Steueralgorithmen bei autonomen Fahrzeugen

von Christin Laßow

Structure

- Level of autonomy
- History
- Algorithm
- Difficulties
- further research

Level of autonomy

1. basic driving assistance

2. advanced assistance

3. system monitors environment and can drive with full autonomy under certain conditions

4.fully autonomous driving in certain conditions

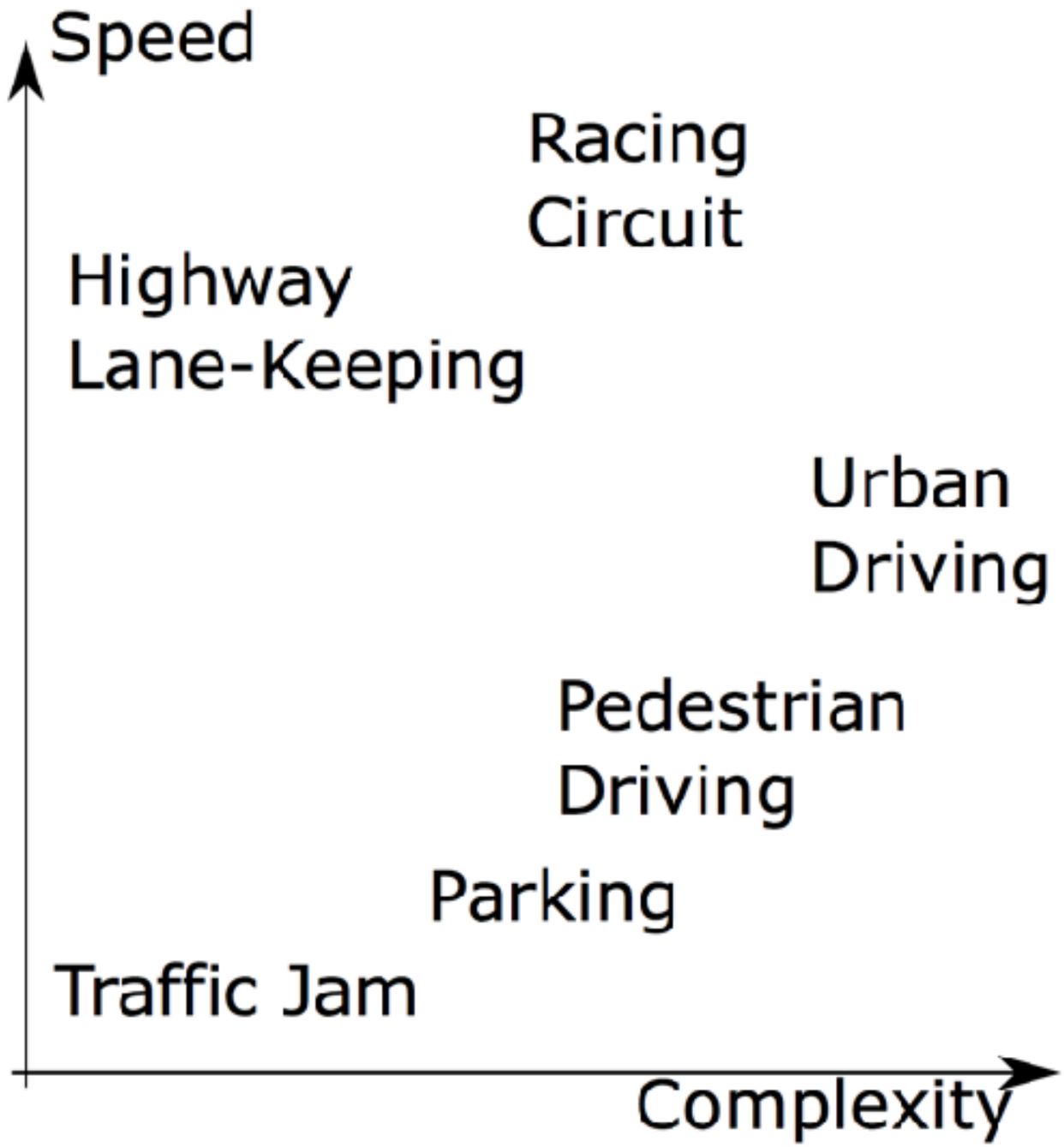
5.fully autonomous in all driving modes

History

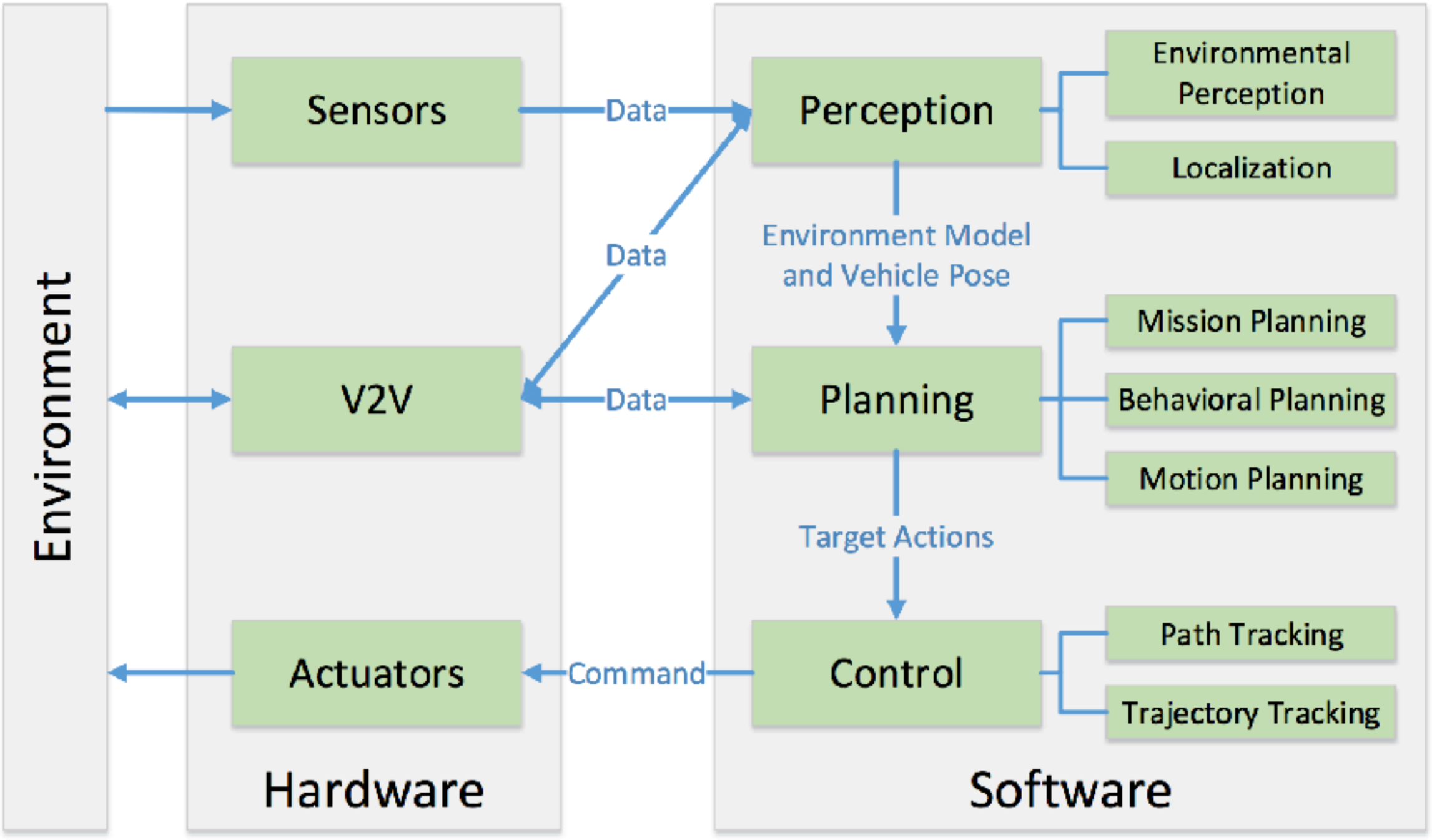
- 1918: first vision of self-driving cars
- 1958: first mention of them on TV
- 1988: vehicle performs lane-following
- 2004: DARPA Grand Challenge
- 2005: DARPA Grand Challenge
- 2007: DARPA Urban Challenge

Difficulties

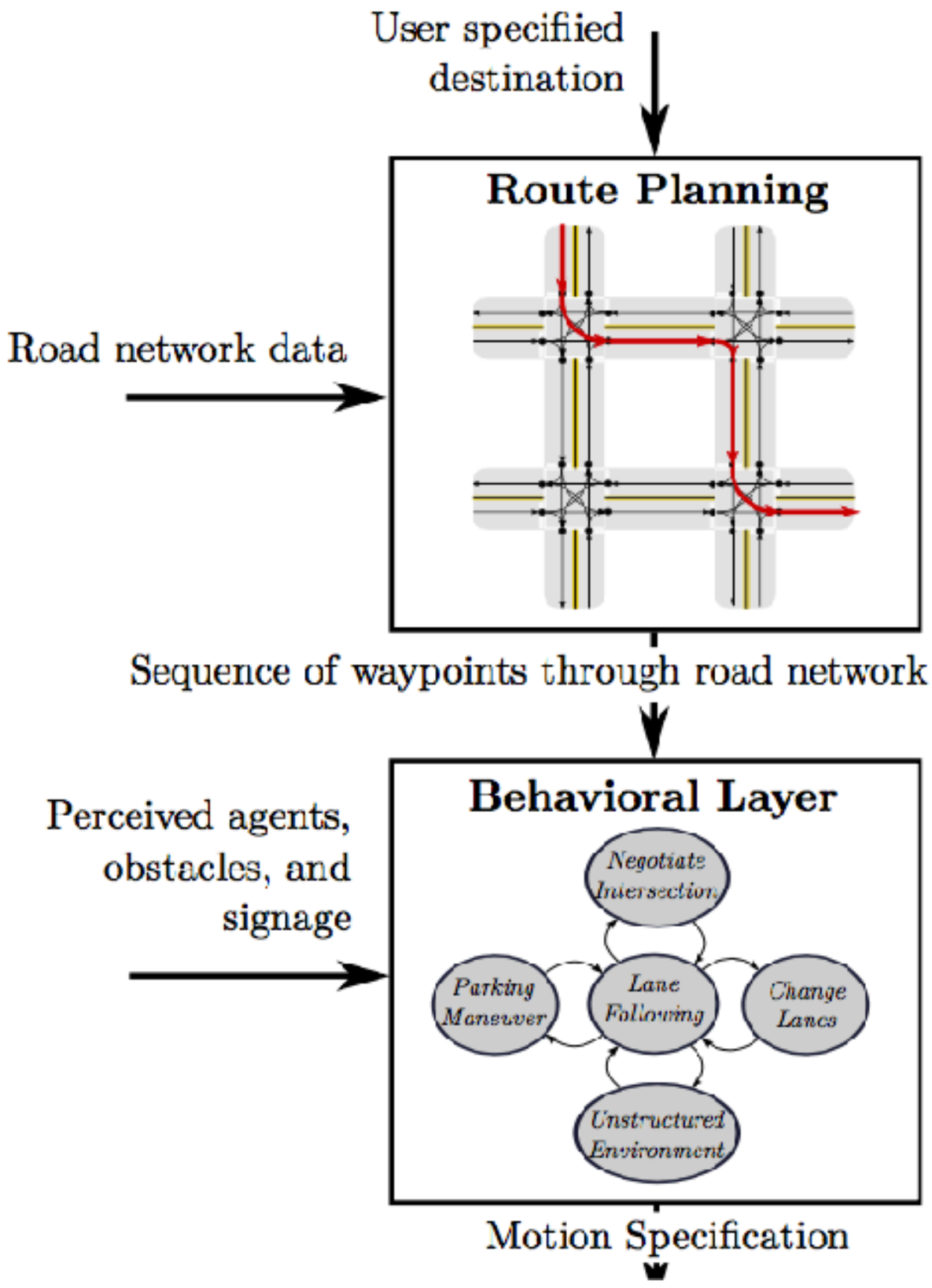
- density of vehicles
- various area-specific traffic rules
- Unpredictable behaviour of human drivers
- limited perception range

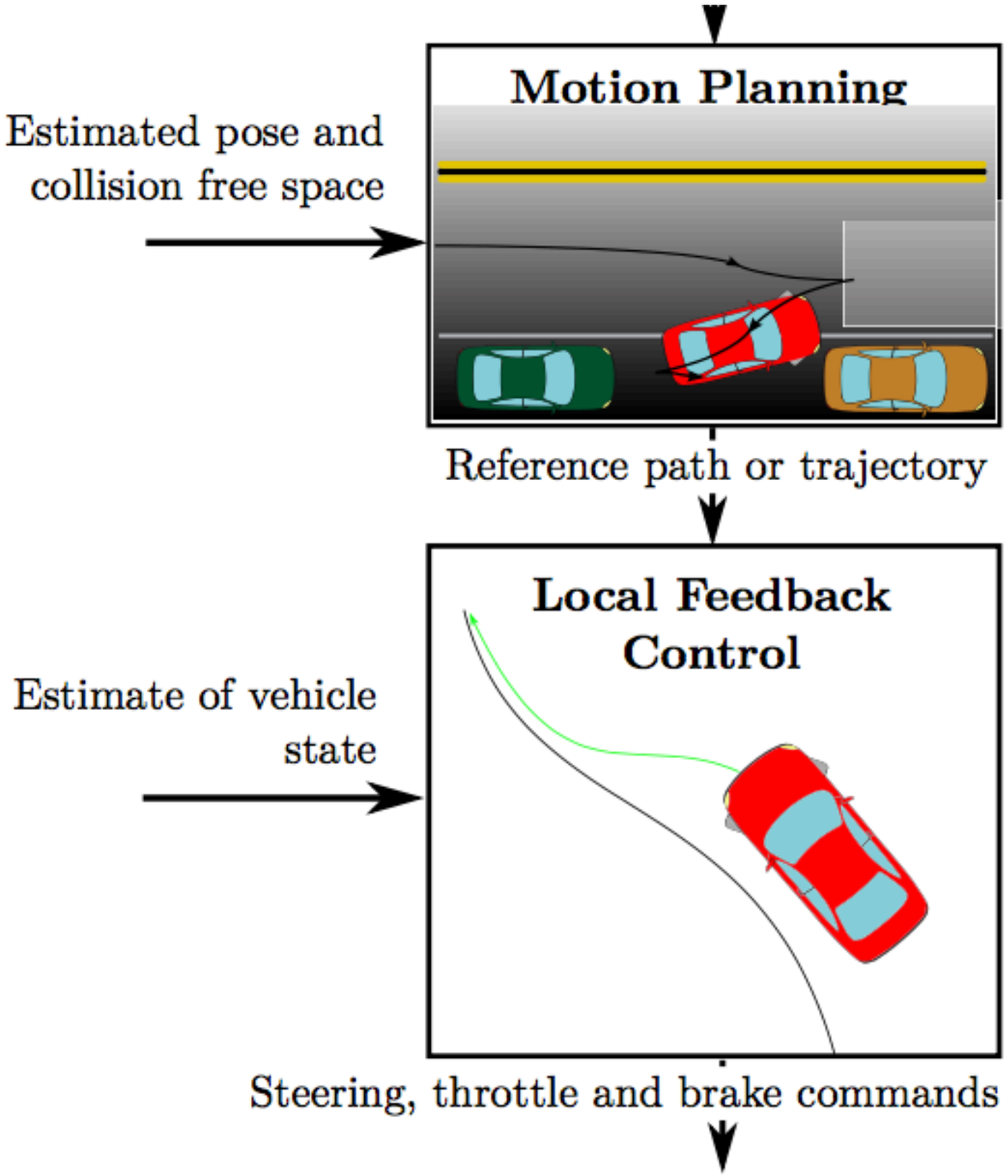


speed vs complexity



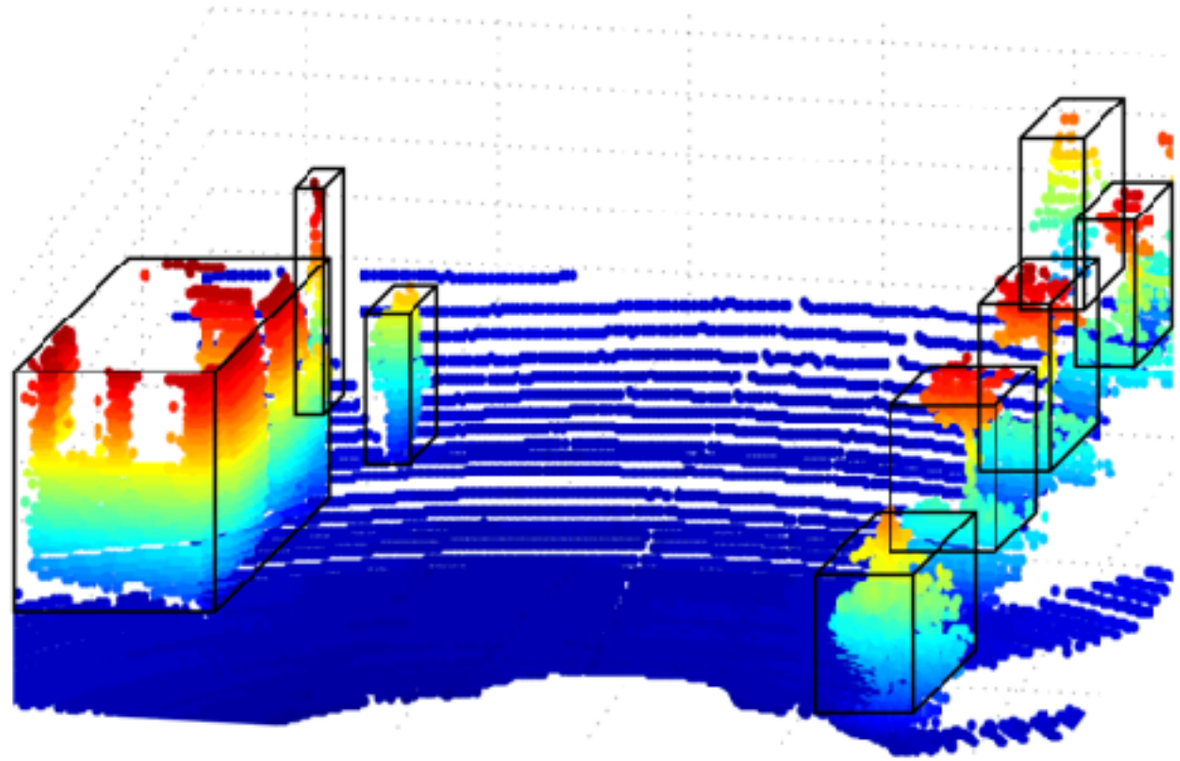
typical autonomous vehicle system


User specified
destination
Route Planning
Road network data
Sequence of waypoints through road network
Behavioral Layer
Perceived agents,
obstacles, and
signage
Negotiate Intersection
Parking Maneuver
Lane Following
Change Lanes
Unstructured Environment
Motion Specification


Motion Planning
Estimated pose and
collision free space
Reference path or trajectory
Local Feedback
Control
Estimate of vehicle
state
Steering, throttle and brake commands

Perception - LIDAR

- Light Detection and Ranging
- used to gather information about environment
- algorithms used can be divided in five categories:
  - edge based
  - region based
  - attributes based
  - model based
  - graph based



Planning

- mission planning
- behavioural planning
- motion planning

mission planning

- generally performed through graph search
- Dijkstra oder A* effective for small neighbourhoods

behavioural planning

- responsible for decision making in following road rules and safe interaction with others

- usually realised through a finite state machine

motion planning

- high-level control problem
- control limitations may be ignored to various degrees
- tasks:
    - finding the way around obstacles
    - finding the most efficient way

first Algorithm - PRM

- originally consists of these steps:

1. random configuration is selected and tested

2. finds closest neighbours

3. connecting the node with the closest neighbour

second Algorithm - RRT

- originally consists of these steps:

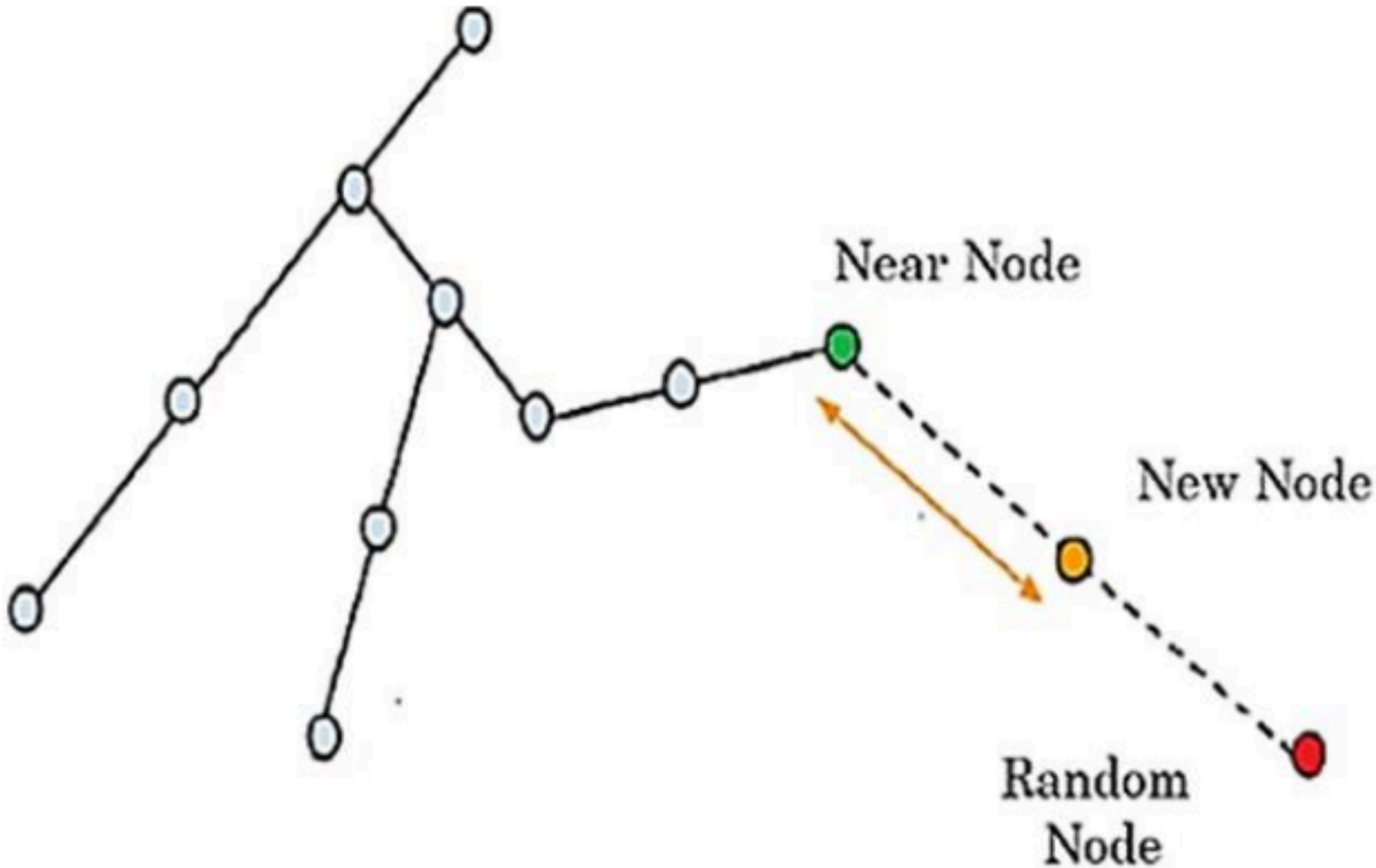
1. random example is picked

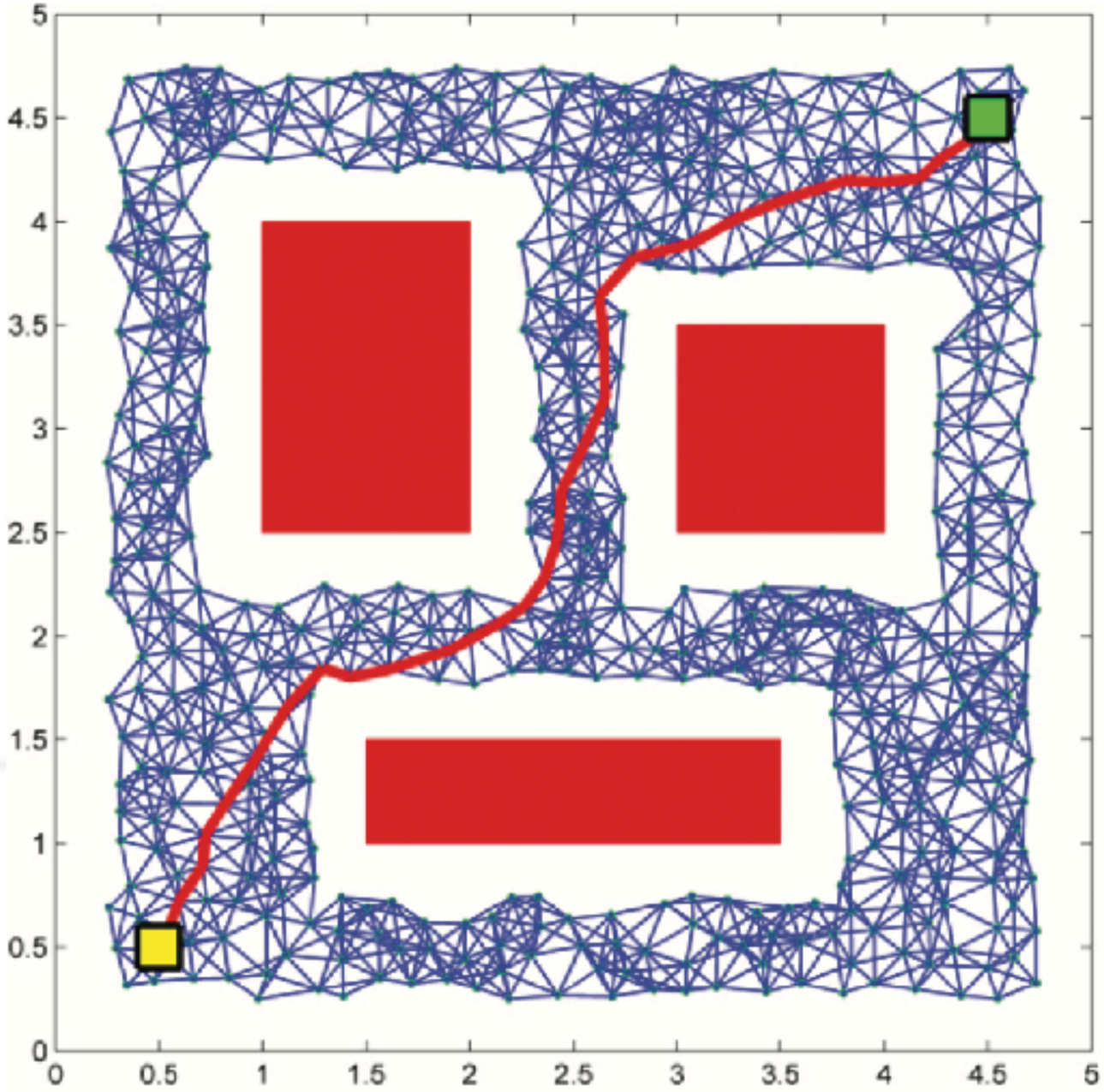
2. finds closest node

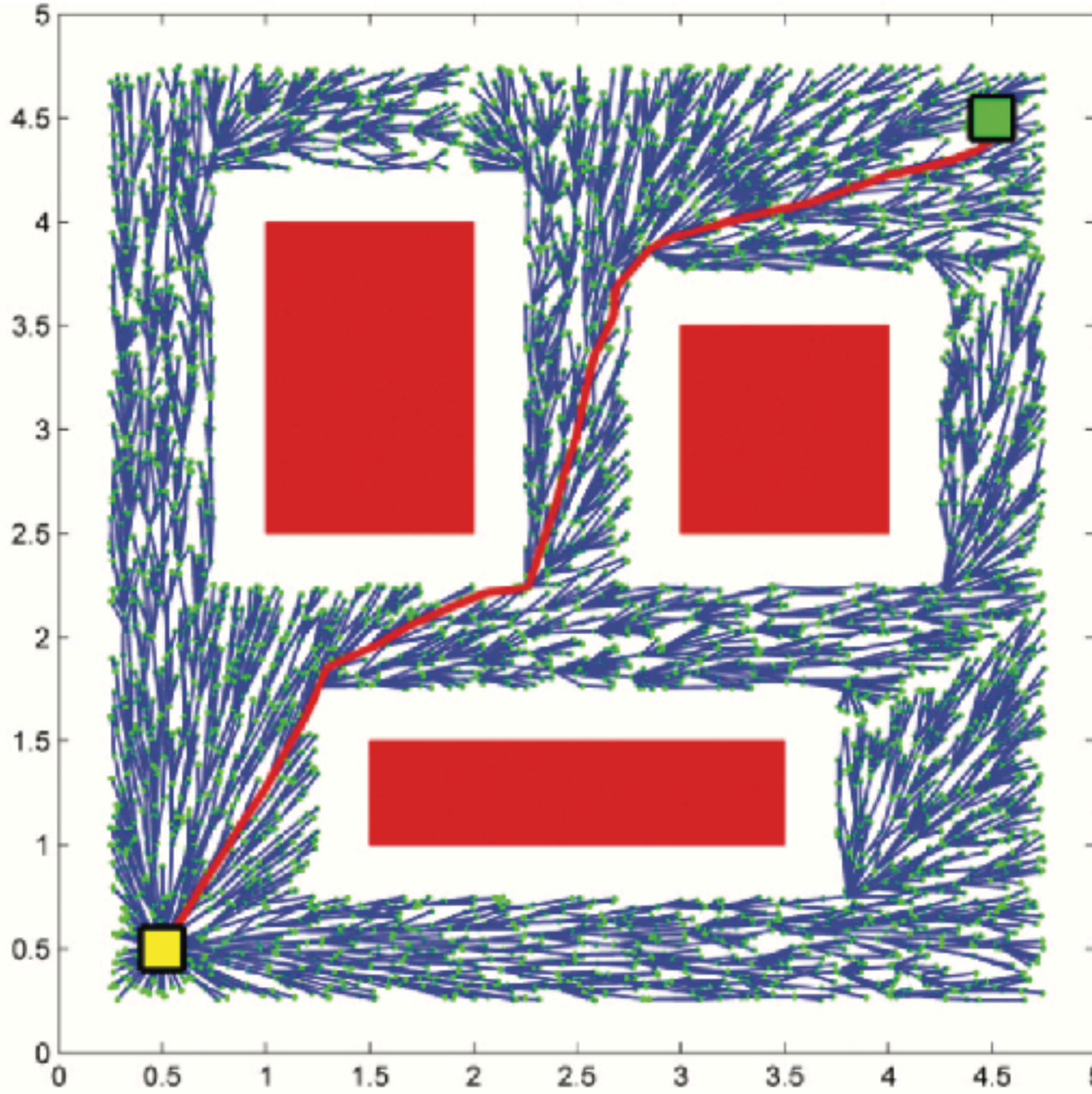
3. create new node

4. test if the new node is in free space and has collision free local path

second Algorithm - RRT







PRM vs RRT

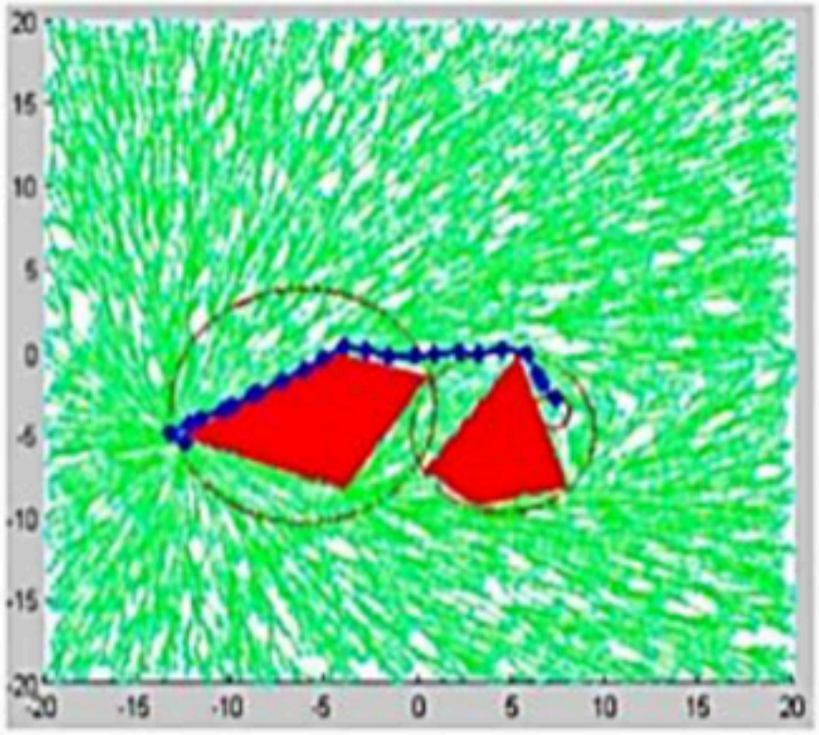
Algorithm 1. $T = (V, E) \leftarrow$ **RRT***(z_{init})

1 T ← InitializeTree();
2 T ← InsertNode(Ø, z_{init}, T);
3 **for** i=0 to i=N **do**
4 z_{rand} ← Sample(i);
5 $z_{nearest}$ ← Nearest(T, z_{rand});
6 (z_{new}, U_{new}) ← Steer ($z_{nearest}$, z_{rand});
7 **if** Obstaclefree(z_{new}) **then**
8 z_{near} ← Near(T, z_{new}, $|V|$);
9 z_{min} ← Chooseparent (z_{near}, $z_{nearest}$, z_{new});
10 T ← InsertNode(z_{min}, z_{new}, T);
11 T ← Rewire (T, z_{near}, z_{min}, z_{new});
12 **return** T

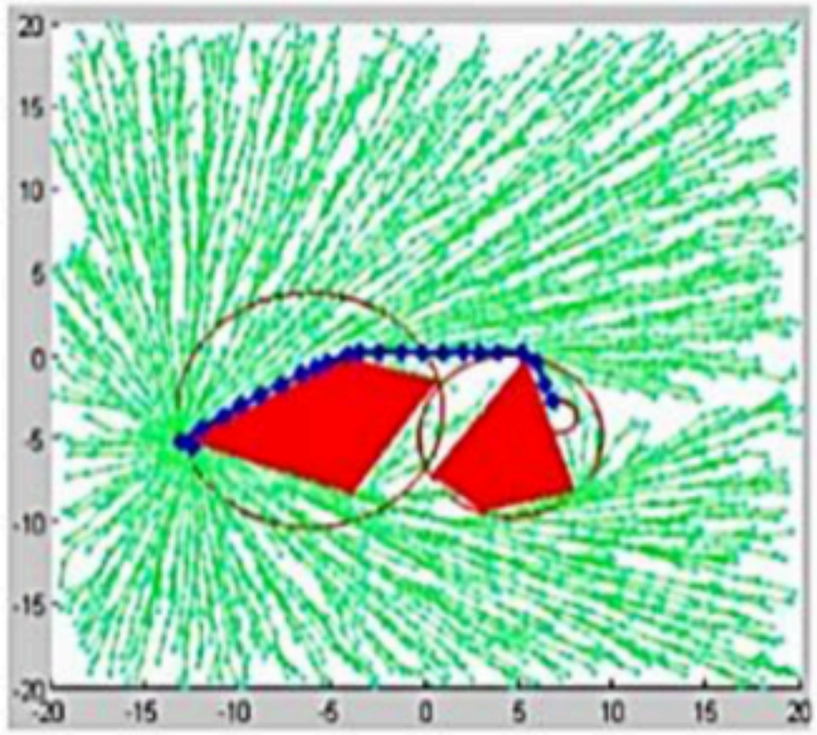
RRT*

RRT* improvements

- edge is only added to tree if it has minimal cost
- fixed number of nodes in a tree



(a) RRT*

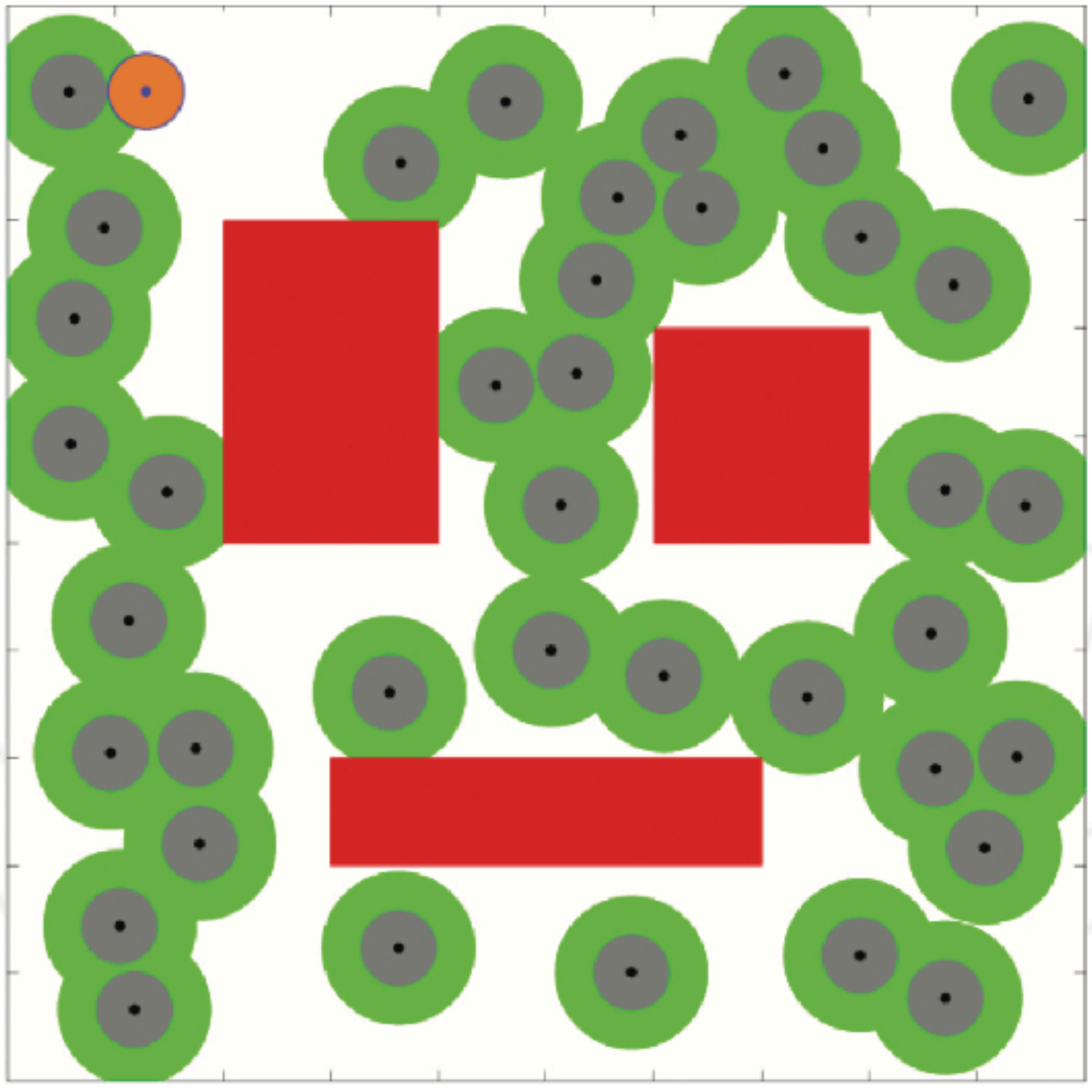


(b) RRT*FN

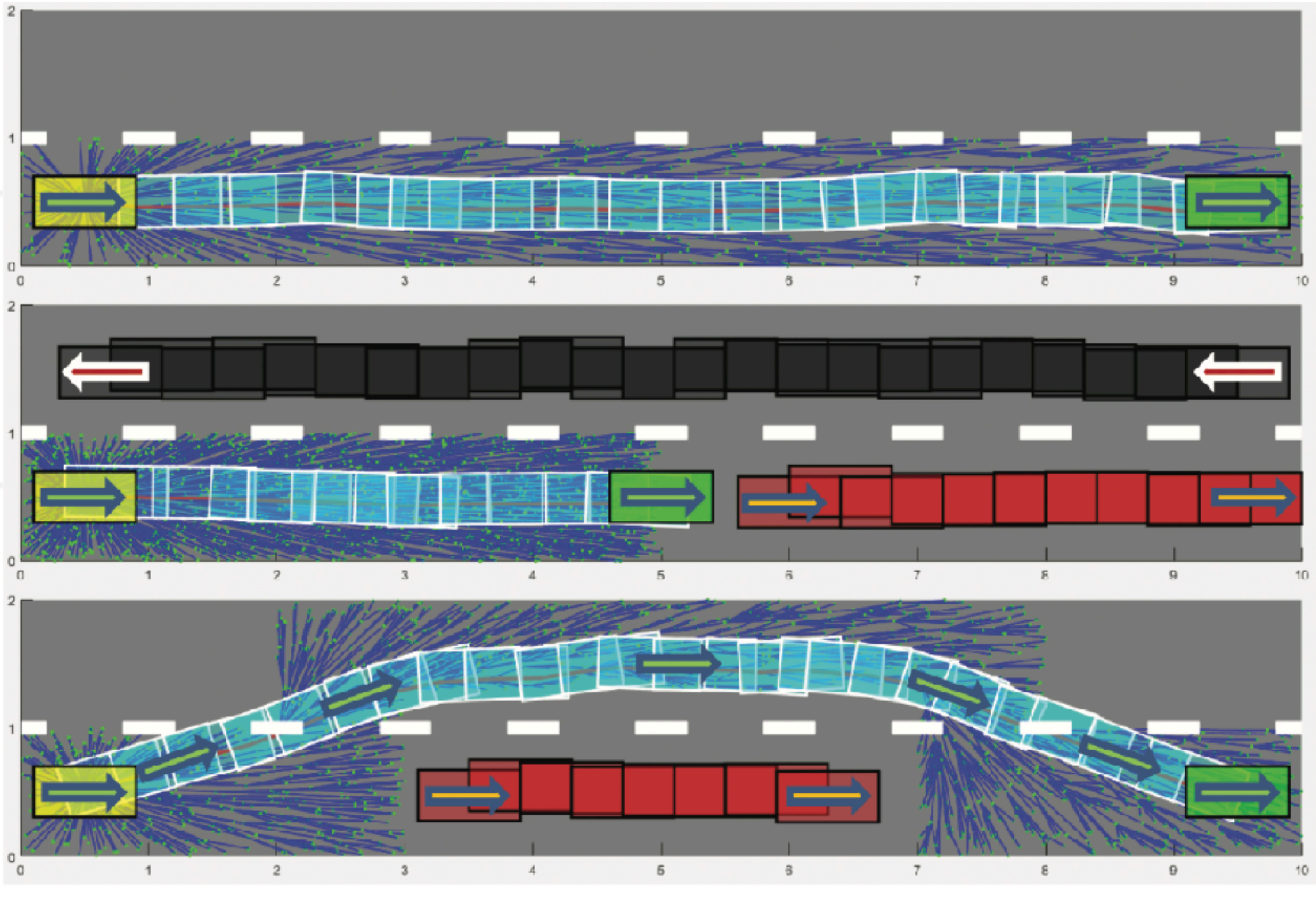
RRT* and improvements

- RRT# — global replanning procedure to keep track of promising nodes
- bidirectional RRT*

RRT* and improvements



Simulation



Simulation

Scenario	Variable	RRT	RRT*	Proposed algorithm
Free drive	Average number of samples	314.2	203.7	138.8
	Average optimality (%)	63.4	93.8	93.9
	Average runtime (s)	87.6	114.0	48.3
Takeover	Average number of samples	627.2	808.5	316.4
	Average optimality (%)	50.7	91.4	91.4
	Average runtime (s)	134.2	168.9	93.5
Follow	Average number of samples	223.1	418.5	108.8
	Average optimality (%)	68.6	93.5	93.4
	Average runtime (s)	73.5	108.6	46.4

incremental planning

- replanning is necessary to adjust for dynamic changes

- if replanning uses information from previous planning, better solutions might be found faster

- safety mechanisms need to be carefully set up

classical control

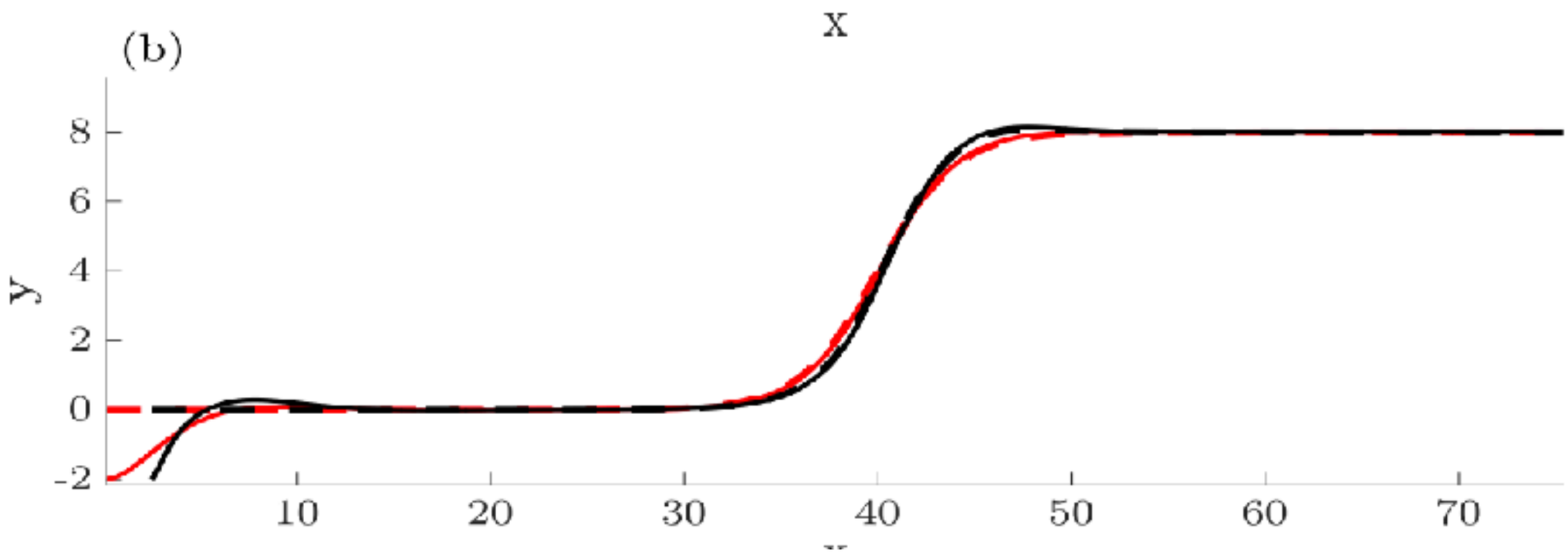
- feedback control is most common controller structure
- role of feedback control is to stabilise to the reference path
- most common form: PID controller
- feedback has several limitations

feedback control

- rear wheel based feedback

- front wheel based feedback

feedback control



rear wheel based feedback

- erhält das Feedback von den Hinterrädern und kann darauf reagieren
- vergleicht den eigentlichen Kurs mit dem vorgegebenen Optimalkurs, da die Stabilität nicht wesentlich von der Schnelligkeit abhängt, eignet sich dieses Feedback auch für das Rückwärtsfahren

front wheel based feedback

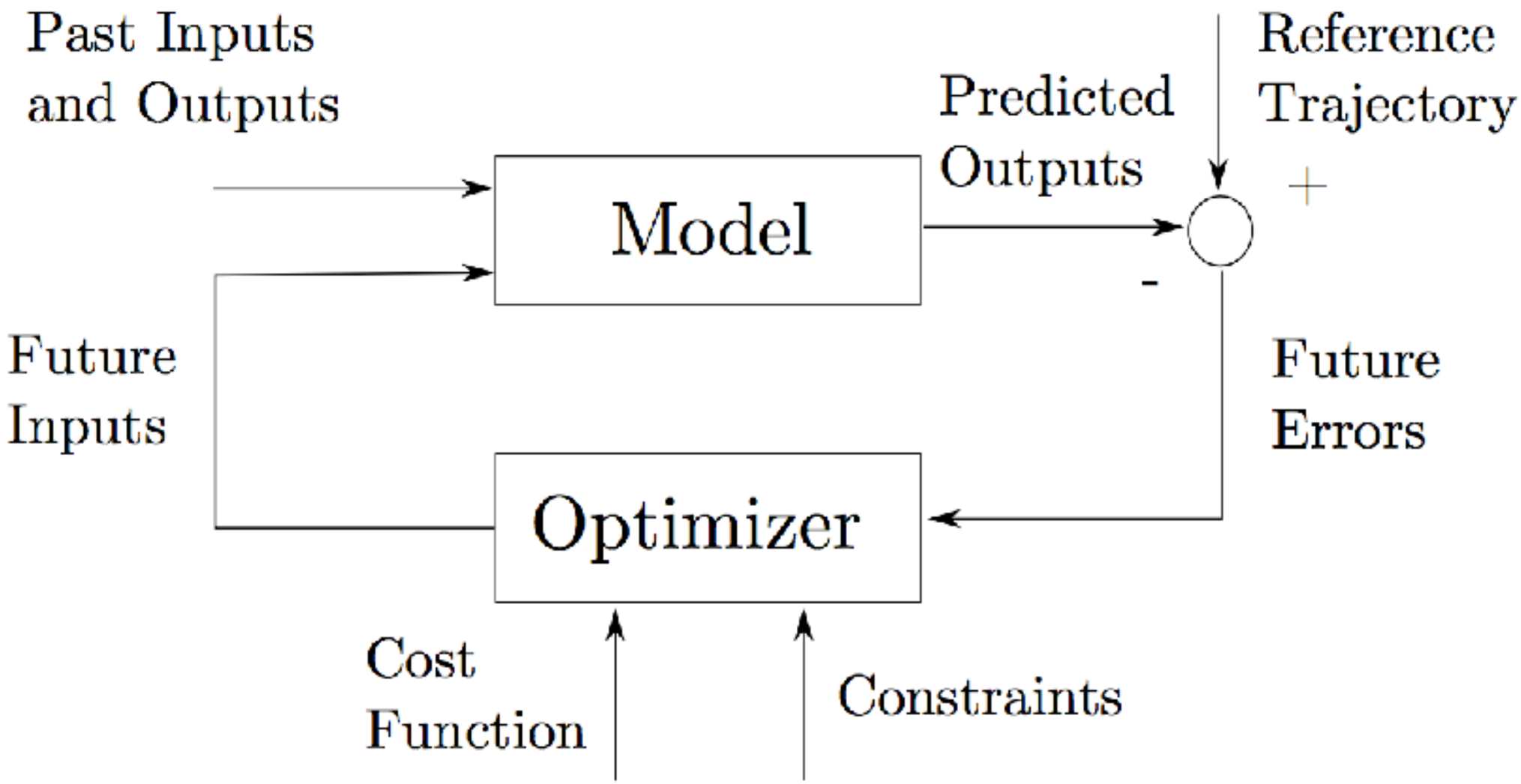
- eigentlich wie rear wheel nur mit den Vorderreifen, front wheel feedback eignet sich nicht für das Rückwärtsfahren
- es ist dabei nicht stabil und eignet sich somit nicht zum Parken

classical control

- state space control
    - examines system states to control entire vector
- linear state space model:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) y(t) = C(t)x(t) + D(t)u(t)$$

Model predictive control



Questions?

Sources

- Paden et al. A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles (2016)

- Khaksar et al. Application of Sampling-Based Motion Planning Algorithms in Autonomous Vehicle Navigation (2016)

- Noreen et al. Optimal Path Planning using RRT* based Approaches: A Survey and Future Directions (2016)

- Pendleton et al. Perception, Planning, Control, and Coordination for Autonomous Vehicles (2017)