From Theory to Physical Systems: Putting It Together

Automotive Cyber-Physical Systems

How do we go from talking about C-spaces, DOF, and control theory to designing a real-life autonomous car?

Team MIT's Approach to the DARPA Urban Challenge

http://people.csail.mit.edu/albert/pubs/2007-dgc-tech-report.pdf

System Architecture

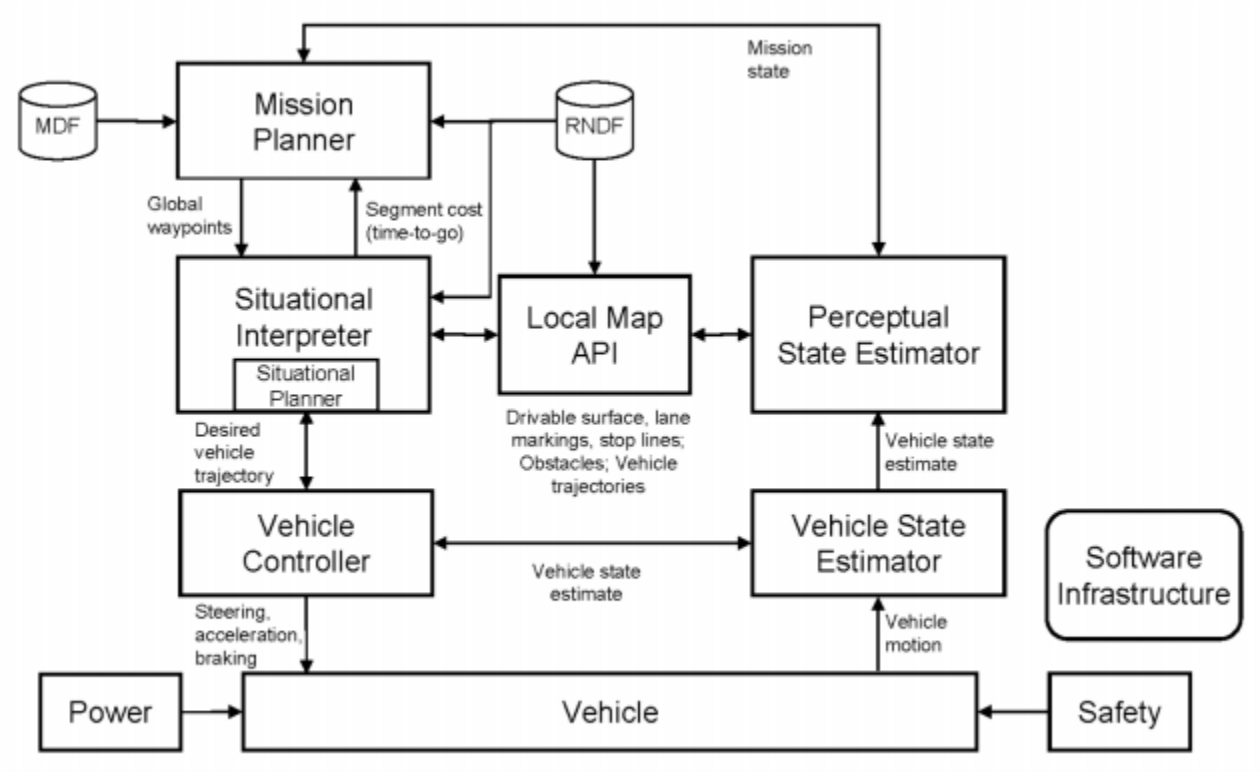


Fig. 1. System Architecture for Team MIT

Perception

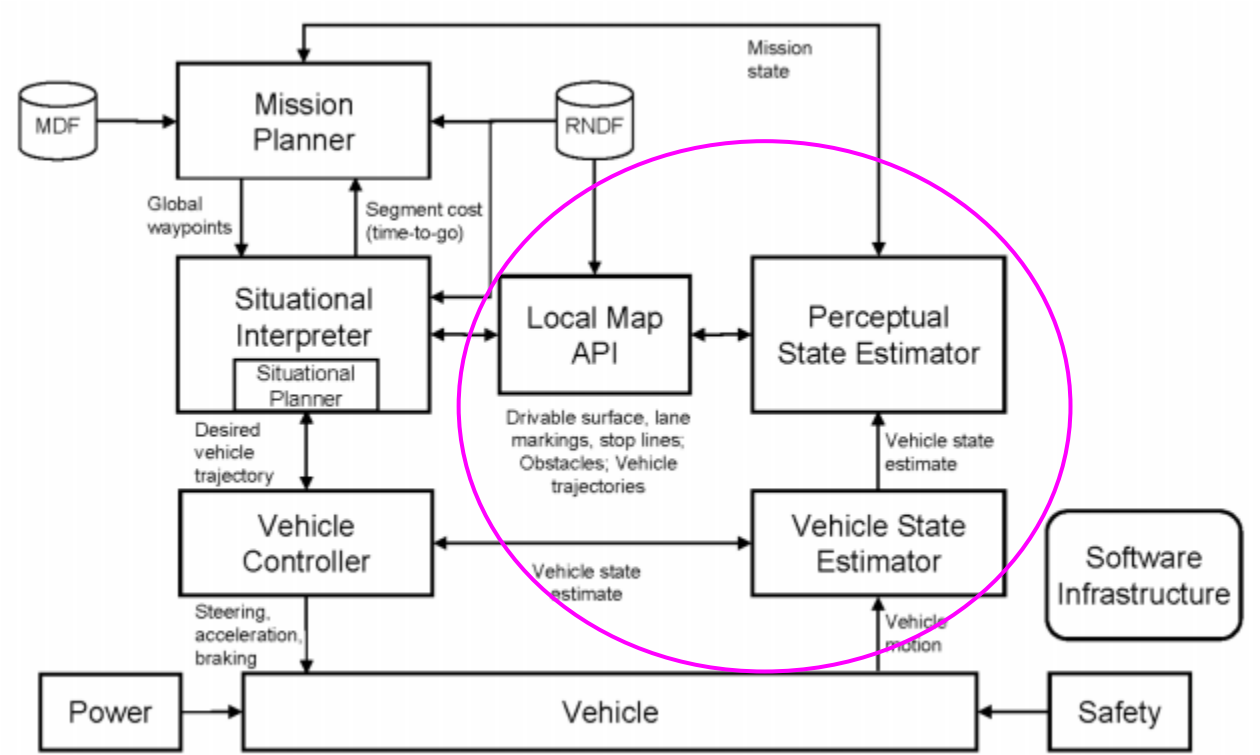


Fig. 1. System Architecture for Team MIT

Perception module

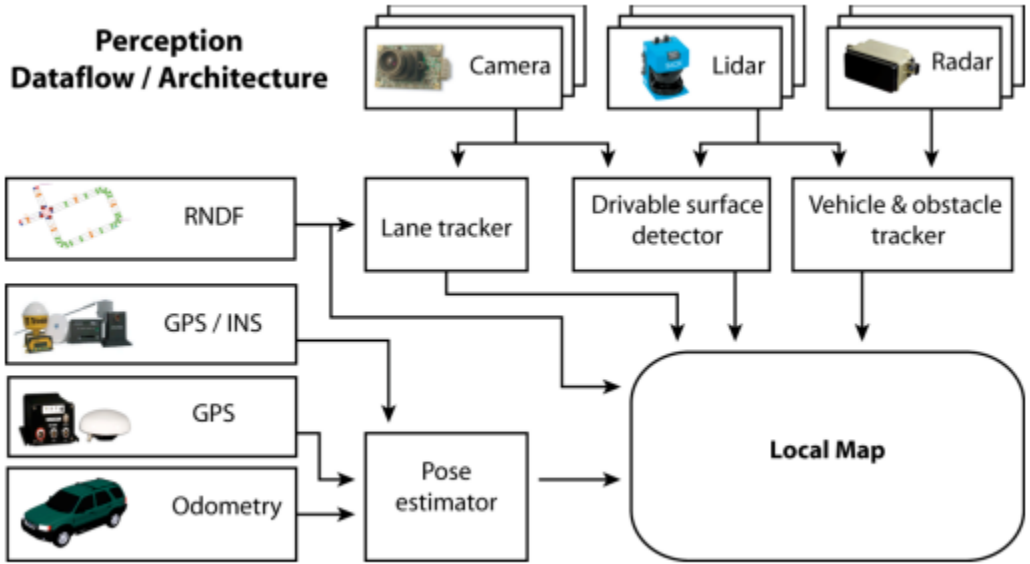
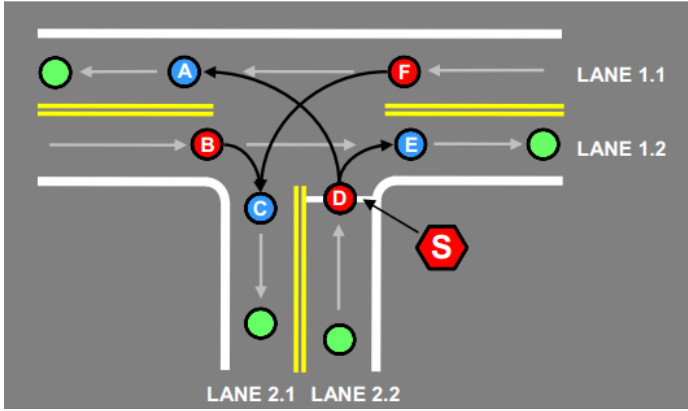


Fig. 5. Dataflow within the perception subsystem. Data from the sensors and RNDF are accumulated into the local map.

Sidenote: RNDF



Route Network Definition File

- specifies accessible road segments
- provides info such as waypoints, stop sign locations, lane widths, checkpoint locations, and parking spot locations
- no implied start or end point

RNDF_name *filename (string)*
num_segments *number_of_segments (integer>0)*
num_zones *number_of_zones (integer≥ 0)*
<optional file header>
<segment 1>
.
.
<segment M>
<zone M+1>
.
.
<zone M+N>
end_file

Sidenote: MDF

Mission Data File

- describes a set of checkpoints the vehicle must visit
- corresponds with a specific RNDF

MDF_name *filename (string)*
RNDF *RNDF_name (string)*
<optional file header>
checkpoints
num_checkpoints *number_of_checkpoints (integer>0)*
<checkpoint 1>
.
.
<checkpoint L>
end_checkpoints
speed_limits
num_speed_limits *number_of_speedlimits (integer>0)*
<speed_limit 1>
.
.
<speed_limit M>

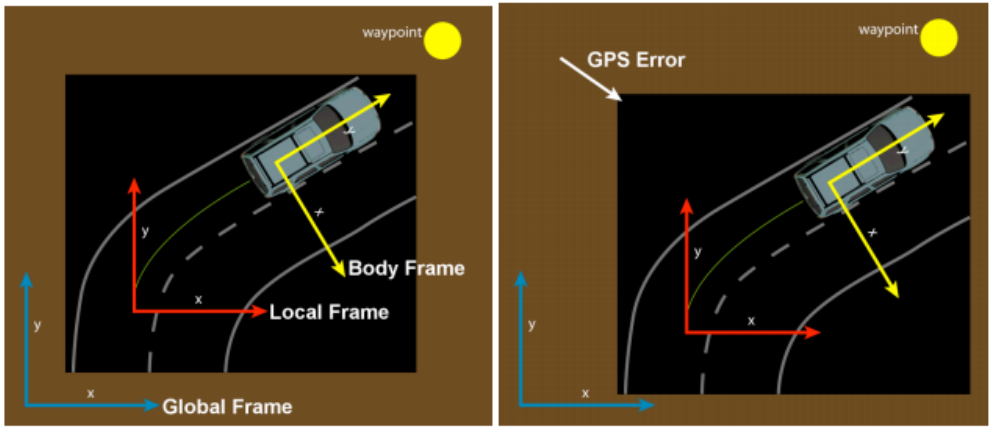
Vehicle State Estimation

How do we determine its 6-DOF pose in the coordinate frame?

- hybrid particle filter and Extended Kalman filter using odometry, inertial, and GPS inputs

Separate Local and Global Coordinate Systems

GPS is unreliable and introduces error in trajectory planning → use a local frame for sensor fusion and trajectory planning



Planning

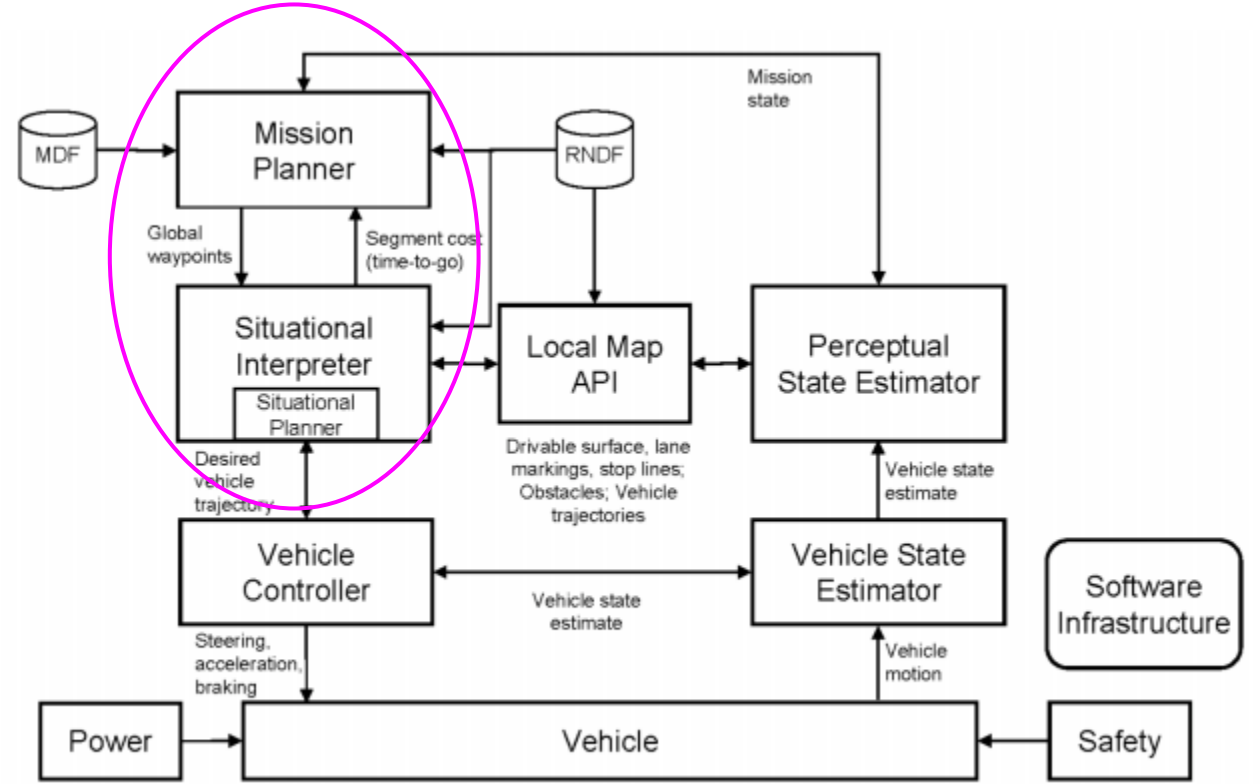


Fig. 1. System Architecture for Team MIT

What sequence of segments should be followed in the RNDF?

→ Mission Planner

What is the best path through this intersection?

→ Situational Planner

Planning Modules

1. Mission Planner
2. Situational Interpreter
3. Situational Planner

Resilient Planning

- Communication between components in the architecture is two-way, allowing “lower” blocks to request that the plan be recomputed if an infeasible problem/situation is encountered.
- Mission Planner uses exponentially forgetting edge costs, so previously rejected plans can be re-explored later

Mission Planner

- works in the global frame
- generates plan that minimizes the expected mission completion time
- Runs A* on RNDF and MDF
  - route segments weighted based on *a priori* info like intersections, turns, speed limits
  - weights updated based on sensor readings

Situational Interpreter

Inputs:

- high-level mission plan (Mission Planner)
- current local map state (Local Map API)
- vehicle state (Vehicle State Estimator)

→ decides next course of action

Situational Interpreter

- Handles mode switching, e.g. PARKING, CRUISE, WAIT, PASSING, STOP as a finite state machine
- Converts global waypoints from MP into local frame coordinates, which are then used as inputs to Situational Planner

Situational Planner

Generates trajectories guaranteed to be:

- collision-free
- dynamically feasible
- consistent with traffic rules

based on inputs from the Situational Interpreter.

Output: list of waypoints and speed limit for controller

Modified RRT

- incrementally constructs a tree of feasible trajectories, starting from the current configuration of the car
- biased random sampling strategy, allowing the tree to efficiently explore region reachable by car
- replanning rate of 10 Hz allows it to quickly react to sensed changes in the environment

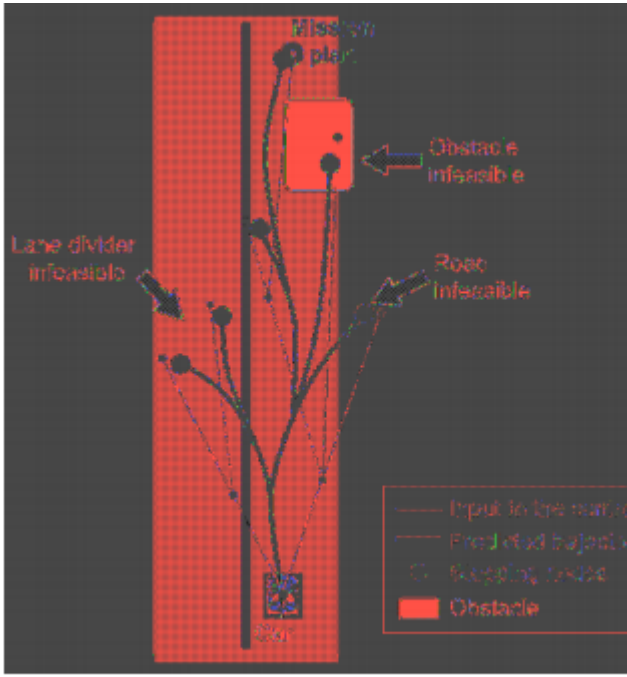


Fig. 10. RRT-based motion planning approach.

To grow tree of feasible trajectories:

1. Sample a reference trajectory (e.g. a point and direction)
2. Simulate closed-loop behavior of the car when this reference is fed into low-level controller
3. Check resulting trajectory for feasibility WRT obstacles and traffic rules
4. If trajectory is feasible, add it to the tree
5. Iterate

Advantages of Modified RRT

1. Provable probabilistic completeness in a static environment
2. Can easily handle complex vehicle dynamics
3. Provides and maintains a large number of possible routes, allowing for efficient re-planning as new obstacles are detected
4. RRTs work just as well in open spaces as in cluttered environments

Control

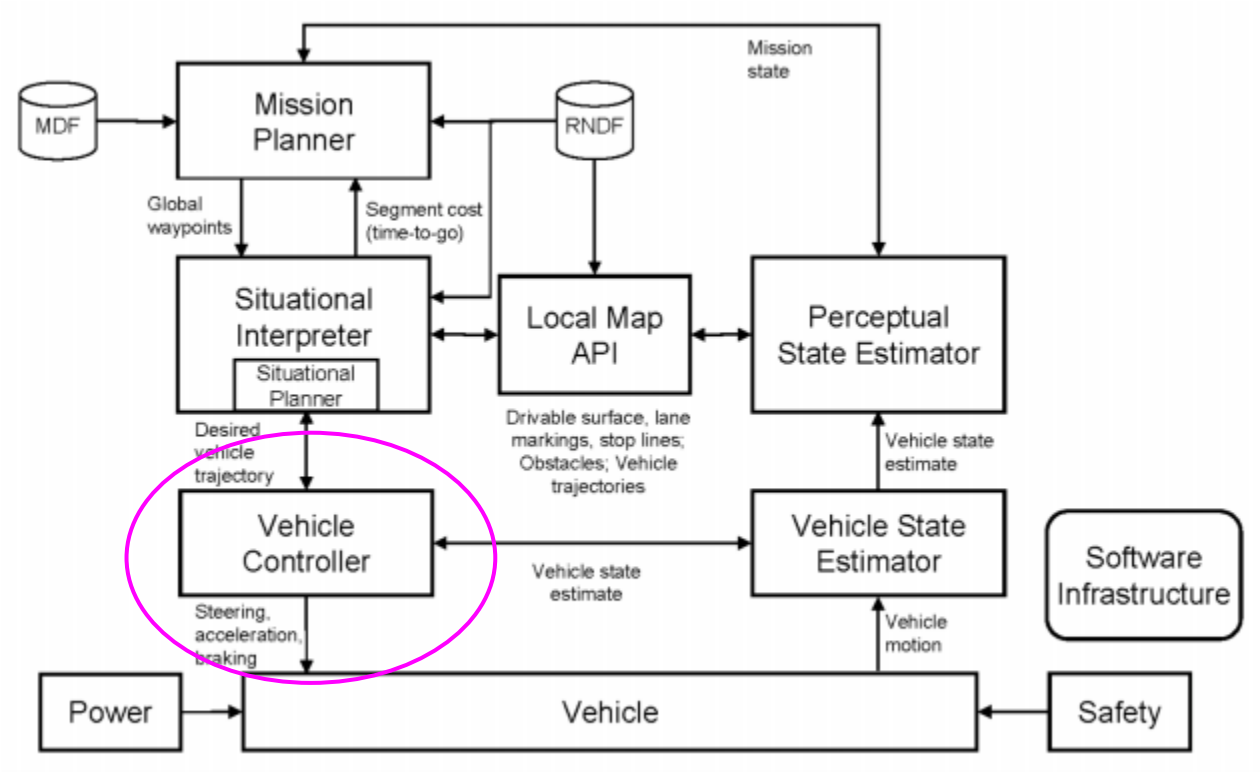


Fig. 1. System Architecture for Team MIT

Low-Level Vehicle Controller

Generates gas, brake, and throttle commands required to track the trajectory specified by the situational planner

Speed

- linear PID controller
- Gains tuned to produce minimal overshoot for step inputs
- For braking, gains scaled to account for dynamic differences between braking and accelerating

Gas and Braking

- EMC actuation system that accepts input signal of 0-5V and places a single servo on the gas and brake pedal
- better transition between braking and accelerating

Forward Steering Control

- variant of pure-pursuit algorithm
- well suited for tracking non-smooth piece-wise linear paths
- specifies a constant curvature path for the vehicle to travel from the current location to the goal → calculate desired steering angle

Robotics Components

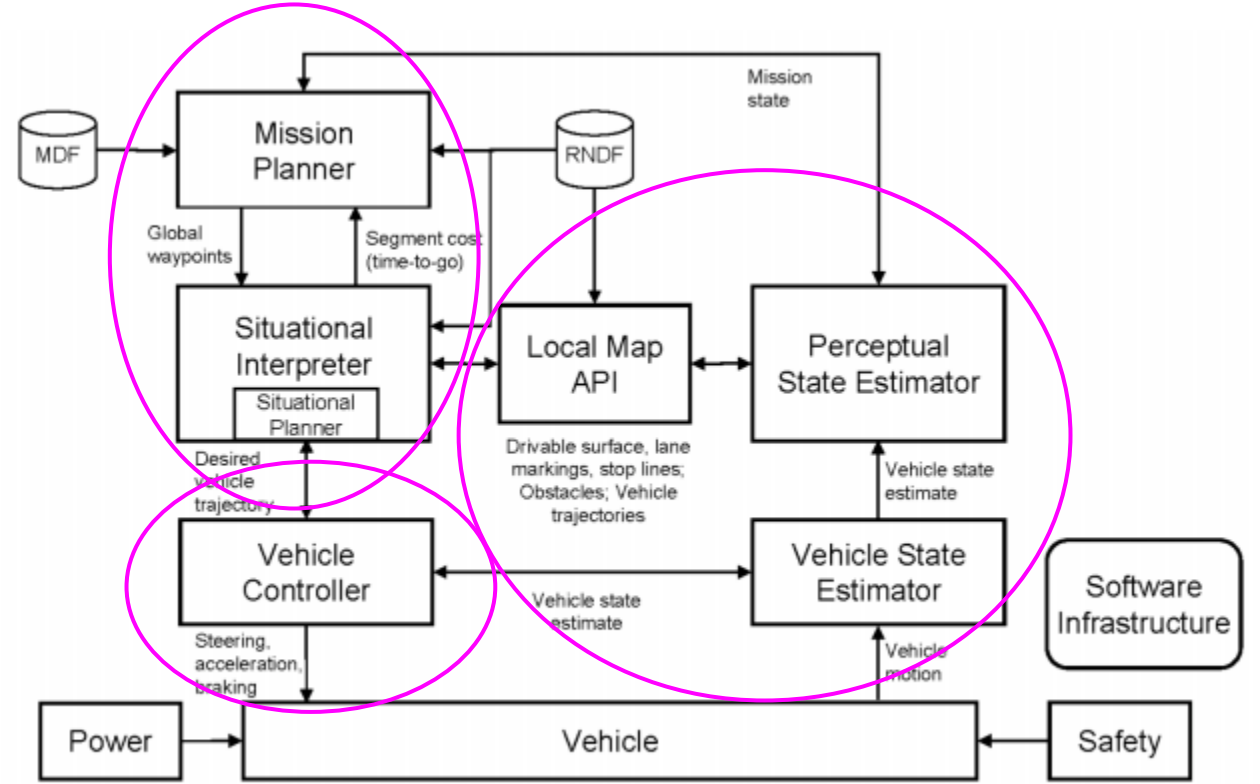


Fig. 1. System Architecture for Team MIT