -w, y, z; \theta_1), f_2(v, \beta, w, y, z; \theta_2) = -f_2(v, -\beta, -w, y, z; \theta_2).$$

To achieve such property, we introduce two auxiliary neural networks $f_1^{'}(v, \beta, w, y, z; \theta_1), f_2^{'}(v, -\beta, -w, y, z; \theta_2)$ which can have arbitrary outputs. We further combine these networks to achieve $f_1(v, \beta, w, y, z; \theta_1) = [f_1^{'}(v, \beta, w, y, z; \theta_1) + f_1^{'}(v, -\beta, -w, y, z; \theta_1)]/2$ and $f_2(v, \beta, w, y, z; \theta_2) = [f_2^{'}(v, \beta, w, y, z; \theta_2) - f_2^{'}(v, -\beta, -w, y, z; \theta_2)]/2$ so that f_1 and f_2 satisfy the symmetric property mentioned before. Specifically, both $f_1^{'}$ and $f_2^{'}$ are 2-layer feed-forward neural networks with 32 hidden units for each layer. The activation function is $\tanh$. The weights of the first layers are shared for basic representation.

Notably, the prediction of the velocity v should always be positive, for which we further rewrite $f_1(v, \beta, w, y, z; \theta_1) = [f_1^{'}(v, \beta, w, y, z; \theta_1) * (2\sqrt{v} + f_1^{'}(v, \beta, w, y, z; \theta_1)) + f_1^{'}(v, -\beta, -w, y, z; \theta_1) * (2\sqrt{v} + f_1^{'}(v, -\beta, -w, y, z; \theta_1))]/(2\Delta t)$ so that $v + \Delta t * f_1(v, \beta, w, y, z; \theta_1) \geq 0$ is always true. In general, $\theta = [\theta_1, \theta_2]$ is a set of parameters to describe the complete dynamics model.

B.2. Model Predictive Control

As mentioned in Section 3.2, the distance to the reference trajectory term is written as $d(p_t, q_t, \mathcal{G}) = \min_{k \in \{1,2,\ldots,N\}} \|(p_t, q_t)^T - g_k\|_2$. However, the minimization operation is difficult in practical gradient-based optimization methods. Instead, we utilize a differential function $d(p_t, q_t, \mathcal{G}) = -\log\left(\frac{1}{N}\sum_{k=1}^{N} \exp\{-\|(p_t, q_t)^T - g_k\|_2\}\right)$ as the distance function, which can be viewed as an approximate and soft version of the original minimization calculation (Chen et al., 2017). Other than that, we absorb the constraints on the controls as a sinusoidal activation function on the original controls.

B.3. POMDP Setup

The reward function is a combination of velocity error ϵ_v and route error ϵ_r, informed as $r_t = \text{clip}(1 + 0.2(1 - e_v/\epsilon_v), 0, 1) \times \text{clip}(1 + 0.2(1 - e_r/\epsilon_r), 0, 1$. The state x_t can be directly acquired from the CARLA simulator. Specifically, position p, q, heading angle ψ and speed v are read directly, while velocity angle β is calculated by $\arctan(v\_y/v\_x)$ and smoothed closed to 0.

Regarding the recurrent policy, an LSTM-based neural network (Hochreiter and Schmidhuber, 1997) with a 256-dim recurrent layer is used. The hidden layer in LSTM can be seen as the belief state in the PODMP framework. The output of the recurrent policy is a 64-dim vector, including 32 parameters in $f_1^{'}$ and $f_2^{'}$ respectively (as we only modify the last layer of $f_1^{'}$ and $f_2^{'}$ for efficieny). To successfully train this recurrent policy, we further adopt an actor-critic algorithm PPO (Schulman et al., 2017) as it is one of the best performing on-policy RL algorithms suitable to the training of dynamics hidden states and continuous action space. We conclude all model structures and hyperparameters in Table 2

Appendix C. Extra Experimental Results

C.1. Ablation Study

In this section we execute an ablation study on the three most essential designs in the framework: RNN structure (RNN for short), system identification loss (SI for short) and episode reward (ER for short). Notably, these variants, which already cover a bunch of adaptive methods, might not have identical structures as previous research but are suitable to this certain autonomous driving task. "-RNN" replaces recurrent policy with a feed-forward policy, which is optimized by 2 objectives as

Table 2: The hyperparameters of MPC-RRL.

HYPERPARAMETERS	VALUE
MPC HORIZON	20
MPC CONTROL INTERVAL	0.1
TARGET SPEED	8.0
SAMPLING STEPS PER UPDATE	512
BATCH SIZE	512
DISCOUNT FACTOR (γ)	0.99
LEARNING RATE	1E-4
ENTROPY COEFFICIENT	0.01
GAE LAMBDA	0.98

dual control. "- SI Loss" trains with cumulative reward maximization alone. "- ER" means only minimizing system identification loss, which is a practical way to fit system's parameters online. We follow the same task setup as Section 5.2 and report the median goal error under all testing values per parameter in Table 3. It indicates that RNN structure plays the most fundamental role in the framework. The combination of these three designs empowers the controller with a more consistent improvement over the adaption ability.

Table 3: The median goal error and average rank of all ablation settings. Less goal error turns to a lower rank.

Ablation Settings			**final ratio**	**moi**	**tire friction**	**damping rate**	**drag coefficient**	**town**	**AVG RANK**
RNN	SI	ER							
+	+	+	9.09	8.44	9.13	8.33	23.50	8.03	**1.3**
+	+	-	10.65	9.40	9.98	9.12	23.89	8.80	2.8
+	-	+	10.44	9.43	9.97	9.19	10.68	8.83	2.5
-	+	+	10.82	9.40	9.94	9.28	10.74	8.94	3.0