Comp 790-058 Lecture 10: Autonomous Driving: Control, Traffic, Predictions

October 24, 2017

Andrew Best

University of North Carolina, Chapel Hill

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Administrative

- Homework 2 due:
  - 11:59 PM October 30th
- Homework 3:
  - Not today! But this week.
- Project Updates:
  - Remember to work consistently on projects
  - It WILL sneak up on you
- AutonoVi Updates:
  - Git setup
  - If you need access, please see me after class

Structure

- Recap
  - Perception
  - Localization
  - State / Kinematics / Dynamics
  - Planning
- Control
- Traffic-Sim
- Prediction

Autonomous Driving

- Autonomous vehicle: a motor vehicle that uses artificial intelligence, sensors and global positioning system coordinates to drive itself without the active intervention of a human operator
- Focus of enormous investment [$1b+ in 2015]

Tesla

Waymo

Nutonomy

Autonomous Driving: Levels of Autonomy

- 0: Standard Car
- 1: Assist in some part of driving
  - Cruise control
- 2: Perform some part of driving
  - Adaptive CC + lane keeping
- 3: Self-driving under ideal conditions
  - Human must remain fully aware
- 4: Self-driving under near-ideal conditions
  - Human need not remain constantly aware
- 5: Outperforms human in all circumstances

Autonomous Driving: Main Components

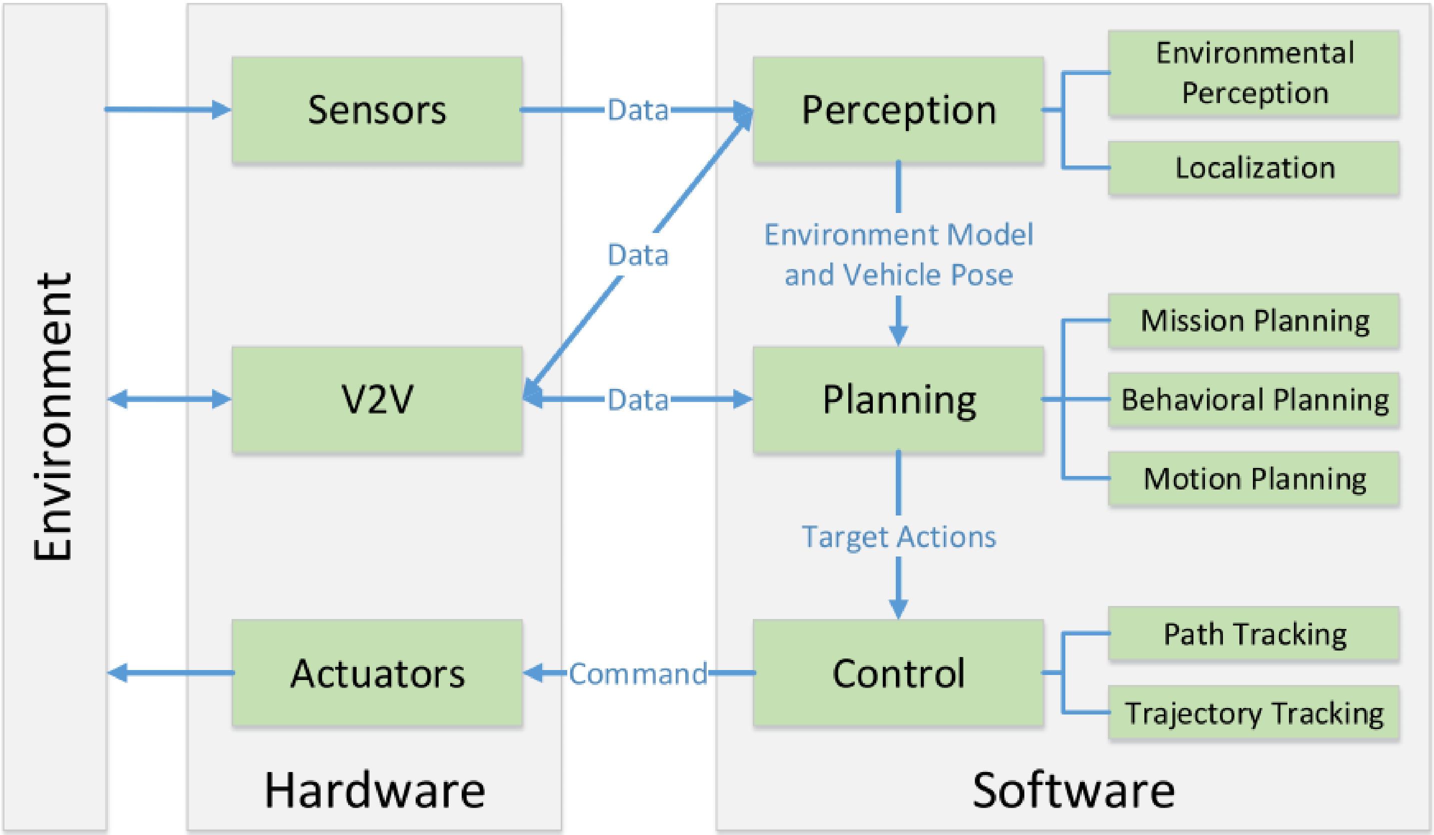


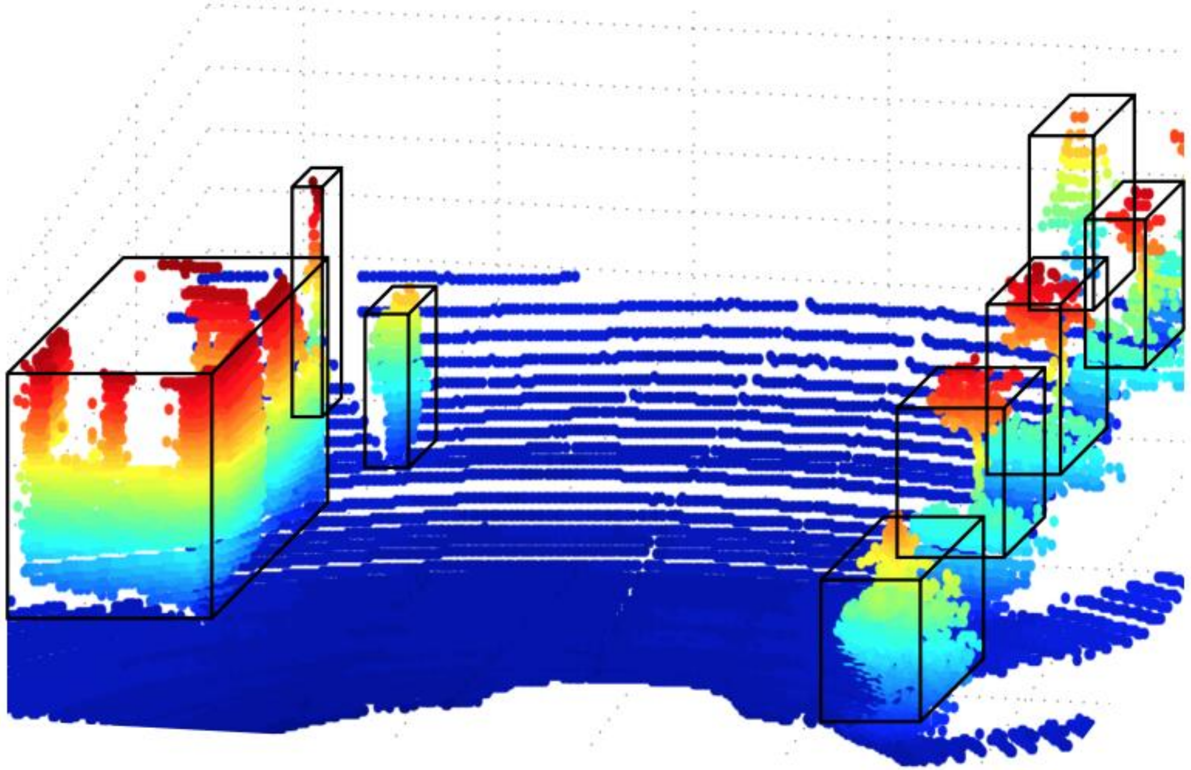
Figure 2. A typical autonomous vehicle system overview, highlighting core competencies.

Structure

- Recap
  - **Perception**
  - Localization
  - State / Kinematics / Dynamics
  - Planning
- Control
- Traffic-Sim
- Prediction

Autonomous Driving: Perception using LIDAR

- Light Detection and Ranging
  - Illuminate target using pulsed laser lights, and measure reflected pulses using a sensor

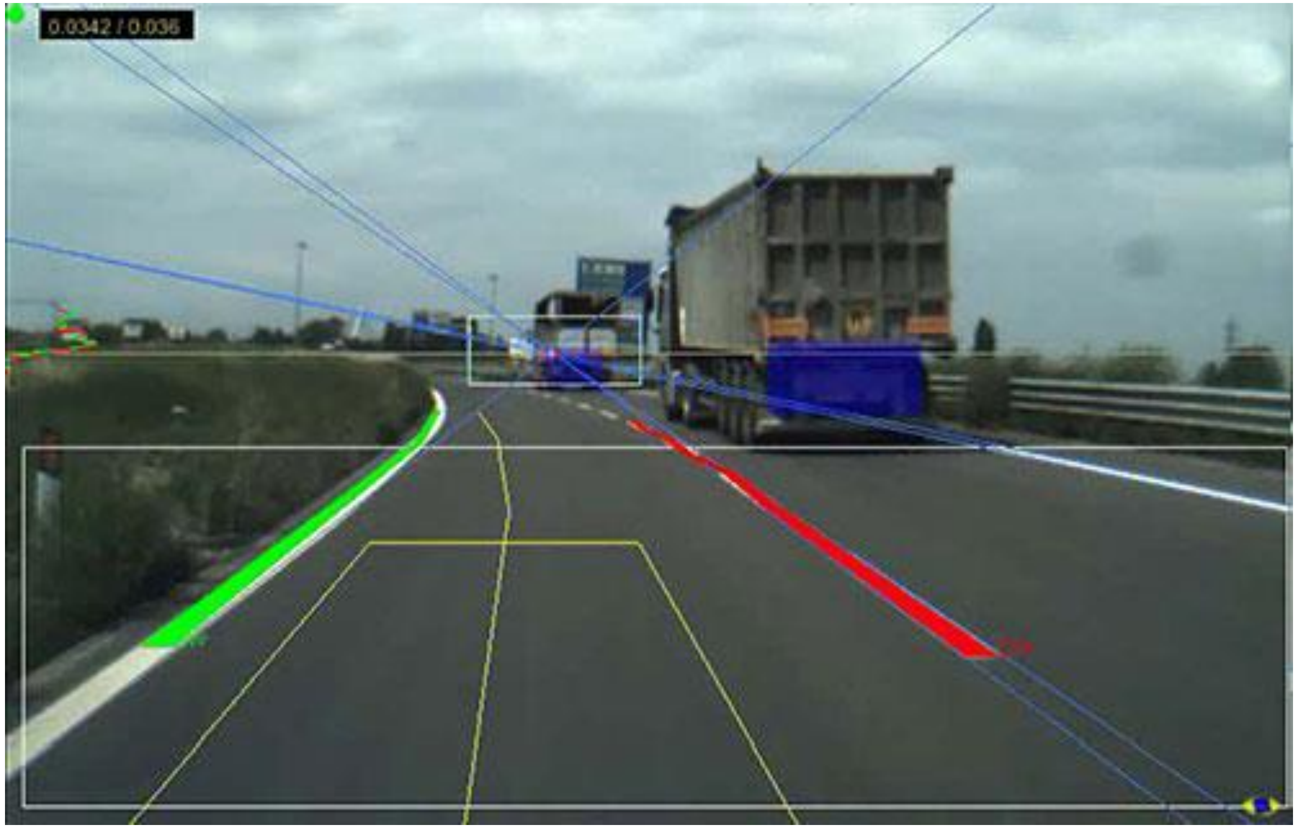


Autonomous Driving: Perception using LIDAR

- LIDAR in practice
  - Velodyne 64HD lidar
    - https://www.youtube.com/watch?v=nXlqv_k4P8Q

Autonomous Driving: Perception using Cameras

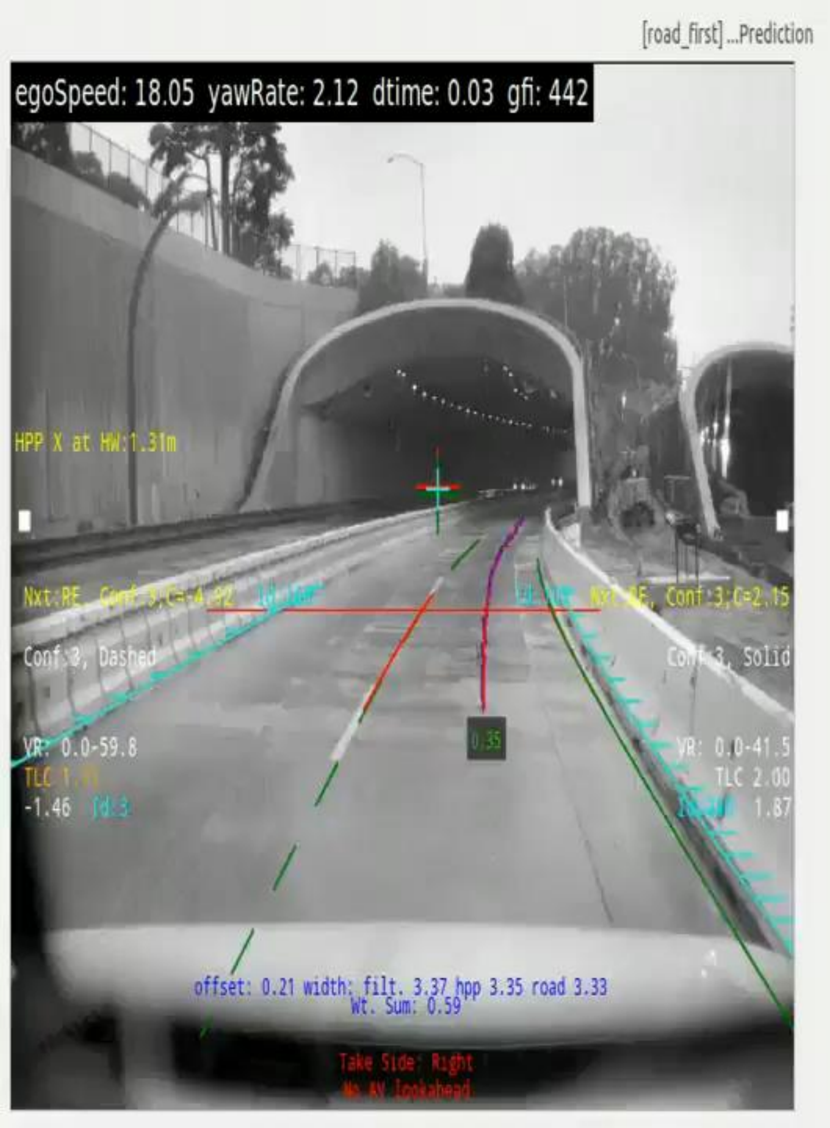
- Camera based vision
  - Road detection
    - Lane marking detection
    - Road surface detection
  - On-road object detection



THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Autonomous Driving: Perception using Cameras

- Sensing Challenges
  - Sensor Uncertainty
  - Sensor Configuration
  - Weather / Environment



Structure

- Recap
  - Perception
  - **Localization**
  - State / Kinematics / Dynamics
  - Planning
- Control
- Traffic-Sim
- Prediction

Autonomous Driving: Vehicle Localization

- Determining the pose of the ego vehicle and measuring its own motion
- Fusing data
  - Satellite-based navigation system
  - Inertial navigation system
- Map aided localization
  - SLAM

Structure

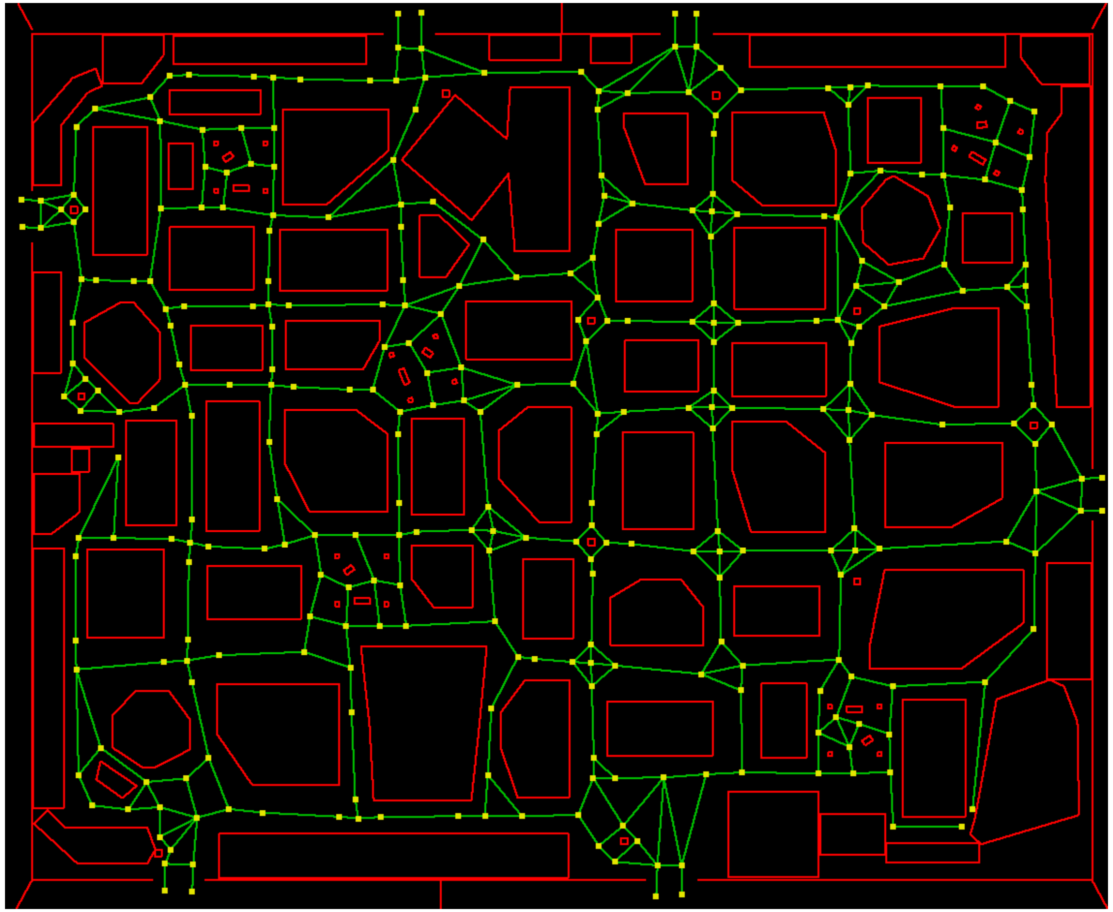
- Recap
  - Perception
  - Localization
  - **State / Kinematics / Dynamics**
  - Planning
- Control
- Traffic-Sim
- Prediction

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

- Recall Pedestrian Planning:
  - Roadmap is essential a graph of potential agent states



THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Autonomous Driving: State Space

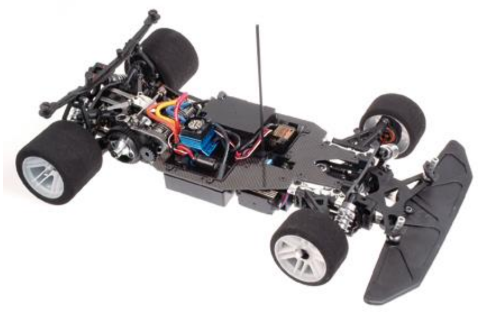
- Examples:
  - 2D space with blinker booleans
    - $(\vec{p}, \theta, \vec{v}, \omega, bl_l, bl_r)$
  - State contains everything we need to describe the robot's current configuration!
  - Neglect some state variables when planning

Structure

- Recap
- State, Kinematics, and Dynamics Models
    - State Space
    - Kinematic constraint models of the vehicle
    - Dynamic constraint models of the vehicle
- Planning
- AutonoVi-Sim

Autonomous Driving: Holonomicity

- Cars are “non-holonomic” robots
  - Typically 5 values describing physical
    - (2 Cartesian coordinates, orientation, linear speed, angular speed)
  - 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

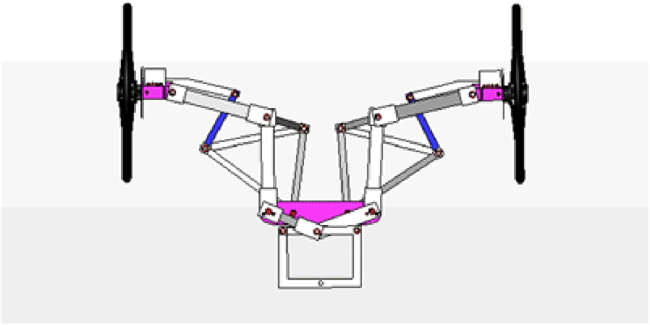
Autonomous Driving: Holonomicity

- kinematic and dynamic constraints can be considered "rules" governing the state evolution function
- For state $s_t \in S$, control input $u_t \in \mathrm{U}$, time $t \in T$:
  - $\mathrm{F}(\mathrm{s_t}, \mathrm{u}_t, \Delta t) \rightarrow \mathrm{s_{t+1}}$
- 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,θ)
    - 2-DOF control: steering (φ), speed (v)
    - Full state: (x,y,θ,v, φ,L)
  - Equations of motion:
    - $\dot{p}_x = v * cos(\theta) \quad \dot{p}_y = v * sin(\theta)$

$$\dot{\theta} = \frac{\tan(\phi)}{L}$$



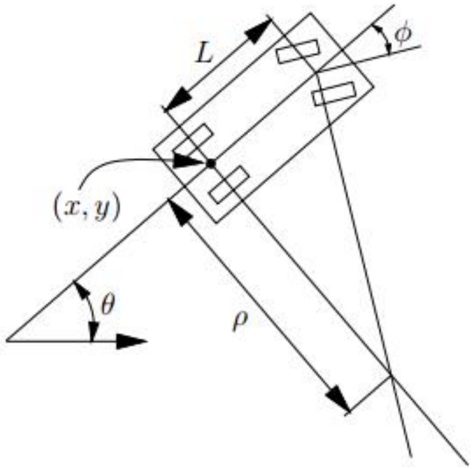
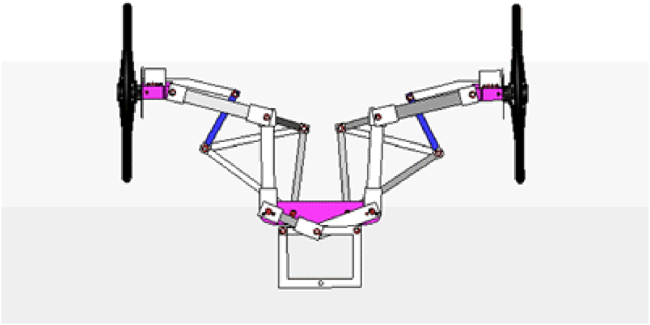


Figure 13.1: The simple car has three degrees of freedom, but the velocity space at any configuration is only two-dimensional.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Kinematic Constraints

- Single-track bicycle example
  - ❿ [github link to my project]



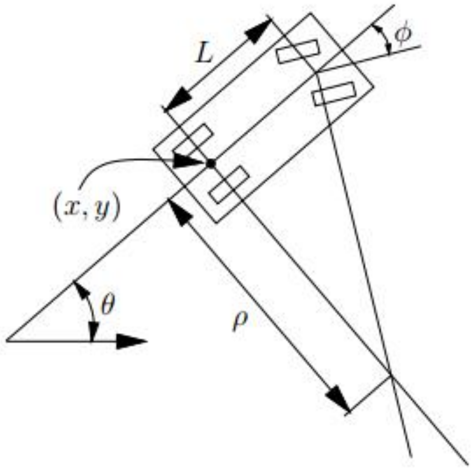


Figure 13.1: The simple car has three degrees of freedom, but the velocity space at any configuration is only two-dimensional.

Structure

- Recap
- State, Kinematics, and Dynamics Models
  - State Space
  - Kinematic constraint models of the vehicle
  - Dynamic constraint models of the vehicle
- Planning
- AutonoVi-Sim

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
    - Peel out / skid

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
  - No load transfer between tires
  - Larger state space including tire stiffness
- F_x longitudinal force
- F_y lateral force
- m mass
- I_z yaw moment of intertia

$$
\begin{aligned}
F_{xf}\cos\delta - F_{yf}\sin\delta + F_{xr} &= m(\dot{v}_x - v_y\dot{\psi}) \\
F_{xf}\sin\delta + F_{yf}\cos\delta + F_{yr} &= m(\dot{v}_y + v_x\dot{\psi}) \\
(F_{xf}\sin\delta + F_{yf}\cos\delta)b - F_{yr}c &= I_z\ddot{\psi}
\end{aligned}
$$

$$F_y = C_\alpha \alpha$$

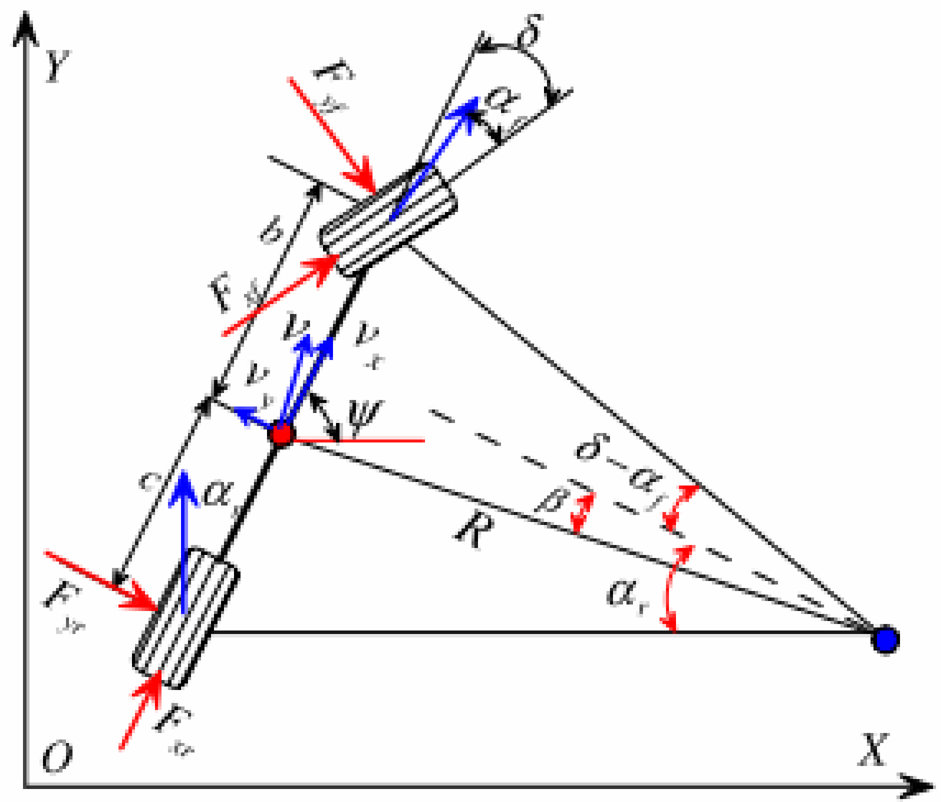
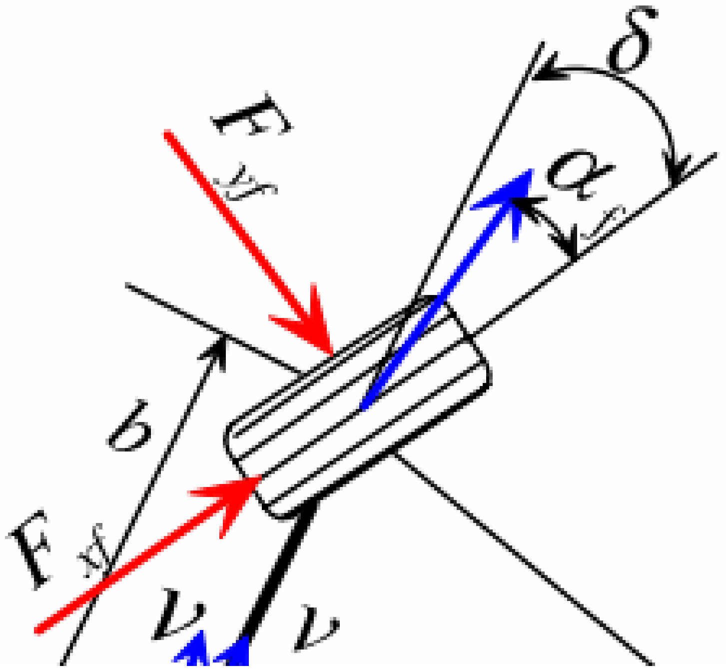


Fig. 1. Bicycle model of vehicle

Dynamic Constraints

- Dynamic Bicycle model with linear tires
  - F_y lateral force on tire
  - F_x longitudinal force on tire
  - α_f "slip angle" of tire
  - δ steering angle



Dynamic Constraints

- Models increase in complexity as needed for performance tuning
  - Aerodynamic drag force $F_{wind} = (C_w A_w v_t^2 g)/16$
  - Maximum engine torque $$\frac{F_{max}}{m} = 1 + \frac{3}{1 + e^{\left(\frac{v_t - 12}{4}\right)}}$$
- Each layer of dynamics:
  - Increases accuracy of model
  - Increases computational complexity

Structure

- Recap
  - Perception
  - Localization
  - State / Kinematics / Dynamics
  - **Planning**
- Control
- Traffic-Sim
- Prediction

Autonomous Driving: Main Components

- Planning
  - Making purposeful decisions in order to achieve the robot's higher order goals

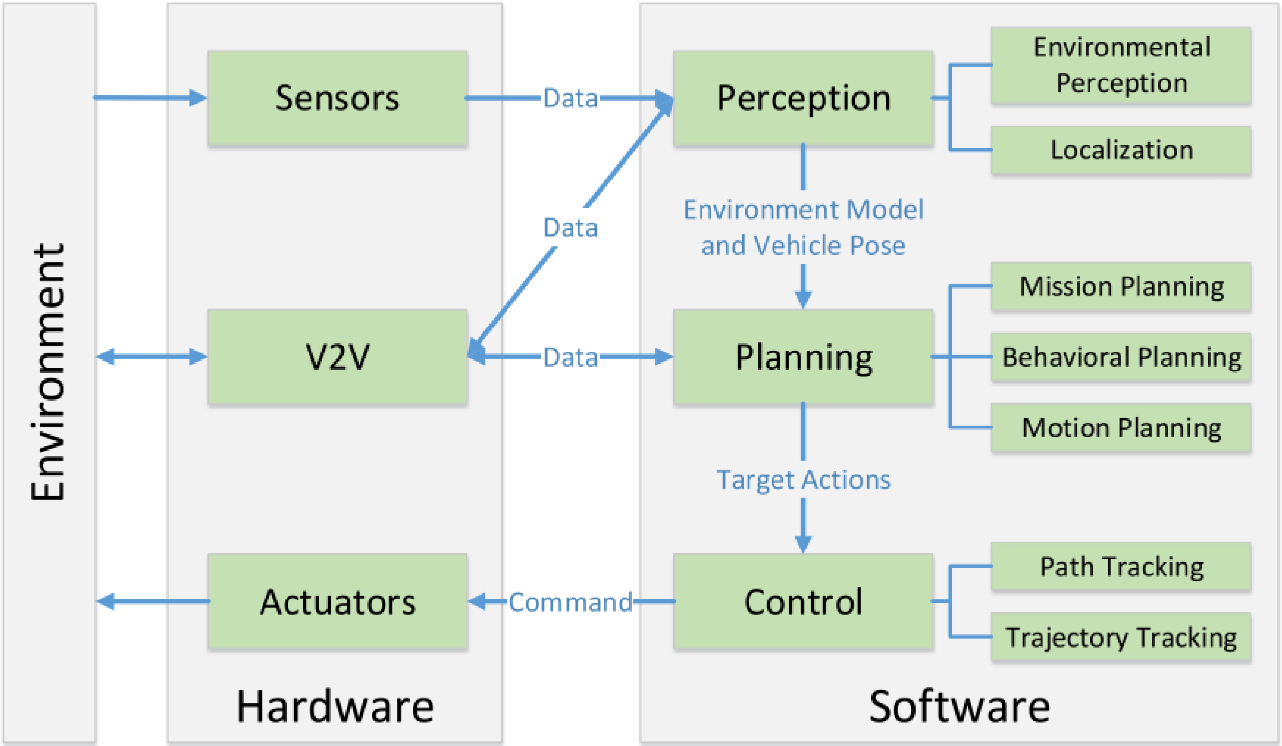


Figure 2. A typical autonomous vehicle system overview, highlighting core competencies.

Main Idea

- Motion Planning: term used in robotics for the process of breaking down a desired movement task into discrete **motions** that satisfy movement constraints and possibly optimize some aspect of the movement

Autonomous Driving: Planning

- Compare to Pedestrian Techniques:
  - Route Planning: road selection (global)
  - Path Planning: preferred lanes (global)
  - Maneuver-search: high level maneuvers (local)
  - Trajectory planning: Lowest level of planning (local)

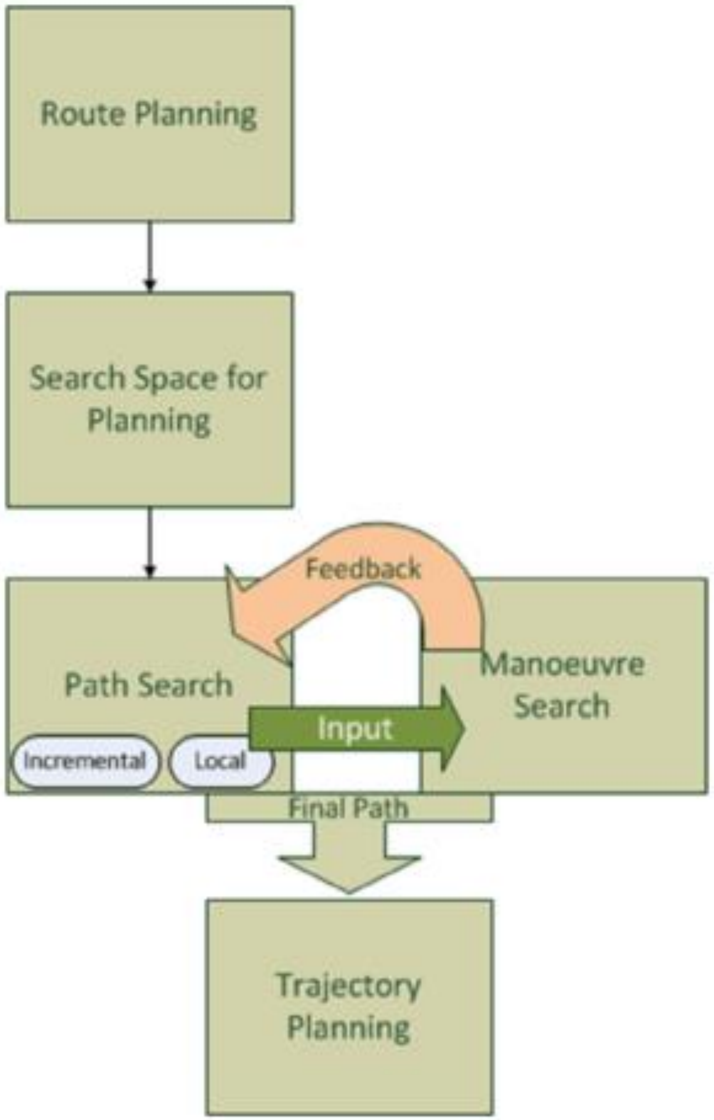
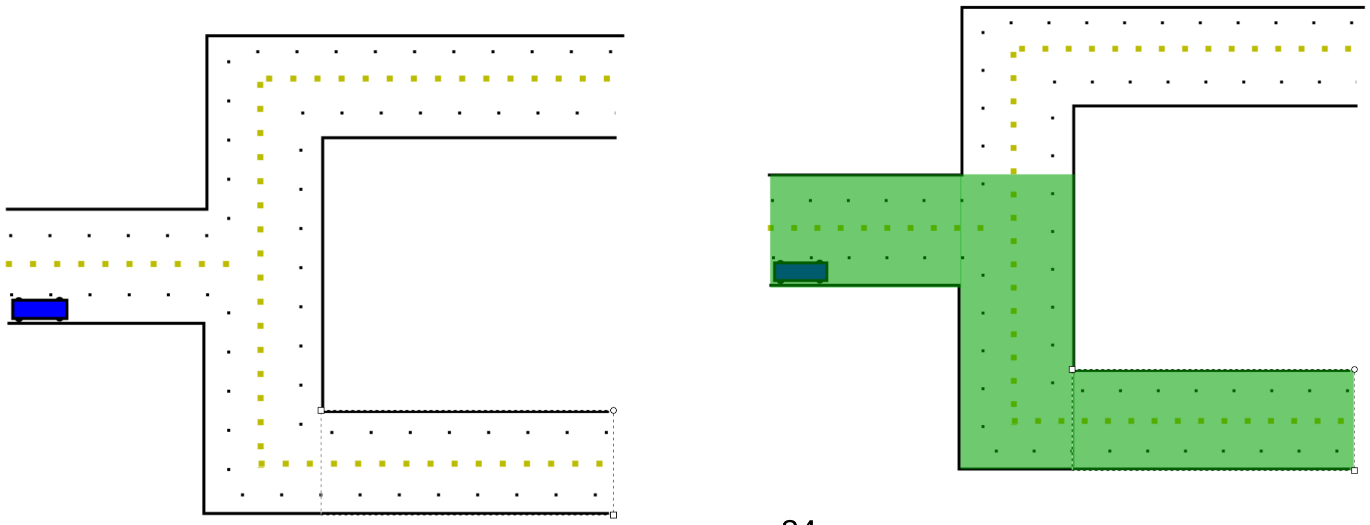


Fig. 2. A flow chart of planning modules.

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

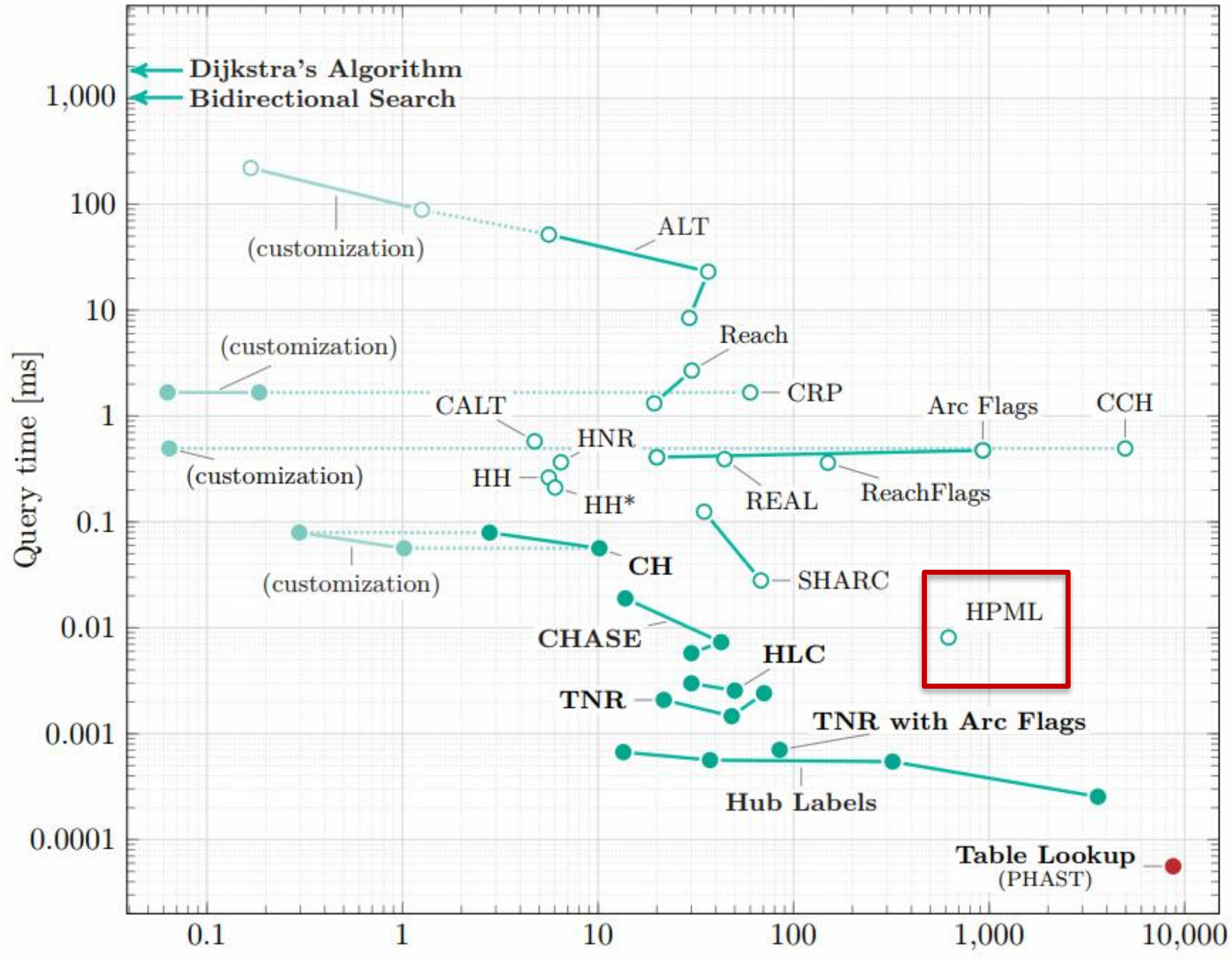
Mission Planner (Route Planning)

- 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



Behavior Planner

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

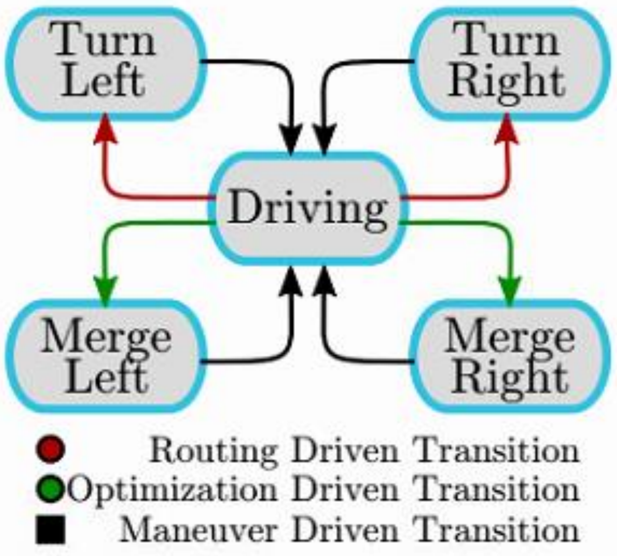
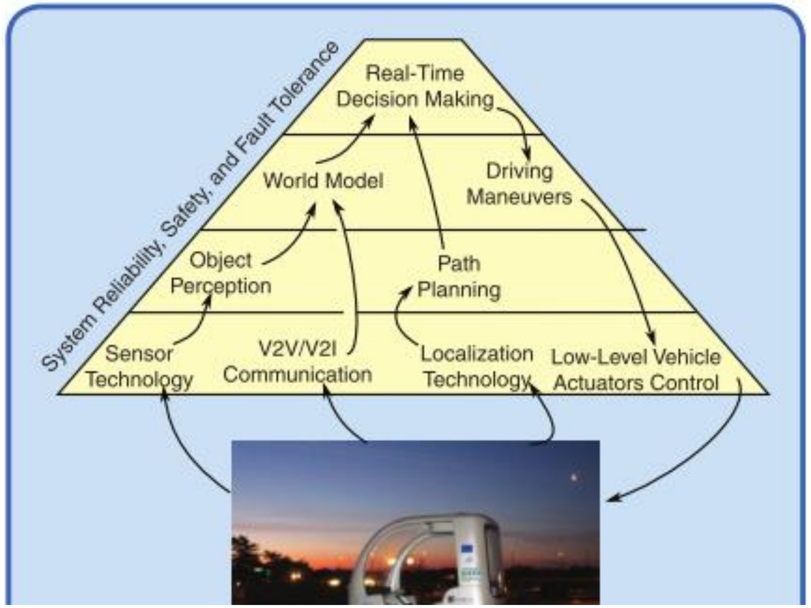
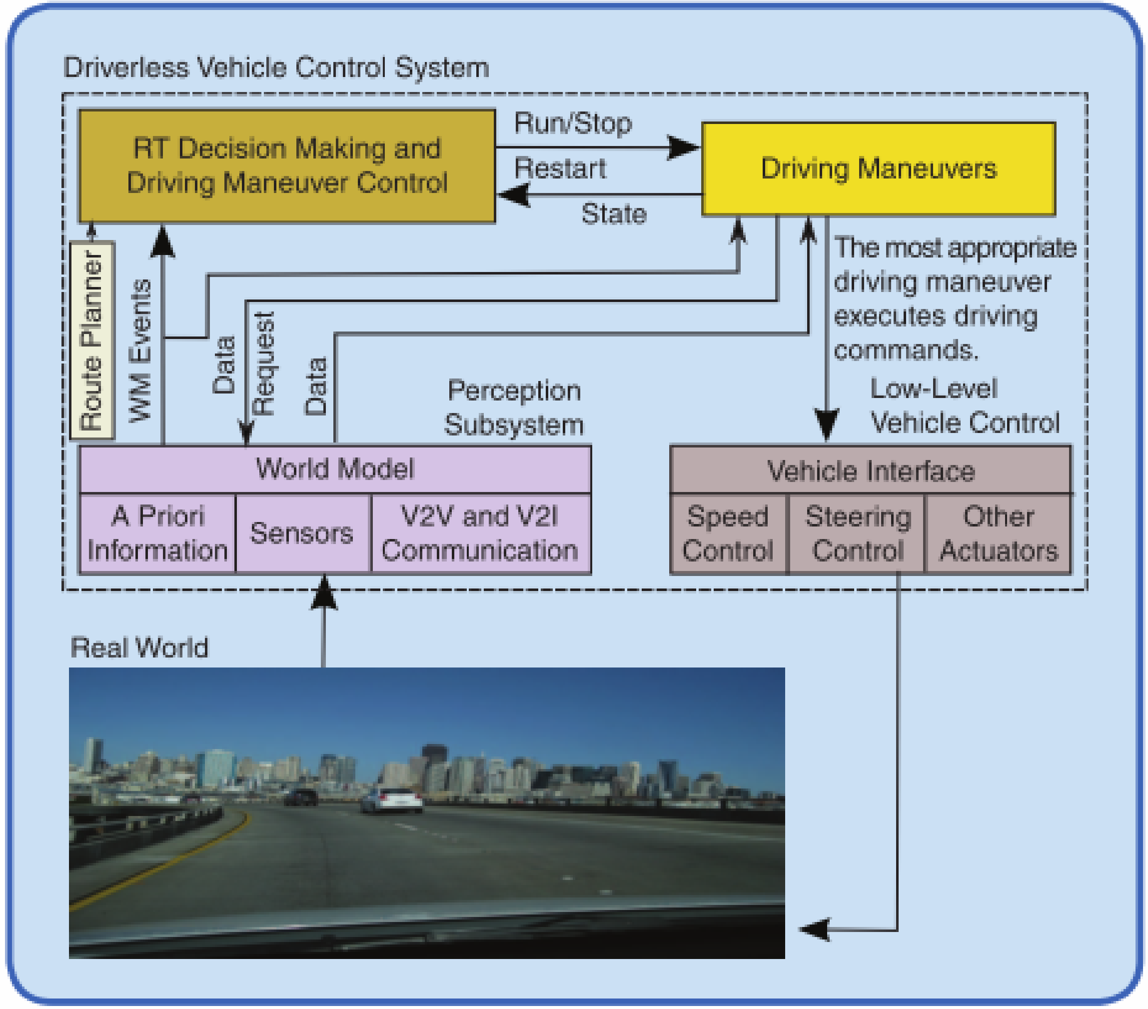


Fig. 2. **Finite State Machine:** We highlight different behavior states that are determined by the routing and optimization algorithms. When executing turns, the routing algorithm transitions the behavior state to a turning state. When the optimization-based maneuver algorithm plans a lane change, the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

Behavior Planner

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





Motion Planner

- Generally two stages:
  - Path planner - Computes the geometric representation of the path to be followed. I.e. the curve, spline, track, line, etc. we are following
  - Trajectory Planner / Path tracker - Computes the specific physical targets for following the path. I.e. velocity, acceleration, heading, steering, etc.

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

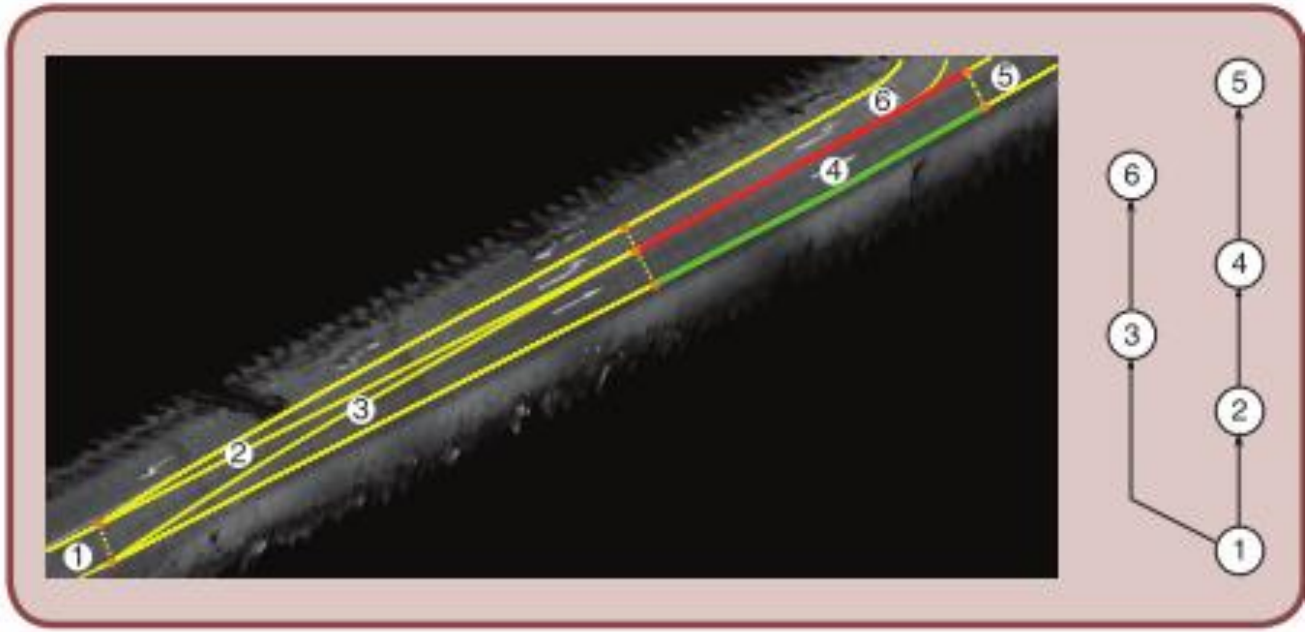
- Basic overview
  - Complete planning
  - Combinatorial Planning - discrete planning over an exact decomposition of the configuration space
    - Exponential in dimensions of c-space discretization (curse of dimensionality)
    - "resolution complete"
  - Sample-Based planning

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

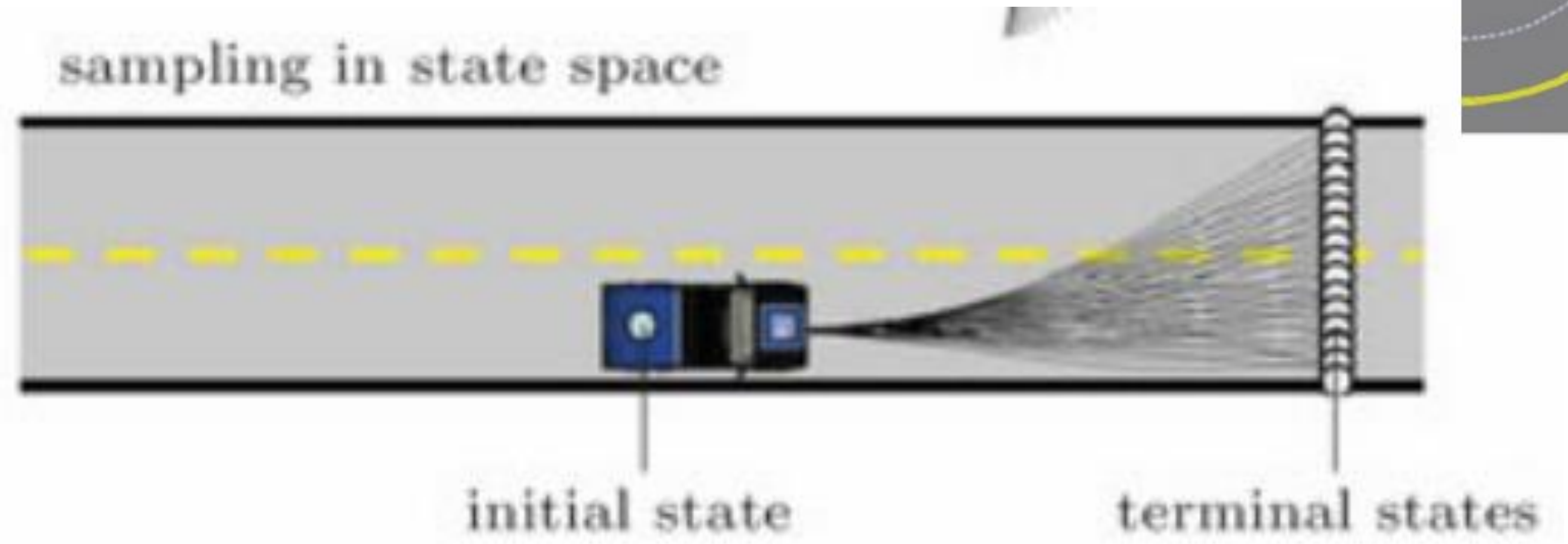
Motion Planner: Combinatorial Planners

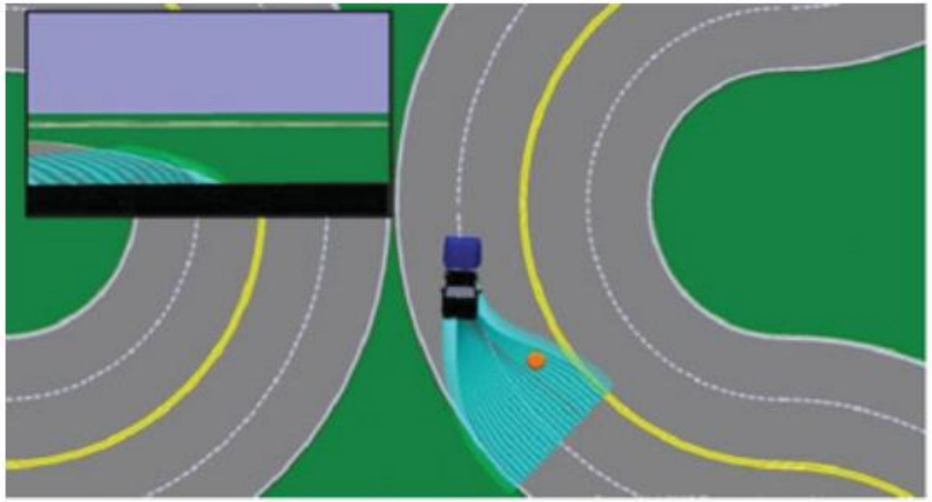
- Driving Corridors:
  - Decompose lanes into polygonal lanelets
  - Represent obstacles as polygonal bounding boxes or overlapping discs
  - Adjust lanelets to obstacle constraints



Motion Planner: Combinatorial Planners

- Darpa Urban Challenge:
  - ⑩ BOSS: kinodynamic reachable set





Maneuver Planner: Sample-based Planners

- Sample-based Planning specifically for cars:
  - Dynamics computation
  - Inevitable collision states
  - “Space-time planning approaches”
- 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

- RRT:
  - ➓ 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 δ towards the point
    - If n to $n + \delta$ is CLEAR, connect to the tree

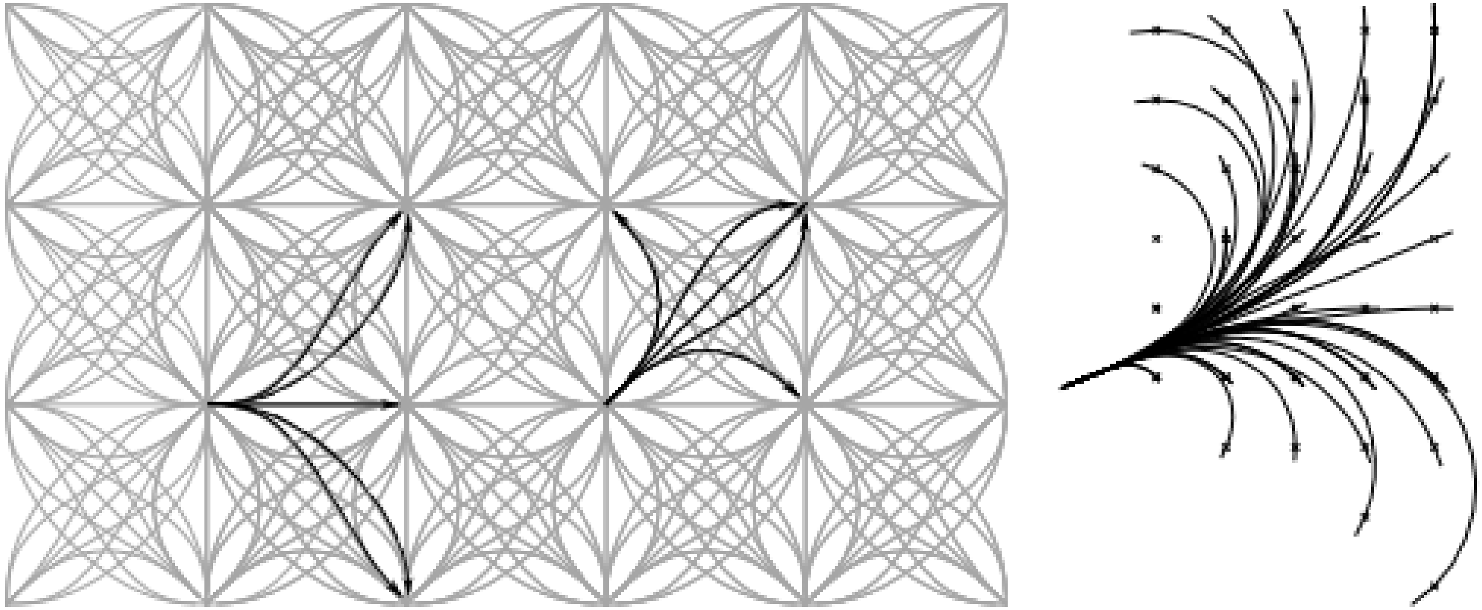
Maneuver Planner: Sample-based Planners

- State-lattice planners
  - Generate set of potential future states through solving boundary-value problem
  - Generate connected "lattice" of potential future states expanding in time and space

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Maneuver Planner: Sample-based Planners

- State-lattice planners
  - Ex: Configurations in space

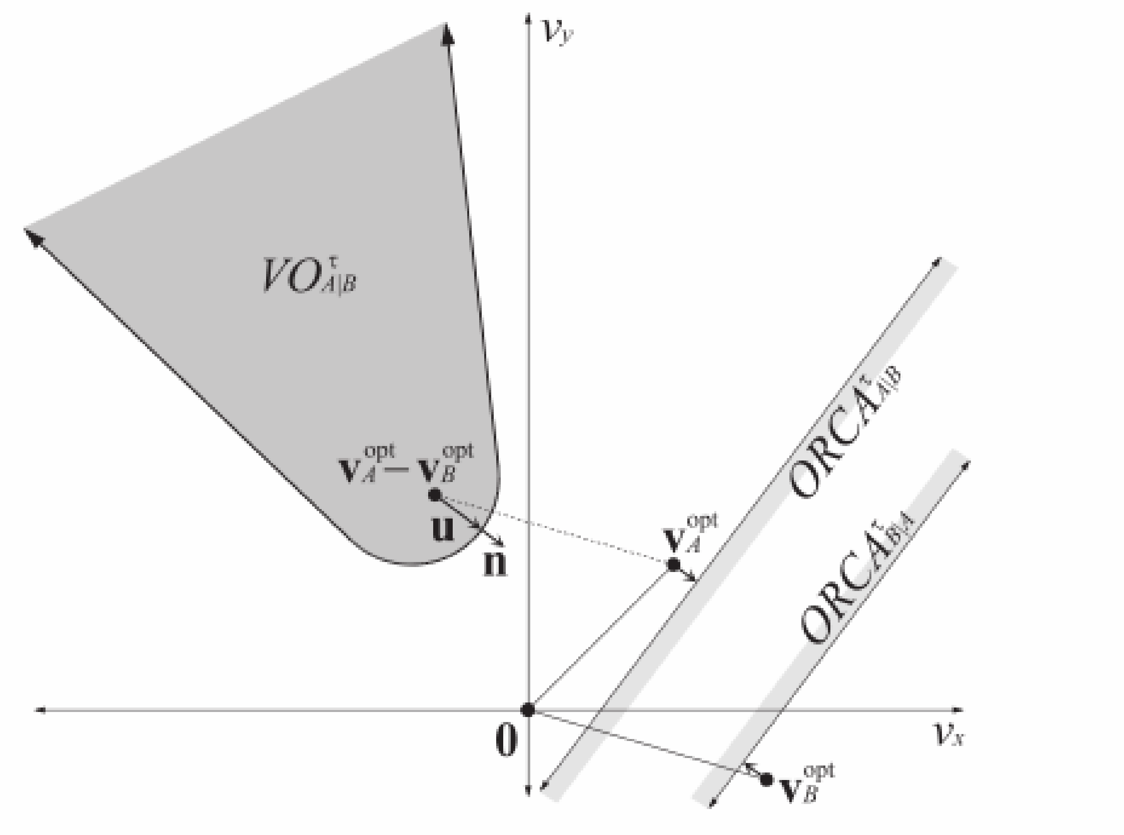


(a) (b)

Maneuver Planner: Obstacle Representation

- RVOs: Reciprocal-velocity Obstacles
  - Constructs mutually exclusive velocity set choices for multiple robots

  - https://youtu.be/1Fn3Mz6f5xA?t=1m24s



Structure

- Recap
- Control
    - **Core concepts**
    - PID
    - MPC
- Traffic-Sim
- Prediction

Autonomous Driving: Main Components

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

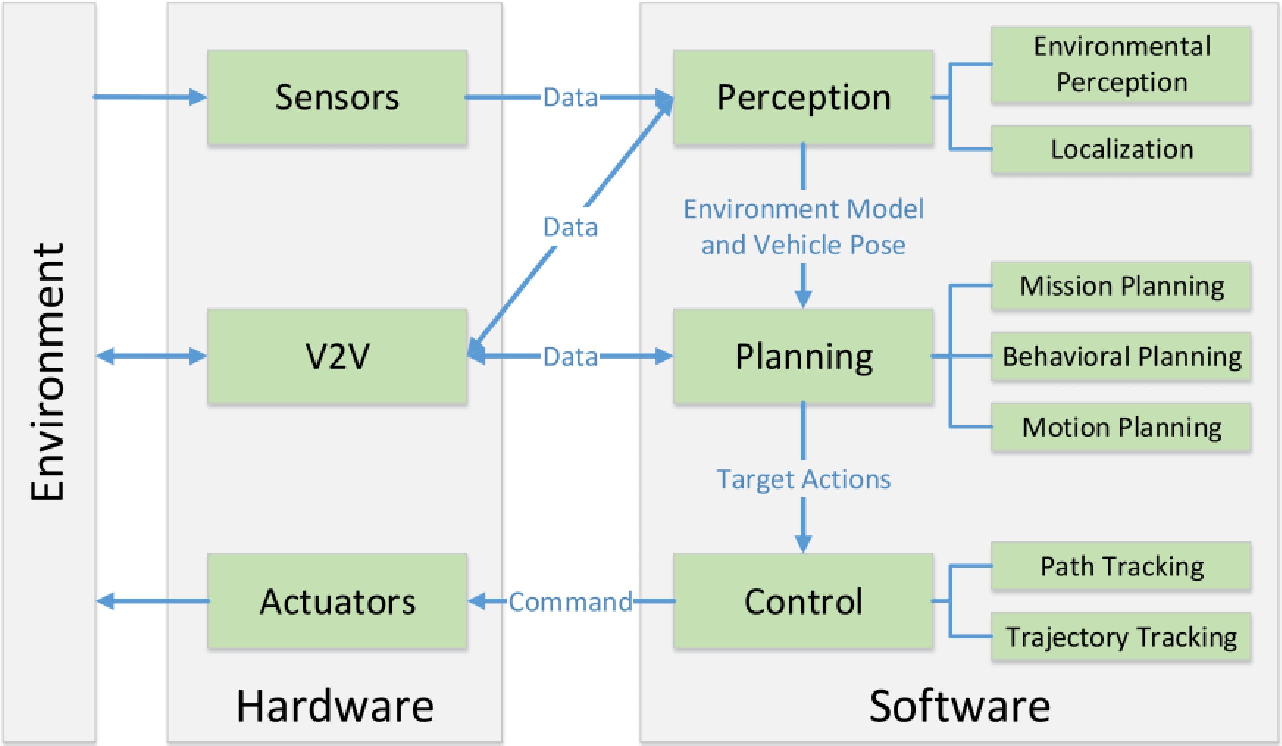


Figure 2. A typical autonomous vehicle system overview, highlighting core competencies.

Control: Core Concepts

- Automatic control in engineering and technology is a wide generic term covering the application of mechanisms to the operation and regulation of processes without continuous direct human intervention
  - Open-loop control: Control input delivered independent of measurements
  - Closed-loop control: Control input determined by system outputs

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

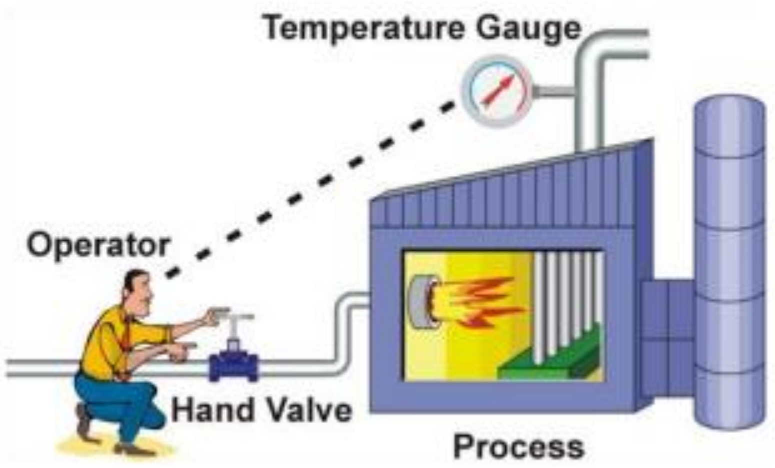


Figure 1
An Operator Performing Manual Control

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

Control: Core Concepts

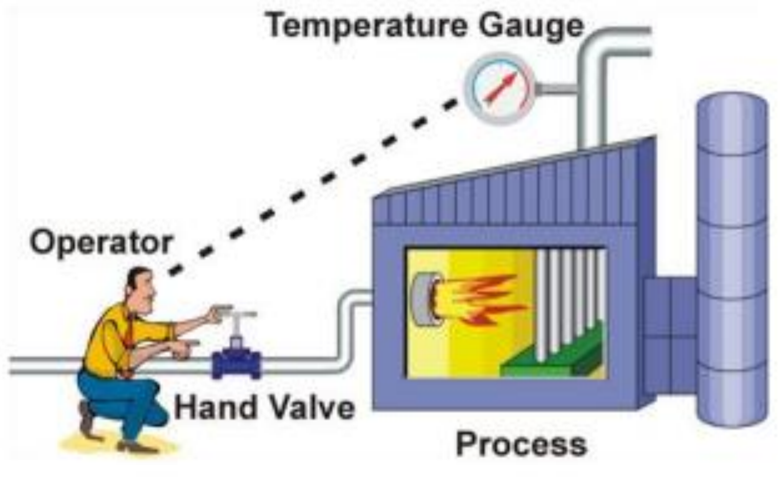
- Can we replace the manual control with automatic controller?
- 

Figure 1
An Operator Performing Manual Control

->

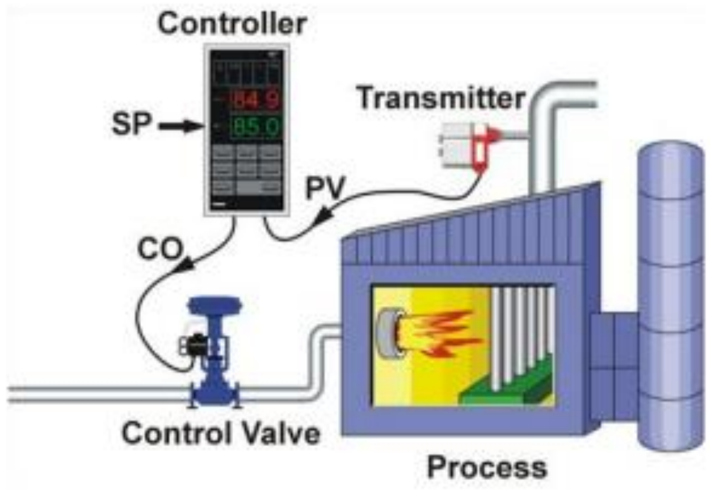


Figure 2
A PID Controller Performing Automatic Control

- Of course, we can!

Structure

- Recap
- Control
  - Core concepts
  - **PID**
  - MPC
  - Path Tracking
- Traffic-Sim
- Prediction

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

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.

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(\tau)\, d\tau + K_\mathrm{d} \frac{de(t)}{dt}$$

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

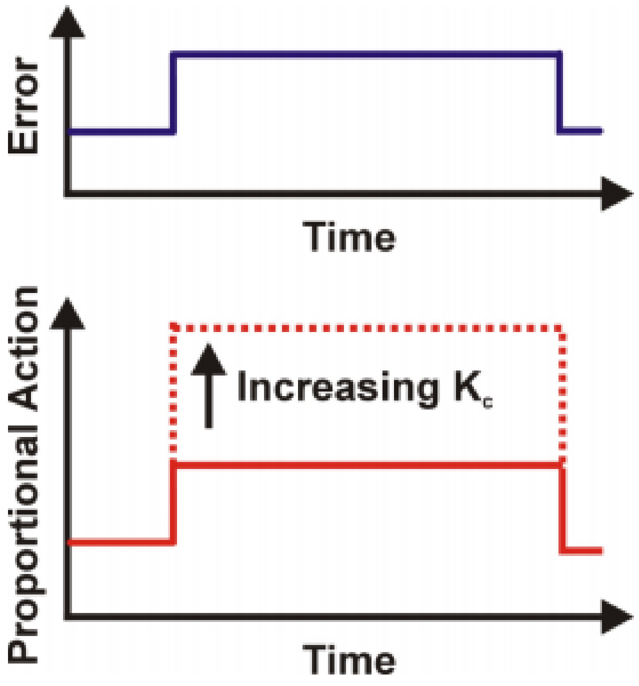


Figure 3
Proportional Control Action

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

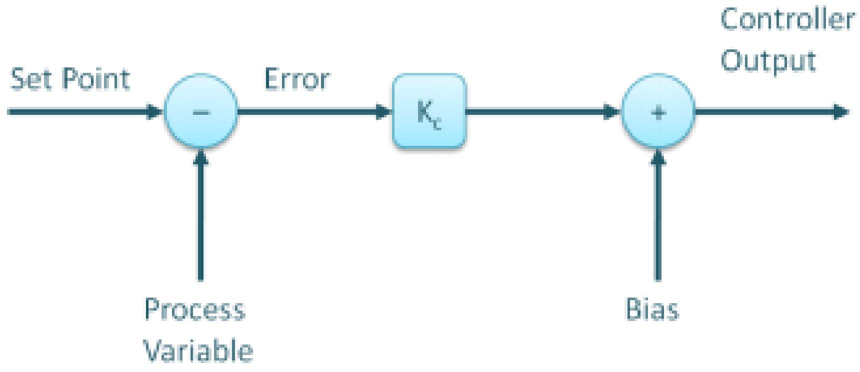


Figure 4
A Proportional-Only Controller Algorithm

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Control: PID

- P-only controller
  - Bias controls steady output

- https://sites.google.com/site/fpgaandco/pid

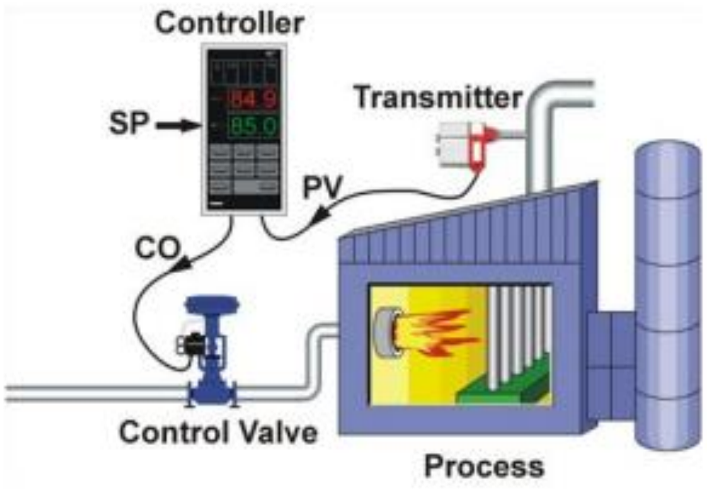


Figure 2
A PID Controller Performing Automatic Control

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

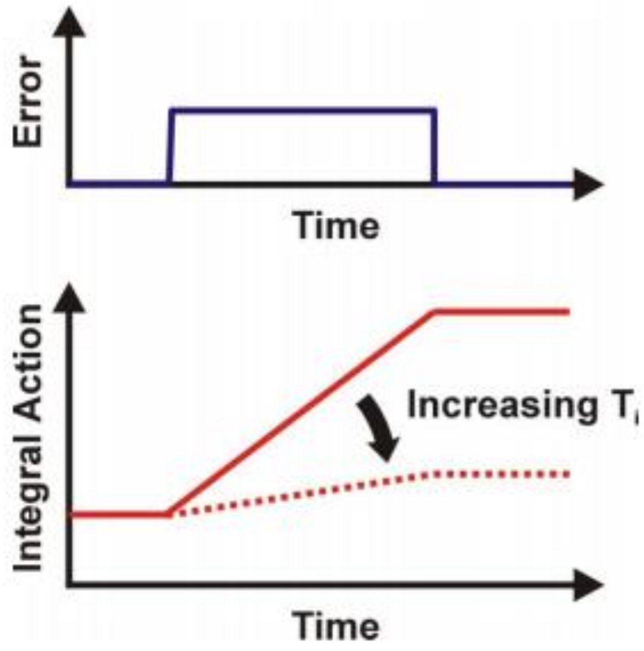


Figure 7
Integral Control Action

Control: PID

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

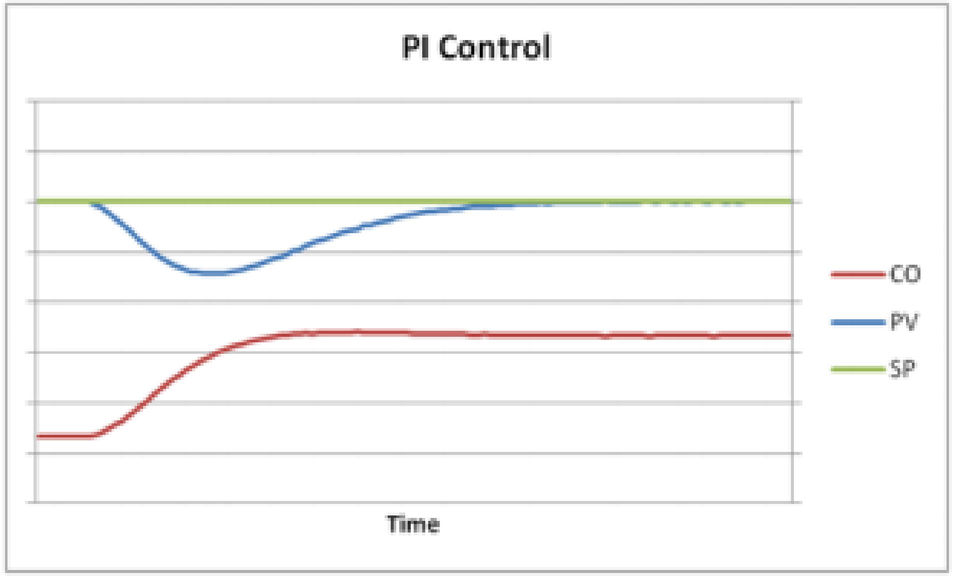


Figure 9
A PI Controller's Response to a Disturbance

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

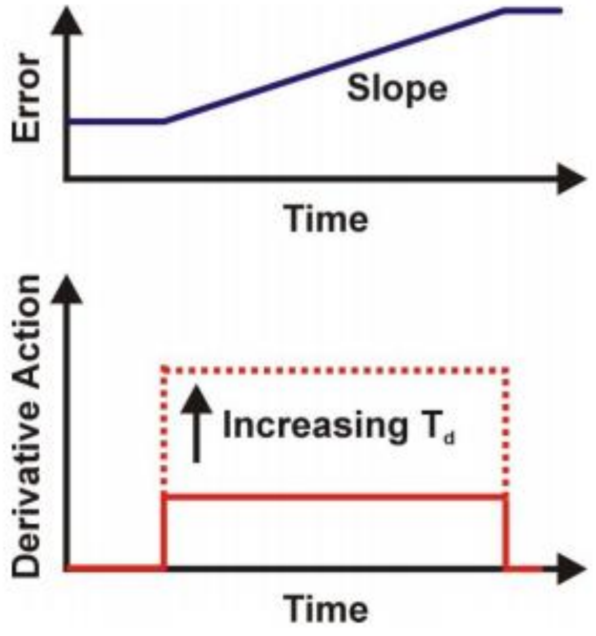


Figure 10
Derivative Control Action

Control: PID

- PID Controller: Proportion, Integral, Derivative terms
- Complete closed-loop controller
  - Used in AutonoVi and countless applications

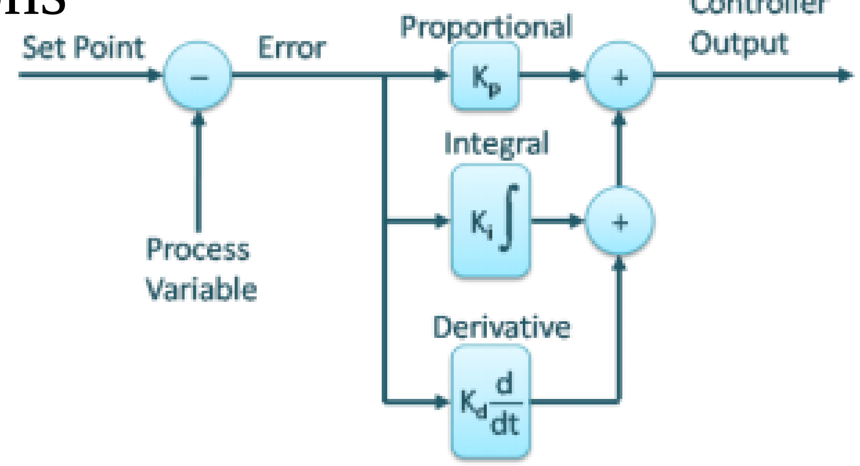


Figure 12
The Parallel PID Controller Algorithm

THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Control: PID Tuning

- Rules of thumb for tuning a PID controller:
  - https://upload.wikimedia.org/wikipedia/commons/3/33/PID_Compensation_Animated.gif

Effects of *increasing* a parameter independently[20][21]

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if K_d small

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

Ziegler–Nichols method

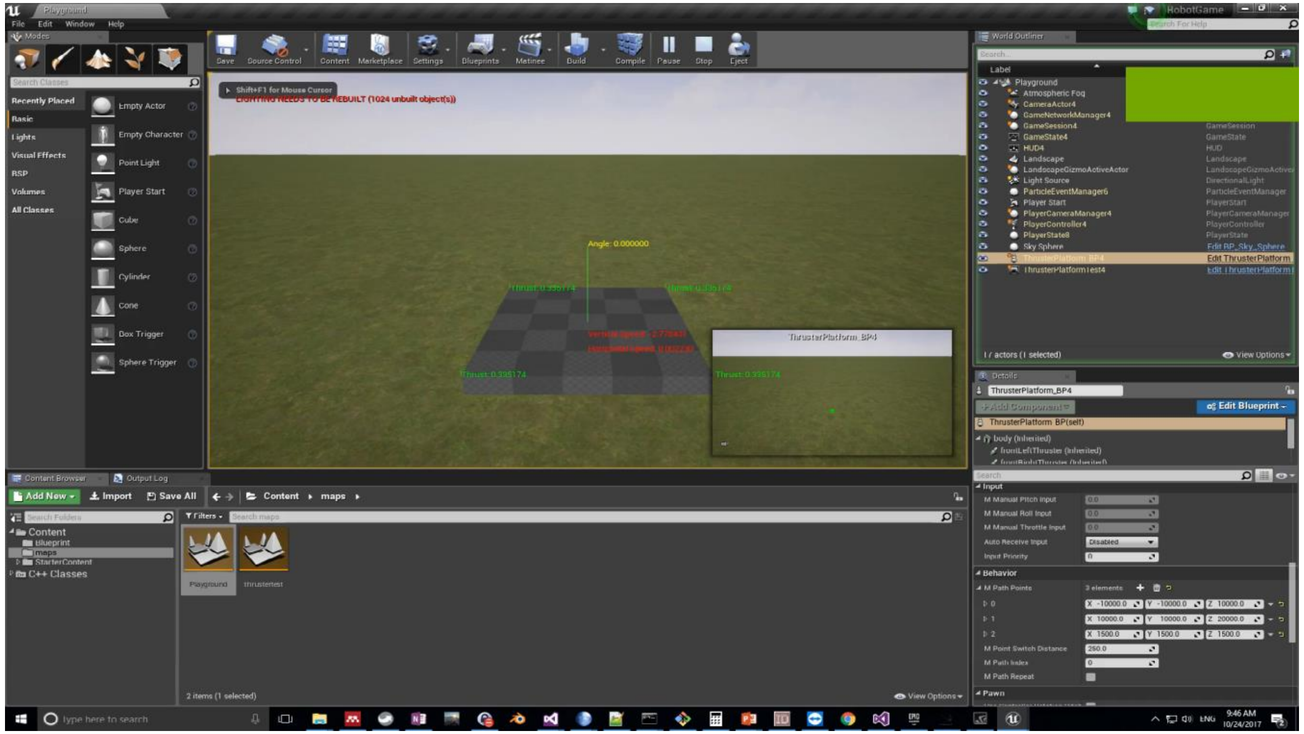
Control Type	K_p	K_i	K_d
P	$0.50K_u$	—	—
PI	$0.45K_u$	$0.54K_u/T_u$	—
PID	$0.60K_u$	$1.2K_u/T_u$	$3K_uT_u/40$

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

Control: PID Examples

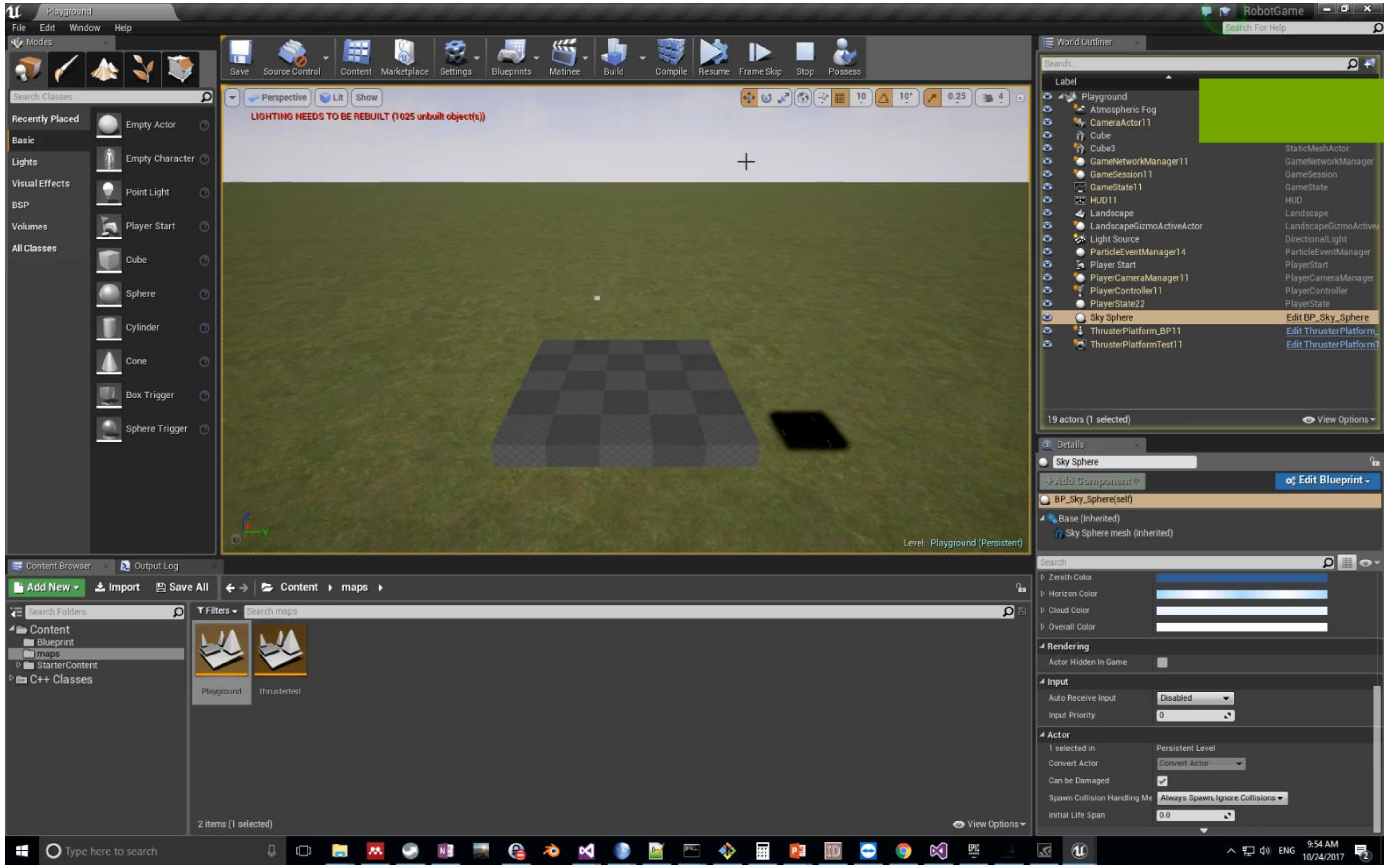
- PID for QuadRotor
  - Pure pursuit
  - Target speed specified
  - 2 layer PID
    - 1. Mix rotors for vertical speed
    - 2. Mix rotors for horizontal speed



THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Control: PID Examples

- PID for QuadRotor
  - Robust to perturbation



Structure

- Recap
- Control
  - Core concepts
  - PID
  - **MPC**
  - Path Tracing
- Traffic-Sim
- Prediction

Control: MPC

- Model-Predictive Controller: control loop relying on an underlying system model to generate feed-forward control
  - Augment feedback control system to generate predicted future values and predicted control outputs
  - Non-linear systems typically linearized over small timescales of MPC
  - https://www.youtube.com/watch?v=oMUtYZOgsng
    - Very good introduction
  - https://www.youtube.com/watch?v=DFqOf5wbQtc
    - Lecture series is helpful for MPC

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

- Recap
- Control
  - Core concepts
  - PID
  - MPC
  - **Path Tracking**
- Traffic-Sim
- Prediction

Control: Path tracking with controllers

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

Control: Path tracking with controllers

- Pure-pursuit
  - Given a geometric path, track a point ahead of the vehicle according to a fixed lookahead (can be a function of speed)
  - https://www.youtube.com/watch?v=qG70QJJ8Qz8
  - https://www.youtube.com/watch?v=vlyTthJugRQ

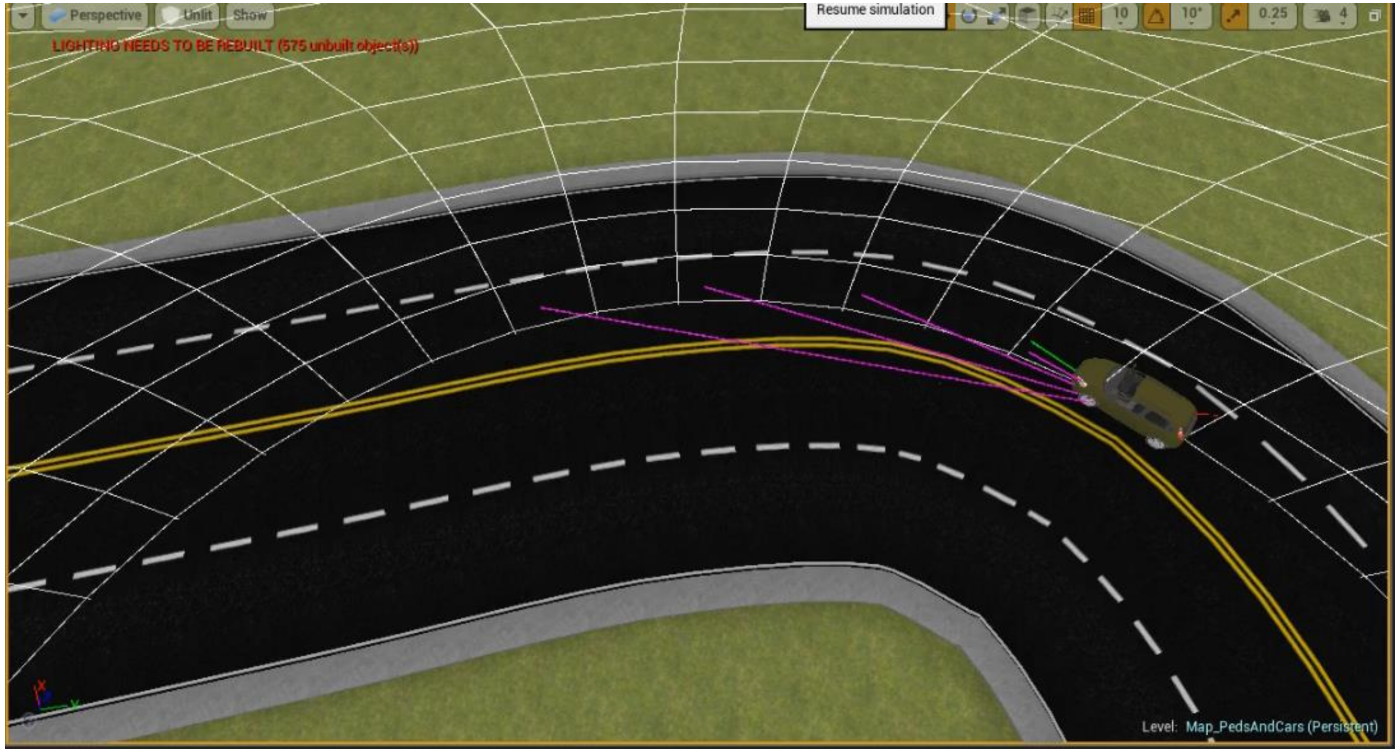
- Advantages: simple, robust to perturbation
- Disadvantages: Corner-cutting, oscillation for non-holonomic robots

Control: Path tracking with controllers

- AutonoVi
  - 2nd order pure-pursuit PID
  - Vehicle position + 2 points ahead on center of lane, trace arc between them
- Advantages: simple, robust to perturbation, can represent kinematic limits in computed curves
- Disadvantages: oscillation, prone to wide-turns, curvature prone to large shifts

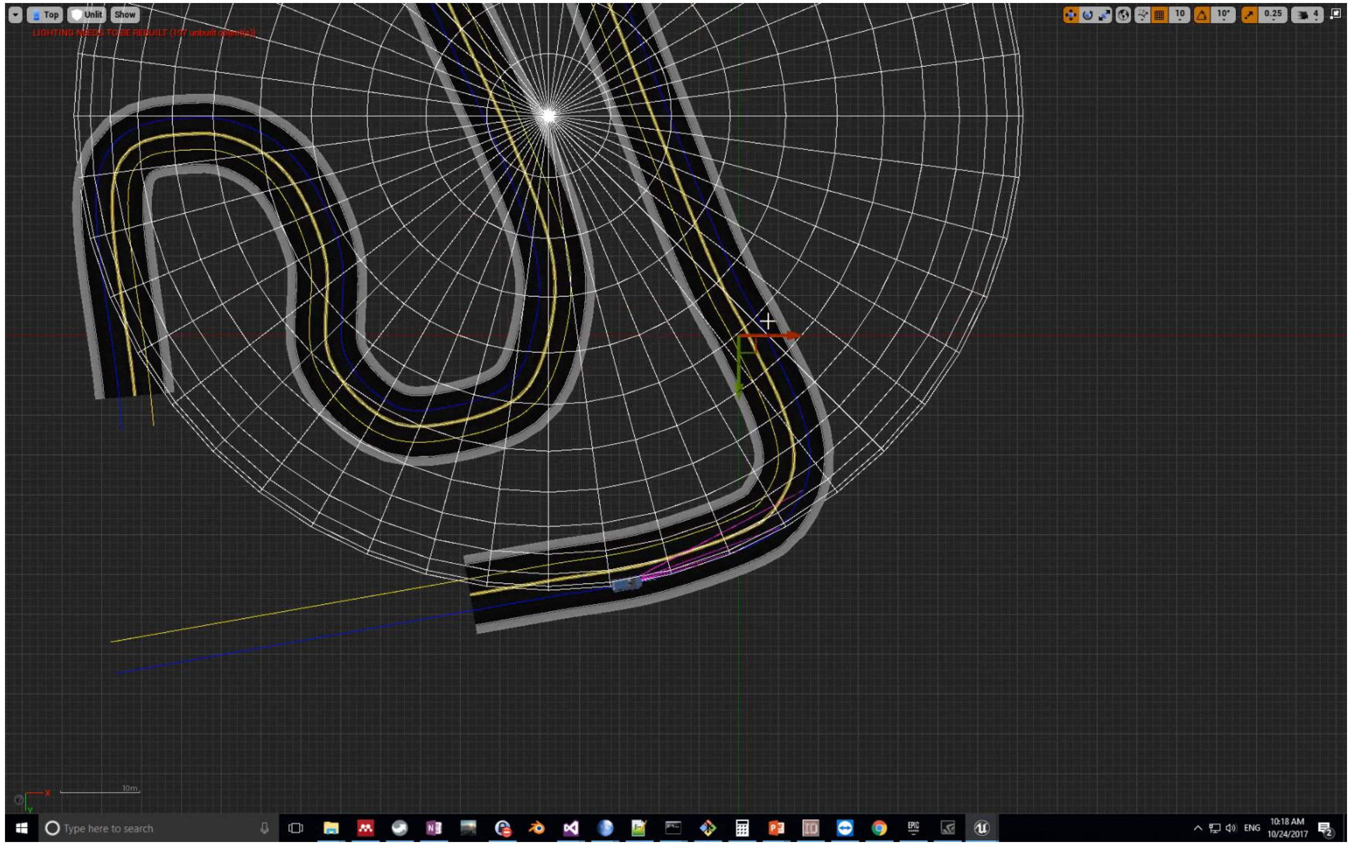
Control: Path tracking with controllers

- AutonoVi



Control: Path tracking with controllers

- AutonoVi



Control: Path tracking with controllers

- AutonoVi
  - ⑩ NOTE: controllers have been demonstrated using arbitrary degree polynomials from N points on the path
    - Trade-offs in computational speed, robustness to perturbation, look-ahead computation

Control: Path tracking with controllers

- Kinematic Car [De Luca 1998]
  - Attempts to simultaneously minimize heading error and cross-track error (distance to reference point on path)
  - Heading measured as path tangent orientation

$$\theta_e = \theta - \theta_p(s)$$
$$\dot{s} = v\cos(\theta_e) + \dot{\theta}_p e_{ra}$$
$$\dot{e_{ra}} = v\sin(\theta_e)$$

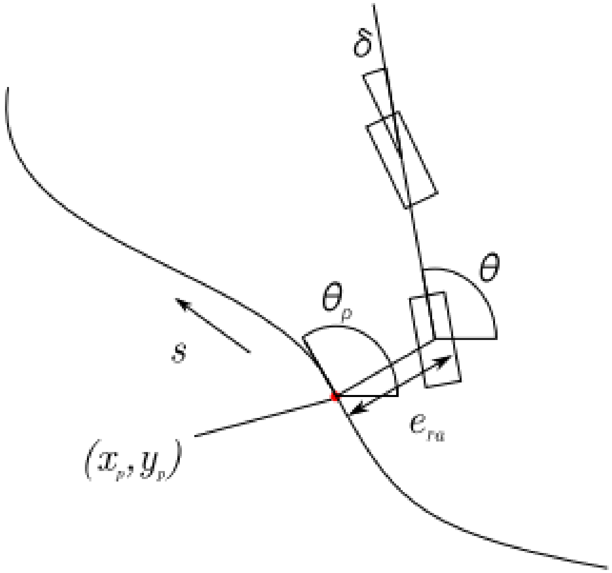


Figure 9. Path representation for kinematic car model

Control: Path tracking with controllers

- Kinematic Car [De Luca 1998]
  - Rewrite kinematics in "path coordinates"

$$\begin{bmatrix} \dot{s} \\ \dot{e}_{ra} \\ \dot{\theta}_e \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \frac{cos(\theta_e)}{1-\dot{e}_{ra}\kappa(s)} \\ sin(\theta_e) \\ (\frac{tan\delta}{L}) - \frac{\kappa(s)cos(\theta_e)}{1-\dot{e}_{ra}\kappa(s)} \\ 0 \end{bmatrix} v \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \dot{\delta}$$

  - Goal becomes maximizing $\dot{s}$ while minimizing $\dot{e}_{ra}$ and $\dot{\Theta}_e$

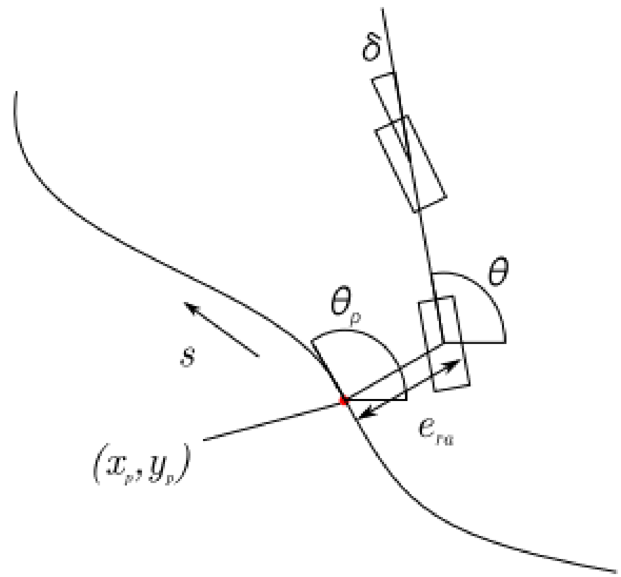


Figure 9. Path representation for kinematic car model

De Luca, A., Oriolo, G., & Samson, C. (1998). Feedback control of a nonholonomic car-like robot, 171–253. http://doi.org/10.1007/BFb0036073

Control: Path tracking with controllers

- Model-predictive
  - Given a model, i.e. kinematic car, perform repeated optimization over future states to determine optimal control
- Advantages:
  - Robust to disturbance, reduces oversteer, requires model
- Disadvantages:
  - Computationally expensive, model mismatch exacerbates errors

- In my experience: a bad model in MPC performs worse than PID!

Control: Path tracking with controllers

- Model-predictive
  - Examples:
  - https://youtu.be/Bk7ES3Qd53s
  - https://youtu.be/C5UILYChPAc
  - https://youtu.be/5-hvtxeZNbo

  - Code at: https://github.com/parilo/CarND-MPC-Project

Structure

- Recap
- Control
- Traffic-Sim
  - MATSim
  - Sumo
  - Hybrid Simulation
- Prediction

Traffic-Sim: Rationale

- Understand infrastructure
  - Evaluate efficiency of proposed changes to roads
  - Evaluate congestion points, failures, and improvements for existing roads
  - Test traffic control algorithms

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Traffic-Sim: Methods

- Agent-based:
  - Macroscopic: agents represented without physics or kinematics
    - Roads treated as edges in directed graph
    - Many agents supported, limited interactions
  - Microscopic: agents represented with kinematics or physics
    - Roads modelled with physical dimensions
    - Few agents supported, interactions can be modelled dynamically

Traffic-Sim: Methods

- Flow-based:
  - Agents not explicitly represented
  - Flow computed over network, system evolves as “fluid” simulation

Structure

- Recap
- Control
- Traffic-Sim
  - MATSim
  - Sumo
  - Hybrid Simulation
- Prediction

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Traffic-Sim: MATSim

- Agent-based, Macroscopic simulation
- Supports millions of vehicles
- https://vimeo.com/124704874
- https://youtu.be/VowP4f9ntCA?t=42s
- https://youtu.be/VowP4f9ntCA?t=5m28s
- https://youtu.be/o60A4r6sSsE?list=PLLGIZCXnKbU6-9vy_rKZ6gW7E_ra42hfX

Traffic-Sim: MATSim

- Features:
  - Millions of agents
  - Route import from loop detectors / traffic data
  - OpenStreetmap Import
- Benefits:
  - Macro-scale modelling replicates usage data gathered over long periods
  - Simulation of alternate routes and large time-scales simply
  - Evaluate macro changes: for example, starting school 30m later

Structure

- Recap
- Control
- Traffic-Sim
    - MATSim
    - **SUMO**
    - Hybrid Simulation
- Prediction

Traffic-Sim: SUMO

- Agent-based, Microscopic simulation
- Allows for modeling lane configuration, route-planning, vehicle size and shapes, preliminary pedestrians
- Online control and modification of network
- https://youtu.be/KgPSREMmA_0
- https://youtu.be/a52U6CQQRcw?t=24s
- https://youtu.be/qewufs0Xsq0

Traffic-Sim: SUMO

- Notable Features:
  - OpenStreetmap Import, automatic processing of lane connectivity
  - Control and physics free
  - Multiple driver models, “person level” transport options

- Benefits:
  - Allows detailed testing of traffic-lights and intersections
  - Widely used for V2X communication research

Structure

- Recap
- Control
- Traffic-Sim
  - MATSim
  - SUMO
  - **Hybrid Simulation**
- Prediction

Traffic-Sim: Hybrid & Flow Models

- Non-agent based models
  - Treat traffic as flow model, like liquid
  - Continuum formulation evolves road network
  - Allows for immense networks, but limits the ability to represent agentive phenomena

Traffic-Sim: Hybrid & Flow Models

Traffic-Sim: Hybrid & Flow Models

- Hybrid models
  - “Best of both worlds”
    - continuum evolution for “distant” traffic phenomena
    - Agent-based simulation for nearby vehicles
  - Captures driver behavior in micro-scale and accurately models aggregate information
  - https://www.youtube.com/watch?v=eEnGFxfN2tE
  - see me after class for more papers

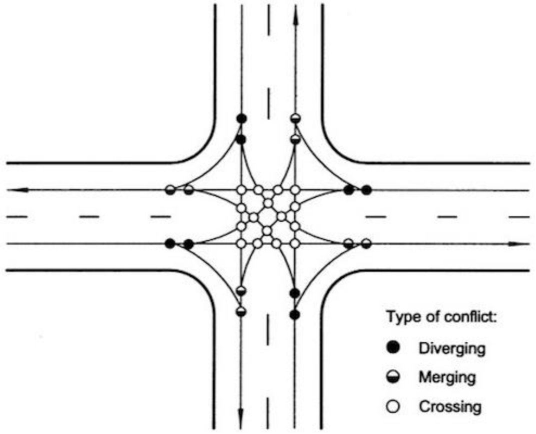
Structure

- Recap
- Control
- Traffic-Sim
  - MATSim
  - SUMO
  - Hybrid Simulation
- Prediction

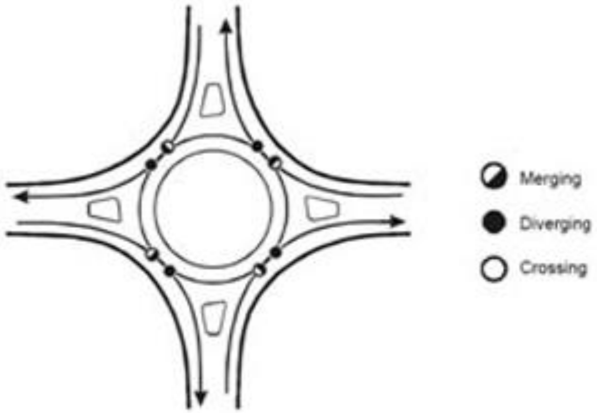
Traffic-Sim: Recent Applications

https://www.citylab.com/solutions/2013/01/could-these-crazy-intersections-make-us-safer/4467/?utm_source=SFFB
good article

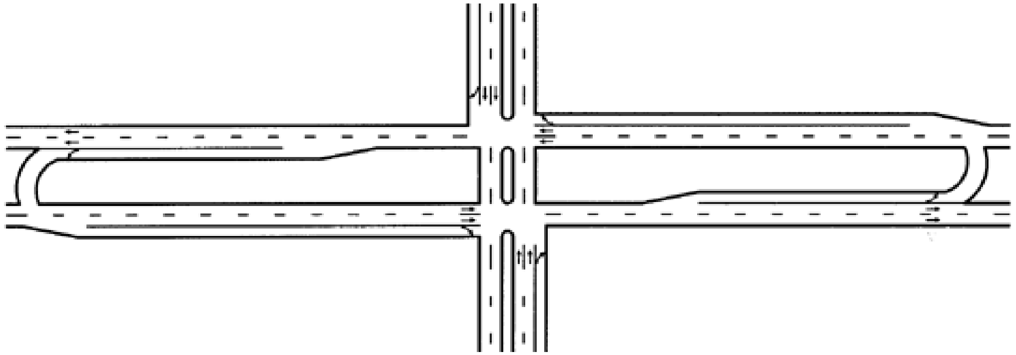
- Safer intersections:
  - Geometric analysis performed to determine intersection danger
  - Design intersections which optimize flow & limit intersection points



32 collision points



8 collision points

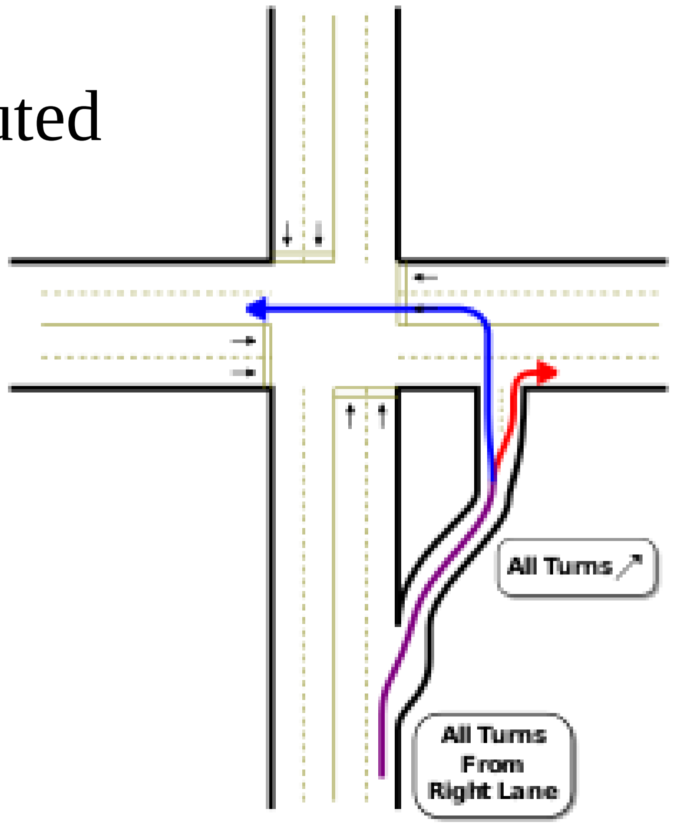


18 collision points?

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

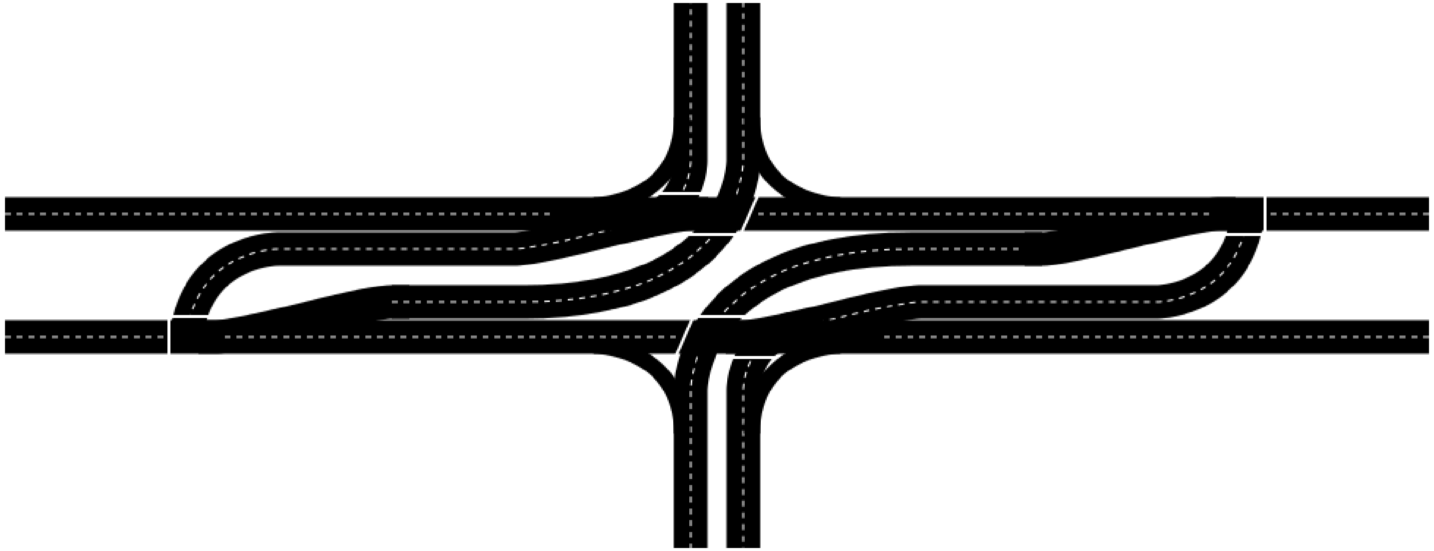
Traffic-Sim: Recent Applications

- Jughandle:
  - Turns from minor road executed at special “handle”
  - https://vimeo.com/58011852



Traffic-Sim: Recent Applications

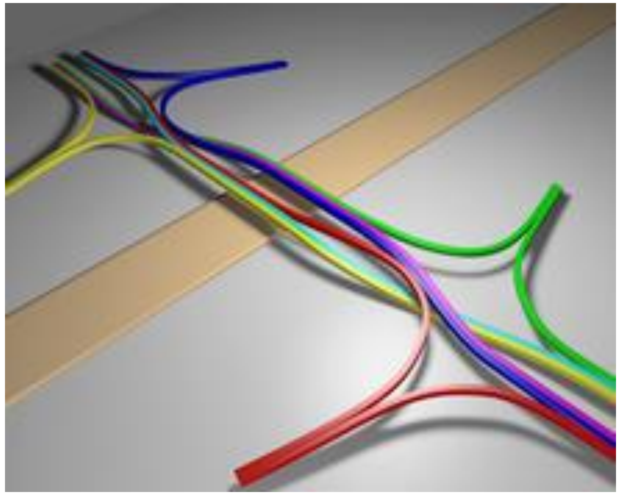
- Superstreet:
  - Minor road NOT ALLOWED to cross major road
  - https://vimeo.com/57973069

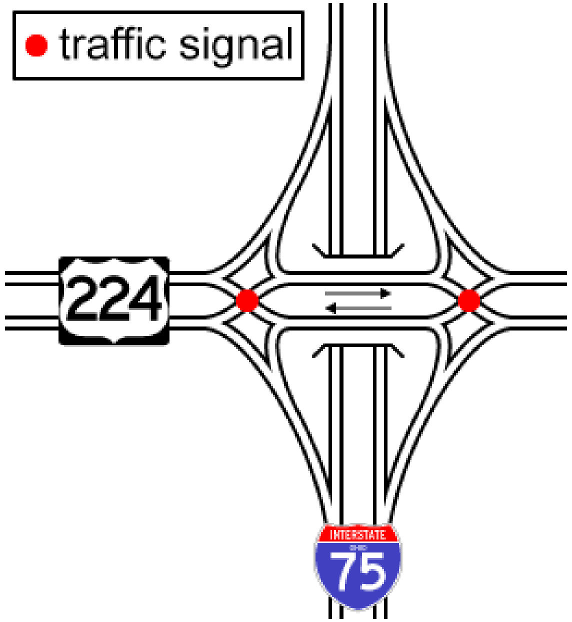


THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Traffic-Sim: Recent Applications

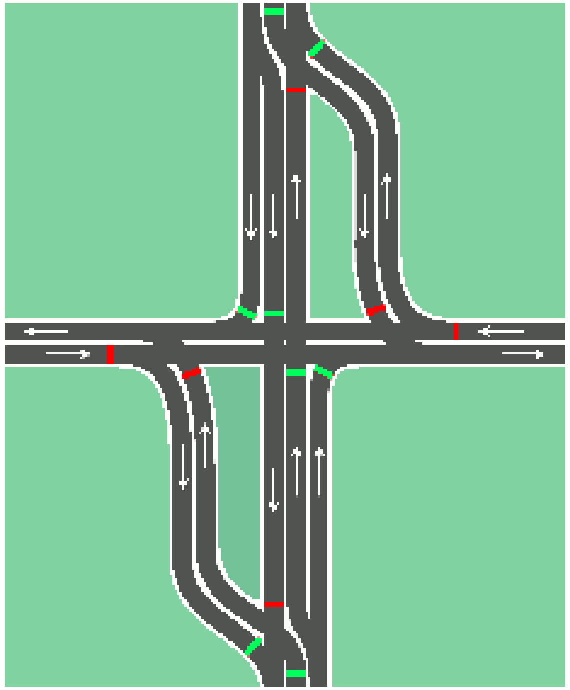
- Diverging Diamond:
  - Minor road crosses in X pattern
  - Allows continuous flow in 2 directions
  - https://vimeo.com/57972903





Traffic-Sim: Recent Applications

- Continuous Flow:
  - Left turns pre-cross oncoming lanes
  - I grew up with one of these
  - https://vimeo.com/57973241
  - https://vimeo.com/57973040



THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Traffic-Sim: Recent Applications

- What should traffic lights look like for AVs?
  - Are they needed at all?
  - How do we optimize for mixed AV and non-AV traffic?
  - https://vimeo.com/37751380
- Great resources at https://goo.gl/3YUY2o

Structure

- Recap
- Control
- Traffic-Sim
- **Prediction**

Limitations in Planning

- Most autonomous navigation algorithms
  - Defensive
  - Opaque
  - Do not consider “interactions” with other participants
  - Assume a very simple model for estimating movement of other cars

- Drivers have a tendency to rear end self-driving cars on the road [Consumer Affairs]
  - 19 such crashes out of 285 Waymo vehicles in CA in 2017

Structure

- Interaction-based planning
  - Formal framework for 2-way interactions
  - Probabilistic reasoning for multi-vehicle interactions

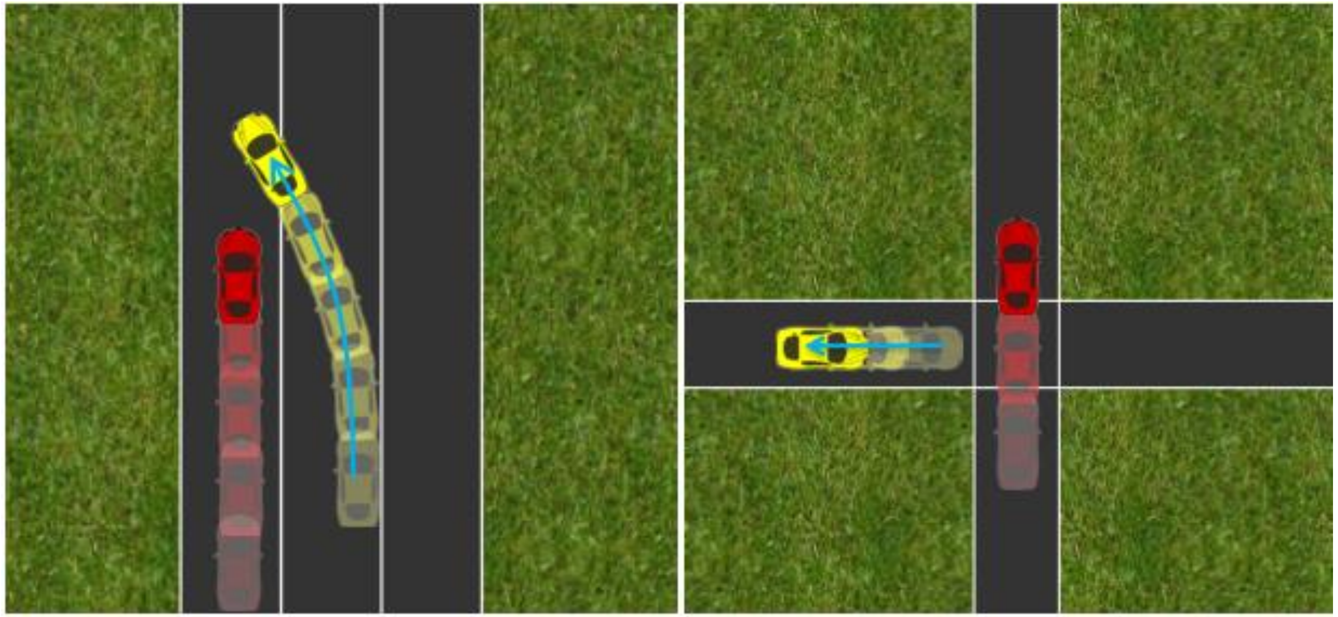
Formal Framework for 2-way interactions

- Dorsa Sadigh, Shankar Sastry, Sanjit A. Seshia, and Anca D. Dragan. **Planning for Autonomous Cars that Leverages Effects on Human Actions.** In Proceedings of the Robotics: Science and Systems Conference (RSS), June 2016.

- Our 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.

Formal Framework for 2-way interactions

- We model the interaction between an autonomous car and a human driver as a dynamical system, in which the robot's actions have immediate consequences on the state of the car, but also on human actions.



(a) Car merges *ahead* of human; anticipates human *braking*

(b) Car *backs up* at 4way stop; anticipates human *proceeding*

Formal Framework for 2-way interactions

- Let x represent the state of the system, which includes positions and velocities of the human and autonomous robot.
- Effect of robot controls: $x' = f_{\mathcal{R}}(x, u_{\mathcal{R}})$
- Effect of human actions: $x'' = f_{\mathcal{H}}(x', u_{\mathcal{H}})$

- Overall dynamics of the system: $x^{t+1} = f_{\mathcal{H}}(f_{\mathcal{R}}(x^t, u^t_{\mathcal{R}}), u^t_{\mathcal{H}})$

Formal Framework for 2-way interactions

- Formulate choosing robot controls as a reward maximization problem
- Reward function of robot

$$r_{\mathcal{R}}(x^t, u^t_{\mathcal{R}}, u^t_{\mathcal{H}}).$$

  - Reward function depends on u_H

- MPC used at every iteration
  - Let x^0 be the current state
  - Reward over MPC time horizon t is :

$$R_{\mathcal{R}}(x^0, \mathbf{u}_{\mathcal{R}}, \mathbf{u}_{\mathcal{H}}) = \sum_{t=1}^{N} r_{\mathcal{R}}(x^t, u^t_{\mathcal{R}}, u^t_{\mathcal{H}})$$

Formal Framework for 2-way interactions

- At every iteration, the robot needs to find the u_R that maximizes this reward:

$$\mathbf{u}_{\mathcal{R}}^* = \arg\max_{\mathbf{u}_{\mathcal{R}}} R_{\mathcal{R}}(x^0, \mathbf{u}_{\mathcal{R}}, \mathbf{u}_{\mathcal{H}}^*(x^0, \mathbf{u}_{\mathcal{R}}))$$

  - $\mathbf{u}_{\mathcal{H}}^*(x^0, \mathbf{u}_{\mathcal{R}})$ is what the human would do over the next N steps if the robot were to execute u_R .
- Typical solutions assume that the human will maintain current velocity

$$\mathbf{u}_{\mathcal{H}}^*(x^0, \mathbf{u}_{\mathcal{R}}) = \mathbf{u}_{\mathcal{H}}^*(x^0)$$

  - Instead they assume that humans would maximize their own reward function.

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Formal Framework for 2-way interactions

- Human driver reward
  - Use Inverse Reinforcement Learning (IRL) over driver demonstrations in simulation.
  - Assume a simple parameterization of human reward
- Given a human reward function
  - Solve the optimization problem using quasi-Newton methods like L-BFGS

$$\mathbf{u}_{\mathcal{R}}^{*} = \arg\max_{\mathbf{u}_{\mathcal{R}}} R_{\mathcal{R}}(x^0, \mathbf{u}_{\mathcal{R}}, \mathbf{u}_{\mathcal{H}}^{*}(x^0, \mathbf{u}_{\mathcal{R}}))$$

Experiments

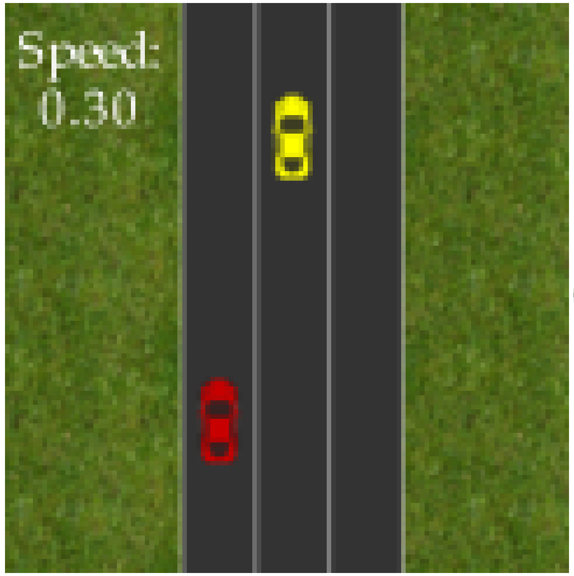
- Assume a simple dynamics model of the car.

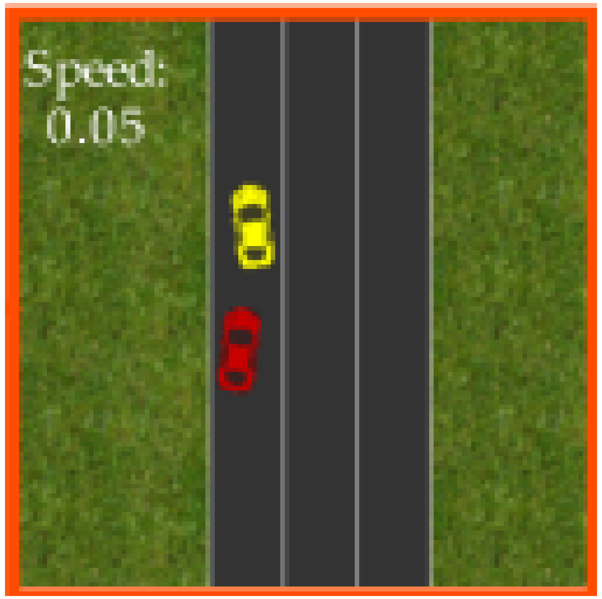
- 3 Scenarios
  - Make human slow down
  - Make human change lanes
  - Make human go first through intersection

- In each case, hand engineer the robot reward function achieve the desired effect on human behavior

Experiments: Make human slow down

- The robot plans to move in front of the person, expecting that this will make them slow down.
- Achieved by augmenting the robot's reward with the negative of the square of the human velocity.

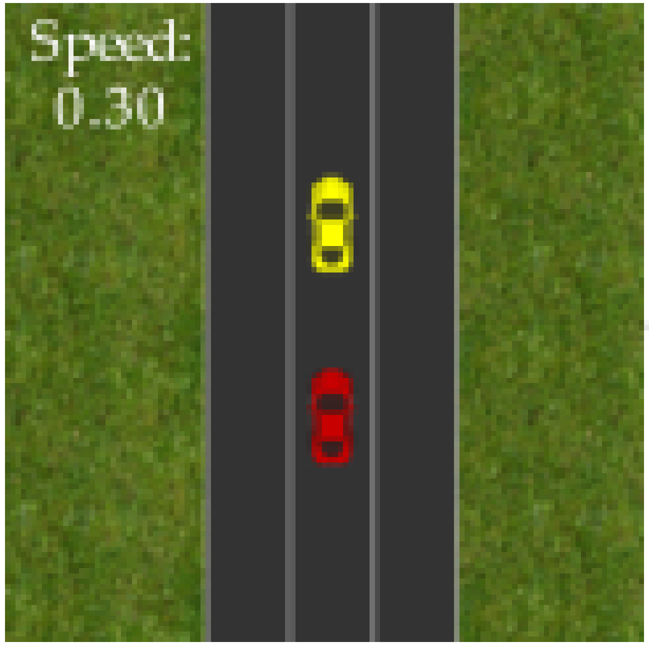


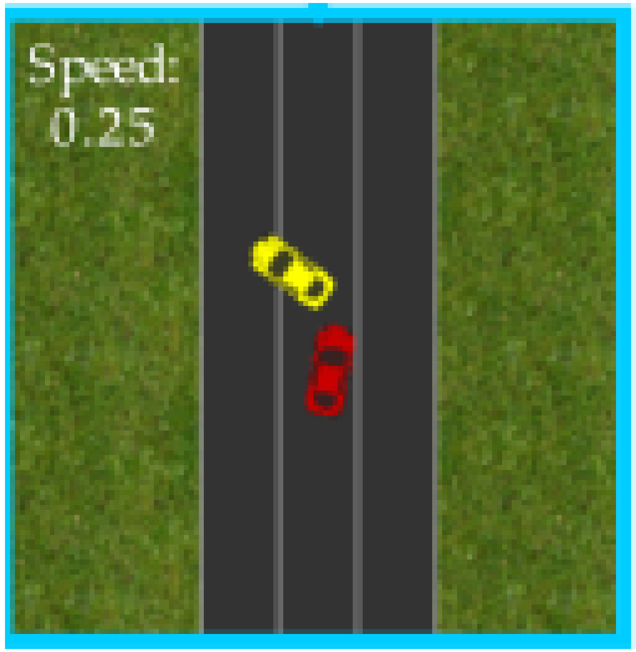


THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

Experiments: Make human change lanes

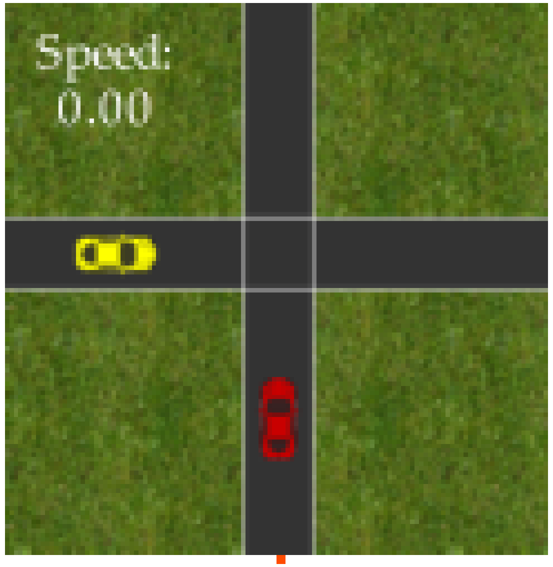
- The robot plans to purposefully occupy two lanes, expecting this will make the human move around it by using the unoccupied lane.
- Achieved by augmenting the robot's reward with the lateral position of the human

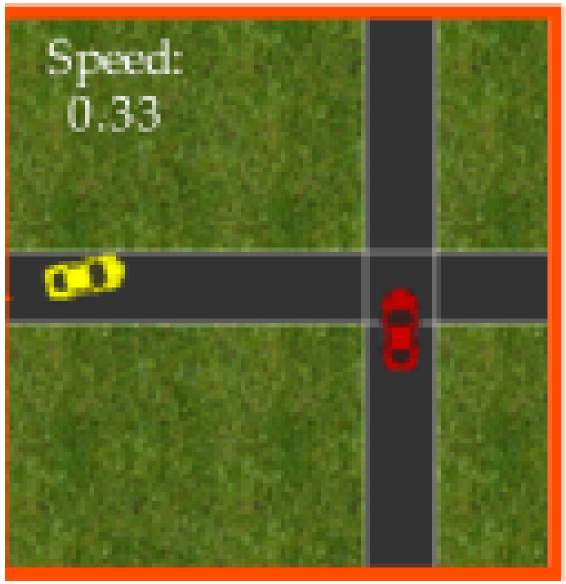




Experiments: Make human cross intersection first

- The robot plans to purposefully back up slightly, expecting this will make the human cross first.
- Achieved by augmenting the robot's reward with a feature based on the position of the human car relative to the middle of the intersection.
  - Communication behavior emerges naturally out of reward optimization.





Structure

- Interaction-based planning
  - Formal framework for 2-way interactions
  - Probabilistic reasoning for multi-vehicle interactions

Behavior Prediction

- Galceran, E., Cunningham, A. G., Eustice, R. M., & Olson, E. (2017). **Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment.** *Autonomous Robots*, 1-16.

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

Behavior Prediction

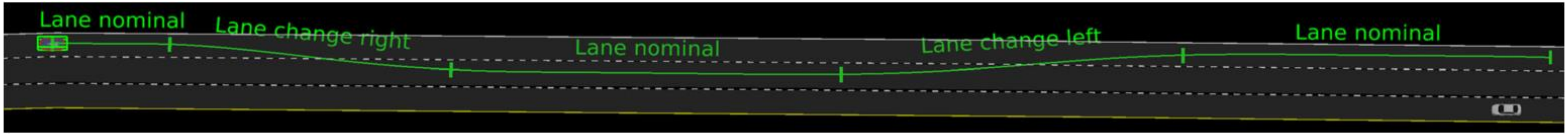
- Assume a set of a priori known policies
  - Go straight, change lanes, merge left, merge right etc

- Leverage Bayesian change-point detection to estimate the policy that a given vehicle was executing at each point in its history of actions.
  - Given current policy, infer the likelihood of actions or intentions.

- Statistical test for detecting anomalous behaviors.

Behavior Prediction

✦ Bayesian change-point detection over 30 s windows

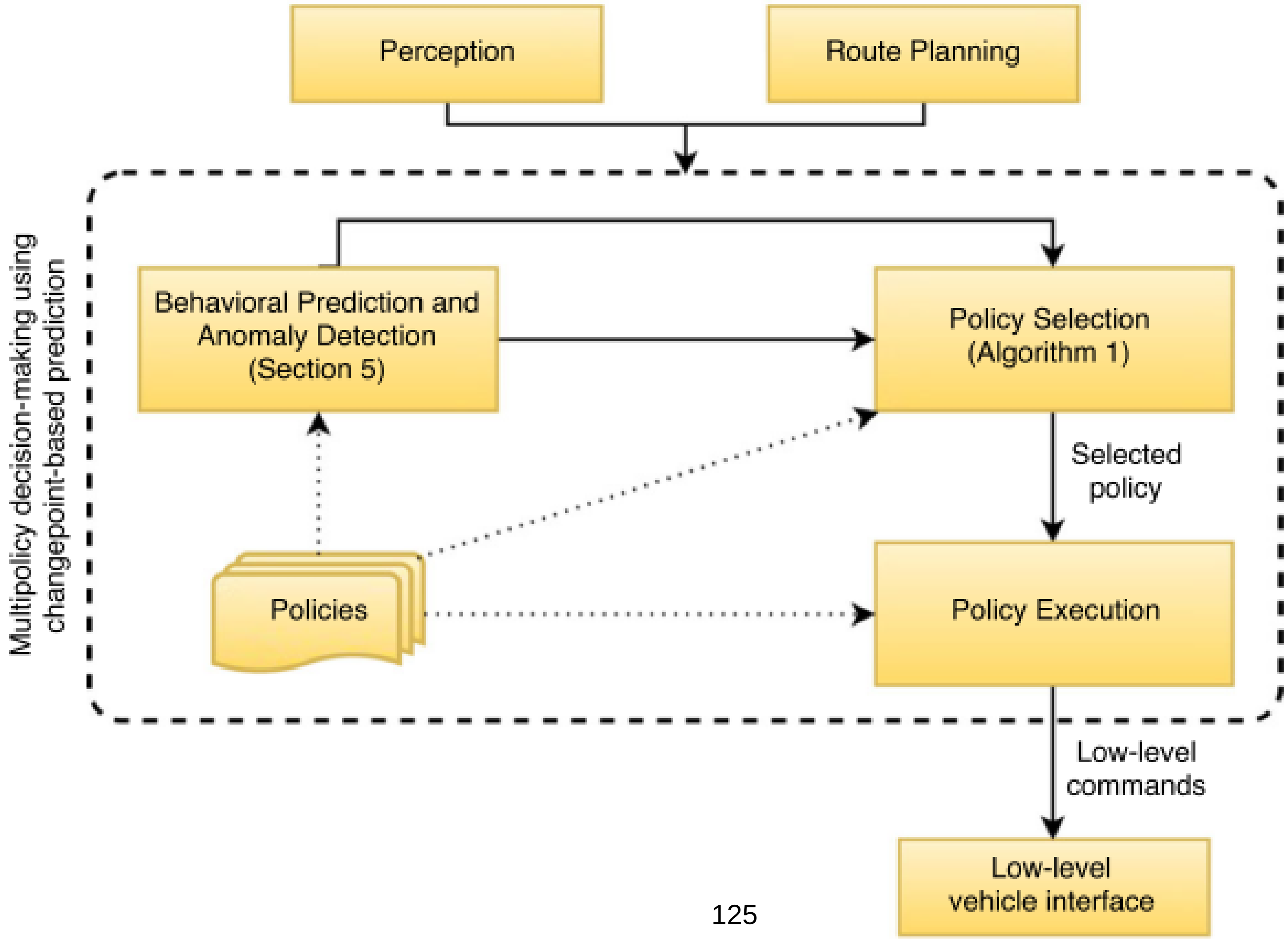


THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Multi-policy decision making

- Draw set of sample policies over distribution for neighbor vehicles
  - For each sample set
    - Simulate other vehicles forward
    - Choose a policy for ego vehicle that maximizes reward in this instance
    - Track best reward
- Choose policy for ego vehicle with the best reward

Approach



Perception
Route Planning
Multipolicy decision-making using changepoint-based prediction
Behavioral Prediction and Anomaly Detection (Section 5)
Policy Selection (Algorithm 1)
Selected policy
Policies
Policy Execution
Low-level commands
Low-level vehicle interface

Conclusion

- Proof of concept tests in real world and simulation
  - Good real world results in “offline” behavior prediction
- Limitations
  - No guarantees in decision making
  - A very coarse approximation of POMDP
  - Decision making is slow
    - 2-4 neighbors with small set of policy samples
    - Anomalous detection does not influence decision making

Questions?

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL