FFStreams: Fast Search with Streams for Autonomous Maneuver Planning

Mais Jamal[1] and Aleksandr Panov[2]

***Abstract*— In autonomous driving, maneuver planning is essential for ride safety and comfort, involving both motion planning and decision-making. This paper introduces FFStreams, a novel approach combining high-level decision-making and low-level motion planning to solve maneuver planning problems while considering kinematic constraints. Addressed as an integrated Task and Motion Planning (TAMP) problem in a dynamic environment, the planner utilizes PDDL, incorporates Streams, and employs Fast-Forward heuristic search. Evaluated against baseline methods in challenging overtaking and lane-changing scenarios, FFStreams demonstrates superior performance, highlighting its potential for real-world applications.**

I. INTRODUCTION

In autonomous driving, the system mimics human drivers by navigating, following a planned route, and adhering to traffic rules. Core components include localization, perception, prediction, planning, and control modules.

Lane changing, especially on highways (Fig. 1 (a)), is a safety-critical maneuver in autonomous driving, requiring safe and comfortable trajectories. Overtaking a moving obstacle is another critical maneuver involving two lane changes, which adds complexity, particularly on two-way roads with potential oncoming traffic (Fig. 1 (b)). In such scenarios, the planning system prioritizes safety and human-like driving behavior to minimize disruption to other drivers.

The planning module has two layers: high-level for driving action and low-level for trajectory planning. The final trajectory must ensure safety and adhere to vehicle kinematic constraints. The choice of integration or decoupling depends on the approach. Decoupling decision-making and trajectory planning may lead to infeasible trajectories, requiring repetitive planning with low-level constraints. Integrated Task and Motion Planning (TAMP) combines high-level task planning with low-level motion planning (Fig. 2), generating feasible solutions for long-horizon tasks with geometric constraints and logical reasoning. Recent research [1], [2] has focused on advancing TAMP.

Unlike traditional TAMP with restrictive assumptions [3], our approach relaxes some constraints, moving from abstract to numeric planning and adapting to dynamic environments. We integrate Task and Motion Planning using Fast-Forward heuristic search and Streams, which were introduced by [4]. This allows for the generation of jerk-optimized trajectories,

[1]Mais Jamal is with the Center for Cognitive Modeling, Moscow Institute of Physics and Technology, 141701 Dolgoprudny, Russia mayssjamal@phystech.edu

[2]Aleksandr I. Panov is with Federal Research Center Computer Science and Control of the Russian Academy of Sciences, 119333 Moscow, Russia, and AIRI, 105064 Moscow, Russia panov@airi.net

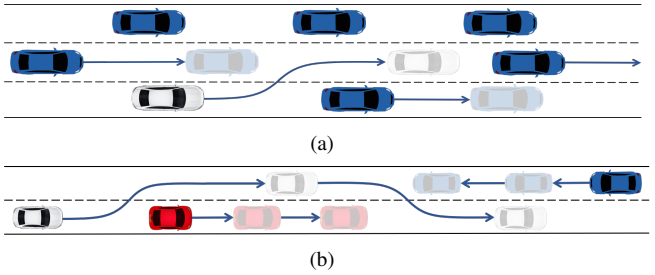

Fig. 1. Critical scenarios: (a) Lane-changing on a highway. (b) Overtaking a low-speed front obstacle on a two-way road.

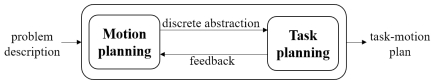

Fig. 2. Integrated task planning and motion planning.

discretized and added to the task problem, enabling the Fast Forward heuristic planner to search for an optimal plan for autonomous driving in a changing environment.

This work focuses on lane-keeping, overtaking, and lane-changing maneuvers. Two primary criteria guide our planning process: ensuring a safe trajectory and generating a comfortable maneuver by optimizing jerk values with acceleration and curvature constraints. This paper makes the following contributions: (1) Integration of Task and Motion Planning in the dynamic environment of autonomous driving, using Fast Forward Heuristic Search with streams for planning lane-keeping, lane-changing, and overtaking safe maneuvers. (2) Validation of our approach through performance comparisons with recent baseline methods in planning overtaking and lane-changing maneuvers on CommonRoad real-life scenarios. Results demonstrate superior performance over baseline methods. (3) Significant reduction in runtime compared to other TAMP approaches, making our approach well-suited for real-time applications.

The remainder of the paper is structured as follows: Section II discusses recent related work, Section III provides background on PDDL and streams, Section IV describes the methods, Section V presents experiments and results, and Section VI concludes with a discussion of future work.

II. RELATED WORK

Some studies [5], [6] focus solely on making the decision for overtaking or lane-changing maneuvers at high speeds, while others [7], [11] also address trajectory planning alongside decision-making.

In [5], the overtaking decision is modeled as an MDP, with the policy learned using dynamic fuzzy logic. The

TABLE I

CHARACTERISTIC COMPARISON OF MANEUVER PLANNING METHODS

Method	Strategy	Decision-making	Trajectory planning	Kinematic constraints	Curvature constraint	Overtake planning	Lane Change	Open source
Fuzzy RL [5]	MDP & RL	✓	✗	✗	✗	✓	✗	✗
Adaptive Behavior Tree [6]	Behavior tree	✓	✗	✗	✗	✓	✗	✗
Semantic Sampling [7]	Decision tree&Opt.	✓	✓	✓	✓	✓	✗	✗
Iterative Path&Speed Opt. [8]	Piecewise-Jerk Opt.	✓	✓	✓	✓	✗	✗	✓
MCTS-OPD [9]	Tree search&RL	✓	✓	✗	✓	✓	✓	✓
IRP [10]	Interval-based planning	✓	✓	✗	✓	✓	✓	✓
CLF-CBF-QP [11]	Rule-based FSM	✓	✓	✓	✗	✗	✓	✓
FFStreams	Search-based&Jerk Opt.	✓	✓	✓	✓	✓	✓	-

scenario involves four surrounding obstacles, and the MDP model has nine states. Experiments are simulated in SUMO. In [6], an adaptive behavior tree is used for overtaking decisions, with a genetic programming algorithm learning the structure. Successful tree structures emerge, but the learning process has significant execution time constraints. Kinematic constraints are not considered in both studies, questioning the method's applicability and leaving a gap between decision-making and trajectory planning.

[7] planned the overtaking maneuver including decisions and trajectories in the Frenet coordination system by constructing a semantic decision tree with nodes defining which vehicles to follow and which to overtake, generating multiple longitudinal and lateral trajectory candidates for each obstacle considered to be overtaken. Then, of all trajectory candidates, the trajectory of the minimum cost is selected. However, the maximum longitudinal acceleration reaches 4.5 m/s^2, leading to uncomfortable rides.

In [11], a rule-based planner using a FMS manages the lane-changing maneuver through five states: Change to the right/left lane, Back to the left/right lane, and adaptive cruise control. Trajectory planning employs a quadratic program optimization with control Lyapunov functions and control barrier functions (CLF-CBF-QP). Tested in Urban Road and Highway scenarios, the proposed method achieves a 62.46% success rate in the urban setting and 55.58% on the highway.

In [12], various RL baseline agents for autonomous driving in Highway-Env environments [13] are introduced. The primary baseline, Monte-Carlo Tree Search (MCTS), employs Optimistic exploration for deterministic systems [9] to search the trajectory space for the best driving action. Another essential baseline is Interval-based Robust Planning (IRB), addressing safe decision-making by considering uncertain non-linear systems and using an interval-based predictor of drivers' behaviors [10]. MCTS-OPD and IRB, cover various maneuvers, including overtaking, and can be integrated into Highway-Env environments for trajectory planning, with actions like Idle, Faster, Slower, Lane Right, and Lane Left.

In Baidu Apollo's open-source autonomous driving platform [14], a planner integrates decision-making with path smoothing and Piecewise-Jerk speed optimization techniques [8]. It selects the minimum-cost path in the Frenet coordinate system and optimizes its longitudinal speed for a collision-free trajectory. However, it does not fully address the need for spatiotemporal trajectories crucial for overtaking or lane-changing maneuvers with high-speed dynamic obstacles, as it prioritizes spatial path planning over temporal profile adjustments.

To overcome challenges, we adopted FFStreams, an integrated TAMP approach leveraging human experiences for decision-making while considering kinematic constraints. Using FastForward heuristic search, this approach addresses autonomous driving problems modeled in PDDL2.1 and modified through Streams. Table I compares the maneuver planning methods, including ours, across decision-making, trajectory planning, curvature, kinematic constraints, and overtaking/lane-changing planning. Our planner uniquely covers both overtaking and lane-changing while considering curvature and kinematic constraints.

III. BACKGROUND

Formulating a TAMP problem involves defining key components: the task domain (states and actions), the motion domain (configuration space), and their interactions. Choosing the de-facto standard [15], the Planning Domain Definition Language (PDDL) [16], specifically its extension PDDL2.1 for numeric planning. Additionally, we use Streams to implement the configuration space, generating new variables and predicates associated with the world's current state.

The maneuver-planning task involves a planning domain and problem $(Dom, Prob)$ in PDDL. The domain Dom defines predicates, object types, and actions for constructing the world model, focusing on autonomous driving with maneuvers like overtaking and lane-change. The planning problem $Prob$ specifies objects, current state, and goal state, with a heuristic planner searching for a plan using the relaxed plan length heuristic h^{FF} [17]. A relaxed plan is designed for a simplified version of a problem, ignoring the negative consequences of actions. While determining the optimal relaxed plan is NP-complete, it is possible to calculate it efficiently in polynomial time. The h^{FF} is an admissible heuristic; it returns infinity if no relaxed plan exists, and otherwise, it returns the relaxed plan's length, indicated by the number of action layers in the relaxed planning graph.

PDDL2.1 [18] introduced numeric fluents representing the ego vehicle's configurations, aiding in collision-checking with obstacle configurations. Additionally, plan metrics enable the integration of optimization metrics, crucial for selecting plans with minimum cost (e.g., driving time).

The planning task has a finite set of objects X. The domain Dom is a tuple $\langle P, F, A \rangle$, where P is a finite set of predicates, which are binary-valued functions of one or more

objects, F is a finite set of functions (variables) also of one or more objects, and A is a finite set of actions (operators).

A predicate p is a positive literal if it is evaluated on its associated objects $\bar{x} = \langle x_1, x_2, ..., x_k \rangle$ to true and a negative literal if evaluated to false. A state I is a set of literals. An action a is given by a tuple of object arguments $\bar{X} = \langle X_1, X_2, ..., X_k \rangle$ and a set of preconditions $pre(a)$ of $\bar{X}$, which are positive literals $pre^+(a)$ and negative literals $pre^-(a)$ that must hold for the operator to apply, and a set of effects $eff(a)$ on $\bar{X}$, which are positive literals $eff^+(a)$ and negative literals $eff^-(a)$ that is the result of applying the operator. The action a is applicable at the state I if

$$\left(pre^+(a(\bar{x})) \subseteq I \right) \wedge \left(pre^-(a(\bar{x})) \cap I = \emptyset \right). \quad (1)$$

The resulting state I' after applying action a in a state I:

$$I' = \left(I \backslash eff^-(a(\bar{x})) \right) \cup eff^+(a(\bar{x})). \quad (2)$$

The planning problem $Prob$ is a tuple $\langle X, I_0, G \rangle$, where X is a set of objects in the domain, I_0 is a set of positive literals expressing the initial state, and G is a set of both positive and negative literals expressing the goal state. A plan $\pi = [a_1(\bar{x}_1), \ldots, a_k(\bar{x}_k)]$ is a finite sequence of k action instances such that each $a_i(\bar{x}_i)$ is applicable in the state I_{i-1} leading to the state I_i. A stream $s(\bar{x})$ is a conditional function of an *input* set of object arguments $\bar{x} = \langle x_1, x_2, ..., x_k \rangle$. This function can modify the planning problem $Prob$ by generating an *output* tuple of new objects $\bar{y} = \langle y_1, y_2, ..., y_k \rangle$, and a set of certified facts associated with them, where $s.cert = \{p \mid \forall \bar{x} \in \bar{X}, \forall \bar{y} \in s(\bar{x}).p(\bar{x} + \bar{y})\}$. The stream can yield None if generating new objects is impossible. The stream can be applied on input parameters $\bar{x}$ only if a set of positive literals p related to them exists in the domain, where $s.dom = \{p \mid \forall \bar{x} \in \bar{X} \cdot p(\bar{x})\}$.

Integrated TAMP balances sampling continuous state variables by streams and iteratively searching discrete action plans. The Incremental Algorithm increases the planning level iteratively until reaching the maximum level N. At each level $l \leq N$, all stream instances $s(\bar{x})$ are initiated and evaluated. Upon evaluation, if the stream is applicable, its certified facts are added to the current problem state, and the search planner is called.

Online collision checking at each future time step is vital in autonomous driving tasks and is implemented as an action in the PDDL domain. The time variable implicitly incorporates changes in obstacle states into the planning domain. Illustrating its importance, consider an example (Fig. 3): during plan search, after examining a lane-change action and transitioning the car from one configuration state q_a to another q_b via trajectory T_{ab}, the current driving time t_c must be updated before subsequent collision checking. The current driving time t_c is increased by the needed time to follow the trajectory T_{ab} . Subsequently, the trajectory T_{bc} is evaluated for collision with obstacles at the updated t_c, and if collision-free, the following lane-change action is applied.

Compared to the PDDLStream framework [4], which performs *offline* collision checking in a dedicated stream before constructing the PDDL problem and calling the search

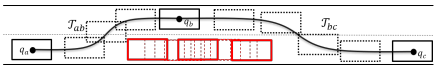

Fig. 3. Online collision checking of trajectories (T_{ab}, T_{bc}) with obstacle's predicted trajectory (in red). It is crucial to update the current time after applying any action during the plan search.

planner, assuming the robotic agent handles static objects with state changes only during agent-performed actions. In the FFStreams planner, we adapt numeric planning using PDDL2.1 with Streams, integrating online collision-checking actions and utilizing the Fast-Forward heuristic planner (FF)[17], which supports the numeric-fluents feature.

IV. Methods

To address autonomous driving problems, we consider the integrated TAMP problem. We model the planning domain using PDDL2.1, make modifications with Streams, and employ the FastForward heuristic search for an iterative search of the optimal plan. We enhance the Metric-FF-v2.1 version, improving the parser's capability for extensive input and addressing comparison issues related to operations on the Float data type in C.

Modeling task planning in PDDL helps interpret the solution plan's searching process. Streams allow the sampling of new configurations and trajectories for the vehicle, considering autonomous vehicle kinematics. Using the FF planner, numerical operations in the domain associate a cost with each action. The Incremental algorithm iterates between sampling and searching processes. One domain covers autonomous driving tasks, including lane-keeping, yielding for obstacles, lane-changing, and overtaking maneuvers.

Motion planning relies on prior knowledge of the autonomous vehicle's state and other obstacles' states, including coordination, heading, and speed. Ideal perceptional observation of surrounding obstacles' states is assumed, with dynamic obstacles moving at a constant speed.

Illustrated in Algorithm 1 and depicted in Fig. 4, the planning task takes inputs are the ego's initial configuration q_0, goal predicates P_{goal}, which includes moving forward and positioning finally on either: current lane in the case of following speed, yield, and overtaking maneuver; neighbor lane in the case of lane-changing maneuver, initial observations of obstacles' coordination and speeds $O(0)$, planning domain Dom, and the maximum number of levels N, set to five experimentally to compromise performance and runtime.

At each timestep, t, an initial PDDL problem is generated, and for N-levels, applicable streams update the problem until finding a plan. Configurations are constructed from the initial configuration q_0 and sampled configurations on the current and neighbor lanes: $Q(t) = [q_0, q_1, ..., q_n]$. The **Configuration stream** generates new configurations $[q_1, q_2, ..., q_n]$ with 2D Frenet coordinates and speed. Different speeds enable planning for varied trajectories needed for maneuvers. Simultaneously, the **Trajectory stream** generates discretized trajectories $traj(q_i, q_j)$ between every two configurations, constructing a connectivity graph. If a kinematic trajectory exists, it consists of a set of in-between configurations

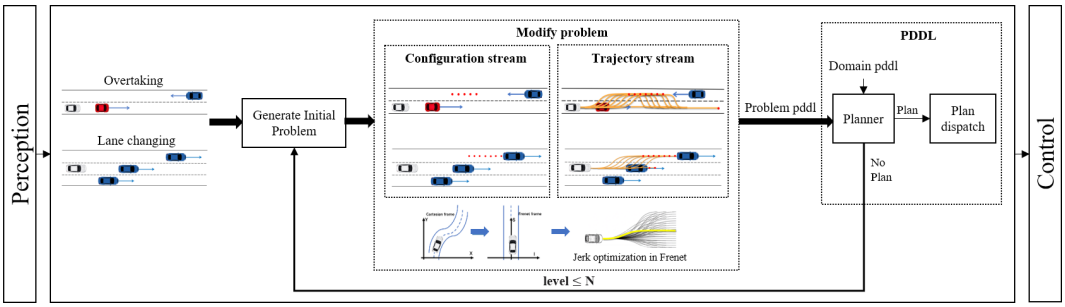

Fig. 4. The scheme of the FFStreams maneuver planner.

$[q_{i\_j\_1}, q_{i\_j\_2}, ..., q_{i\_j\_s}]$, where s is constant and equals the planning time divided by Δt.

Algorithm 1 Autonomous Maneuver Planning

$INPUT : q_0, O(0), N(levels), P_{goal}, Dom$
$OUTPUT : traj = [q_1, ..., q_k], [a_1, \ldots, a_k]$
$initialize : t \leftarrow 0$, $traj \leftarrow \emptyset$
while $t \leq t_k$ **do**
  $Q(t) \leftarrow q_0$
  $Prob(t) \leftarrow GenerateInitialProb(q_0, O(t), P_{goal})$
  for $l = 1, 2, ..., N$ **do**
    $ApplyApplicableStreams(Prob(t))$
    $Prob(t) \leftarrow Prob(t) \cup s.cert$
    $\pi \leftarrow FFPlanner(Dom, Prob(t))$
    if π **then**
      $a_t, q_1 \leftarrow PlanDispatch(\pi)$
      $traj \leftarrow traj \cup q_1$
    end if
  end for
  $q_0 \leftarrow q_1$
  $O(t) \leftarrow ObserveEnv()$
  $t \leftarrow t + 1$
end while

After calling each applicable stream, its certified facts are added to the PDDL problem. At each level, the updated problem and domain are handed to the FF planner to find a plan π. Once a plan is found, the first action and its corresponding trajectory are extracted. The action is executed, updating the ego vehicles state and the obstacles state with a new observation. Possible actions include following speed, yielding, moving to the left or right lane, and overtaking. The algorithm outputs the entire trajectory from the initial to the goal state $traj = [q_1, ..., q_k]$, the followed plan, and its set of actions: $\pi = [a_1, \ldots, a_k]$.

A. Maneuver-Planning Domain

Representing our planning domain in PDDL, we define two types of objects in the planning domain: car configuration $-conf$ and obstacle $-obstacles$. In addition, the set of predicates including: *(traj* $?q_1$ $?q_2$*)* indicating the existence of a trajectory from configurations $?q_1$ to $?q_2$, *(next* $?q_1$ $?q_2$ $?q_{end}$*)* indicating a sequence of sub-configurations on a trajectory to the final configuration $?q_{end}$, *idle* stating that the collision checking process is idle to start a new check, *(checking_traj* $?q_1$ $?q_2$ $?o$*)* indicating that the trajectory between configurations $?q_1$ and $?q_2$ is under checking with obstacle $?o$, *(checked_traj* $?q_1$ $?q_2$ $?o$*)* indicating that the trajectory between the two configurations is checked with obstacle $?o$ and is collision-free, *(ego_at* $?q$*)* stating that the ego vehicle is currently at configuration $?q$, *(on_right_lane)* and *(on_left_lane)* stating the ego's current lane. The functions of the maneuver-planning domain include the following changing variables: *(total_cost)* - total cost, *(curr_time)* - current time, *(time_of_traj* $?q_1$ $?q_2$*)* - the duration of following the trajectory, *(at_s* $?q$*) (at_l* $?q$*) (at_time* $?q$*)* - coordination and the reaching time of a certain configuration, *(obst_at_s* $?o$*) (obst_at_l* $?o$*) (obst_at_speed* $?o$*)* - coordination and speed of a certain obstacle when the current time is zero.

The maneuver domain consists of 14 actions. Four moving forward actions through a specific trajectory from first configuration $?q_1$ to the second configuration $?q_2$; follow speed, yield, and change to the left/right lane. For the moving action to occur, the ego vehicle must start at the first configuration *(ego_at* $?q_1$*)*. A collision-free trajectory between the initial and final configurations should be established *(traj* $?q_1$ $?q_2$*)* after checking with all existing obstacles $?o$; *(checked_traj* $?q_1$ $?q_2$ $?o$*)*. The action results in the ego vehicle transitioning to the second configuration*(ego_at* $?q_2$*)*, and incrementing current time by the duration of following the trajectory. In the case of a lane change, the ego vehicle shifts to the neighboring lane (e.g., *(on_left_lane)*), no longer occupying the initial lane.

```
(:action change_lane_left
    :parameters (?q1 ?q2 - conf)
    :precondition (and (ego_at ?q1)
        (traj ?q1 ?q2)
        (> (at_l ?q1)(at_l ?q2))
        (forall (?o - obstacles)
            (checked_traj ?q1 ?q2 ?o))
        (on_init_lane) (idle))
    :effect (and (moved_forward)
        (ego_at ?q2)(not (ego_at ?q1))
        (on_left_lane) (not (on_init_lane))
        (increase (curr_time) (time_of_traj ?q1 ?q2))
        (increase (total_cost) 3)))
```

In the current work, we implement the prediction in the collision-checking actions of the domain. Due to the linearity restriction in PDDL2.1 functions, we model the motion of the surrounding vehicles as $\Delta x = V.\Delta t$, where V is constant. Real-time planning accounts for changes in the speed of surrounding vehicles in each planning cycle, modifying the plan accordingly. Each trajectory is discretized on Δt and associated with configurations. For checking a current trajectory with an existing obstacle, we use one begin_check action, eight actions for eight checking situations, and one end_check action. The collision-checking actions iterate through all trajectory configurations, comparing the vehicle's longitudinal and lateral coordination with the predicted coordination of the obstacle at each t_i. The checking

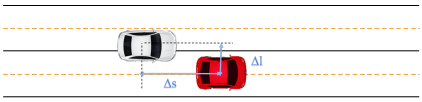

Fig. 5. Linear collision checking on Δs and Δl in the Ferent system.

process is exemplified by one of the eight actions, each corresponding to a situation related to the obstacle's lane, ego's lane, Δs, and Δl in the Frenet system (Fig. 5).

```
(:action begin_check
    :parameters(?first_q ?last_q -conf ?o -obstacles)
    :precondition (and
       (idle)(ego_at ?first_q)(traj ?first_q ?last_q)
       ( not (checked_traj ?first_q ?last_q ?o )))
    :effect (and
       (is_first ?first_q ?o)(is_last ?last_q ?o)
       (checking_traj ?first_q ?last_q ?o)(not(idle)))
       )

(:action check_forword_diff_lane_upper_bigger_delta_y
    :parameters (?most_first ?first ?after_first
        ?last - conf ?o - obstacles)
    :precondition (and (is_first ?first ?o)
        (is_last ?last ?o)
        (next ?first ?after_first ?last)
        (checking_traj ?most_first ?last ?o)
        (not (= ?first ?last))
        (> (at_l ?after_first)(obst_at_l ?o))
        (>= (- (at_l ?after_first)(obst_at_l ?o)) 3))
    :effect (and
        (not (is_first ?first ?o)) ; update is_first
        (is_first ?after_first ?o)))
```

B. Configuration stream

The configuration stream Conf generates new configurations, sampling forward positions on the current and neighboring lanes with varying speeds. Configurations in the current lane with higher speeds facilitate acceleration, while those with lower speeds aid in deceleration, essential for following a slow-moving front obstacle. Sampling configurations in neighboring lanes with speeds higher than the current speed enables lane-change and overtaking maneuvers within acceleration limits.

```
(:stream conf
    :inputs (?q1)
    :domain (and (ego_at ?q1) (at_s ?q1) (at_l ?q1)
        (at_time ?q1))
    :outputs (?q2)
    :certified
        (and (at_s ?q2)(at_l ?q2) (at_time ?q2))  )
```

C. Trajectory stream

The Trajectory stream generates feasible trajectories between two configurations in the Frenet. Trajectory optimization involves minimizing squared jerk over a time interval T[19]. This process includes defining quintic polynomials for lateral movement and quartic polynomials for longitudinal movement, calculating their costs, and selecting the trajectory with the minimum cost. The cost of each trajectory is a weighted sum of costs on longitudinal and lateral movements, expressed by the equation:

$$C_{total} = k_{lat}C_l + k_{lon}C_s. \tag{3}$$

The cost of the trajectory on the lateral movement is a function of the lateral jerks J_l, trajectory duration, and lateral error, expressed by the equation:

$$C_l = k_j \sum J_l^2 + k_t T + k_d dl^2, \tag{4}$$

TABLE II
VEHICLE AND TRAJECTORY OPTIMIZATION SOFT AND HARD PARAMETERS

Parameter	Symbol	Value(soft)	Value(hard)
Maximum acceleration	a_{max}	1.0 $[m/s^2]$	15 $[m/s^2]$
Jerk weight	k_j	0.1	0.08
Trajectory duration weight	k_t	0.1	0.9
Error weight	k_d	1.0	1.0
Maximum speed	v_{max}	33.33 $[m/s]$	
Maximum Curvature	K	1 $[1/m]$	
Time step	Δt	0.2 $[s]$	
Lane width	l	3.4 $[m]$	
Lateral weight	k_{lat}	1.0	
Longitudinal weight	k_{lon}	1.0	

where k_j is the jerk weight, T is the time interval, and dl is the lateral error of the final trajectory's point. The cost of the trajectory on the longitudinal movement is a function of the longitudinal jerks, trajectory duration, and longitudinal error, expressed by the equation:

$$C_s = k_j \sum J_s^2 + k_t T + k_d ds'^2, \tag{5}$$

where ds' is the longitudinal speed error of the final trajectory's point. The parameters of optimization are stated in Table II.

```
(:stream motion_trajectory
    :inputs (?q1 ?q2)
    :domain (and (at_s ?q1) (at_l ?q2) (at_time ?q1)
        (at_s ?q2) (at_l ?q2) (at_time ?q2) )
    :outputs (?q_{1_2_1} ... ?q_{1_2_24})
    :certified
        (and (traj ?q1 ?q2) (time_of_traj ?q1 ?q2)
        (next ?q1 ?q1_2_1 ?q2) ...
        (next ?q1_2_23 ?q1_2_24 ?q2)))
```

V. EXPERIMENTS AND RESULTS

The proposed method has been validated through simulation experiments in two main driving scenarios and real-life scenarios: (1) a single forward lane and a bidirectional neighboring lane with varying obstacle speeds and initial positions. (2) a multi-lane highway. (3) Three CommonRoad scenarios. The experiments were conducted locally on an Ubuntu system with an Intel Core i7-10700KF CPU:8x3GHz computer with 32 GB RAM and NVIDIA GeForce RTX 3060 Ti video card.

Scenario 1: Two lanes, different directions

The scenario involves two obstacles: a low-speed dynamic obstacle in the same lane as the ego vehicle, positioned 50 m ahead with a random speed of $[7m/s, 8m/s]$, and another oncoming obstacle in the neighboring lane moving in the opposite direction. The oncoming obstacle starts at a random position on the x-axis $[50\ m, 350\ m]$ with a speed $[4\ m/s, 12\ m/s]$. The ego vehicle, with an initial speed v_0 of 10 m/s, aims to overtake the front obstacle. It can either accelerate to overtake the front obstacle before the oncoming one passes or yield, adjusting speed until the oncoming obstacle passes, and then accelerate to overtake.

In 100 random scenarios, our method, with soft parameters optimized for comfort (table II), achieved a 92% success rate in overtaking. In 9% of cases, overtaking was completed before being passed by the neighboring-lane obstacle, while

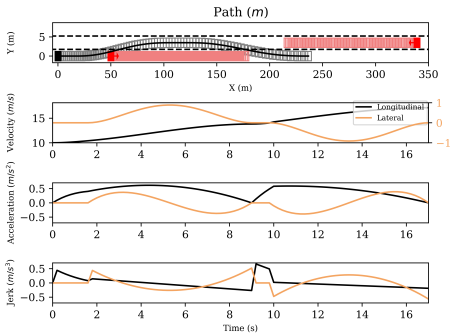

Fig. 6. A successful overtaking by FFStreams planner with soft parameters at a critical scenario where oncoming obstacle's velocity is 7.22 m/s.

in 83%, overtaking was performed after the obstacle passed, aligning with a safer and human-like driving attitude. In 8% of cases the planner failed to find a plan within the maximum level. A critical scenario is illustrated in fig. 6, showcasing the planned trajectory, velocity, acceleration, and jerk profiles. The planned actions include maintaining the lane and accelerating to overtake when a safe trajectory is available, ensuring smooth paths, and a maximum jerk of 0.8 m/s^3.

We compared FFStreams to open-source planners Monte Carlo Tree Search with Optimistic Exploration for Deterministic Systems (MCTS-OPD) and Interval-based Robust Planning (IRP)[10], implemented in[12]. We configured our method with optimization hard parameters (table II). Three behaviors were differentiated: 1) overtaking before the backward obstacle passes; 2) yielding until the backward obstacle passes, then overtaking; and 3) following an unsafe trajectory leading to collision. Trajectories were evaluated using the Occupant's Preference Metric (OPM) [20], considering comfort based on the maximum lateral and longitudinal jerk.

In 100 random experiments, FFStreams achieved a 94% success rate in overtaking, while MCTS-OPD had an 82% success rate, and IRP achieved 62%. Unsuccessful overtaking attempts in MCTS-OPD (18%) and IRP (38%) often resulted from decisions made when the oncoming obstacle had high speed, leading to collisions. FFStreams, consistently provided candidate trajectories for lane-keeping or yielding to the front obstacle's speed, except of 6% of cases where no plan was found within the maximum planning level. FFStreams outperformed in overtaking before being passed by the oncoming vehicle in 44% of experiments, compared to 40% for MCTS-OPD and 37% for IRP.

On the OPM metric, all three planners exhibited highly aggressive driver behavior. Thus, we designate the FFStreams planner with soft optimization parameters as the safest, ensuring overtaking only when safe and comfortable for passengers, achieving a Normal drivers behavior on the OPM metric. In Fig. 7(a), a critical scenario is presented, where the ego's velocity is 7.5 m/s, and the oncoming vehicle's velocity is 4.47 m/s. The planned trajectory by FFStreams with hard parameters is demonstrated, including velocity, acceleration, and jerk profiles. Planned actions involve maintaining the lane and accelerating to overtake the front obstacle when a safe collision-free trajectory for overtaking exists. Despite the extreme acceleration limit, the planned optimized trajectory has a maximum absolute acceleration of 4.3 m/s^2.

In Fig. 7 (b), the planned trajectory by MCTS-OPD in a critical scenario demonstrates a rapid and unsafe overtaking decision with unrealistic values, including a maximum acceleration of 15.0 m/s^2 and a maximum jerk of 25.3 m/s^3, posing a high risk to the passengers of the autonomous vehicle and other vehicles. In Fig. 7 (c), the planned trajectory by IRP in a critical scenario reveals inconsistent and abnormal decisions, alternating between changing to the left and right lanes multiple times. These decisions pose a high risk of affecting other vehicles' driver attitudes. Additionally, the planned trajectory exhibits unrealistic values of acceleration and jerk, reaching a maximum absolute acceleration of 15.0 m/s^2 and a maximum absolute jerk of 52.5 m/s^3, which are impractical for real-life applications. The acceleration and jerk values in both MCTS-OPD and IRP deviate significantly from human driver behavior, posing safety concerns. Additionally, the decision-making process in these methods exhibits inconsistency, potentially influencing other drivers negatively.

Scenario 2: Highway

To show the efficiency of our approach, we tested it on the same experiments introduced in the rule-based Control CLF-CBF-QP approach [11] in the highway environment. The ego has an initial speed of $v_{ego}(0) = 29$ m/s. The scenario consists of one front obstacle placed randomly $x_1(0) = [50\ m, 65\ m]$, moving at a random constant speed $[26\ m/s, 32\ m/s]$. In addition to the front obstacle, there are four obstacles in the neighboring lane on a random x coordination $[-85\ m, 85\ m]$, moving at random speeds $[26\ m/s, 32\ m/s]$ and with random accelerations $[-3\ m/s^2, 3\ m/s^2]$, and an additional obstacle in the third lane, $x6(0) = [-85\ m, 85\ m]$, which changes to the neighboring lane at a random time. The goal is to change to the left lane. After running 50 experiments, in 88.00% of the experiments, the ego successfully changed the lane. In 84.00% of them, the lane-change happened under 60 seconds (Table IV). By contrast, the rate has not exceeded 56% in the CLF-CBF-QP method. We have also evaluated the trajectories of the two methods on the Occupants Preference Metric (OPM) [20]. The results show that both methods are classified under the Public Transportation driving type classification.

Fig. 8 demonstrates an example of a successful experiment of lane-changing. In this experiment, one obstacle changes lanes, and the planner changes the lane when it is safe to change. In Fig. 8, the speed, acceleration, and jerk profiles are shown; the acceleration ranges in $[-0.5\ m/s^2, +0.5\ m/s^2]$ and the jerk ranges in $[-0.5\ m/s^3, +0.5\ m/s^3]$ which leads to a comfortable ride. The results show that the FFStreams planner has a higher success rate and outperforms the Rule-based control CLF-CBF-QP method. The FFStreams planner runs with a frequency of ≈ 6 Hz.

Additional scenarios: CommonRoad

To showcase the effectiveness of our approach, we tested

TABLE III

RESULTS OF OVERTAKING SIMULATIONS

Method	Overtake	Yield&Overtake	Failure rate	Success rate	OPM
MCTS-OPD baseline	40%	42%	18%	82%	Extremely Aggressive Driver
IRP baseline	37%	25%	38%	62%	Extremely Aggressive Driver
FFStreams - Opt. hard params	44%	50%	6%	94%	Extremely Aggressive Driver
FFStreams - Opt. soft params	9%	83%	8%	92%	Normal Driver

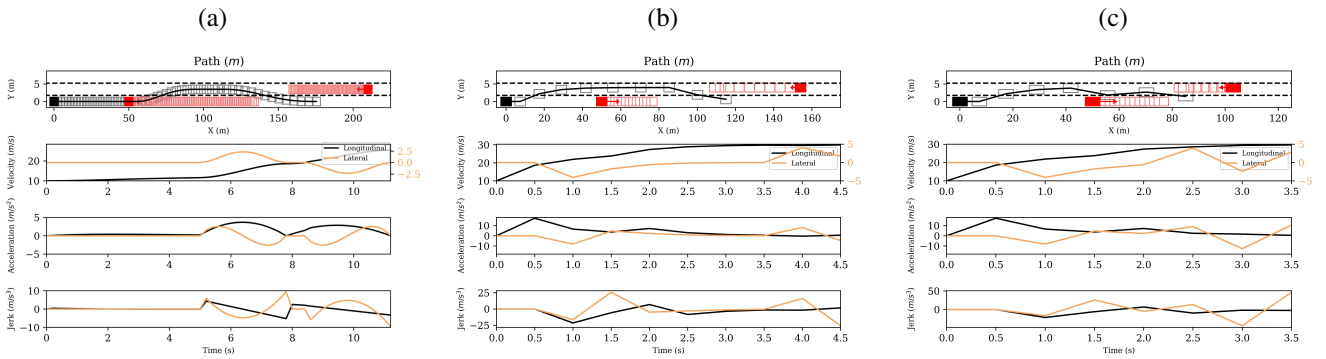

Fig. 7. (a) A successful overtaking by FFStreams planner(hard parameters). (b) A successful but risky overtaking was planned by MCTS-OPD. In a critical scenario where the oncoming obstacle's velocity is 10.7 m/s. (c) A successful but very risky overtaking was planned by IRP, where the oncoming obstacle's velocity is 6.0m/s.

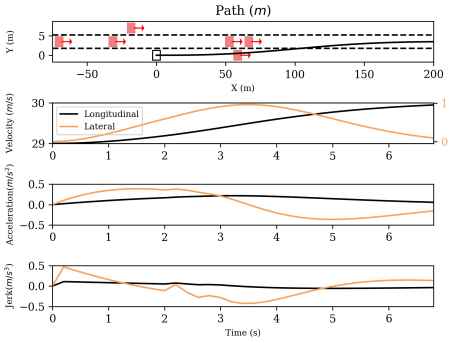

Fig. 8. A successful lane-changing, when the ego vehicle changes the lane when it is safe to change and merges into the middle of four obstacles.

TABLE IV

RESULTS OF LANE-CHANGE SIMULATIONS

Method	Lane-Change under 60s	OPM
FFStreams	84.00%	Public Transportation
CLF-CBF-QP	55.58%	Public Transportation

it on three CommonRoad scenarios USA_US101-1_1_T-1, ESP_Monzon-2_1_T-1, ITA_Empoli-18_1_T-1 derived from real-life situations. These scenarios present diverse challenges an autonomous driving system might face. The first involves rapid lane changes and merging with fast-moving obstacles. The second requires following a decelerating bus, necessitating speed adjustments. The third involves following an accelerating vehicle, demanding speed adaptations. We compared our method with the Search-Based planner in CommonRoad, which utilizes 2697 motion primitives and is based on the Search-Based Planning Library (SBPL) [21].

In Fig. 9, FFStreams excels over the Search-based planner in various scenarios. In the second scenario, FFStreams avoids collision, and adjusts speed to follow a bus, while the Search-based planner results in the ego vehicle stopping. In the last scenario, FFStreams accelerates to follow the leading vehicle, whereas the Search-based planner maintains zero speed, causing the ego vehicle to stop throughout. FFStreams excels in acceleration and jerk values, offering a more comfortable trajectory. Its superiority in comfort, safety, and human-like criteria arises from its ability to integrate semantic knowledge for human-like behavior.

To assess planning efficiency, we analyzed average runtime in experiments with up to eleven obstacles comparing our algorithm with two baselines; MCTS-OPD and IRP (Fig. 10). While runtime increases with the number of checked obstacles, focusing on a limited number of relevant obstacles is practical for executing autonomous overtaking or lane-changing maneuvers in diverse scenarios.

VI. CONCLUSION

The FFStreams planner efficiently handles maneuver planning by integrating TAMP, interpreting decisions with logic rules, and considering kinematic constraints to avoid infeasible trajectories. Currently supporting lane-keeping, yielding, lane-changing, and overtaking maneuvers, future work may expand the domain to include diverse intersection scenarios. The use of a standard planning language allows performance comparisons with different heuristic search planners. Enhancements may involve replacing the heuristic planner for increased efficiency, and testing the method with uncertain information in the observable environment.

ACKNOWLEDGMENT

This work was partially supported by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730321P5Q0002; grant No. 70-2021-00138).

REFERENCES

[1] B. Kim, L. Shimanuki, L. P. Kaelbling, and T. Lozano-Pérez, "Representation, learning, and planning algorithms for geometric task and motion planning," *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 210–231, 2022.

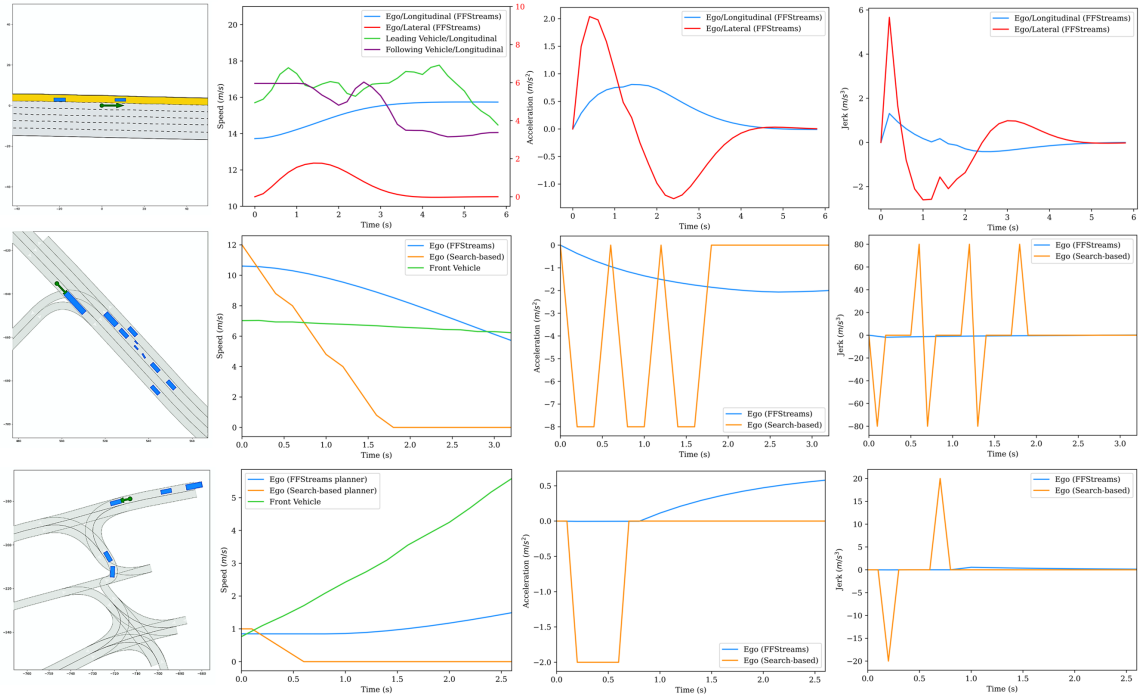

Fig. 9. Speed, acceleration, and jerk graphs comparing FFStreams and Search-based planners in CommonRoad scenarios (USA_US101-1_1_T-1, ESP_Monzon-2_1_T-1, ITA_Empoli-18_1_T-1). Green arrows mark the initial ego vehicle position, and blue rectangles denote initial obstacle positions.

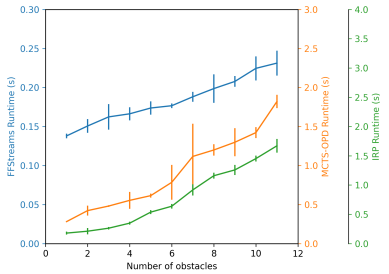

Fig. 10. Mean runtime and standard deviation for FFStreams (blue), MCTS-OPD (orange), and IRP (green) algorithms with up to eleven obstacles.

[2] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2motion: From natural language instructions to feasible plans," *arXiv preprint arXiv:2303.12153*, 2023.

[3] D. Nau and M. Ghallab, "Measuring the performance of automated planning systems," in *Performance Metrics for Intelligent Systems Workshop (PerMIS04)*, 2004.

[4] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 440–448.

[5] Q. Wu, S. Cheng, L. Li, F. Yang, L. J. Meng, Z. X. Fan, and H. W. Liang, "A fuzzy-inference-based reinforcement learning method of overtaking decision making for automated vehicles," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 236, no. 1, pp. 75–83, 2022.

[6] M. Jamal and A. Panov, "Adaptive maneuver planning for autonomous vehicles using behavior tree on apollo platform," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 2021, pp. 327–340.

[7] H. Li, G. Yu, B. Zhou, P. Chen, Y. Liao, and D. Li, "Semantic-level maneuver sampling and trajectory planning for on-road autonomous driving in dynamic scenarios," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1122–1134, 2021.

[8] J. Zhou, R. He, Y. Wang, S. Jiang, Z. Zhu, J. Hu, J. Miao, and Q. Luo, "Autonomous driving trajectory optimization with dual-loop iterative anchoring path smoothing and piecewise-jerk speed optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 439–446, 2020.

[9] J.-F. Hren and R. Munos, "Optimistic planning of deterministic systems," in *European Workshop on Reinforcement Learning*. Springer, 2008, pp. 151–164.

[10] E. Leurent, D. Efimov, T. Rassi, and W. Perruquetti, "Interval prediction for continuous-time systems with parametric uncertainties," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 7049–7054.

[11] S. He, J. Zeng, B. Zhang, and K. Sreenath, "Rule-based safety-critical control design using control barrier functions with application to autonomous lane change," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 178–185.

[12] E. Leurent, "rl-agents: Implementations of reinforcement learning algorithms," https://github.com/eleurent/rl-agents, 2018.

[13] ——, "An environment for autonomous driving decision-making," https://github.com/eleurent/highway-env, 2018.

[14] "Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving, howpublished = https://github.com/apolloauto/apollo, note = Accessed: 2019-02-11."

[15] P. Haslum, N. Lipovetzky, D. Magazzeni, C. Muise, R. Brachman, F. Rossi, and P. Stone, *An introduction to the planning domain definition language*. Springer, 2019, vol. 13.

[16] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson *et al.*, "Pddl— the planning domain definition language," *Technical Report, Tech. Rep.*, 1998.

[17] J. Hoffmann, "Ff: The fast-forward planning system," *AI magazine*, vol. 22, no. 3, pp. 57–57, 2001.

[18] M. Fox and D. Long, "Pddl2. 1: An extension to pddl for expressing temporal planning domains," *Journal of artificial intelligence research*, vol. 20, pp. 61–124, 2003.

[19] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 987–993.

[20] I. Bae, J. Moon, J. Jhung, H. Suk, T. Kim, H. Park, J. Cha, J. Kim, D. Kim, and S. Kim, "Self-driving like a human driver instead of a robocar: Personalized comfortable driving experience for autonomous vehicles," *arXiv preprint arXiv:2001.03908*, 2020.

[21] M. Likhachev, "Search-based planning library (sbpl) including a collection of graph searches and planners that utilize these graph searches," November 2018.