REACTIVE MOTION PLANNING

HIGHWAY DRIVING WITH LOCAL OPTIMIZATION

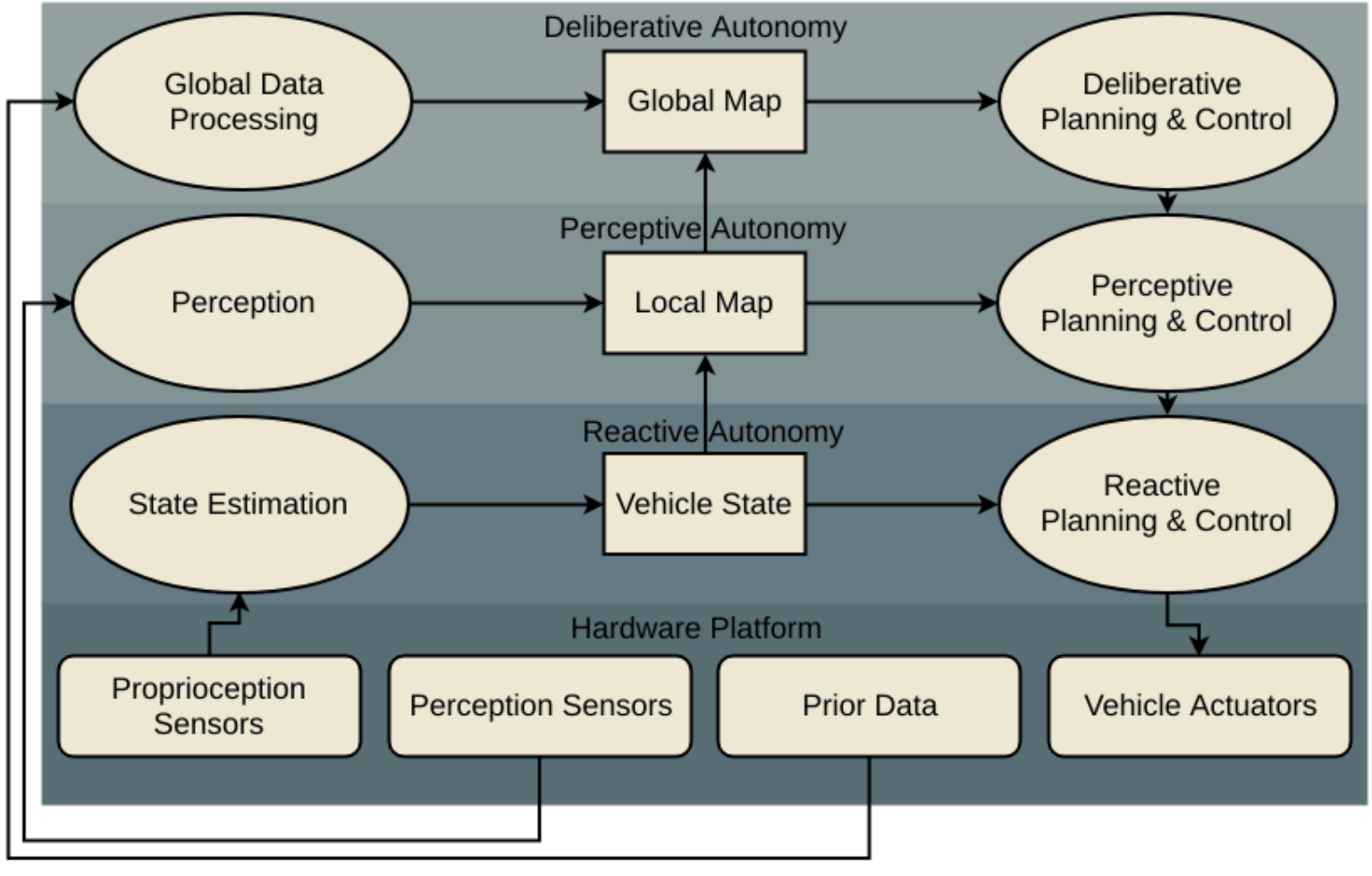
Jordan Ford / @jsford

OUTLINE

- Background
  - Planning Stack
  - Motion Planning
  - Speed, Quality, Cost
  - Problem Statement
- Prior Art
  - Survey
  - Graph Planning
  - Variational Planning
- Pattern Planning
  - Motivation
  - Control Flow
  - Examples
- Trajectory Generation
- Planner Evaluation
  - Simulation
  - Vehicle Testing
- Conclusion

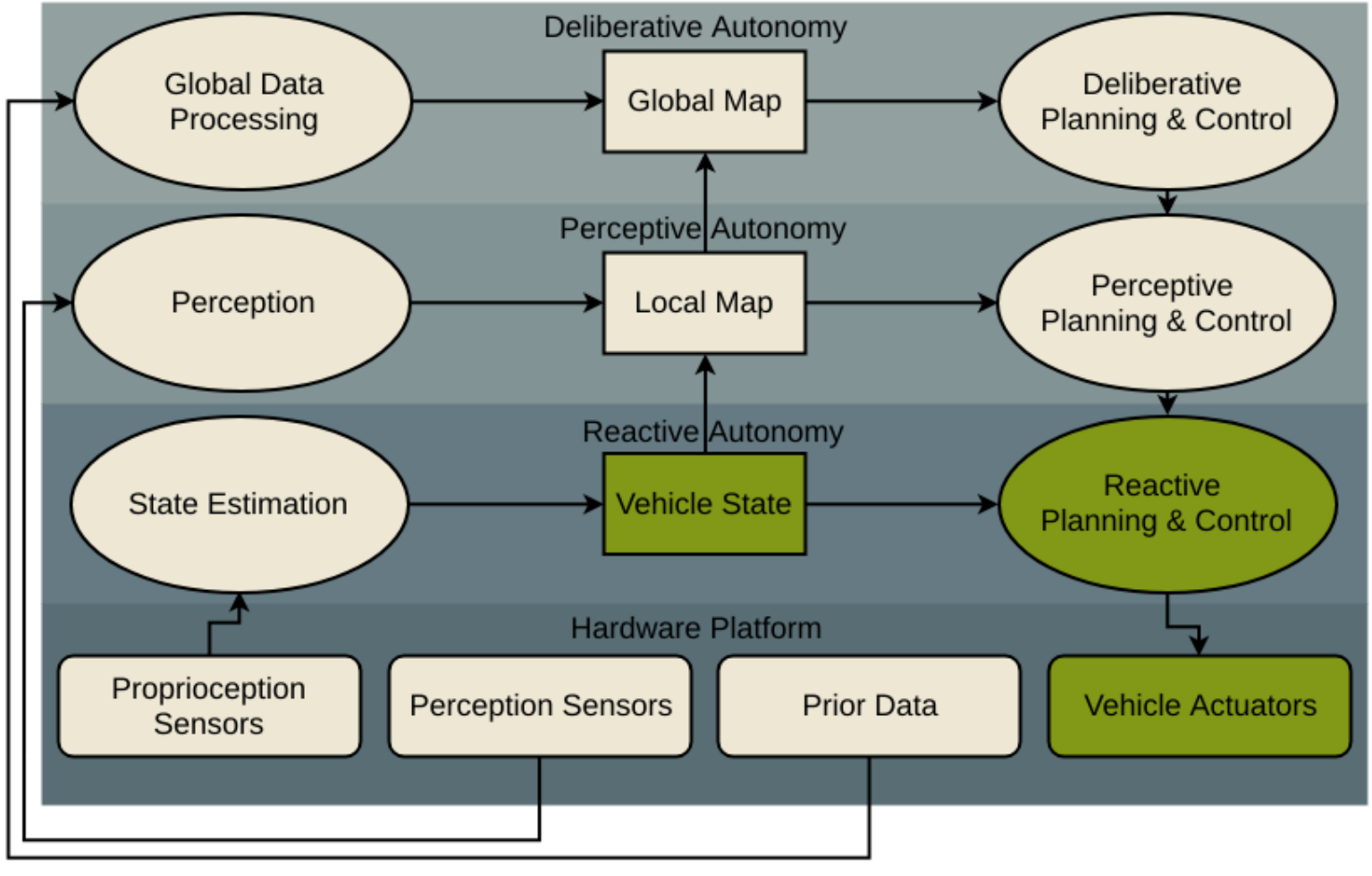
BACKGROUND

BACKGROUND PLANNING STACK



Kelly, A. (2013). *Mobile Robotics: Mathematics, Models, and Methods.* Cambridge University Press

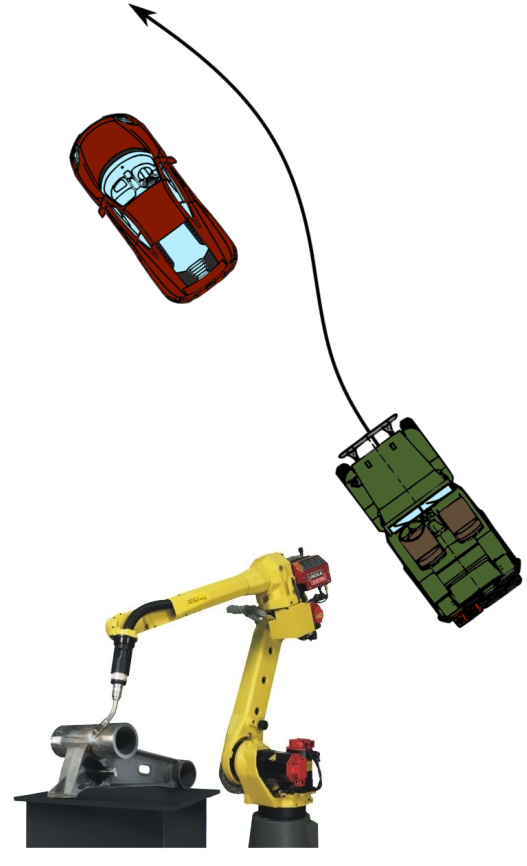
BACKGROUND PLANNING STACK



Kelly, A. (2013). *Mobile Robotics: Mathematics, Models, and Methods.* Cambridge University Press

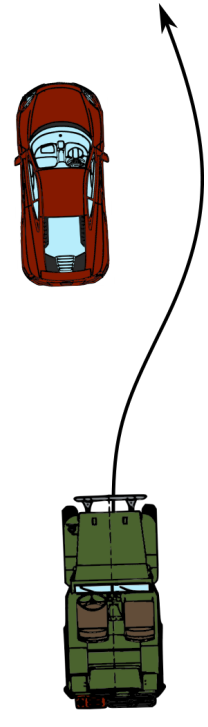
BACKGROUND MOTION PLANNING

- Accomplish a movement task.
- Adapt.
  - Real-Time Demands
  - Unknown/Dynamic Environments
  - Sensor Anomalies (Noise/Occlusion/Failure)
- Respect physical constraints.
  - Joint/Torque Limits
  - Obstacle Avoidance
- Optimize.
  - Energy, Time
  - Safety, Comfort



BACKGROUND MOTION PLANNING

- Inputs
  - Vehicle State
  - Local Road Geometry
  - Obstacle Predictions
  - Behavioral Commands
- Outputs
  - A trajectory valid for a short time into the future (~10sec).
- Plan vs. Trajectory
  - By convention, a path contains spatial waypoints e.g. $[x, y, \theta, \kappa]^\intercal$, and a trajectory includes both spatial and velocity waypoints e.g. $[x, y, \theta, \kappa, v, a]^\intercal$.



BACKGROUND PLANNING SPEED

- Planners checkpoint the world state and solve the instantaneous planning problem.
  - Worst case, the vehicle cannot react to a change for nearly two planning cycles.
- Highway driving requires rapid replanning.
  - cycle time $\leq$ 33ms (30Hz) is preferred. (~4ft. at 80mph).
  - cycle time $\geq$ 100ms (10Hz) is considered poor. (~12ft. at 80mph).

BACKGROUND TRAJECTORY QUALITY

- Quantifying trajectory quality is difficult

  -

  no golden standard.
- But many factors are obviously desirable.
  - Obstacle Avoidance.
  - Reaching the destination.
  - Occupant comfort.
  - Minimum Travel Time.
  - etc.

BACKGROUND PLANNING COST

- For the GM fleet, **computing $\approx \frac{1}{2}$ of total power!**
- Many existing motion planners use high-end CPUs and GPUs to achieve dense state space sampling.
- Ideally, the motion planner would consume **one thread on a commodity CPU.**

PRIOR ART

PRIOR ART

GRAPH-SEARCH METHODS

- Conformal Graph Planning
- Augmented Graph Planning

VARIATIONAL METHODS

- Parametric Optimal Control
- Iterative Linear Quadratic Requlator

GEOMETRIC METHODS

- Visibility Graph
- Cell Decomposition

INCREMENTAL SEARCH

- Rapidly exploring Random Trees (RRT)
- Probabilistic Road Maps

PRIOR ART

GRAPH-SEARCH METHODS

- Conformal Graph Planning
- Augmented Graph Planning

VARIATIONAL METHODS

- Parametric Optimal Control
- Iterative Linear Quadratic Requlator

GEOMETRIC METHODS

- Visibility Graph
- Cell Decomposition

INCREMENTAL SEARCH

- Rapidly exploring Random Trees (RRT)
- Probabilistic Road Maps

PRIOR ART GLOBAL VS. LOCAL OPTIMIZATION

GLOBAL OPTIMIZATION

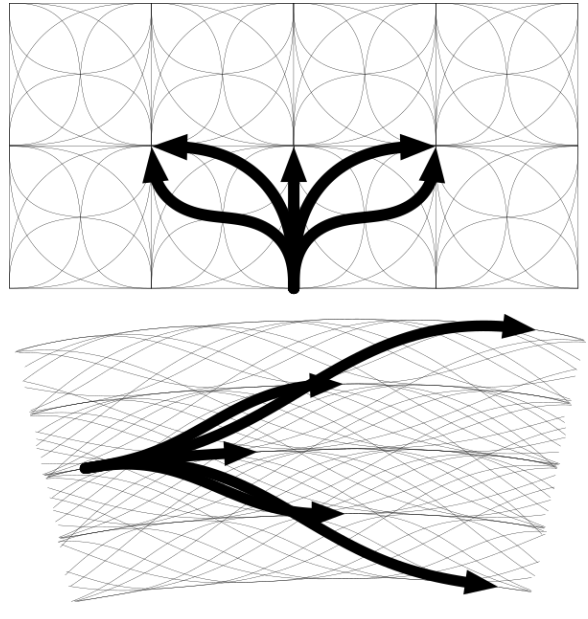
- Graph-Search planners attempt to search the full space of trajectories for the global optimum.
- Graph-Search planners are *resolution-complete*. If a feasible plan exists and the graph resolution is fine enough, they are guaranteed to find it (eventually).

LOCAL OPTIMIZATION

- Optimization-Based planners make no global guarantees. Given an initial plan, they will refine it to a local optimum.
- If the initial plan is poor, the optimization will likely converge to a poor result.

PRIOR ART CONFORMAL GRAPH PLANNING

1. Sample points in (s, l) road coordinates.
2. Generate kinematically feasible paths connecting sampled coordinates.
3. For each path, generate a set of trajectories - each with a constant acceleration profile.
4. Assign costs to each trajectory. (Exact formulation not specified).
5. Choose final vertex to maximize forward progress and minimize time and cost.
6. Backtrack from final vertex to construct the minimum cost trajectory.

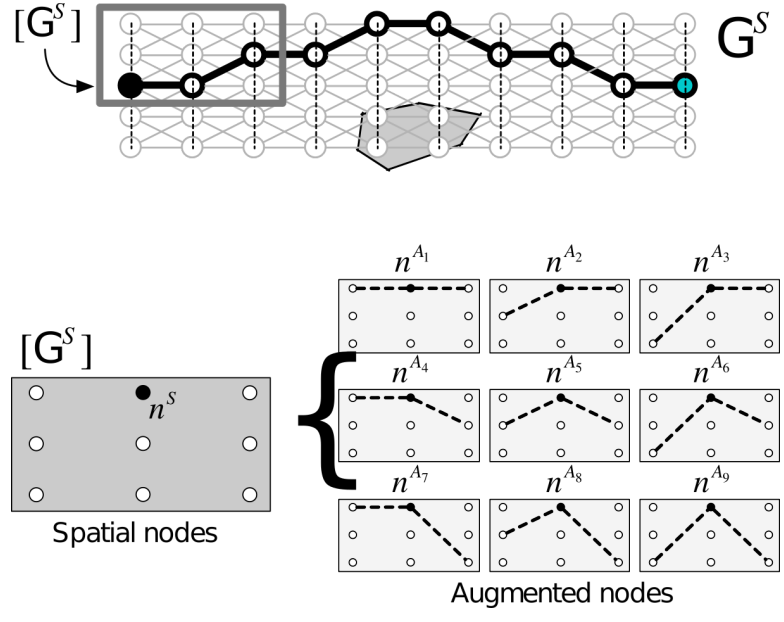


Note: GPU acceleration required to meet realtime demands.

"Motion Planning for Autonomous Driving with Conformal Spatiotemporal Lattice" McNaughton et al.

PRIOR ART AUGMENTED GRAPH PLANNING

1. Construct a graph by sampling points (spatial nodes) at regular intervals along the road.
2. Assign obstacle and lane-centering costs to each spatial node in the graph.
3. Use the Bellman-Ford algorithm to construct a min. cost path through the graph.
4. Use a path following algorithm (pure pursuit) to generate a kinematically feasible smooth path.
5. Use constraint satisfaction to generate a reference velocity profile based on the speed limit and curvature of the road.
6. Generate a pool of similar trajectories by sampling in space and velocity.
7. Rank the pool of trajectories, and select the first ranked trajectory.



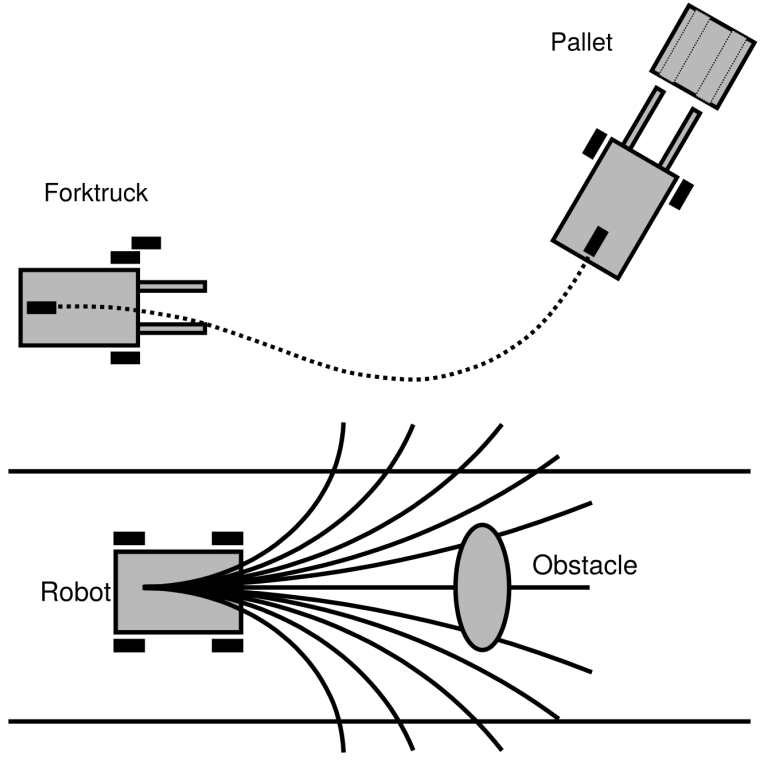
"Runtime-Bounded Tunable Motion Planning for Autonomous Driving" Gu et al.

PRIOR ART OPTIMIZATION-BASED TRAJECTORY GENERATION

Solve the Boundary Value Problem connecting $\mathbf{X_s} = [x_0, y_0, \theta_0, \kappa_0]^\intercal$ to $\mathbf{X_e} = [x_e, y_e, \theta_e, \kappa_e]^\intercal$ while respecting vehicle kinematics.

1. Parameterize vehicle controls as polynomials with unknown coefficients.
2. Initialize the polynomial coefficients by solving the terminal heading and terminal curvature constraints explicitly.
3. Use a shooting method to forward simulate the parameterized trajectory.
4. Optimize over the polynomial coefficients to bring the final path state $\mathbf{X_f}$ into agreement with $\mathbf{X_e}$.
5. More detail to follow soon.

"Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control" Kelly et al.



PRIOR ART

COMPUTATIONAL COMPLEXITY

GRAPH-SEARCH METHODS

- The Conformal Lattice runs at 10Hz, using a GTX 260 to reoptimize and evaluate ~400,000 trajectories per cycle.
- The Augmented Graph Planner runs at 15-20Hz and evaluates ~4,000 spatial nodes and ~20,000 augmented nodes per cycle.

VARIATIONAL METHODS

- While these methods can be made arbitrarily complex, the trajectory generator is extremely lightweight. Expect at least $650,000 \frac{\text{paths}}{\text{sec}}$ per CPU thread.

PROBLEM STATEMENT

1. Until recently, the hardware and power costs of planning have not played a major part in planning algorithm design. **Researchers can afford expensive hardware and high power consumption, but consumers cannot.**

2. **Quantitative comparison of planner performance is difficult.** There are desirable and undesirable features, but no universal agreement on how to fuse them into a single scalar cost.

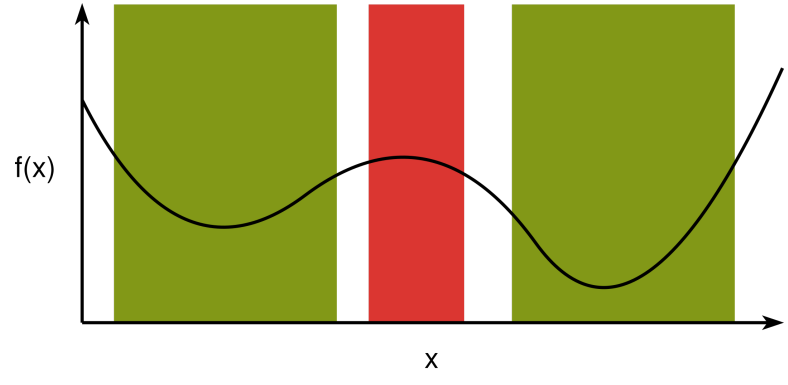
PATTERN PLANNING

PATTERN PLANNING

GUIDED LOCAL OPTIMIZATION

- Probe the global search space with a series of local optimizations.

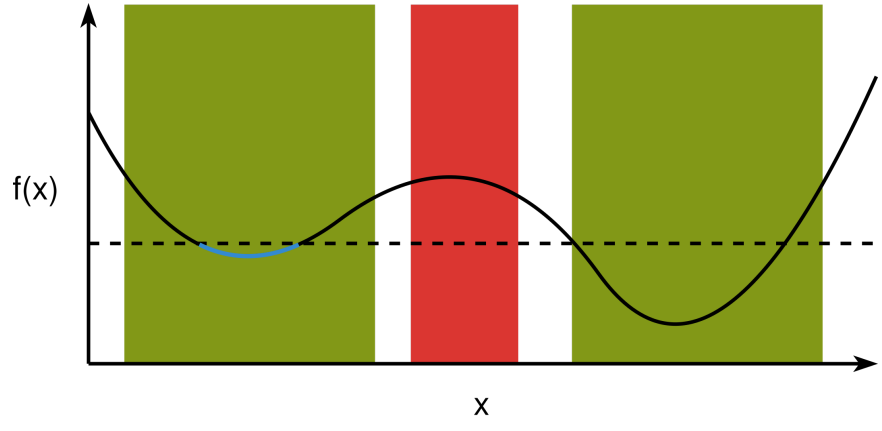
- Leverage the structure of the highway environment to prioritize exploration.



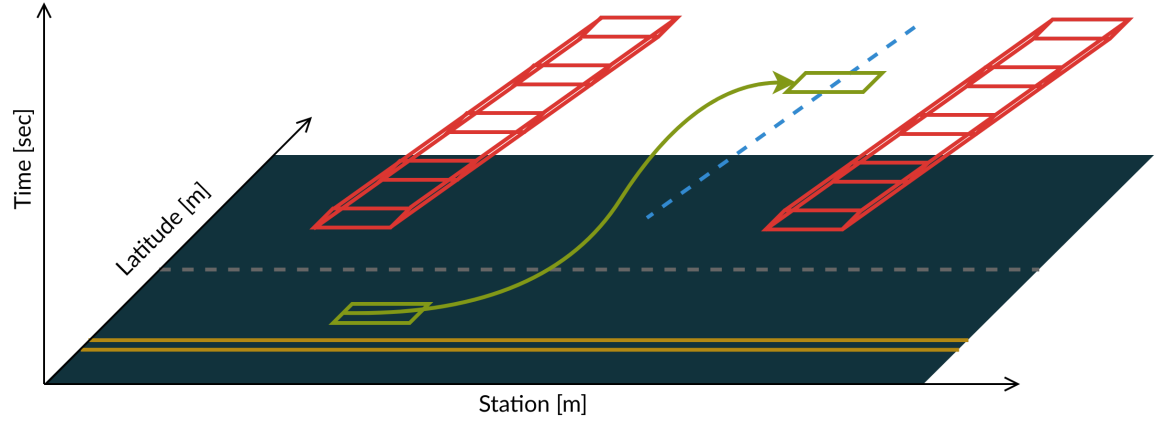
PATTERN PLANNING EARLY EXIT

- Execute the first trajectory that achieves the current goal and is found to be sufficiently safe and comfortable.

- Perfect answers are useless if they are found too late.



PATTERN PLANNING MANEUVER PATTERN

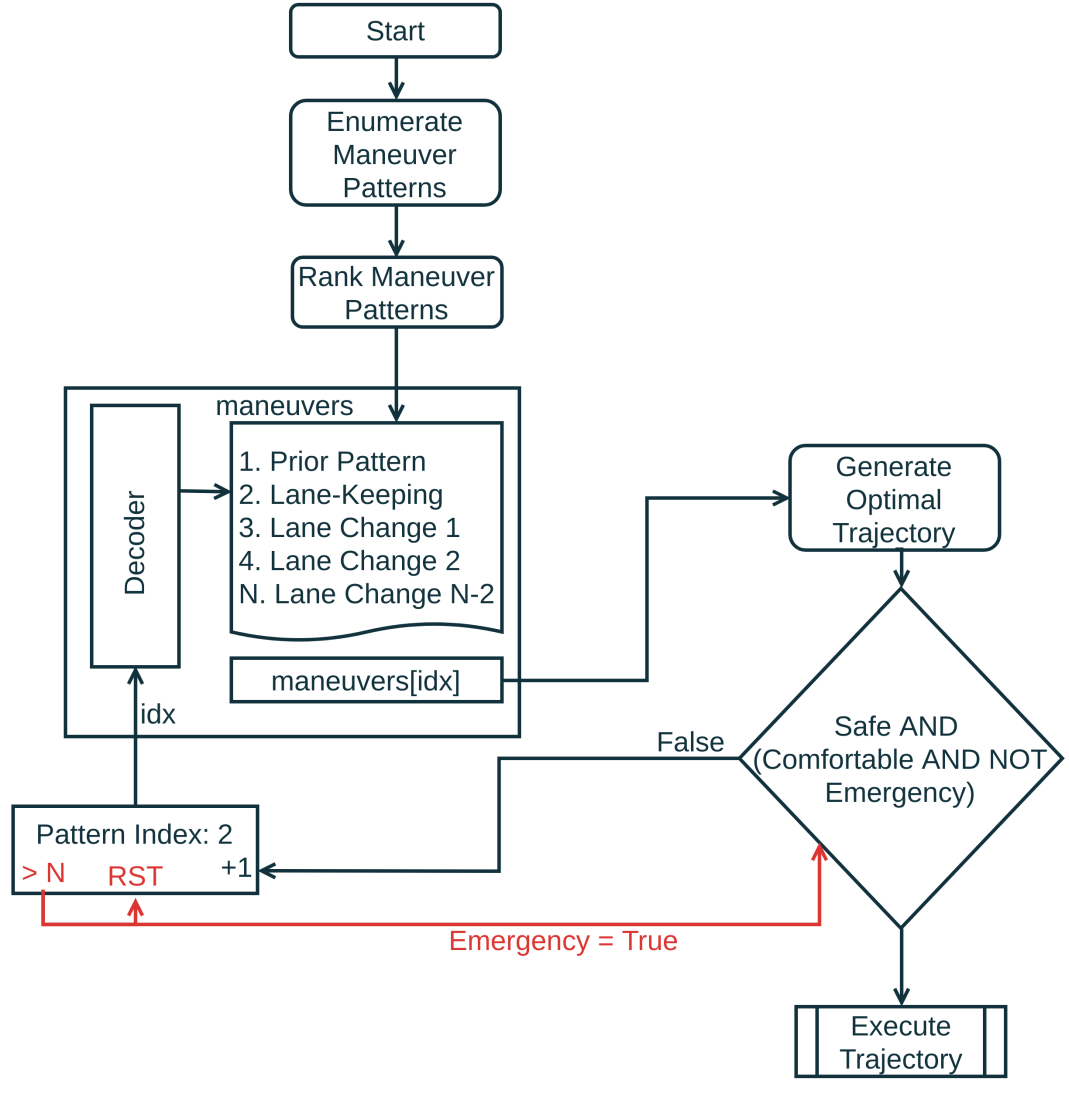


- The highway driving environment is naturally segmented by local traffic.
- It is convenient to abstract away from individual trajectories.
- All trajectories that terminate on the blue line are considered part of the same pattern.

PATTERN PLANNING

CONTROL FLOW

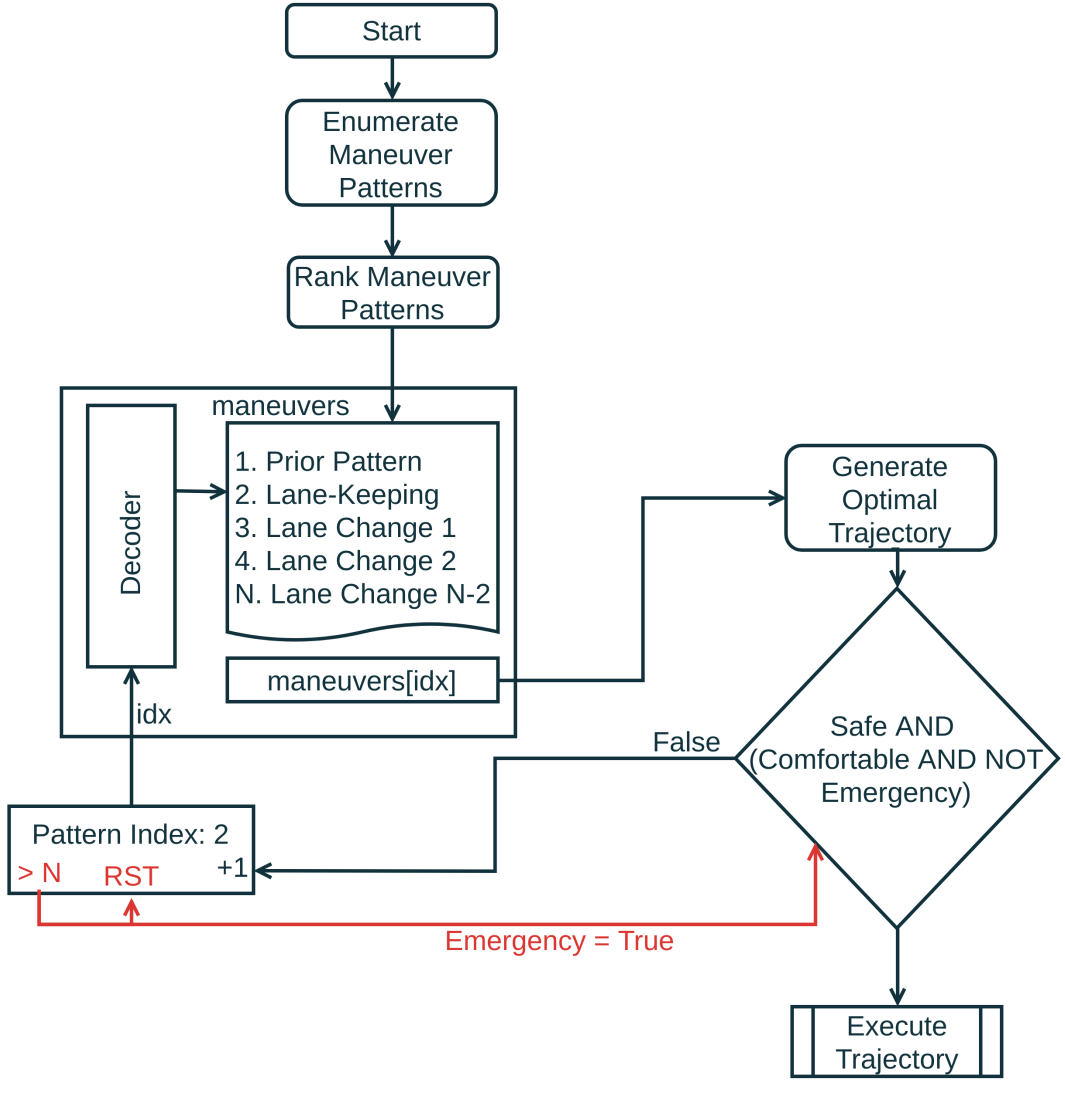
- Generate a ranked list of maneuver patterns.
- Take the top ranked pattern, and generate a trajectory using it.
- If the trajectory is safe and (optionally) comfortable, execute it.
- Else, return to the list of patterns and evaluate the next ranked pattern.



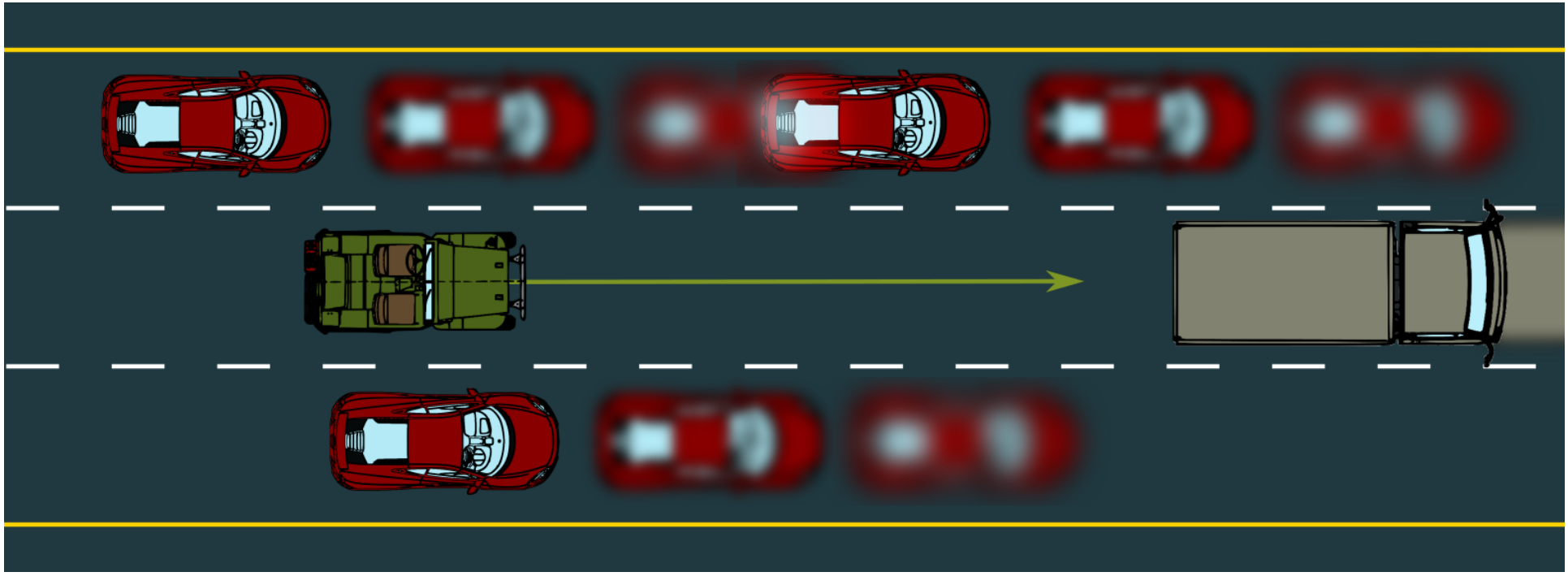
PATTERN PLANNING

PATTERN RANKING

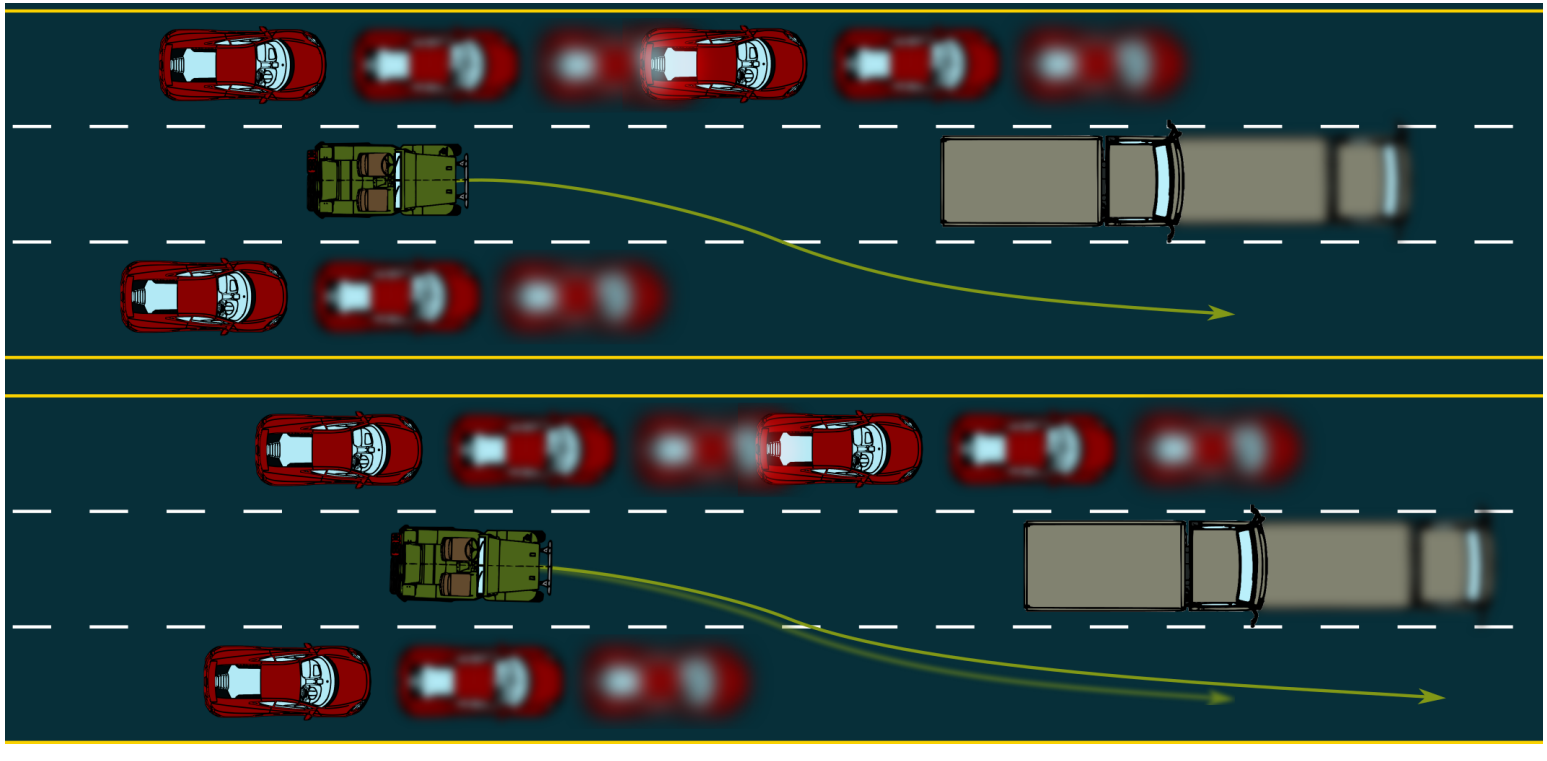
- Define a maneuver pattern for each neigboring gap.
- At steady state, consider the pattern from the previous planning cycle first.
- At steady state, consider the lane-keeping pattern next.
- At steady state, consider lane-change maneuvers in order by estimated safety.
- If the behavioral planner commands a lane change, consider those maneuvers first.



PATTERN PLANNING
LANE-KEEPING EXAMPLE

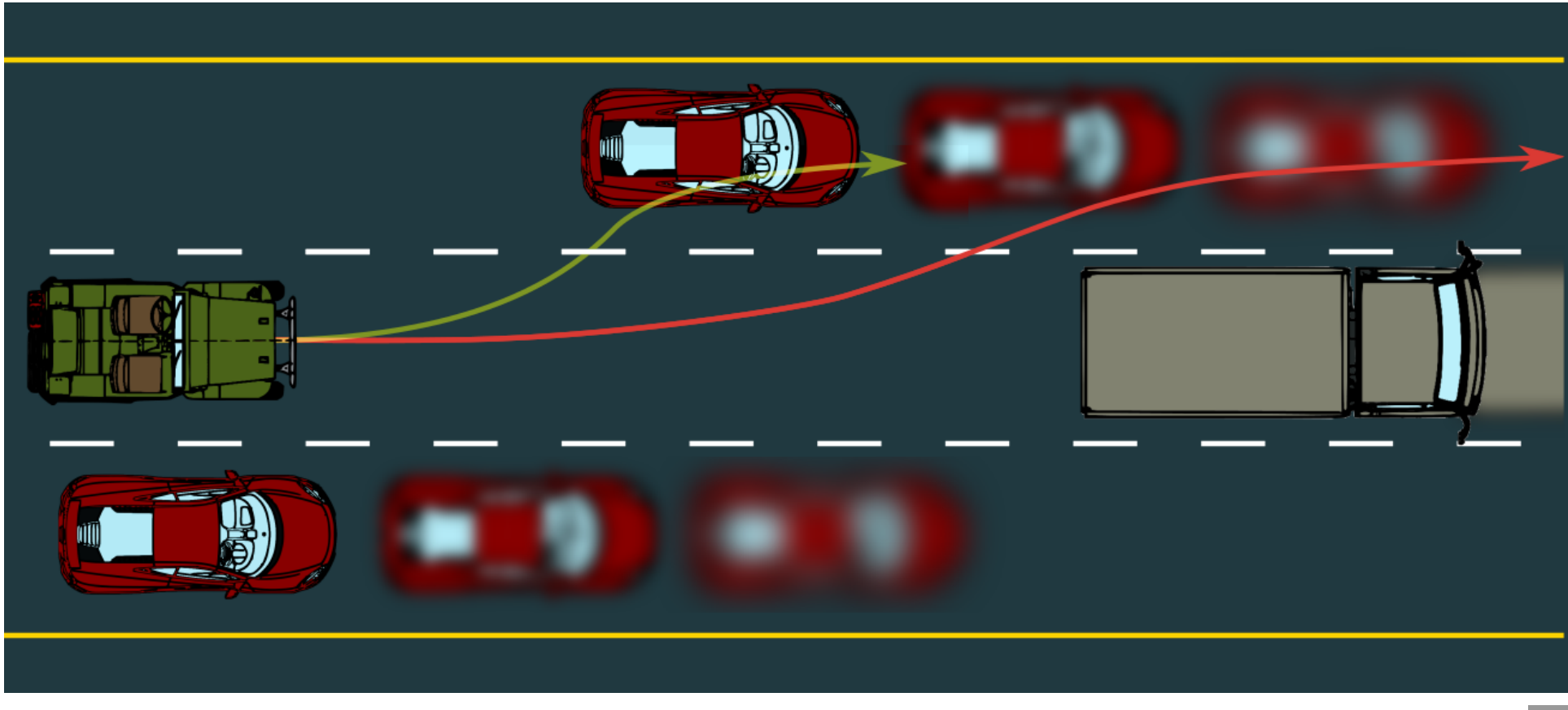


PATTERN PLANNING
HYSTERESIS EXAMPLE



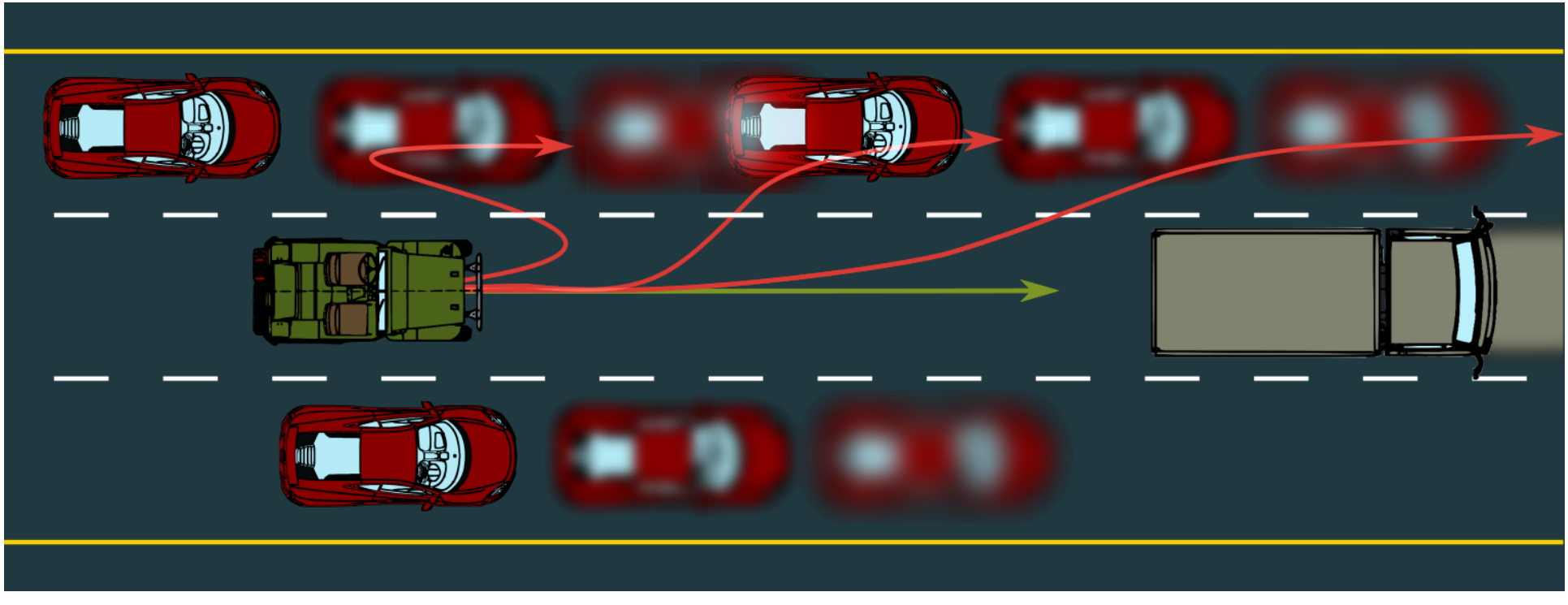
PATTERN PLANNING

SUCCESSFUL LANE CHANGE EXAMPLE



PATTERN PLANNING

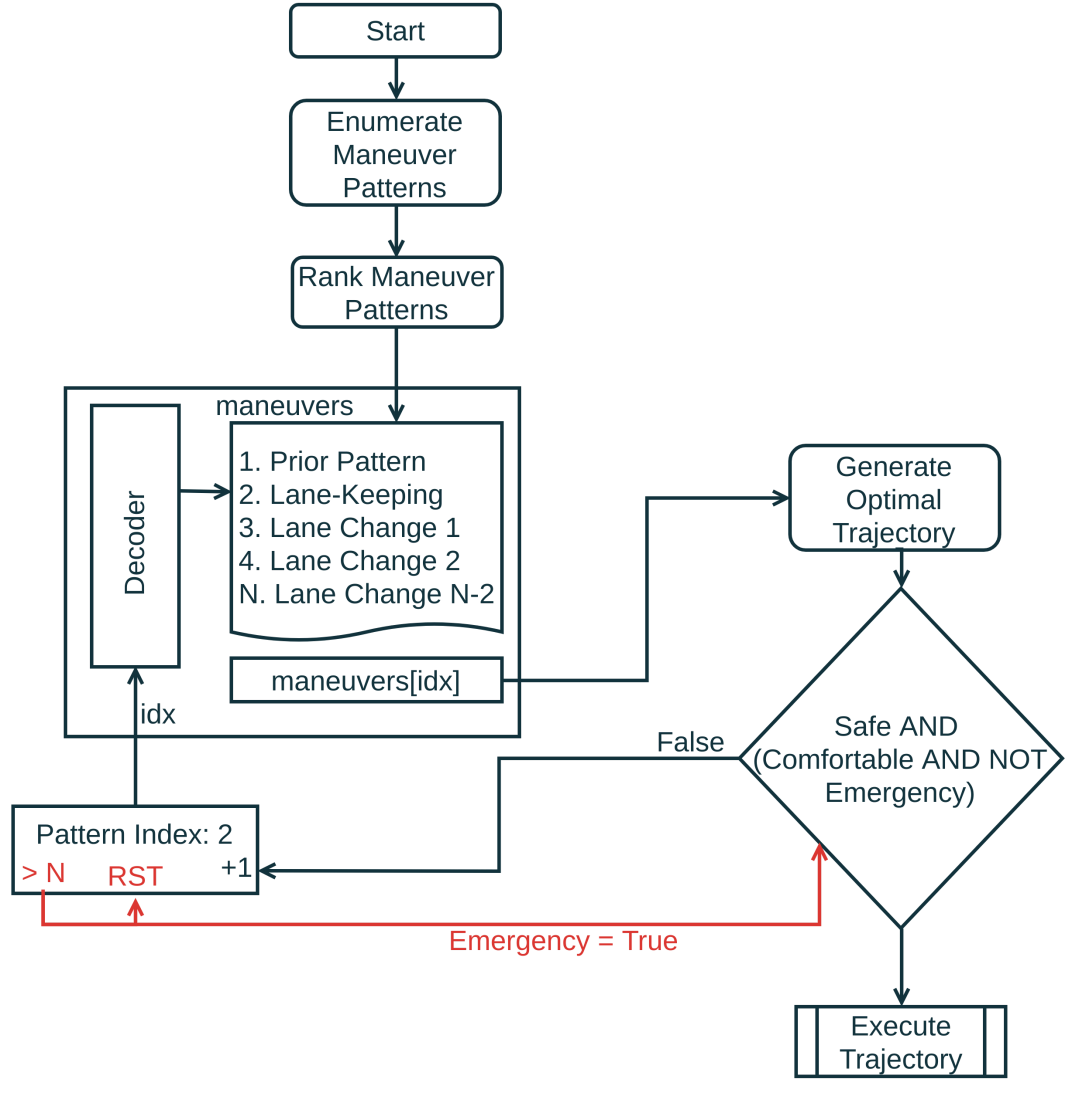
FAILED LANE CHANGE EXAMPLE



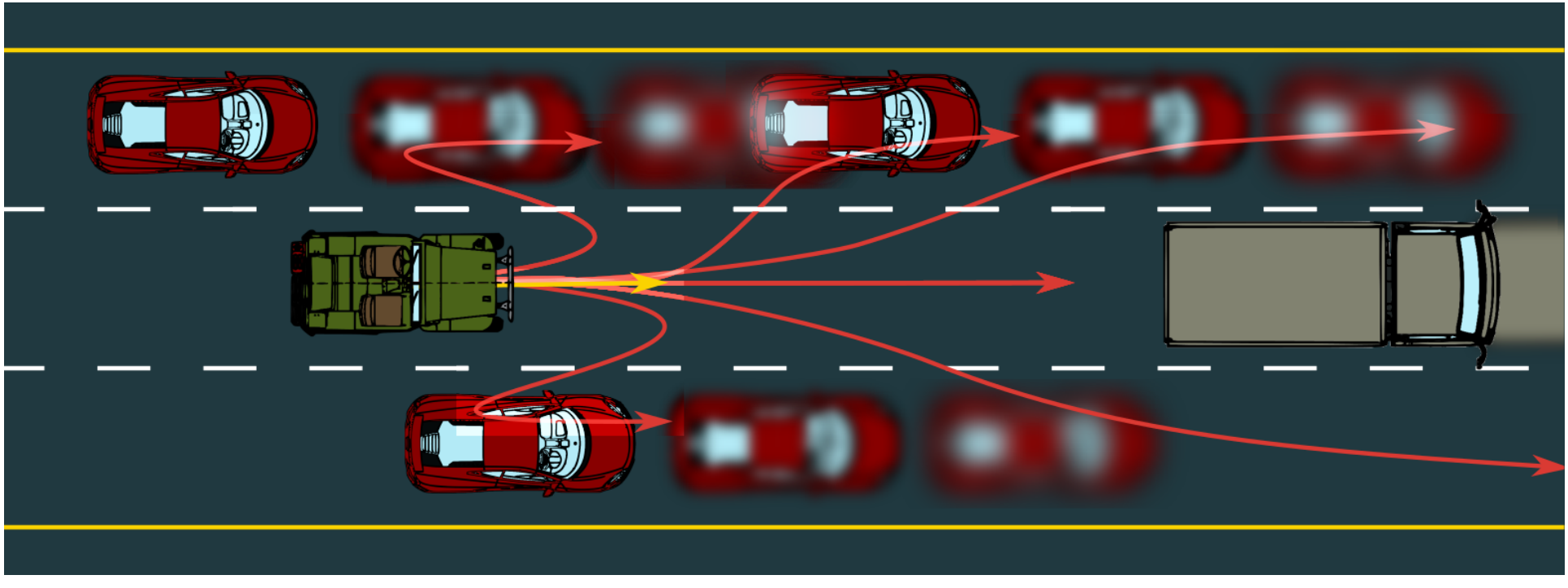
PATTERN PLANNING

EMERGENCY RESPONSE

- If all maneuvers are infeasible, remove the comfort requirement.
- Prioritize emergency braking maneuvers.
- Re-evaluate the maneuver pattern list and immediately execute the first safe trajectory that is found.



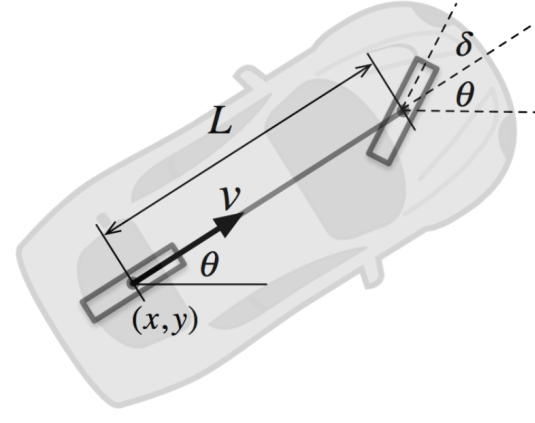
PATTERN PLANNING
EMERGENCY EXAMPLE



TRAJECTORY GENERATION DETAILS

DETAILS KINEMATIC BICYCLE MODEL

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) = \begin{bmatrix} v \cdot cos(\theta) \\ v \cdot sin(\theta) \\ v \cdot \kappa \\ u_k(t) \\ a \\ u_{\dot{a}}(t) \end{bmatrix}$$



$$\kappa = \frac{tan(\delta)}{L}$$

$$\boldsymbol{x} = \begin{bmatrix} x & y & \theta & \kappa & v & a \end{bmatrix}^\intercal \quad \boldsymbol{u} = \begin{bmatrix} \dot{\kappa} & \dot{a} \end{bmatrix}^\intercal$$

DETAILS PARAMETRIC CONTROLS

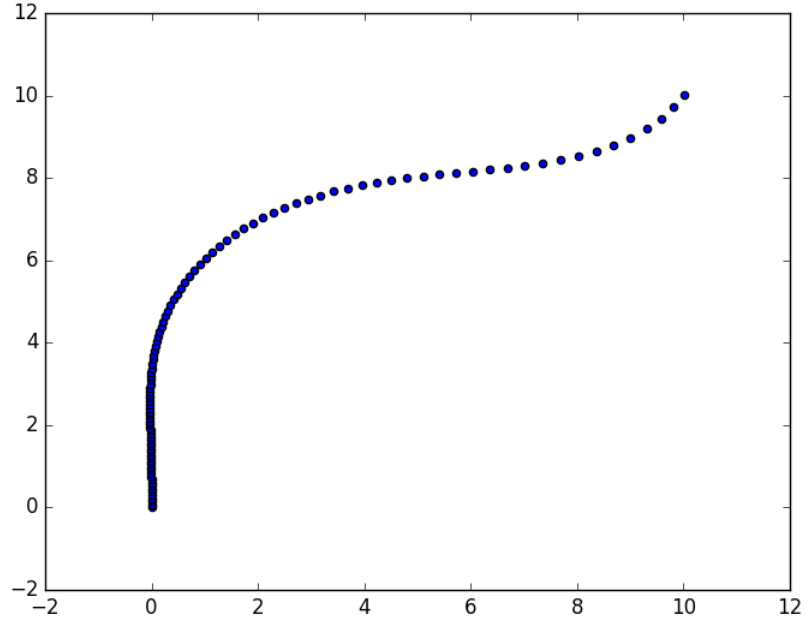
Parameterize the controls $\boldsymbol{u}(t) = \begin{bmatrix} \dot{\kappa}(t) & \dot{a}(t) \end{bmatrix}^\mathsf{T}$ as polynomials defined on an interval $0 \leq t \leq T_f$.

$$\boldsymbol{u}(t) = \begin{bmatrix} c_0 + c_1 t + c_2 t^2 + \ldots + c_{n-1} t^{n-1} \\ d_0 + d_1 t + d_2 t^2 + \ldots + d_{m-1} t^{m-1} \end{bmatrix}^\mathsf{T}$$

DETAILS FORWARD SHOOTING

Given a control sequence parameterized by $\begin{bmatrix} c_0 \dots c_{n-1} & d_0 \dots d_{n-1} & T_f \end{bmatrix}^\intercal$ and an initial state $\mathbf{X_s}$, the vehicle state can be found by numerically integrating the kinematic equations:

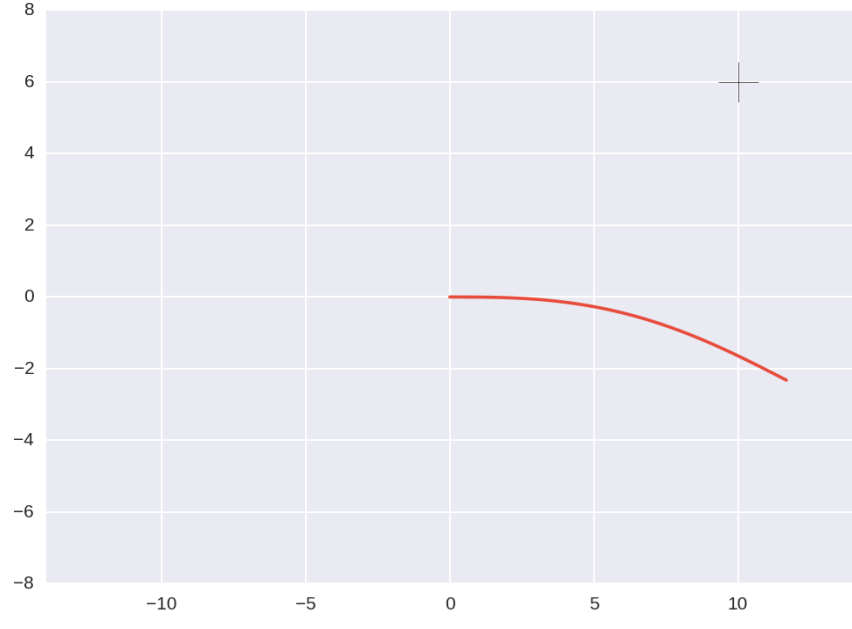
$$\mathbf{X}(t) = \mathbf{X_s} + \int_0^t \mathbf{f}(\mathbf{x}(t'), \mathbf{u}(t'), t')dt'$$



$$\mathbf{X_s} = \begin{bmatrix} 0 & 0 & \frac{\pi}{2} & 0 & 1 & 0 \end{bmatrix}^\intercal$$
$$\dot{\kappa} = 0.078 - 0.123t + 0.017t^2 + 0.001t^3$$
$$\dot{a} = -0.094 + 0.143t + 0.029t^2 - 0.009t^3$$
$$0 \leq t \leq 7.478$$

DETAILS PATH OPTIMIZATION

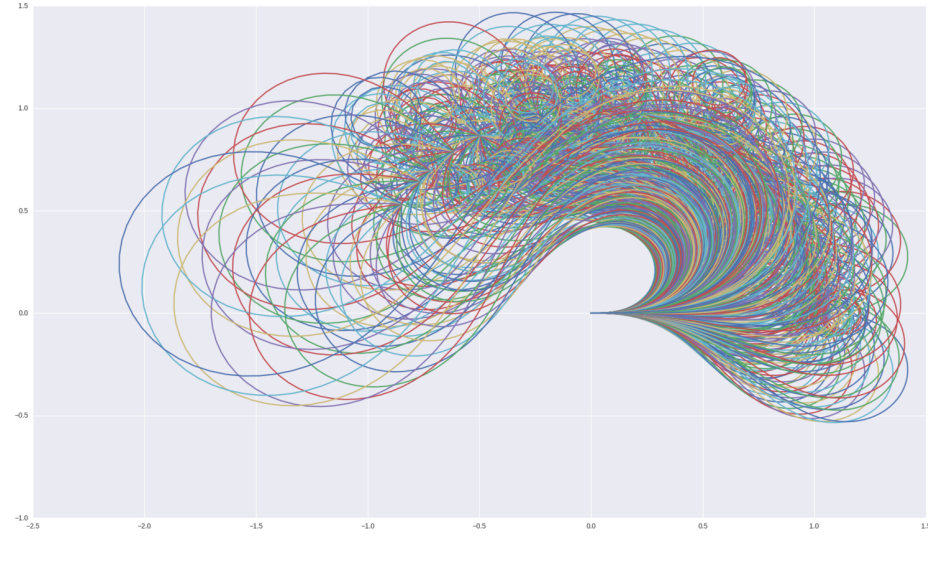
1. Set $\mathbf{u}_{\dot{a}}(t) = 0$
2. Set $T_f = R(1 + \frac{\theta^2}{5})$
3. Select coefficients of $\mathbf{u}_{\dot{a}}(t)$ to satisfy endpoint heading and curvature constraints.
4. Use Newton's Method with the cost function $(\mathbf{X}(T_f) - \mathbf{X_e})^\intercal \mathbf{D}(\mathbf{X}(T_f) - \mathbf{X_e})$ to iteratively move the trajectory endpoint to the desired position $\mathbf{X_e}$ by perturbing T_f and $c_0 \ldots c_{n-1}$.



DETAILS PATH CACHING

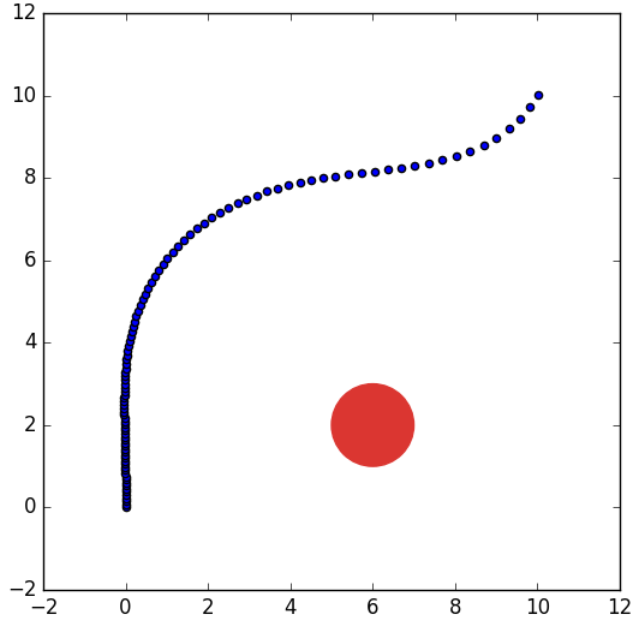
- Paths can be scaled to the unit circle and cached for future use.

- At runtime, scale the problem to the unit circle and select the most similar path. Scale the cached parameters to the size you need and use them to warm-start the optimization.



DETAILS TRAJECTORY OPTIMIZATION

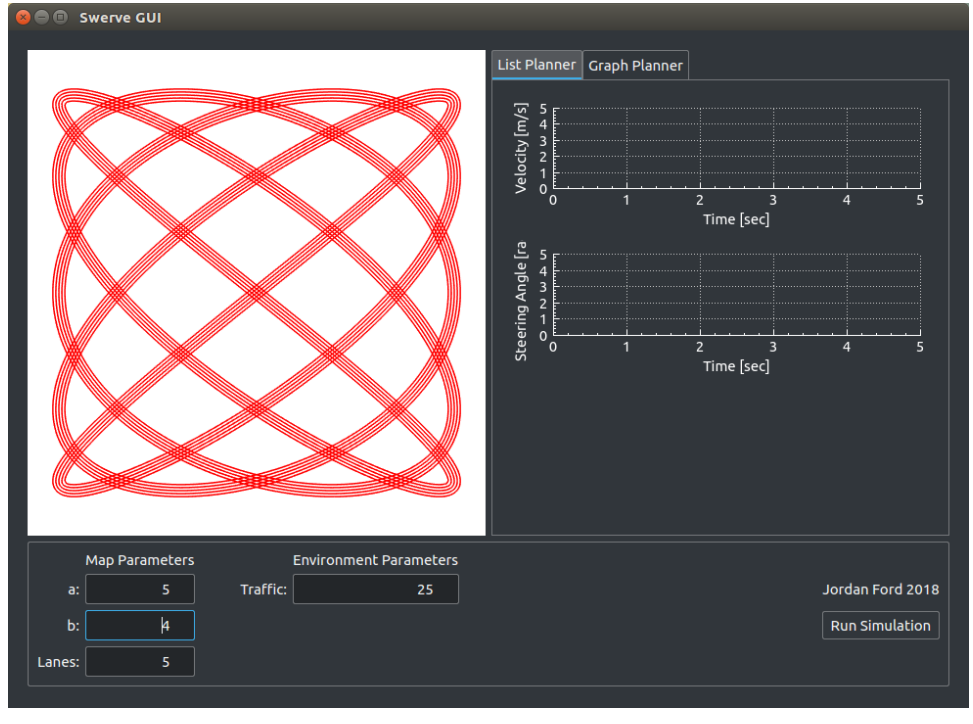
- Use the procedure from the previous slide to generate a path without regard for obstacles, velocities, or accelerations.
- Reoptimize with initial parameters $\left[c_0 \ldots c_{n-1} \quad 0 \ldots 0 \quad T_f \right]$.
- Use a cost function that measures the following path costs in addition to endpoint state matching costs.
  - min. distance to an obstacle
  - max. lateral acceleration
  - max. deviation from the road centerline
  - etc.



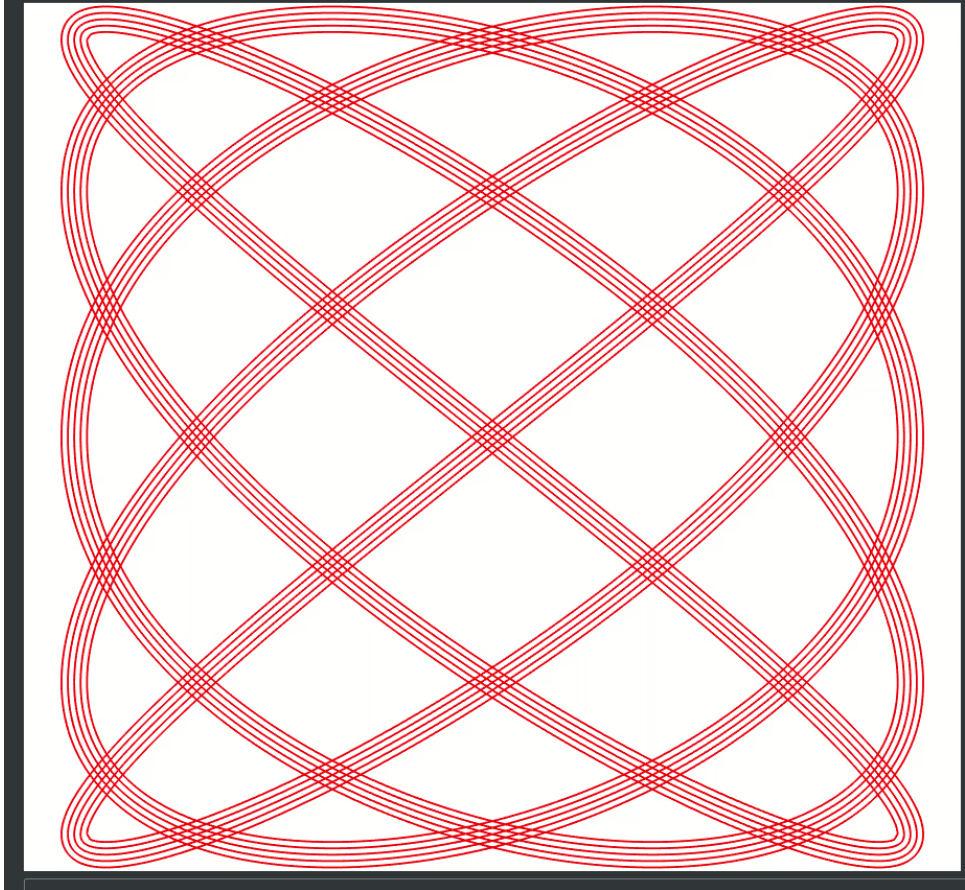
PLANNER EVALUATION

EVALUATION SWERVESIM

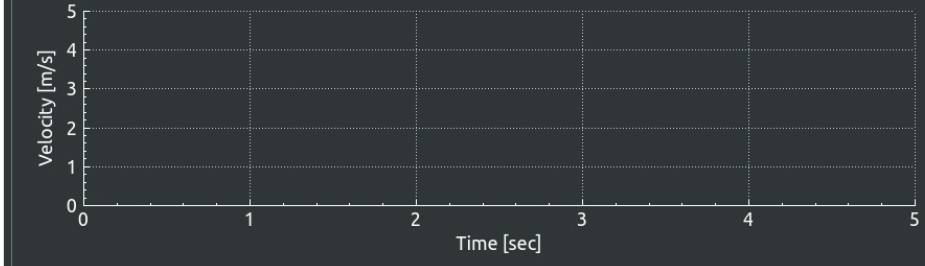
- Comparing Planners is difficult because there is no universally applicable objective standard for trajectories.
- SwerveSim places two or more vehicles with different planning algorithms in otherwise identical vehicles in an identical traffic-filled world.
- Vehicles under test are not aware of each other. They are competing to incur the lowest cost as they traverse the SwerveSim track.
- SwerveSim gives the user control of the cost function used to score the planners.
  - Available cost terms will include common trajectory cost terms as well as computational cost metrics.

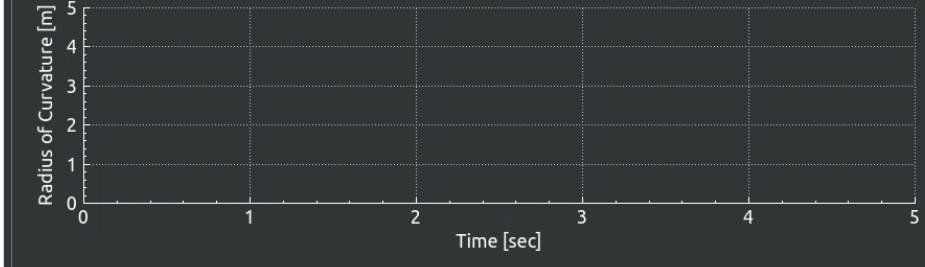


EVALUATION SWERVESIM



List Planner | Graph Planner | Statistics





Map Parameters

a: 5

b: 4

Lanes: 5

Environment Parameters

Traffic: 25

Jordan Ford 2018

Run Simulation

EVALUATION VEHICLE TESTING

Testing on GM's autonomous Chevy Volts is tentatively scheduled for early June.

CONCLUSION

CONCLUSION

1. Motion planning is expensive - time, money, power.
2. Pattern Planning is an algorithm to reduce planning expense.

3. Quantitative comparison of planner performance is difficult.

4. SwerveSim allows users to compare algorithms with their own cost function.

CITATIONS

1. A. Kelly, *Mobile robotics: mathematics, models, and methods*. New York, NY: Cambrige Univ. Press, 2013.
2. M. McNaughton, C. Urmson, J. M. Dolan, and J. W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," *in 2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4889–4895.
3. B. Paden, M. p, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, March 2016.
4. T. Gu, J. M. Dolan, and J. W. Lee, "Runtime-bounded tunable motion planning for autonomous driving," in 2016 IEEE Intelligent Vehicles Symposium (IV), June 2016, pp. 1301–1306.