Autonomous Driving: Planning, Control & Other Topics

Jan 8th, 2018
Sahil Narang
University of North Carolina, Chapel Hill
Autonomous Driving: Main Components

Figure 2. A typical autonomous vehicle system overview, highlighting core competencies.
LIDAR Interference

- Probability of LIDAR interference is very low
  - LIDAR uses very focused light pulses
  - Broad range of “Pulse Repetition Frequency”
  - Most approaches fuse data from from various sensors

Autonomous Driving: Main Components

- Perception
  - collect information and extract relevant knowledge from the environment.

![Diagram of Autonomous Driving Components](image)

Figure 2. A typical autonomous vehicle system overview, highlighting core competencies.
Autonomous Driving: Main Components

Planning

Making purposeful decisions in order to achieve the robot’s higher order goals

Figure 2. A typical autonomous vehicle system overview, highlighting core competencies.
Autonomous Driving: Main Components

Control

Executing planned actions

Figure 2. A typical autonomous vehicle system overview, highlighting core competencies.
Structure

- Modeling a car
  - State Space
  - Kinematic constraints
  - Dynamic constraints
- Motion Planning
- Control
- Interaction with other Drivers
- Case Studies
Autonomous Driving: State Space

- “The set of attribute values describing the condition of an autonomous vehicle at an instance in time and at a particular place during its motion is termed the ‘state’ of the vehicle at that moment”
- Typically a vector with position, orientation, linear velocity, angular velocity
- **State Space**: set of all states the vehicle could occupy
Autonomous Driving: State Space

Examples:

10 3D space with velocity
   ✤ \((p_x, p_y, p_z, \theta_x, \theta_y, \theta_z, v_x, v_y, v_z, \omega_x, \omega_y, \omega_z)\)
   ✤ \((\ddot{p}, \ddot{\theta}, \ddot{v}, \ddot{\omega})\)

10 2D space with acceleration
   ✤ \((p_x, p_y, \theta, v_x, v_y, \omega, a_x, a_y, \alpha)\)
   ✤ \((\ddot{p}, \ddot{\theta}, \ddot{v}, \omega, \ddot{a}, \alpha)\)
Autonomous Driving: State Space

Examples:

- 2D space with blinker booleans
  \[ (\hat{p}, \theta, \hat{v}, \omega, bl_l, bl_r) \]

- State contains everything we need to describe the robot’s current configuration!

- Neglect some state variables when planning
Autonomous Driving: Holonomicity

“Holonomic” robots

- Holonomic system are systems for which all constraints are integrable into positional constraints.
- Holonomic system where a robot can move in any direction in the configuration space.
- Controllable DOF = total DOF

Examples:

- Omni-drive base
- https://youtu.be/9ZCUxXajzXs
Autonomous Driving: Holonomicity

- Cars are “non-holonomic” robots
  - Typically 3 values describing C-Space
    - x, y, \( \theta \)
  - 2 “kinematic” constraints
    - Can only move forward or backward, tangent to body direction
    - Can only steer in bounded radius
Kinematic Constraints

Kinematics of Motion

“the branch of mechanics that deals with pure motion, without reference to the masses or forces involved in it”

Equations describing conversion between control and motion

Control: inputs to the system

- In vehicle: steering and throttle
- Also referred to as “Action” in literature
Autonomous Driving: Holonomicity

- Kinematic and dynamic constraints can be considered "rules" governing the state evolution function.
- For state $s_t \in S$, control input $u_t \in U$, time $t \in T$:
  $$F(s_t, u_t, \Delta t) \rightarrow s_{t+1}$$
- Ex:
  - A car cannot turn in place. No amount of steering will accomplish this.
  - A Roomba can turn in place.
Kinematic Constraints

- Kinematic models of a car
- **Single-track Bicycle (or simple car model)**
  - 3-DOF configuration: \((x, y, \theta)\)
  - 2-DOF control: steering \((u_{\phi})\), speed \((u_S)\)
  - Full state: \((x, y, \theta, u_S, u_{\phi}, L)\)
- Equations of motion:
  - \(\dot{p}_x = u_S \cdot \cos(\theta)\)
  - \(\dot{p}_y = u_S \cdot \sin(\theta)\)
  - \(\dot{\theta} = \frac{u_S}{L} \cdot \tan(u_{\phi})\)
Kinematic Constraints

- Kinematic models of a car
  - Single-track Bicycle example
  - https://www.youtube.com/watch?v=TyW1BPpHy18
Kinematic Constraints

- Kinematic models of a car
  - Extended Car w. linear integrators
  - Assume speed is fixed to 1
  - 6-DOF configuration \((x, y, \theta, \phi, \omega)\)
    - 1-DOF Control:
      - Angular acceleration \((u_a)\)
    - Full state \((x, y, \theta, v, \phi, \omega, u_s, u_v, L)\)
Kinematic Constraints

- Extended Car w. linear integrators
  - Equations of motion
    - $\dot{p}_x = \cos(\theta)$
    - $\dot{p}_y = \sin(\theta)$
    - $\dot{\theta} = \frac{\tan(\phi)}{L}$
    - $\dot{\phi} = \omega$
    - $\dot{\omega} = \mu_\alpha$

- Steering is continuous $C^1$
- Control is more complex
Kinematic Constraints

- Example: Stopping the car
  - Let $u_s$ and $u_v$ represent speed and velocity controls
  - Simple-car: $u_s = 0$
  - LI-car $u_v = -v$ iff $\max(u_v) \geq v$ else $\max(U_v)$
  - Car will not necessarily stop right away

- Error increases as we increase the number of integrators
Dynamic Constraints

- “the branch of mechanics concerned with the motion of bodies under the action of forces.”
- Tires subject to lateral and longitudinal force during steering / accelerating
  - If lateral force exceeds friction force
    - Fishtailing
  - If longitudinal force exceeds friction force
    - Skidding
Dynamic Constraints

- No longer directly control acceleration and steering
  - Apply engine force
  - Apply steering force
- Diminishing returns on each force at limits of control
Dynamic Constraints

- Dynamic Bicycle model with linear tires
  - $F_y$ lateral force on tire
  - $F_x$ longitudinal force on tire
  - $\alpha_f$ “slip angle” of tire
  - $\delta$ steering angle
Dynamic Constraints

- Dynamic Bicycle model with linear tires
  - No load transfer between tires
  - Larger state space including tire stiffness

\[
\begin{align*}
F_x f \cos \delta - F_y f \sin \delta + F_{xr} &= m(\dot{v}_x - v_y \dot{\psi}) \\
F_x f \sin \delta + F_y f \cos \delta + F_{yr} &= m(\dot{v}_y + v_x \dot{\psi}) \\
(F_x f \sin \delta + F_y f \cos \delta)b - F_{yr}c &= I_z \ddot{\psi}
\end{align*}
\]

- \( F_x \) longitudinal force
- \( F_y \) lateral force
- \( m \) mass
- \( I_z \) yaw moment of inertia
Dynamic Constraints

- Dynamic constraints
  - Correcting for slip
  - [https://www.youtube.com/watch?v=itggGQu_ECc](https://www.youtube.com/watch?v=itggGQu_ECc)
Dynamic Constraints

- Models increase in complexity as needed for performance tuning
  - Aerodynamic drag force
    \[ F_{\text{wind}} = \frac{(C_w A_w v^2 g)}{16} \]
  - Maximum engine torque
    \[ F_{\text{max}} = 1 + \frac{3}{1 + e^{\frac{v - 12}{4}}} \]

- Each layer of dynamics:
  - Increases accuracy of model
  - Increases computational complexity

https://youtu.be/tesD4F-HOxs?t=1m24s
Structure

- Modeling a car
- Planning
  - Route Planning
  - Behavior Planning
  - Motion Planning
- Control
- Interaction with other Drivers
- Case Studies
Autonomous Driving: Planning

- Route Planning
- Behavior Planning
- Motion Planning
Mission Planner (Route Planning)

- Determine the appropriate macro-level route to take
- Typically road level i.e. which roads to take
- Katrakazas: “Route planning is concerned with finding the best global route from a given origin to a destination, supplemented occasionally with real-time traffic information”
Mission Planner (Route Planning)

- Pendleton: “considers high level objectives, such as assignment of pickup/dropoff tasks and which roads should be taken to achieve the task”
- Typical approaches:
  - RNG (Road-network Graph)
  - A*
  - Dijkstras
- Scale poorly!
Mission Planner (Route Planning)

- Massive-scale algorithms needed for routing
- 18 million vertices, 42.5 million edges
- Partial Western Europe dataset

Structure

- Modeling a car
- Planning
  - Route Planning
  - Behavior Planning
  - Motion Planning
- Control
- Interaction with other Drivers
- Case Studies
Behavior Planner

“makes ad hoc decisions to properly interact with other agents and follow rules restrictions, and thereby generates local objectives, e.g., change lanes, overtake, or proceed through an intersection”

- Finite State Machines
- Finite time maneuvers
**Behavior Planner**

- Example from crowd sim
- AI Technique
  - Defines a set of States and Transition functions between them
  - Allows us to represent complex behaviors with simple components

```
Start -> Find Luggage (50%) -> Get Luggage
          | Luggage Reached
Wait For Help (50%) -> Attendant Arrives
          | No Luggage
Exit Plane -> Luggage Obtained
```
Behavior Planner

- Finite State Machines
  - Set of “states” and transition functions between them
  - Separate from configuration state

Fig. 2. **Finite State Machine**: We highlight different behavior states that are determined by the routing and optimization algorithms. When executing turns, the routing algorithm transitions the behavior state to a turning state. When the optimization-based maneuver algorithm plans a lane change, the behavior state is transitioning to merging.
Behavior Planner

- FSMs limited in some cases
  - What to do in unseen situations?
- Real-time decision making [Furda et al 2011]

Structure

- Modeling a car
- Planning
  - Route Planning
  - Behavior Planning
  - Motion Planning
- Control
- Interaction with other Drivers
- Case Studies
Motion Planning

* Pendleton: generates appropriate paths and/or sets of actions to achieve local objectives, with the most typical objective being to reach a goal region while avoiding obstacle collision
Motion Planning

- **Path Planning**
  - A path $\sigma(\alpha): [0,1] \rightarrow \mathcal{X}$

- **Trajectory Planning**
  - A path $\pi(t): [0,T] \rightarrow \mathcal{X}$

\[ \text{Problem IV.1 (Optimal path planning). Given a 5-tuple } (\mathcal{X}_{\text{free}}, x_{\text{init}}, X_{\text{goal}}, D, J) \text{ find } \sigma^* = \]

\[
\begin{align*}
&\arg\min_{\sigma \in \Sigma(\mathcal{X})} J(\sigma) \\
&\text{subj. to } \sigma(0) = x_{\text{init}} \text{ and } \sigma(1) \in X_{\text{goal}} \\
&\quad \sigma(\alpha) \in \mathcal{X}_{\text{free}} \\
&\quad D(\sigma(\alpha), \sigma'(\alpha), \sigma''(\alpha), \ldots) \quad \forall \alpha \in [0, 1].
\end{align*}
\]

\[ \text{Problem IV.2 (Optimal trajectory planning). Given a 6-tuple } (\mathcal{X}_{\text{free}}, x_{\text{init}}, X_{\text{goal}}, D, J, T) \text{ find } \pi^* = \]

\[
\begin{align*}
&\arg\min_{\pi \in \Pi(\mathcal{X}, T)} J(\pi) \\
&\text{subj. to } \pi(0) = x_{\text{init}} \text{ and } \pi(T) \in X_{\text{goal}} \\
&\quad \pi(t) \in \mathcal{X}_{\text{free}} \\
&\quad D(\pi(t), \pi'(t), \pi''(t), \ldots) \quad \forall t \in [0, T].
\end{align*}
\]
Motion Planning

How do we evaluate them?

- Complexity (computation cost)
  - limits how frequently we can replan
  - NEVER get it perfectly right, so we focus on replanning as fast as possible

- Completeness (likelihood that a solution will be found if one exists)
  - The piano-movers problem is PSPACE-HARD
Motion Planner

- Basic overview
  - Complete planning
  - Combinatorial Planning
  - Sample-Based planning
Motion Planner

+ Basic overview

- **Complete planning** - continuous plan in configuration space
  - Exponential in dimensions of c-space (curse of dimensionality)
  - "Complete"
- **Combinatorial Planning** - discrete planning over an exact decomposition of the configuration space
- **Sample-Based planning**:
Motion Planner

- Basic overview
  - Complete planning
  - **Combinatorial Planning** - discrete planning over an exact decomposition of the configuration space
    - Exponential in dimensions of c-space discretization (curse of dimensionality)
    - "resolution complete"
  - Sample-Based planning
Motion Planner

- Basic overview
- Complete planning
- Combinatorial Planning
- Sample-Based planning - Sample in space to find controls / positions which are collision free and linked
  - Probabilistically complete
    - Some "probabilistically optimal"
  - NOT exponential in configuration space
Motion Planner: Combinatorial Planners

✨ Driving Corridors:

1️⃣ Decompose lanes into polygonal lanelets
2️⃣ Represent obstacles as polygonal bounding boxes or overlapping discs
3️⃣ Adjust lanelets to obstacle constraints

http://doi.org/10.1109/MITS.2014.2306552
Motion Planner: Combinatorial Planners

Driving Corridors:

- Decompose lanes into polygonal lanelets
- Represent obstacles as polygonal bounding boxes or overlapping discs
- Adjust lanelets to obstacle constraints

Motion Planner: Combinatorial Planners

- Driving Corridors:
  - [Video](https://youtu.be/GfXg9ux4xUw?t=2m5s)
Motion Planner: Combinatorial Planners

- Darpa Urban Challenge:
  - BOSS: kinodynamic reachable set
  - Trajectory planner generates candidate trajectories and goals
    - Done by precomputation of many curves
  - “best” trajectory chosen by optimization

Motion Planner: Combinatorial Planners

**Grid Decomposition approaches:**

- Generate cellular-grid representation of local space
- Cells encode probability of occupancy
- Moving obstacles propagate occupancy probability

**Grid Decomposition approaches:**

- [YouTube Video 1](https://youtu.be/CRQfhhICSj0)
- [YouTube Video 2](https://youtu.be/MzpBzrtEGrA)
Motion Planner: Sample-based Planners

- Pendleton: popular for their guarantees of probabilistic completeness, that is to say that given sufficient time to check an infinite number of samples, the probability that a solution will be found if it exists converges to one.

- General approaches:
  - PRM: Probabilistic Roadmaps
  - RRT: Rapidly-Exploring Random Tree
  - FMT: Fast-Marching Trees
Motion Planner: Sample-based Planners

- **PRM: Probabilistic Roadmaps G(V, E)**
  - Repeat until n collision-free samples found i.e. |V| = n
  - Sample a point \( p \) in configuration space
  - \( V = V \cup p \) if \( p \) is collision-free

- For each vertex \( v_i \in V \)
  - Find k nearest neighbors: \( N = \{v_{i1}, v_{i2}, \ldots, v_{ik}\} \)
  - For each neighbor \( v_{ij} \in N \)
    - Determine if edge \( e = (v_i, v_{ij}) \) is collision-free
    - \( E = E \cup e \)
Maneuver Planner: Sample-based Planners

- **RRT: Rapidly-Exploring Random Tree**
  - Given at-least one initial configuration in free-space and a goal configuration
    - Sample a point \( p \) in configuration space, determine if it is collision free
    - If so, find nearest node \( n \) to the point, move some \( \delta \) towards the point
    - If \( n \) to \( n + \delta \) is CLEAR, connect to the tree
Motion Planner: Sample-based Planners

- RRT:
  1. https://www.youtube.com/watch?v=rPgZyq15Z-Q
  2. https://www.youtube.com/watch?v=mEAr2FBUJEI
  3. https://www.youtube.com/watch?v=p3p0EWT5lw
Motion Planner: Sample-based Planners

- Sample-based Planning specifically for cars:
  - Dynamics computation
  - Complex state evolution

- Pendleton: “Incorporating differential constraints into state-sampling planners is still a challenging matter, and requires a steering function to draw an optimal path between two given states which obeys control constraints (if such a path exists), as well as efficient querying methods to tell whether a sampled state is reachable from a potential parent state"
Maneuver Planner: Sample-based Planners

- PRM with dynamics
  - Extends existing PRM framework
  - Sampling directly from admissible controls
  - State $\times$ time space formulation
  - Set of differential equations describing all possible local motions of a robot

Algorithm 1 Control-driven randomized expansion.

1. Insert $m_b$ into $T$; $i ← 1$.
2. repeat
3. Pick a milestone $m$ from $T$ with probability $\pi_T(m)$.
4. Pick a control function $u$ from $U_\ell$ uniformly at random.
5. $m' ← \text{PROPAGATE}(m, u)$.
6. if $m' \neq \text{nil}$ then
7. Add $m'$ to $T$; $i ← i + 1$.
8. Create an edge $e$ from $m$ to $m'$; store $u$ with $e$.
9. if $m' ∈ \text{ENDGAME}$ then exit with SUCCESS.
10. if $i = N$ then exit with FAILURE.
Maneuver Planner: Sample-based Planners

- State-lattice planners

  Ex: Configurations in space

Maneuver Planner: Sample-based Planners

- State-lattice planners
  - [https://www.youtube.com/watch?v=I5hL8vSo6DI](https://www.youtube.com/watch?v=I5hL8vSo6DI)
  - Notice the discrete maneuver points
Structure

- Modeling a car
- Planning
- Control
  - PID
  - Model Predictive Control
  - Path/Trajectory tracking
- Interaction with other Drivers
- Case Studies
Autonomous Driving: Control

Control

- Executing the planned maneuvers accounting for error / uncertainty
- Commands sent to actuators

Figure 2. A typical autonomous vehicle system overview, highlighting core competencies.
Autonomous Driving: Control

Motion Specification

Estimated pose and collision free space

Motion Planning

Reference path or trajectory

Local Feedback Control

Estimate of vehicle state

Steering, throttle and brake commands
Control: Core Concepts

- **Open-loop** control examples
  - **Timers**:
    - Electronic timing switches
    - Clothes Dryer
  - Simple throttle (non-electronic)
    - Motorbikes, go-karts
    - Stove-top gas
  - Sinks / simple valves
    - Hot water / cold water
Control: Core Concepts

- **Closed-loop control examples**
  - **Thermostat:**
    - Engages air-conditioning depending on temperature
  - **Oven:**
    - Heating element controlled by temperature
  - **Cruise-control:**
    - Throttle controlled by current speed / acceleration
  - Used EXTENSIVELY in plant control (i.e. chemical, energy)
Control: Core Concepts

- **Process Variable (PV):** The system output we wish to control
- **Set Point (SP):** Target value of the process Variable
- **Control Output (CO):** Output of the controller (input to the system)
- **Error (E):** Difference between SP and PV

https://www.dataforth.com/introduction-to-pid-control.aspx
Control: Core Concepts

- Example: Water Plant Thermal Control
  - Water kept at constant temperature by gas heater
  - If level rises, gas reduced to stabilize
- PV: Temperature of water
- SP: Desired Temperature
- CO: Level of gas applied to burner

https://www.dataforth.com/introduction-to-pid-control.aspx
Control: Core Concepts

Can we replace the manual control with automatic controller?

Of course, we can!
Structure

- Modeling a car
- Planning
- Control
  - PID
  - Model Predictive Control
  - Path/Trajectory tracking
- Interaction with other Drivers
- Case Studies
Control: PID

Proportional-Integral-Derivative Controller: control loop feedback mechanism widely used in industrial control systems and a variety of other applications requiring continuously modulated control.

- Continuously calculates E, applies correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively.
- Proportion (P): Current error, E (typically SP – PV)
- Integral (I): integral of E (sum of errors over time)
- Derivative (D): derivative of E (typically finite difference)
Control: PID

- **Proportional-Integral-Derivative** Controller: control loop feedback mechanism widely used in industrial control systems and a variety of other applications requiring continuously modulated control.

\[ u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \]
Control: PID

- Proportion: Output controlled by error and Controller Gain ($K_p$)
- Control output proportional to error
  - Choice of error function, but typically SP – PV
  - High gain: can cause oscillation
  - Low gain: fails to correct to Set Point

![Proportional Control Action Diagram](image)

Figure 3
Proportional Control Action
Control: PID

- Proportion-only controller: Output controlled by error and Controller Gain ($K_p$)
- Control output proportional to error
- Choice of error function, but typically SP – PV
- Add bias point for steady output at 0 error
Control: PID

- Integral Control: Output term controlled by integral of error and Integral Gain ($K_i$)
- Corrects “steady-state” error
- Requires a “time” factor for integration ($T_i$)
- Longer time = less integral action
Control: PID

- PI Controller: Proportion and integral terms
- Corrects steady-state error, converges rather than oscillates
Control: PID

- Derivative: Output term controlled by derivative of error and Derivative Gain ($K_d$)
- Assists in rapid response to disturbance
- Requires time parameter to operate

Figure 10
Derivative Control Action
Control: PID

- PID Controller: Proportion, Integral, Derivative terms
- Complete closed-loop controller

Used in AutonoVi and countless applications

Figure 12
The Parallel PID Controller Algorithm
Control: PID Tuning

_rules of thumb for tuning a PID controller:

- https://upload.wikimedia.org/wikipedia/commons/3/33/PID_Compensation_Animated.gif

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rise time</th>
<th>Overshoot</th>
<th>Settling time</th>
<th>Steady-state error</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_p$</td>
<td>Decrease</td>
<td>Increase</td>
<td>Small change</td>
<td>Decrease</td>
<td>Degrade</td>
</tr>
<tr>
<td>$K_i$</td>
<td>Decrease</td>
<td>Increase</td>
<td>Increase</td>
<td>Eliminate</td>
<td>Degrade</td>
</tr>
<tr>
<td>$K_d$</td>
<td>Minor change</td>
<td>Decrease</td>
<td>Decrease</td>
<td>No effect in theory</td>
<td>Improve if $K_d$ small</td>
</tr>
</tbody>
</table>
Control: PID Tuning

Ziegler–Nichols Tuning

Tune $K_p$ until the control loop begins to oscillate

Called Ultimate control point ($K_u$)

$K_u$ and oscillation period $T_u$ used to tune parameters as follows

<table>
<thead>
<tr>
<th>Control Type</th>
<th>$K_p$</th>
<th>$K_i$</th>
<th>$K_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$0.50K_u$</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>$PI$</td>
<td>$0.45K_u$</td>
<td>$0.54K_u/T_u$</td>
<td>—</td>
</tr>
<tr>
<td>$PID$</td>
<td>$0.60K_u$</td>
<td>$1.2K_u/T_u$</td>
<td>$3K_uT_u/40$</td>
</tr>
</tbody>
</table>
Control: PID Examples

✦ More examples of PID:
  ✦ Cruise-control
  ✦ Quad-rotor Autopilot
  ✦ Mobile robot control
    ✦ PID for steering + PID for speed
  ✦ Spaceships
  ✦ …
  ✦ …
  ✦ Innumerable examples of PID control
Structure

- Modeling a car
- Planning
- Control
  - PID
  - MPC
  - Path Tracking
- Interaction with other Drivers
- Case Studies
Control: MPC

- **Model-Predictive Controller**: control loop relying on an underlying system model to generate feed-forward control
  - At each time step, compute control by solving an openloop optimization problem for the prediction horizon
  - Apply the first value of the computed control sequence
  - At the next time step, get the system state and re-compute

- **Resources**
  - [https://www.youtube.com/watch?v=oMUtYZOgsng](https://www.youtube.com/watch?v=oMUtYZOgsng)
  - [https://www.youtube.com/watch?v=DFqOf5wbQtc](https://www.youtube.com/watch?v=DFqOf5wbQtc)
Control: MPC

- MPC is very useful when process model is available
  - Reduces overshoot substantially
  - Using cached table of input responses, optimization can be done quickly
- MPC uses in automotive context:
  - Traction control [Borelli 2006]
  - Braking control [Falcone 2007]
  - Steering [Falcone 2007]
  - Lane-keeping [Liu 2015]
Structure

- Modeling a car
- Planning
- Control
  - PID
  - MPC
  - Path Tracking
- Interaction with other Drivers
- Case Studies
Control: Path tracking with controllers

- Given a path/trajectory computed by the motion planner, we use controls to follow or “achieve” the path
- Many methods for path/trajectory tracking:
  - Pure-pursuit
  - AutonoVi (Arcs)
  - Kinematic Bicycle
  - Model-Predictive Control
Control: Path tracking with controllers

- **Pure-pursuit**

  1. Given a geometric path, track a point ahead of the vehicle according to a fixed lookahead (can be a function of speed)
  2. [video1](https://www.youtube.com/watch?v=qG70QJJ8Qz8)
  3. [video2](https://www.youtube.com/watch?v=vlyTthJugRQ)

- Advantages: simple, robust to perturbation
- Disadvantages: Corner-cutting, oscillation for non-holonomic robots
Control: Path tracking with controllers

AutonoVi
Control: Path tracking with controllers

AutonoVi
Structure

- Modeling a car
- Planning
- Control
- Interaction with other drivers
  - Formal framework for 2-way interactions
  - Probabilistic reasoning for multi-vehicle interactions
- Case Studies
Formal Framework for 2-way interactions

Key insight is that other drivers do not operate in isolation:
- An autonomous car’s actions will actually have effects on what other drivers will do.
- Leveraging these effects during planning will generate behaviors for autonomous cars that are more efficient and communicative.

Structure

- Modeling a car
- Planning
- Control
- Interaction with other drivers
  - Formal framework for 2-way interactions
  - Probabilistic reasoning for multi-vehicle interactions
- Case Studies
Behavior Prediction

- Choose ego-vehicle actions that maximize a reward function over time within a dynamic, uncertain environment with tightly coupled inter-actions between multiple agents.

- Relies on finite set of a priori known policies

Structure

- Modeling a car
- Planning
- Control
- Interaction with other Drivers
- Case Studies
  - MATSim: Macroscopic Traffic Simulator
  - Autonovi
Autonomous Driving: MATSim

- Large-scale agent-based transport simulations
  - Extendable modules
  - High level agent behaviors/activities
  - Capable of running massive simulations
  - Analysis tools
  - Suitable for road network planning, traffic analysis and routing
Structure

- Modeling a car
- Planning
- Control
- Interaction with other Drivers
- Case Studies
  - MATSim: Macroscopic Traffic Simulator
  - Autonovi
References


AutonoVi-Sim:
Modular Autonomous Vehicle Simulation Platform Supporting Diverse Vehicle Models, Sensor Configuration, and Traffic Conditions

Andrew Best, Sahil Narang, Lucas Pasqualin, Daniel Barber, Dinesh Manocha
University of North Carolina at Chapel Hill
UCF Institute for Simulation and Training
http://gamma.cs.unc.edu/AutonoVi/
Motivation

- 1.2 billion vehicles on the roads today
- 84 million new vehicles in 2015
  - China: 24 m  U.S.:  2.7m
  - India:  3.7 m  S.E Asia: 3.8m
- Many markets expected to grow exponentially through 2030
Motivation

• Majority of new vehicles in developing markets (30+ million)
• Limited infrastructure, loose traffic conventions
• Average vehicle life: 10+ years (17 years in U.S)
Motivation

• Long before autonomy will reach this:

Au et al. 2012

Kabbaj, TED 2016
Motivation

- It will deal with this:
Challenges

- Safety guarantees are critical
- Drivers, pedestrians, cyclists difficult to predict
- Road and environment conditions are dynamic
- Laws and norms differ by culture
- Huge number of scenarios
Challenges

• Development and testing of autonomous driving algorithms
  • On-road experiments may be hazardous
  • Closed-course experiments may limit transfer
  • High costs in terms of time and money

• Solution: develop and test robust algorithms in simulation
  • Test novel driving strategies & sensor configurations
  • Reduces costs
  • Allows testing dangerous scenarios
  • Vary traffic and weather conditions
Contributions

• **AutonoVi-Sim**: high fidelity simulation platform for testing autonomous driving algorithms
  - Varying vehicle types, traffic condition
  - Rapid Scenario Construction
  - Simulates cyclists and pedestrians
  - Modular Sensor configuration, fusion
  - Facilitates testing novel driving strategies
Contributions

- **AutonoVi**: novel algorithm for autonomous vehicle navigation
  - Collision-free, dynamically feasible maneuvers
  - Navigate amongst pedestrians, cyclists, other vehicles
  - Perform dynamic lane-changes for avoidance and overtaking
  - Generalizes to different vehicles through data-driven dynamics approach
  - Adhere to traffic laws and norms
Overview

• Motivation

**Related Work**

• Contributions:
  • Simulation Platform: Autonovi-Sim
  • Navigation Algorithm: Autonovi

• Results
Related work:

• Traffic Simulation
  • MATSim [Horni 2016], SUMO [krajzewicz 2002]

• Autonomous Vehicle Simulation
  • OpenAI Universe, Udacity
  • Waymo Carcraft, Righthook.io

• Simulation integral to development of many controllers & recent approaches [Katrakazas2015].
Related work:

- Collision-free navigation
  - Occupancy grids [Kolski 2006], driving corridors [Hardy 2013]
  - Velocity Obstacles [Berg 2011], Control obstacles [Bareiss 2015],
    polygonal decomposition [Ziegler 2014], random exploration
    [Katrakazas 2015]
  - Lateral control approaches [Fritz 2004, Sadigh 2016]

- Generating traffic behaviors
  - Human driver model [Treiber 2006], data-driven [Hidas 2005],
    correct by construction [Tumova 2013], Bayesian prediction
    [Galceran 2015]
Related work:

• Modelling Kinematics and Dynamics
  • kinematic models [Reeds 1990, LaValle 2006, Margolis 1991]
  • Dynamics models [Borrelli 2005]

• Simulation for Vision Training
  • Grand Theft Auto 5 [Richter 2016, Johnson-Roberson 2017]
Overview

• Motivation
• Related Work
• Contributions:
  • Simulation Platform: Autonovi-Sim
  • Navigation Algorithm: Autonovi
• Results
Autonovi-Sim

- Modular simulation framework for generating dynamic traffic conditions, weather, driver profiles, and road networks
- Facilitates novel driving strategy development
Autonovi-Sim: Roads & Road Network

- Roads constructed by click and drag
- Road network constructed automatically

Road layouts
Autonovi-Sim: Roads & Road Network

- Construct large road networks with minimal effort
- Provides routing and traffic information to vehicles
- Allows dynamic lane closures, sign obstructions

Urban Environment for pedestrian & cyclist testing
4 kilometer highway on and off loop
Autonovi-Sim: Infrastructure

- Infrastructure placed as roads or overlays
- Provide cycle information to vehicles, can be queried and centrally controlled

3 way, one lane
3 way, two lane
4 way, two lane
Autonovi-Sim: Environment

- Goal: Testing driving strategies & sensor configuration in adverse conditions
- Simulate changing environmental conditions
  - Rain, fog, time of day
  - Modelling associated physical changes

Fog reduces visibility

Heavy rain reduces traction
Autonovi-Sim: Non-vehicle Traffic

- **Cyclists**
  - operate on road network
  - Travel as vehicles, custom destinations and routing

- **Pedestrians**
  - Operate on roads or sidewalks
  - Programmable to follow or ignore traffic rules
  - Integrate prediction and personality parameters
Autonovi-Sim: Vehicles

- Various vehicle profiles:
  - Size, shape, color
  - Speed / engine profile
  - Turning / braking
- Manage sensor information

Laser Range-finder | Multiple Vehicle Configurations | Multi-camera detector

Vehicles
  - Physics
  - Control
  - Perception
Autonovi-Sim: Vehicles

• Sensors placed interactively on vehicle
  • Configurable perception and detection algorithms
Autonovi-Sim: Drivers

- Control driving decisions
  - Fuse sensor information
  - Determine new controls (steering, throttle)
- Configurable parameters representing personality
  - Following distance, attention time, speeding, etc.
- Configure proportions of driver types
  - i.e. 50% aggressive, 50% cautious
Autonovi-Sim: Drivers

- 3 Drivers in AutonoVi-Sim
  - Manual
  - Basic Follower
  - AutonoVi
Autonovi-Sim: Results

- Simulating large, dense road networks
- Generating data for analysis, vision classification, autonomous driving algorithms

50 vehicles navigating (3x)
Overview

• Motivation
• Related Work
• Contributions:
  • Simulation Platform: Autonovi-Sim
  • **Navigation Algorithm: Autonovi**
• Results
Autonovi

- Computes collision free, dynamically feasible maneuvers amongst pedestrians, cyclists, and vehicles
- 4 stage algorithm
  - Routing / GPS
  - Guiding Path Computation
  - Collision-avoidance / Dynamics Constraints
  - Optimization-based Maneuvering
Autonovi: Routing / GPS

- Generates maneuvers between vehicle position and destination
- Nodes represent road transitions
- Allows vehicle to change lanes between maneuvers

Autonovi: Guiding Path

- Computes “ideal” path vehicle should follow
- Respects traffic rules
- Path computed and represented as arc
- Generates target controls
Autonovi: Collision Avoidance / Dynamics

- Control Obstacles [Bareiss 2015]
  - “Union of all controls that could lead to collisions with the neighbor within the time horizon, \( \tau \)”
  - Plan directly in control space (throttle, steering)
  - Construct “obstacles” for nearby entities

- Key principles / Assumptions
  - Reciprocity in avoidance (all agents take equal share)
  - Bounding discs around each entity
  - Controls / decisions of other entities are observable
  - New controls chosen as minimal deviation from target s. t. the following is not violated:

\[
\forall (j \neq i, 0 \leq t < \tau) : (\mathcal{O}_i \oplus \{ q_i(g_i(t, x_i, u_i + \Delta u_i)) \}) \cap (\mathcal{O}_j \oplus \{ q_j(g_j(t, x_j, u_j + \Delta u_j)) \}) = \emptyset
\]

[Bareiss 2015]
Autonovi: Collision Avoidance / Dynamics

• Goal: Augment control obstacles with dynamics constraints
• Generate dynamics profile for vehicles through profiling
  • repeated simulation for each vehicle testing control inputs
• Represent underlying dynamics without specific model
• Gather data to generate approximation functions for non-linear vehicle dynamics
  • $S(\mu)$ : target controls are safe given current vehicle state
  • $A(\mu)$ : Expected acceleration given effort and current state
  • $\Phi(\mu)$ : Expected steering change given effort and current state
Augmented Control Obstacles

- Reciprocity is not assumed from others
- Use tightly fitting bounding polygons
- Do not assume controls of others are observable
- New controls chosen from optimization stage

Autonovi: Collision Avoidance / Dynamics
Augmented Control Obstacles
- Reciprocity is not assumed from others
- Use tightly fitting bounding polygons
- Do not assume controls of others are observable
- New controls chosen from optimization stage

Obstacles constructed from avoidance
Autonovi: Collision Avoidance / Dynamics

• Augmented Control Obstacles
  • Reciprocity is not assumed from others
  • Use tightly fitting bounding polygons
  • Do not assume controls of others are observable
  • New controls chosen from optimization stage

• Obstacles constructed from avoidance
• Obstacles constructed from dynamics
Augmented Control Obstacles

- Reciprocity is not assumed from others
- Use tightly fitting bounding polygons
- Do not assume controls of others are observable
- New controls chosen from optimization stage

Obstacles constructed from avoidance

Obstacles constructed from dynamics

New velocity chosen by cost-optimization
Autonovi: Collision Avoidance / Dynamics

- Advantages of augmented control obstacles:
  - Free-space is guaranteed feasible and safe
  - Conservative linear constraints from surface of obstacles
- Disadvantages:
  - Closed-form of surface may not exist
  - Space may be non-convex
  - Computationally expensive
Autonovi: Collision Avoidance / Dynamics

- Sampling approach
  - Construct candidate controls via sampling near target controls
  - Evaluate collision-avoidance and dynamics constraints
  - Forward integrate safe controls to generate candidate trajectories
- Choose “optimal” control set in optimization stage
Autonovi: Optimization-Based Maneuvering

- Choose “optimal” controls through multi-objective cost function
- Path (velocity, drift, progress)
- Comfort (acceleration, yaw)
- Maneuver (lane change, node distance)
- Proximity (cyclists, vehicle, pedestrians)

\[
C = \sum_{i=0}^{I} c_{\text{path}}(i) + c_{\text{cmff}}(i) + c_{\text{mnvr}}(i) + c_{\text{prox}}(i)
\]
Autonovi: Optimization-Based Maneuvering

- Choose “optimal” controls through multi-objective cost function
- Path (velocity, drift, progress)
- Comfort (acceleration, yaw)
- **Maneuver (lane change, node distance)**
  - Static cost for lane changes
  - Cost inverse to distance if vehicle occupies incorrect lane as maneuver approaches
- Proximity (cyclists, vehicle, pedestrians)

\[
C = \sum_{i=0}^{I} c_{path}(i) + c_{cmft}(i) + c_{mnvr}(i) + c_{prox}(i)
\]
Autonovi: Optimization-Based Maneuvering

- Choose “optimal” controls through multi-objective cost function
- Path (velocity, drift, progress)
- Comfort (acceleration, yaw)
- Maneuver (lane change, node distance)
- **Proximity (cyclists, vehicle, pedestrians)**
  - Configurable cost per entity type
  - Generates safe passing buffers

\[
C = \sum_{i=0}^{I} (c_{path}(i) + c_{cmft}(i) + c_{mnvr}(i) + c_{prox}(i))
\]
Overview

• Motivation
• Related Work
• Contributions:
  • Simulation Platform: Autonovi-Sim
  • Navigation Algorithm: Autonovi
• Results
Results: Sudden Hazards @ 20 mph

- Vehicle responds quickly to sudden hazards
  - Braking and swerving to avoid collisions
Results: Sudden Hazards @ 60 mph

- Vehicle responds quickly to sudden hazards
  - Respects unique dynamics of each car
Results: Jaywalking Pedestrian

- Vehicle accounts for pedestrians and comes to a stop
Results: Jaywalking Pedestrian

• Vehicle accounts for pedestrians and comes to a stop
  • Respects unique dynamics of each car
Results: Passing Cyclists

- Vehicle changes lanes to safely pass cyclist
Results: Passing Cyclists

• Vehicle changes lanes to safely pass cyclist
  • Lane change only when possible
Results: Next Steps

- Integrating prediction compensates for uncertainty
- Leverage behavior models to predict behavior
  - Bayesian Behavior prediction
  - Personality Trait Theory

SocioSense approach to pedestrian response

Predicting Future Pedestrian Trajectories

Predicting Future Cyclist Trajectories
Results: Next Steps

- Using real-world training data, behaviors can be optimized to improve realism
  - Ex: Drivers behave more like human drivers
  - Ex: Infrastructure tuned to specific real patterns
- Vehicle sensors can be similarly calibrated
Overview

- Motivation
- Related Work
- Contributions:
  - Simulation Platform: Autonovi-Sim
  - Navigation Algorithm: Autonovi
- Results
- Conclusion
Conclusions

• Simulation of hundreds of vehicles, pedestrians, cyclists
• Configurable sensors and driver behavior
• Collision-free, dynamically feasible maneuvers
• Perform dynamic lane-changes for avoidance and overtaking
• Generalizes to different vehicles through data-driven dynamics profiling
Limitations

- Data-driven dynamics rely on simulation
- Reliance on perfect sensing
- Parameter weights manually optimized
- Manual Sensor Calibration

Future Work

- Data-driven parameter weight learning
- Validation of dynamics modelling
- Improve generation and handling of sensor uncertainty
- Behavior prediction for nearby entities
- Combine with road-network data to generate scenarios
- Additional sensor implementations
References


References