Types of trajectory Models in Autonomous Vehicle

Made by : Mohammed Balkhair

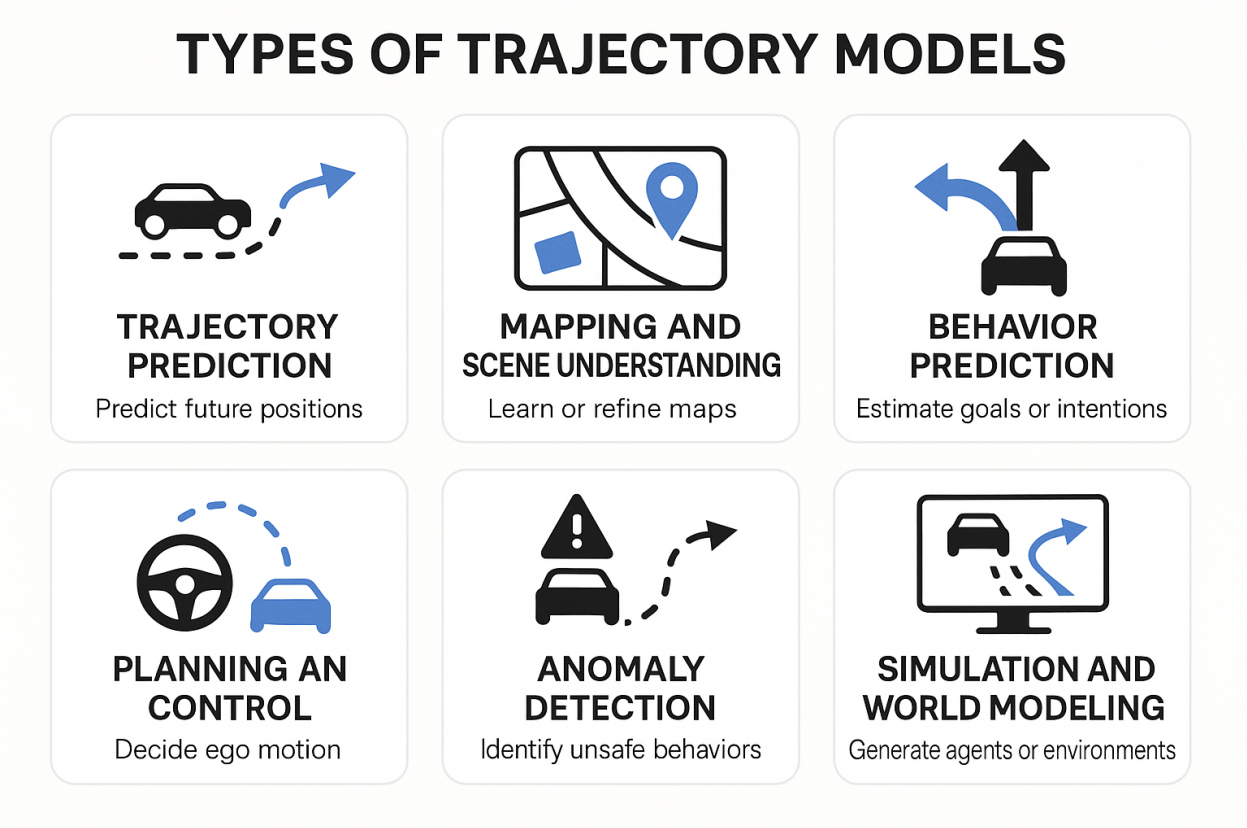

Table of Contents

Types of Trajectory Models in Autonomous Vehicle Research *3*

1. Trajectory Prediction *3*

about this type 3

Problems it solves 3

How it works 4

Notable Models 4

2. Mapping and Scene Understanding *5*

about this type 5

Problems it solves 5

How it works 5

Notable Models/Approaches 6

3. Behavior Prediction *7*

about this type 7

Problems it solves 7

How it works 7

Notable Models/Approaches 8

4. Planning and Control *9*

About this type 9

Problems it solves 9

How it works 10

Notable Models/Approaches 11

5. Anomaly Detection *12*

about this type 12

Problems it solves 12

How it works 13

Notable Models/Approaches 13

6. Simulation and World Modeling *14*

about this type 14

Problems it solves 15

How it works 15

Notable Models/Approaches 16

Conclusion **17**

Refrences ***17***

1. Trajectory Prediction **17**

2. Mapping & Scene Understanding **18**

3 . Behavior Prediction **18**

4. Planning & Control **18**

5. Anomaly Detection **19**

6. Simulation & World Modeling **19**

Types of Trajectory Models in Autonomous Vehicle Research

Autonomous vehicles (AVs) rely on a variety of models and algorithms to understand their environment and plan safe trajectories. This guide explains six key categories of trajectory-related models in AV research. Each section provides an overview of the category's purpose in the AV context and highlights 2–3 notable models or approaches from academia or industry. We assume the reader has intermediate machine learning knowledge but is new to autonomous systems, so we define advanced terms as needed.

1. Trajectory Prediction

about this type

Trajectory prediction models enable an AV to forecast the future paths of surrounding agents (other vehicles, pedestrians, cyclists, etc.) over a short horizon. In dynamic traffic, predicting what others will do next is crucial for the AV to make safe decisions. For example, an AV might estimate the probability of a nearby car merging into its lane or a cyclist making a left turn . Accurate predictions help avoid collisions by informing the planning module of likely future obstacles or conflicts.

Problems it solves

These models address the uncertainty and complexity of road user behavior. Human-driven agents can be unpredictable, and multiple agents may interact (e.g. a car yielding to a pedestrian). Trajectory prediction systems must handle multi-agent interactions, account for social norms (like right-of-way rules), and produce multi-modal outputs (predicting

several possible futures, since human behavior is not deterministic). A key challenge is balancing realism and safety: naive models might assume everyone moves straight at constant speed, whereas more advanced ones incorporate context like maps and agent interactions.

How it works

Early approaches used physics-based models (e.g. constant-velocity or kinematic models) for short-term predictions. Modern approaches leverage machine learning to capture patterns from data. They often take as input the past trajectories of agents and sometimes a map of the surroundings. The output is a set of possible future trajectories for each agent, often with probabilities. Many models use recurrent neural networks (RNNs) or sequence modeling to encode motion over time, and incorporate interaction layers or graph neural networks to model agent-agent influences.

Notable Models

- **Social LSTM** (Alahi et al., 2016): A pioneering deep learning model for human trajectory forecasting. It uses a Long Short-Term Memory network (LSTM) for each agent and introduces a social pooling layer that allows LSTMs for nearby agents to share information. This way, the model learns social behaviors like collision avoidance and grouping. Social LSTM demonstrated that a data-driven approach can learn human navigation conventions (e.g. keeping personal space) better than hand-crafted rules like the “social force” model.

- **Social GAN** (Gupta et al., 2018): An extension using Generative Adversarial Networks. It produces multi-modal trajectory predictions by generating many plausible futures and using an adversarial loss to ensure realism. A variety loss encourages the model to output a diverse set of trajectories, addressing the uncertainty in human behavior. Social GAN was notable for predicting socially acceptable trajectories (e.g. not colliding with others) in crowded scenes by training a generator against a discriminator that learns to distinguish real vs. fake trajectories.

- **VectorNet** (Gao et al., 2020, Waymo): A graph neural network-based approach for vehicle behavior prediction. VectorNet represents map elements (lane lines, road boundaries, crosswalks) and agent histories as sequences of points (polylines) and builds a hierarchical graph. In the first stage it encodes each polyline’s shape, and in the second stage it uses a global interaction graph to capture relationships between moving agents and map elements. By processing the scene in this vectorized form instead of pixels, it achieves accurate predictions with less compute . VectorNet

was deployed to help the Waymo Driver anticipate complex maneuvers like merges or unprotected turns by learning from real driving data.

Other examples: Trajectron++ (Ivanović & Pavone, 2019) which uses a graph-structured recurrent model for multi-agent trajectories, DESIRE (Lee et al., 2017) which uses conditional variational auto-encoders to propose and refine trajectory samples, and various transformer-based predictors in recent literature. Each advances the ability to handle interaction and the inherent uncertainty (outputting a distribution of possible futures rather than one deterministic path). Overall, trajectory prediction is an active research area, bridging computer vision, time-series modeling, and social behavior understanding.

2. Mapping and Scene Understanding

about this type

This category focuses on building models of the static environment and comprehending the overall scene context. In autonomous driving, "mapping" usually refers to creating or using detailed maps of the surroundings (including road geometry and semantics), while "scene understanding" refers to interpreting sensor data (camera images, LiDAR point clouds, etc.) to identify lanes, traffic signs, free space, and objects. Essentially, it gives the AV a semantic and geometric understanding of "what is where" around the vehicle.

Problems it solves

Before an AV can plan a trajectory, it must know the road layout and locate drivable space. High-definition (HD) maps provide a prior model of the world, including lane configurations, road boundaries, crosswalks, traffic lights, etc. This is crucial for contextualizing trajectories – for instance, knowing there's a sharp curve ahead or an upcoming intersection influences how the vehicle should move. Scene understanding helps in obstacle avoidance (detecting other cars, pedestrians, debris) and ensuring the planned path obeys traffic rules (staying in lane, stopping at stop lines). Without accurate mapping and perception, an AV cannot reliably choose a safe trajectory because it might miss important constraints or hazards.

How it works

There are two complementary approaches:

- **Prior maps**: Many AVs use HD maps built through extensive surveying. These maps contain semantic layers (roads, lane connectivity, sign locations). The vehicle localizes itself in this map, then augments it with live sensor data to account for

current obstacles or changes. Maintaining these maps is challenging due to scale and changes (construction, new traffic patterns) . Research is ongoing into automated map construction to reduce manual effort.

- **Online mapping (SLAM) and perception**: Simultaneous Localization and Mapping (SLAM) algorithms let the vehicle build a map on the fly using sensors, while also keeping track of its own position . Visual SLAM (using cameras) and LiDAR SLAM (using 3D laser scans) can produce point cloud maps of the environment. In parallel, scene understanding uses deep learning to label the environment: CNNs or Transformers identify drivable road area, traffic signs, or segment the image into classes (road, sidewalk, vehicles, etc.). Lidar-based algorithms cluster point clouds to detect obstacles and free space. The result is a bird's-eye-view model of the scene with static elements (road geometry) and dynamic objects, which is then used by prediction and planning modules.

Notable Models/Approaches

- **ORB-SLAM (2015) and LOAM (2014**): Classic SLAM systems. ORB-SLAM (for Oriented Fast and Rotated BRIEF features) is a visual SLAM that tracks feature points in camera images to build a 3D map and estimate pose. LOAM (Lidar Odometry and Mapping) is a LiDAR-based method that splits the task into odometry (short-term motion estimation) and mapping (refining the map), achieving accurate real-time 3D maps. These have been fundamental in enabling an AV to localize itself and map the environment without a prior map, though they may struggle in dynamic (moving objects) settings.

- **HD Maps and Semantic Mapping**: Companies like HERE and Waymo create HD maps containing lane-level detail and semantic features relevant to driving (curbs, stop lines, speed limits). HD maps allow an AV to "know" road rules and geometry beyond its sensors' line of sight. For scene understanding, deep networks such as HDMapNet (Li et al., 2021) have emerged to build these maps on the fly. HDMapNet takes camera images (and optionally LiDAR) to predict a bird's-eye-view semantic map around the vehicle. It outputs vectorized elements like lane boundaries and crosswalks in real-time, which can supplement or update static HD maps. This reduces reliance on fully pre-mapped roads and handles changes in the environment dynamically.

- **Semantic Segmentation and Object Detection**: These are perception models that feed into scene understanding. For example, U-Net or PSPNet for segmenting road vs. off-road areas, or YOLO (You Only Look Once) and Faster R-CNN for detecting vehicles and pedestrians in camera images. On the LiDAR side, models like PointPillars and CenterPoint detect 3D objects from point clouds. While not "maps" per se, they populate the world model with dynamic actors and obstacles. Another example is Lift-Splat-Shoot (Philion & Fidler, 2020), which lifts image features into

3D space and “splat” them into a bird’s eye grid to predict semantics like drivable space – a form of scene understanding critical for trajectory planning in urban streets.

Key takeaway: Mapping and scene understanding modules give the AV awareness of static road structure and dynamic agents. This awareness forms the canvas on which trajectory predictions and planning operate. Without an accurate map or scene interpretation, even the best trajectory planner could fail (e.g., planning through a concrete divider it didn’t know was there). Research continues in making mapping more scalable and scene understanding more robust to changes so that AVs can handle complex, changing environments.

3. Behavior Prediction

about this type

While trajectory prediction (Section 1) gives precise future positions, behavior prediction is about inferring higher-level intentions or maneuvers of agents. In simpler terms, it answers questions like: Is the car ahead intending to change lanes? Will that pedestrian likely start crossing the street? It often involves classifying or predicting discrete actions/states (e.g., turn vs. go straight, yield vs. continue) or understanding the latent goals of other road users. This is closely related to trajectory forecasting, but works at an intentional or semantic level of behavior rather than exact coordinates.

Problems it solves

Behavior prediction helps an AV plan proactively. Knowing what another agent plans to do (not just where they will be) provides context for safety and efficiency. For example, if the AV predicts a nearby car will yield at a merge, it can proceed; if the car will cut in aggressively, the AV should be ready to slow down. Similarly, predicting that a pedestrian is about to jaywalk triggers a different response than if the pedestrian will stay on the curb. This category tackles the challenge of understanding implicit cues and decision processes of humans. It often requires integrating multimodal data: vehicle kinematics (speed, acceleration patterns), signals (turn indicators), and possibly vision-based cues (head orientation of pedestrians, eye contact, etc.). By anticipating behaviors, the AV can avoid reactive last-moment maneuvers and instead adjust its trajectory in advance.

How it works

Traditional approaches used models from behavioral psychology or economics, like rule-based systems or game-theoretic models. For instance, the Intelligent Driver Model (IDM)

is a physics-based car-following model that can predict when a human driver would slow down or change lanes given a lead vehicle. Modern ML-based approaches either classify future maneuvers or perform intent inference. Some use sequence models or Hidden Markov Models on the history of an agent's states (position, velocity) to predict a maneuver change. Others use deep neural networks that take raw sensor inputs (camera images of a pedestrian, or Lidar tracks of a vehicle) and output probabilities of discrete behaviors (e.g., turn left, turn right, go straight). Behavior prediction can be seen as a coarse-level trajectory prediction – often feeding into trajectory predictors. In fact, some pipelines first predict a high-level behavior, then condition a trajectory forecast on that behavior.

Notable Models/Approaches

- **Lane-Change Intent Prediction (HMM/SVM models):** A common focus is predicting if a neighboring vehicle will change lanes. Xiong et al. (2018) combine a Support Vector Machine (SVM) to classify the intent to leave the lane vs. stay based on the vehicle's trajectory, then use a Gaussian Mixture Hidden Markov Model (GM-HMM) to predict if that lane change could lead to a collision. The SVM acts as an intent detector and the HMM forecasts the outcome, illustrating a hybrid of classification and probabilistic modeling. Other works use purely HMM or Bayesian networks on time-series of a driver's steering and speed to recognize maneuvers (lane change, braking, etc.) a few seconds before they occur.

- **IntentNet (Casas et al., 2018**): An example of a deep learning approach. IntentNet learns to predict drivers' intentions directly from raw sensor data (e.g., LiDAR and camera). It produces both a discrete intention (like turn vs. go straight) and a set of possible trajectories. By using a neural network, it can fuse visual cues (brake lights, turn signals, road context) with motion history. This was one of the early networks to integrate perception and intention prediction in a single end-to-end model, demonstrating how rich sensor data can improve understanding of what another vehicle will do next.

- **M2I (Sun et al., 2022, MIT):** A modern approach that explicitly models interactions to predict multi-agent behavior. M2I stands for "Multi-agent to Interactive", and it breaks the problem into two parts: First, a relation predictor guesses the relationship between each pair of agents – namely who has right-of-way and who will yield. Next, given a predicted leader-yield relationship, it uses separate trajectory predictors for the passer (agent with right-of-way) and the yielder, conditioning the yielder's prediction on the passer's predicted behavior. By factoring in social dynamics (like yielding), M2I achieved state-of-the-art accuracy on the Waymo Open Dataset, even outperforming Waymo's own model at the time . This

underscores that understanding behaviors like yielding, aggressiveness, caution is key to realistic trajectory forecasts.

Other examples: Pedestrian intent prediction models that use pose or gaze (e.g., a pedestrian making eye contact with the AV might indicate intent to cross), and reinforcement learning or game-theoretic models that predict actions by modeling the interaction as a game (the AV and human drivers influencing each other). In industry, behavior prediction is often folded into the planning stack: companies simulate "agents" with certain behaviors and see how the AV should respond. The take-home point is that behavior prediction adds a semantic layer of understanding on top of raw trajectory paths – it's about the why and what of agent actions, not just the where. This improves the AV's ability to plan safe and courteous maneuvers in a human-populated environment.

4. Planning and Control

About this type

Planning and control models are responsible for deciding the AV's own trajectory and executing it safely. If we think of an autonomous vehicle's software as a pipeline, planning is the stage that takes in all the information (maps, predictions, etc.) and outputs a sequence of actions or a path for the vehicle to follow. Control is the low-level execution: steering, accelerating, and braking to follow that planned trajectory. In essence, this category is about how the AV computes a trajectory for itself (planning) and how it makes the vehicle follow that trajectory (control).

Problems it solves

This is the core of "driving". The planning system must solve a complex optimization problem: find a route and trajectory that gets the vehicle to its goal safely, efficiently, and comfortably. It needs to avoid obstacles, obey traffic laws, and account for dynamic changes (like a suddenly swerving car). The control system then must ensure the actual vehicle motion tracks the planned path within the physical limits of the car (no skidding, maintaining passenger comfort). **Challenges include**:

- **Trajectory planning under uncertainty:** dealing with the unpredictability of other agents (the plan should be safe even if another driver does something unexpected).
- **High-dimensional search:** the continuous space of possible accelerations or steering angles is huge, so planning algorithms must efficiently search or optimize in this space.

- **Dynamic feasibility:** the planned path must respect vehicle kinematics and dynamics (e.g., a car cannot instantaneously change direction or exceed traction limits).

- **Real-time operation:** planning and control must happen in real-time (typically 10-100 milliseconds for control loops, and a few times per second for re-planning) to react promptly to the environment.

How it works

AV planning is often hierarchical:

- **Route Planning (Mission Planning):** high-level, uses road network to choose a route from A to B (like a GPS navigator). We won't focus on this as it's more like standard navigation.

- **Behavior Planning**: mid-level, deciding maneuver or behavior (e.g., "change lane to pass slow vehicle" or "yield to pedestrian"). This is sometimes rule-based or uses a state machine or reinforcement learning to choose a high-level action.

- **Trajectory Planning (Motion Planning**): low-level, computing the actual geometric path and speed profile the vehicle will take over the next few seconds (typically planning a trajectory 5-10 seconds into the future). This is where algorithms and models come in heavily.

- Trajectory planning algorithms can **be search-based** or **optimization-based** (or **hybrids**):

1. **Search-based**: For example, Hybrid A* (a version of the A* algorithm adapted for car kinematics) finds a path on a grid or graph that represents possible motions, ensuring the path is drivable. State lattice planners pre-compute a library of motion primitives (feasible short trajectories) and search through those for a sequence that avoids obstacles.

2. **Optimization-based**: These treat planning as a mathematical optimization problem. A cost function encodes desired behavior (stay in lane, keep distance, smoothness) and constraints ensure safety (no collisions, within road boundaries). The planner then optimizes this cost. Model Predictive Control (MPC) is a common framework: it formulates a trajectory as a sequence of control inputs over a horizon and uses a solver to minimize cost subject to vehicle dynamics and obstacle constraints. Gradient-based methods like

Iterative LQR (iLQR) and Differential Dynamic Programming (DDP) also fall here, iteratively improving a trajectory solution by approximating dynamics and costs .

Notable Models/Approaches

- **Model Predictive Control (MPC):** MPC is widely used in both academia and industry for AV planning and control. In MPC, at each time step the planner solves an optimization (often a quadratic or nonlinear program) to find the best control action sequence for the next few seconds, then applies the first action and repeats. MPC can handle multi-objective costs (like follow the lane center, but also keep a safe distance from others) and constraints (like speed limits, or "don't hit obstacles" enforced via safety margins). For example, an MPC-based planner can incorporate predicted trajectories of others as moving obstacles and optimize the AV's path accordingly . The advantage is the resulting trajectory is smooth and dynamically feasible by design. The challenge is computational: solving an optimization in real-time, especially if nonlinear (for car dynamics) or if there are many obstacles, can be difficult. Nevertheless, with modern solvers and simplifications, MPC is a go-to approach. Many commercial AVs use MPC for lateral and longitudinal control due to its balance of performance and flexibility.

- **Sampling-based Planners (RRT / RRT / Hybrid A):**These planners build many candidate paths and select one that fits criteria. RRT (Rapidly-Exploring Random Tree) randomly searches the space by growing tree branches until it finds a path around obstacles, and RRT* optimizes it to be shorter. They guarantee finding a path if one exists (given enough time) and can handle complex environments. Hybrid A* is designed for car kinematics: it can search a grid of positions and headings, using discrete moves that approximate driving (including turning radius). These were used in early AVs (e.g., the Stanford Stanley car used A*-like planning for off-road paths). They ensure collision-free paths but might produce jerky or suboptimal trajectories if not combined with smoothing. Often a search-based approach provides a coarse path that is then refined by an optimizer for smoothness.

- **Behavioral Cloning and End-to-End Models:** Some models learn to plan by imitating human drivers. For instance, ChauffeurNet (Waymo, 2019) was an LSTM-based imitation model that took in rasterized images of the scene and output a planned path, essentially learning a planning policy from human driving examples. Similarly, Conditional Imitation Learning (Nvidia) took high-level commands (turn left, go straight) and directly output steering. These blur the line between perception and planning, using deep networks to directly compute trajectories or control signals. They can capture complex implicit behaviors (since they learn from what

human drivers actually did), but are hard to guarantee safety for all corner cases without explicit constraints.

- **Low-Level Control (PID, Stanley, Pure Pursuit):** Once a trajectory is planned, the control system must execute it. Classic controllers like PID (Proportional-Integral-Derivative) loops are used to maintain speed (throttle/brake control) or heading. For steering, methods like Pure Pursuit compute a steering angle by looking at a point on the desired path a short distance ahead and turning towards it. The Stanley controller (Stanford's 2005 DARPA challenge winner) is a specific method for lateral control that minimizes the cross-track error and heading error to the path. These classical controllers are simple and robust for keeping a vehicle on a given trajectory. In modern AVs, however, often a form of MPC handles both planning and tracking together (e.g., a trajectory is optimized such that it's inherently trackable, or a separate MPC for tracking adjusts the steering and acceleration continuously).

Key point: Planning and control algorithms ensure the AV's own trajectory is safe and optimal. They are the decision-makers of "how to drive". While some approaches are hand-crafted (e.g., rule-based behaviors or geometric planners), there is a trend toward more integrated and learning-based planning. Regardless of approach, planning must always respect safety constraints and be able to handle the worst-case outcomes (like emergency braking if a pedestrian jumps out). Achieving human-level driving smoothness and foresight, while guaranteeing safety, is an ongoing balancing act in AV research.

5. Anomaly Detection

about this type

Anomaly detection in the context of trajectories refers to identifying unusual or abnormal patterns in vehicle or pedestrian movements. This can apply to the behavior of other road users or the AV itself. Essentially, these models watch streams of trajectory data (positions over time) and try to flag when something "doesn't fit" the normal patterns learned. In autonomous driving, anomalies might mean a car behaving erratically (e.g., a drunk driver weaving), a pedestrian suddenly running onto the road, or even sensor glitches causing ghost obstacles. By detecting anomalies, the AV can trigger special safety responses (like be extra cautious or even disengage automated mode if sensors seem unreliable).

Problems it solves

The "long tail" of driving scenarios is incredibly diverse. An AV is trained on mostly normal driving data, but it must handle rare events (accidents, rule-breaking drivers, unexpected obstacles). An anomaly detection system acts as a guardian to catch things the main models (which expect normal behavior) might miss. For example, trajectory prediction models (Section 1) might assume reasonable driving; if a car goes against traffic or a cyclist

falls, those predictions could be very wrong. An anomaly detector would recognize that "this trajectory is anomalous" and alert the system. Another use is in system monitoring: if the AV's planned trajectory deviates strongly from its typical driving pattern (due to a controller fault), an internal anomaly detector might flag a possible hardware or software issue. In summary, it adds resilience by handling out-of-distribution events.

How it works

Anomaly detection is often unsupervised or self-supervised because by definition anomalies are rare and not well-represented in training data. **Common techniques**

- **Statistical models:** Define what "normal" ranges of behavior are (e.g., typical acceleration and turning rates for cars) and flag outliers. Even simple threshold-based systems (e.g., if a pedestrian's speed suddenly > X or a car's jerk (acceleration change) > Y, flag it) can catch extreme events.
- **One-class classification**: Train a model (like a One-Class SVM or autoencoder) on only normal data, so it learns a compact representation of normal trajectories. At runtime, if a new trajectory doesn't fit that representation (i.e., high reconstruction error in an autoencoder, or SVM outputs it as outside the boundary), it's labeled anomalous.
- **Sequence models for anomaly:** Use RNNs or graph neural networks to encode multi-agent trajectories and then detect anomalies via probability estimates. For instance, a model might learn the joint distribution of all vehicles' motions in typical driving; if a combination occurs that has very low probability under this distribution, it's an anomaly.

Notable Models/Approaches

- **Spatio-Temporal Graph Autoencoder (STGAE) for Trajectories**: Wiederer et al. (Mercedes-Benz, 2021) proposed a graph-based autoencoder to learn normal driving patterns in multi-agent trajectories . They represent each traffic agent as a node in a graph and model their interactions over time, encoding this into a latent vector (using an encoder neural network). By training on lots of driving data with no accidents, the autoencoder learns to reconstruct typical trajectories well. At test time, they perform a density estimation on the latent space; trajectories that fall into low-density regions (i.e. the autoencoder finds them hard to reconstruct, meaning they're unlike anything seen in training) are flagged as anomalies. This method can detect collective anomalies (e.g., a scenario where one car's movement is weird relative to others). Since real traffic anomalies (wrong-way drivers, etc.) are hard to find in public datasets, they even built a simulation to generate synthetic anomalies for evaluation .

- **Diffusion and Generative Models (DiffTAD, 2023)**: Newer approaches apply advanced generative models to anomaly detection. DiffTAD (Trajectory Anomaly Detection via Diffusion Models) trains a denoising diffusion model on normal vehicle trajectories . Diffusion models learn to gradually corrupt and then recover data; by learning the distribution of normal trajectories in this manner, the model can later evaluate how likely a given trajectory is by seeing how well it fits the learned distribution. If a trajectory cannot be "denoised" effectively by the model (i.e., it lies outside the manifold of normal behavior), it's an anomaly . This approach leverages the power of diffusion models (which have been successful in image generation) to model the complex distribution of driving patterns.

- **Rule-based Outlier Detection**: Aside from learning-based methods, many AV systems incorporate heuristic checks for safety. For example, if an agent's predicted trajectory violates physics (teleporting or accelerating unrealistically) due to sensor errors, the system can ignore that prediction as an anomaly. If the AV itself experiences a sudden sensor dropout or a critical subsystem failure, a watchdog might trigger with a fault code – essentially an anomaly detection in the vehicle's self-diagnostics. These aren't "trajectory models" in a learning sense, but they are part of the overall safety mechanism handling anomalous conditions.

Implications: Anomaly detection models contribute to the fail-safe behavior of an AV. When something highly unusual is detected, the AV might slow down, increase following distance, or even trigger a minimal risk maneuver (like pull over and stop) if the anomaly suggests the environment is too unpredictable or sensors are unreliable. By handling edge cases that lie outside normal driving data, anomaly detectors are an essential complement to the primary modules. The field is evolving – from simple thresholds to sophisticated deep models – to cover the ever-expanding edge cases as AVs encounter more real-world miles.

6. Simulation and World Modeling

about this type

This category refers to models that create and use simulated worlds or learned world representations to aid autonomous driving. Simulation provides a virtual environment to test and train AV trajectory models without real-world risk. World modeling, in a learning context, means the AV builds an internal model of how the world works – essentially a predictive model of the environment's dynamics – which can be used for planning or imagination of futures. These models might not directly output a trajectory, but they generate scenarios or context in which trajectories are evaluated. In short, simulation and

world modeling enable an AV to “practice” or envision scenarios beyond what it has directly experienced.

Problems it solves

Real-world testing for AVs is expensive, time-consuming, and potentially dangerous. Simulation addresses the need for scalable and safe testing, including rare events (simulating thousands of possible crash scenarios or extreme weather). It also helps in training data-hungry models by generating additional synthetic data. World modeling, on the other hand, is useful for prediction and planning: if the AV has a model of how the world might evolve in response to its own actions, it can do lookahead planning (a form of “imagination”). For example, a world model could allow the AV to predict sensor readings if it were to take a certain turn, effectively letting it evaluate different options without actually executing them. In reinforcement learning terms, this is akin to a “model-based” approach, where the agent uses a learned model to simulate outcomes. Both simulation and world models aim to cover the vast diversity of driving situations by generating or modeling those situations internally.

How it works

- **High-Fidelity Simulators:** These are physics-based engines (like computer games) that simulate vehicles, pedestrians, sensors, and environments. Examples include CARLA (open-source), LGSVL, Apollo Simulation, and commercial ones like ADAS RP or NVIDIA Drive Sim. They use 3D models of cities, with programmed agent behaviors (traffic cars following rules or scripts, pedestrian models, etc.). Trajectory models can be tested here by placing the AV in various traffic scenarios and checking if it plans and reacts properly. Simulation can also generate trajectory datasets by having virtual agents drive around.

- **Data-Driven Simulation (World Models):** Instead of relying on manual content creation, researchers are using data to learn simulation models. A world model in this sense is a neural network that takes the current state (and possibly an action from the AV) and predicts the next state or sensor output. This can be done at different levels of fidelity:

    1. Some world models predict raw sensor data (e.g., camera images of the next frames) given actions.

    2. Others predict high-level state, like positions of all agents in the next second.

- These models are often generative and may incorporate modern AI techniques like transformers or VAEs.

A prominent idea is to treat the driving environment as something you can learn a generative model for, capturing the joint behavior of all agents.

Notable Models/Approaches

- **GAIA-1 (Wayve, 2023):** A large-scale generative world model that creates realistic driving videos based on a brief context. GAIA-1 is a 9-billion-parameter model trained on video feeds from real driving in cities. Given a few seconds of real video and a text or action prompt, it can imagine possible futures as video. For example, from a clip of approaching an intersection, GAIA-1 might generate a video of a pedestrian suddenly crossing or the weather changing, consistent with what it has learned about driving scenarios. It uses an autoregressive transformer to predict video token sequences, and can incorporate prompts like "a car cuts in from the right" to generate that scenario. The remarkable thing is it isn't just replaying data – it learns rules of driving (like how vehicles and people typically move, or that sidewalks are for pedestrians) and can produce novel but plausible outcomes. This kind of model serves as a neural simulator, allowing infinite variations of events to test planning algorithms or train prediction models (especially for rare dangerous events that are hard to find in real data).

- **DriveDreamer (2022):** Another video-generation world model, trained on the nuScenes dataset. It uses inputs like HD maps and detected object boxes to better control the generation of future scenes. By conditioning on more structured input, DriveDreamer can generate specific outcomes for planning. For instance, an AV could use DriveDreamer to simulate "what if I take this lane change now – show me a likely outcome," and it would output a visual rollout of that scenario. This merges into the concept of imagination for planning: the AV effectively asks its world model to predict consequences of actions, then picks the action that leads to the best outcome.

- **Learning-based Traffic Simulation (TrafficGen, TrafficSim, BITS):** Instead of simulating sensor data, these focus on trajectories of multiple agents. TrafficGen (2021) and SceneGen (2021) are models that learn to generate diverse traffic scenarios – essentially sets of agent trajectories that obey driving norms without manual scripting. NVIDIA's BITS (Bi-Level Imitation for Traffic Simulation, 2022) is a model that learns to control multiple traffic agents realistically, capturing interactions. These allow creating large numbers of realistic trajectories for training or testing. They can be thought of as "world models" that operate at the level of agent motions in the world, rather than pixels.

- **Model-Based Reinforcement Learning for Driving:** In academic research, there's interest in training driving policies with fewer real interactions by learning a world model. One example is applying the World Models concept by Ha & Schmidhuber

(2018) or Dreamer (Hafner et al.) to driving. The idea is the AV (or virtual agent) first learns a model of the driving environment's dynamics (either from data or from a simulator). Then it uses that model to plan actions (e.g., via Monte Carlo tree search or gradient optimization in the imagined space). While not yet common in industry, such approaches could let an AV plan with foresight, considering many hypothetical futures quickly inside its neural network brain.

In practice: Companies extensively use simulation to validate autonomous driving software. Before activating a new trajectory planning algorithm on a real car, they run millions of virtual miles in simulation, including corner cases (like cut-ins, tire blowouts, dense traffic jams) to ensure safety. The more realistic the simulation/world model, the better it can prepare the AV for reality. World modeling research also contributes to understanding counterfactuals – answering "what if?" questions by simulating alternative outcomes. As AVs progress, simulation and learned world models will remain crucial for both development and real-time decision making (e.g., envisioning the outcomes of various maneuvers to choose the best one). They effectively broaden the experience of an AV beyond what it has physically encountered, which is key to robust performance.

Conclusion

The six categories above represent pillars of autonomous vehicle trajectory research. Trajectory prediction and behavior prediction equip the AV with foresight about others. Mapping and scene understanding ground the AV in its environment's reality. Planning and control translate goals into the AV's own driving trajectory. Anomaly detection adds a safety net for the unexpected, and simulation/world modeling expands testing and predictive imagination. Together, these trajectory models and methods allow an autonomous vehicle to navigate complex environments safely and efficiently. As the field advances, we see increasing integration between these components (e.g., prediction tightly coupled with planning, learned world models aiding both perception and decision-making). Understanding each component in depth, as we've outlined in this guide, provides a solid foundation for engaging with current AV research and development. Each area is rich with ongoing innovation, and improving any one of them brings us a step closer to safe, reliable self-driving systems.

Refrences

1. Trajectory Prediction

Alahi et al., 2016 – Social LSTM: https://arxiv.org/abs/1603.09419

Gupta et al., 2018 – Social GAN: https://arxiv.org/abs/1803.10892

Gao et al., 2020 (Waymo) – VectorNet: https://arxiv.org/abs/2005.04259

Lee et al., 2017 – DESIRE: https://arxiv.org/abs/1704.04394

Ivanović & Pavone, 2019 – Trajectron++: https://arxiv.org/abs/2001.03093

2. Mapping & Scene Understanding

Mur-Artal et al., 2015 – ORB-SLAM: https://arxiv.org/abs/1502.00956
Zhang & Singh, 2014 – LOAM: https://ieeexplore.ieee.org/document/6907030

Li et al., 2021 – HDMapNet: https://arxiv.org/abs/2107.06397

Philion & Fidler, 2020 – Lift-Splat-Shoot: https://arxiv.org/abs/2008.05711

3 . Behavior Prediction

Casas et al., 2018 – IntentNet: https://arxiv.org/abs/1806.03833

Sun et al., 2022 (MIT) – M2I: https://arxiv.org/abs/2202.11841

Xiong et al., 2018 – HMM+SVM for Lane Change Prediction:
https://ieeexplore.ieee.org/document/8432046

4. Planning & Control

Kalman & Mayne – Model Predictive Control (MPC): Classic control systems texts

Dolgov et al., 2008 – Hybrid A**: https://ieeexplore.ieee.org/document/4543480

Bojarski et al., 2016 – End-to-End Driving (NVIDIA): https://arxiv.org/abs/1604.07316

Bansal et al., 2018 (Waymo) – ChauffeurNet: https://arxiv.org/abs/1812.03079

5. Anomaly Detection

- Wiederer et al., 2021 (Mercedes-Benz) – STGAE: https://arxiv.org/abs/2111.07972
- DiffTAD (2023) – Trajectory Anomaly Detection via Diffusion Models: https://arxiv.org/abs/2303.07907

6. Simulation & World Modeling

GAIA-1 (Wayve, 2023) – Generative World Models for Driving: https://arxiv.org/abs/2305.16666

DriveDreamer (2022) – Video-based Simulation Model: https://arxiv.org/abs/2206.02069

BITS (NVIDIA, 2022) – Bi-Level Imitation Simulation: https://arxiv.org/abs/2210.03145

Ha & Schmidhuber, 2018 – World Models for RL: https://arxiv.org/abs/1803.10122

Hafner et al., 2020 – Dreamer: https://arxiv.org/abs/1912.01603