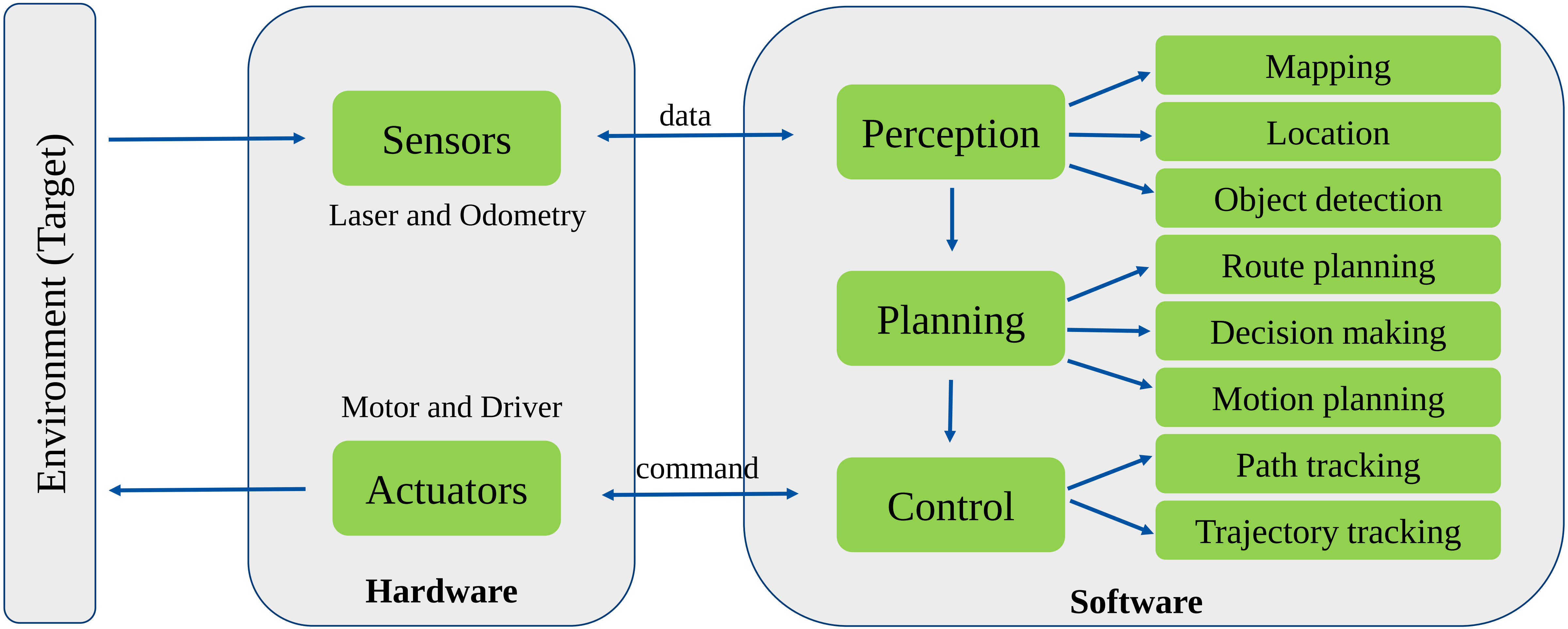
A Review of Motion Planning Techniques

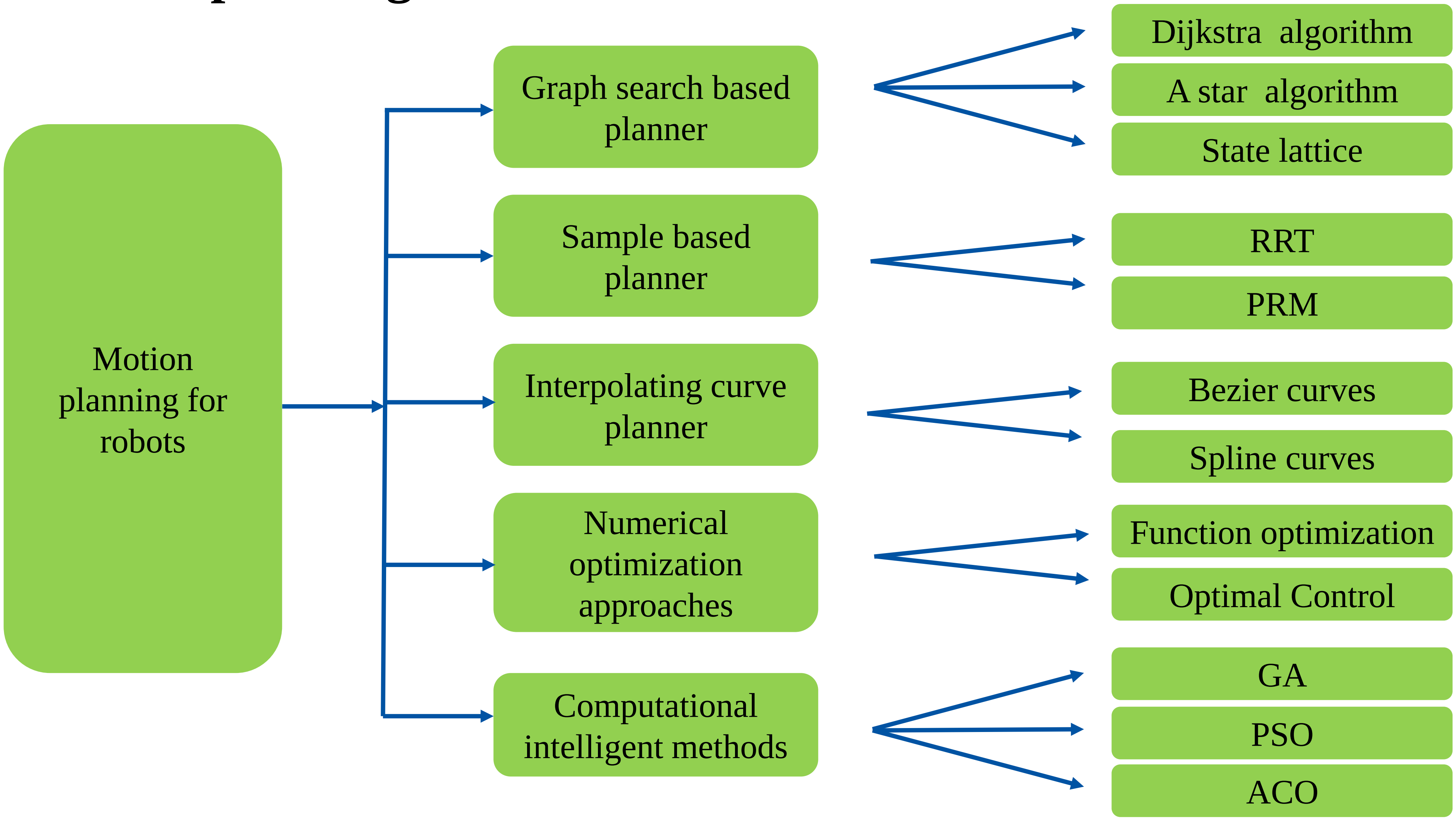
Benchun Zhou

School of Automation Science and Electronic Engineering
Beihang University

- **Framework**


Environment (Target)
Sensors
Laser and Odometry
Motor and Driver
Actuators
Hardware
data
command
Perception
Planning
Control
Mapping
Location
Object detection
Route planning
Decision making
Motion planning
Path tracking
Trajectory tracking
Software

- **Motion planning state of art**

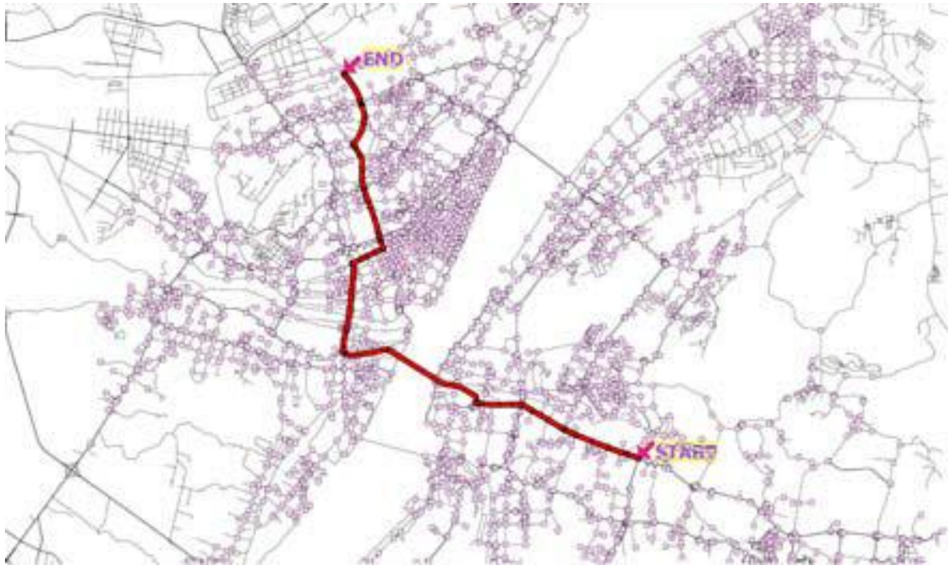


0. Framework

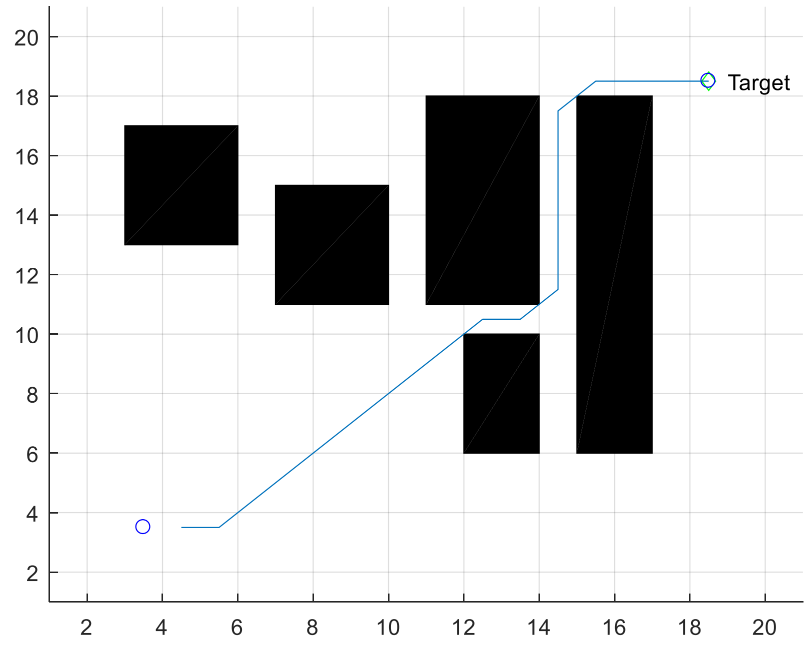
➢ Basic conception

- Path planning
- Motion planning
- Trajectory planning

- **Dijkstra algorithm**



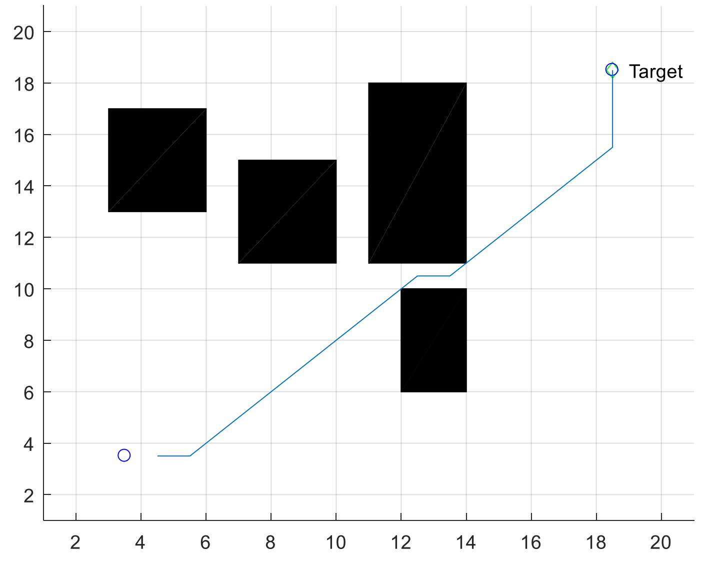
- **A* algorithm**

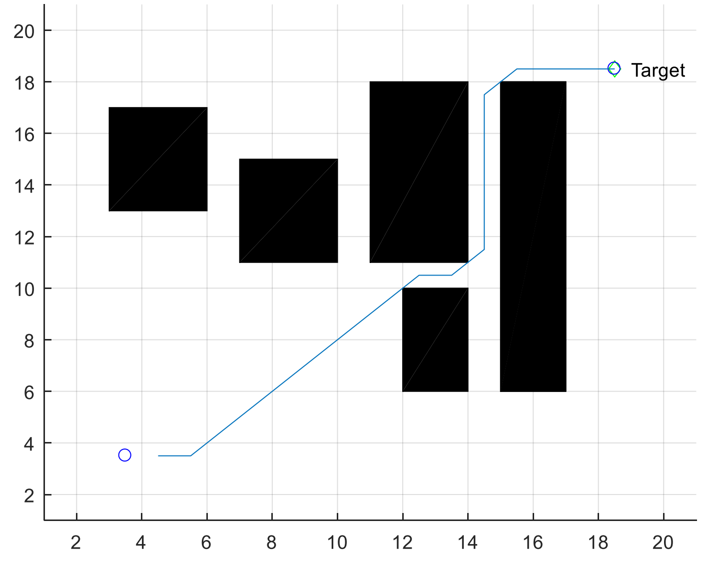


- In A Star we maintain 2 lists valid and in_valid
- Valid List contains
- o All valid nodes as we compute costs for obstacle free cells
- Invalid List contains
- o Obstacle containing cells
- o Cells that are included in path
- Implementation
- o Compute costs of all the adjacent cells to the current cell
- o Add them to valid list
- o Find cell with minimum total cost from valid list, make it parent for next iteration and make it unavailable for next minimum total cost comparison
- o Put that in in_valid list so that we don’t visit it again
- o Keep doing this until the cell with minimum cost is not target cell.

1. Graph Search Based Planner

- **D* algorithm**
- D star is almost similar to *A star, as the name says its robust to dynamic appearing of obstacles.
- If after computing the path an obstacle occurs while traversing the path following will be done:
- o Update in_valid list with new obstacles.
- o Remove all the obstacle cells from valid list.
- o Increase the total cost for all adjacent nodes where first obstacle was found.
- o Iteratively increase the total cost of all the cells which are children to the adjacent nodes.
- o Also make them available (by updating valid list) for reconsideration while comparing total costs.





- **State lattices**
  - *Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios*

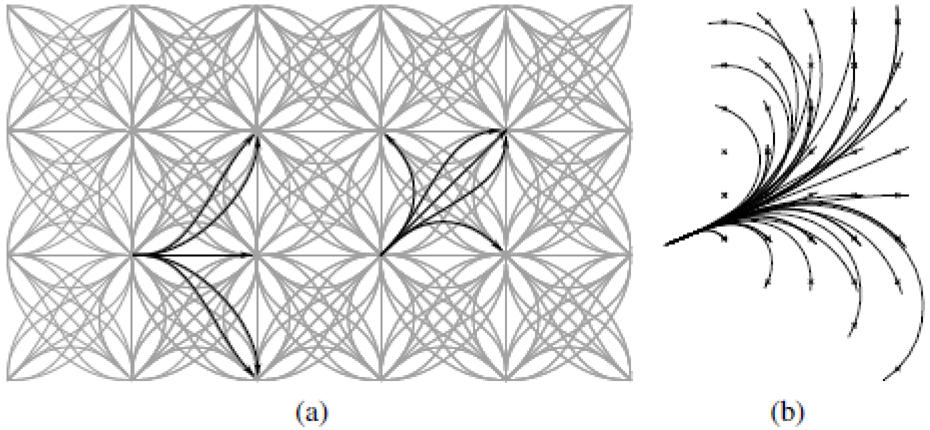
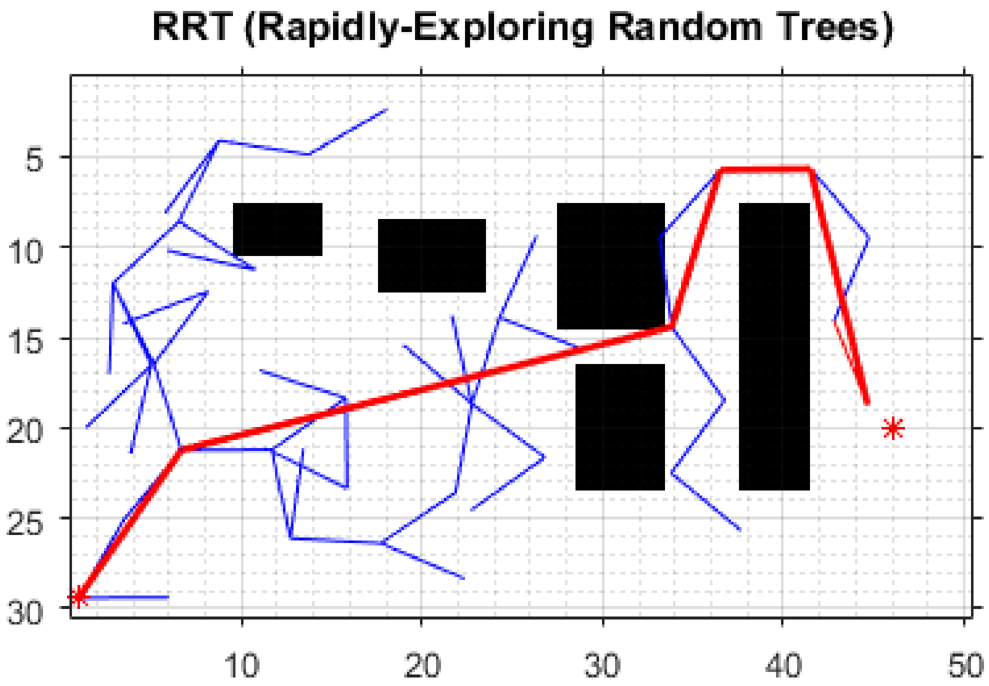


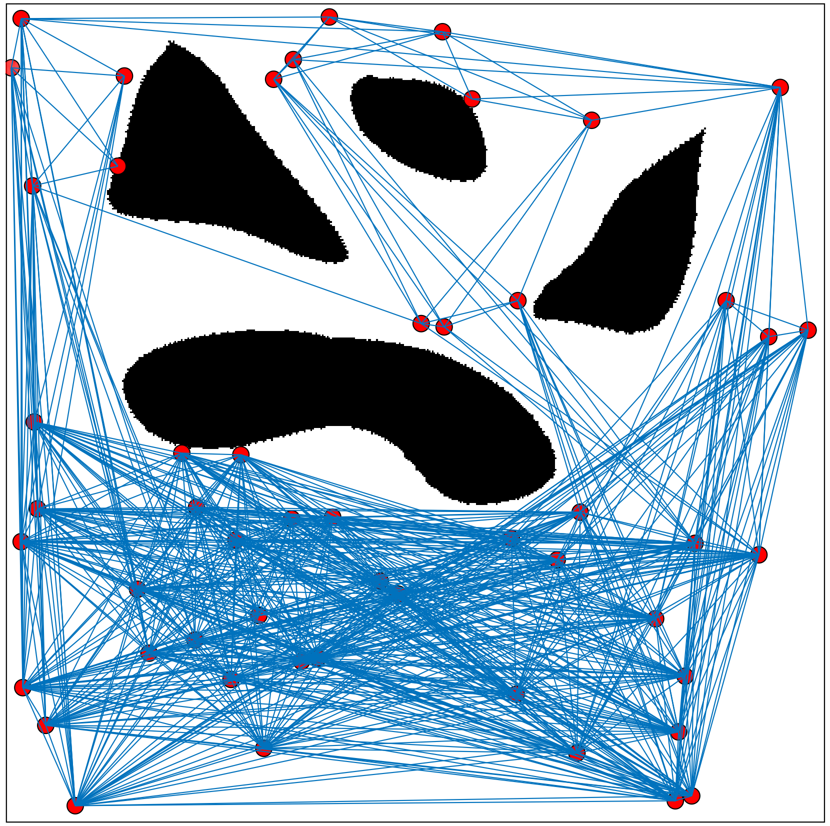
Fig. 1: (a): A nontemporal state lattice on a 2D workspace. Part of the canonical control set is displayed in black. (b): The right hand side shows part of a possible control set at finer discretisation, where the configuration space includes curvature.

• Rapidly-exploring random tree (RRT)

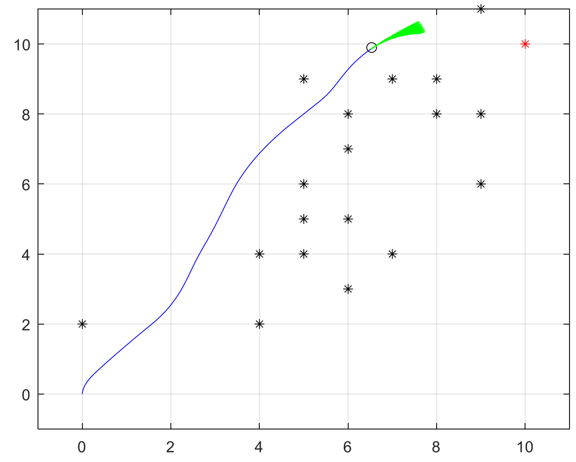
- *Sampling-based Algorithms for Optimal Motion Planning*
- In RRT we maintain two lists Edges and Vertices.
- Vertices contains
- o All valid nodes as we compute new vertex.
- o Vertices must lie in obstacle free space.
- Edges contains
- o Two vertices which constitutes the edge.
- o The edge must lie in obstacle free space.
- Implementation
- o Find a random point within the environment
- o Find a nearest vertex to that random point.
- o Find a new point at the delta distance away from nearest point in the direction of random point.
- o Check if the new point and edge attached to it lies in free space by breaking the edge in intermediate points and checking for each point alone.
- o Keep doing this until the cell with the target is not reached or the edge constitutes the target.

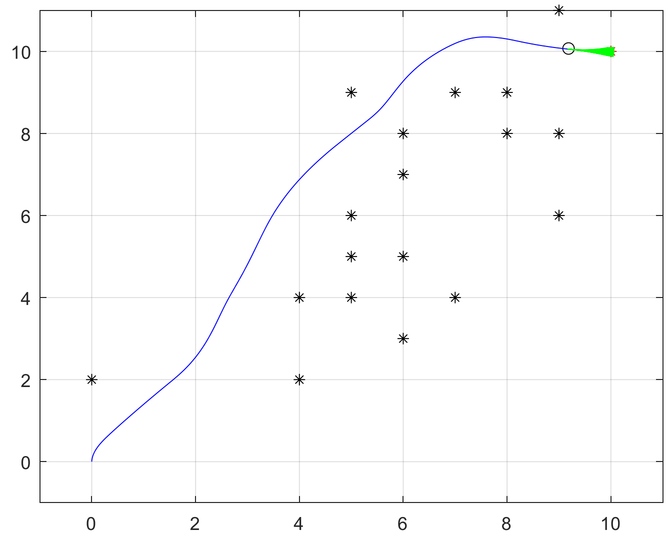


- **Probabilistic roadmap method (PRM)**
- *Path Planning in Complex 3D Environments Using a Probabilistic Roadmap Method*



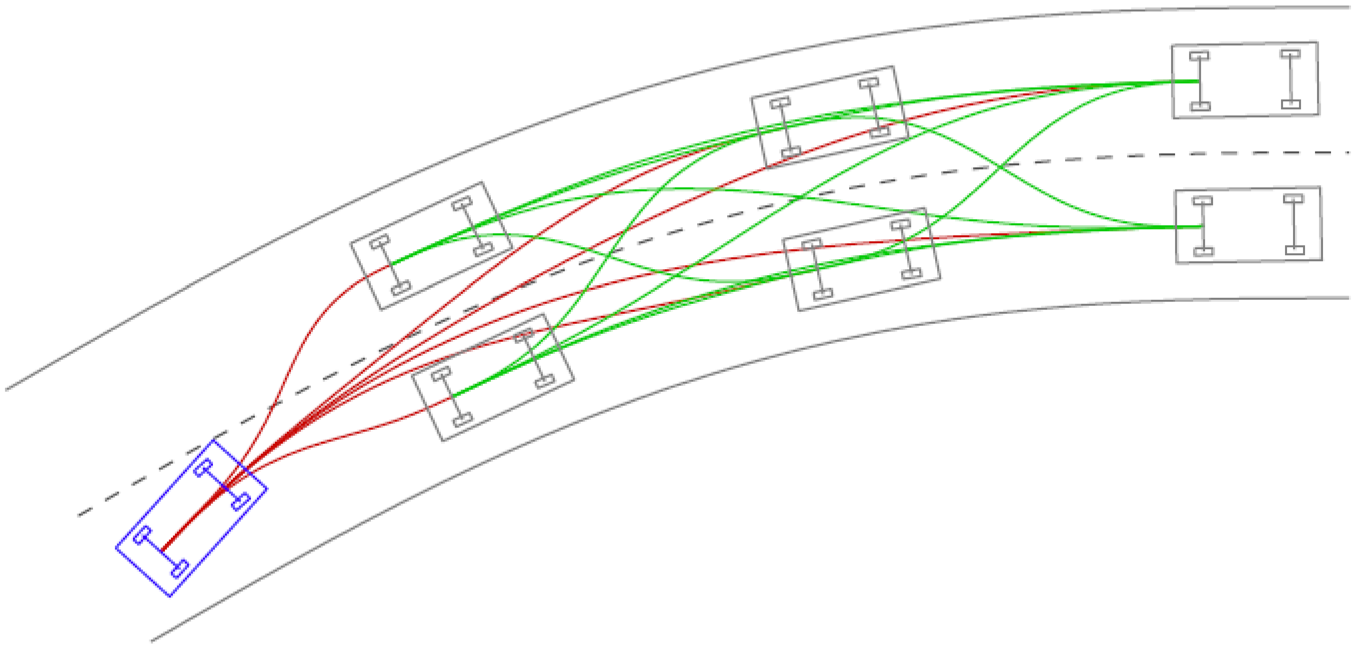
- **Dynamic window approach (DWA)**



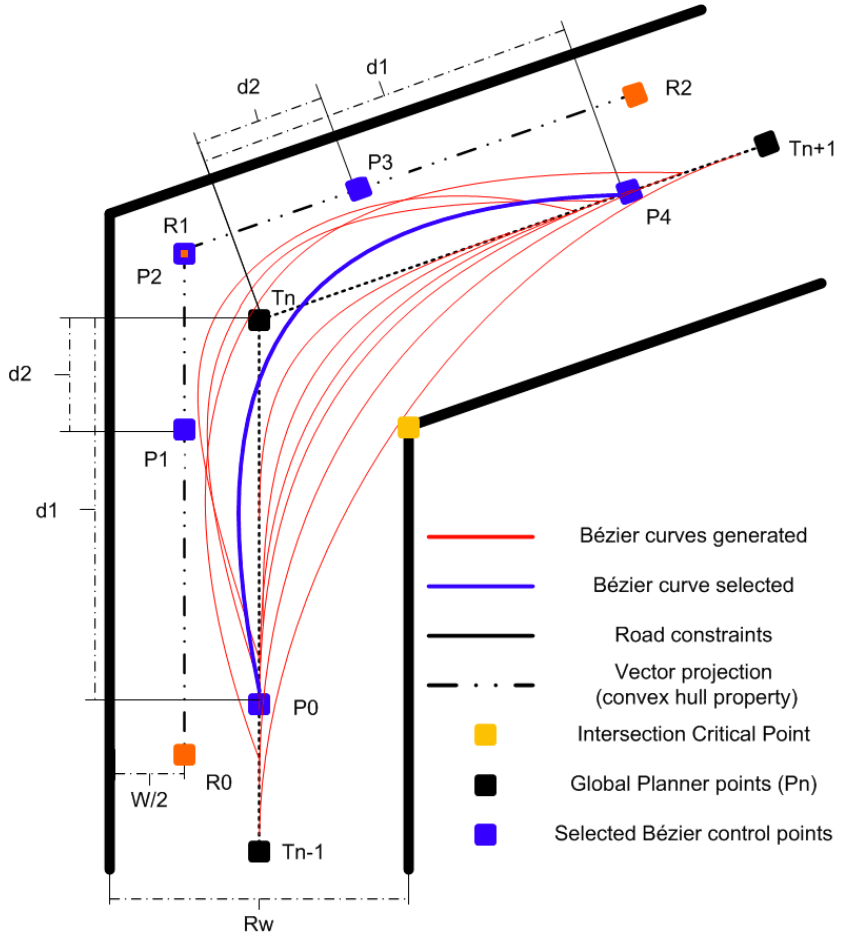


3. Interpolating Curve Planner

- **Polynomial curves**
- *A Real-Time Motion Planner with Trajectory Optimization for Autonomous Vehicles*

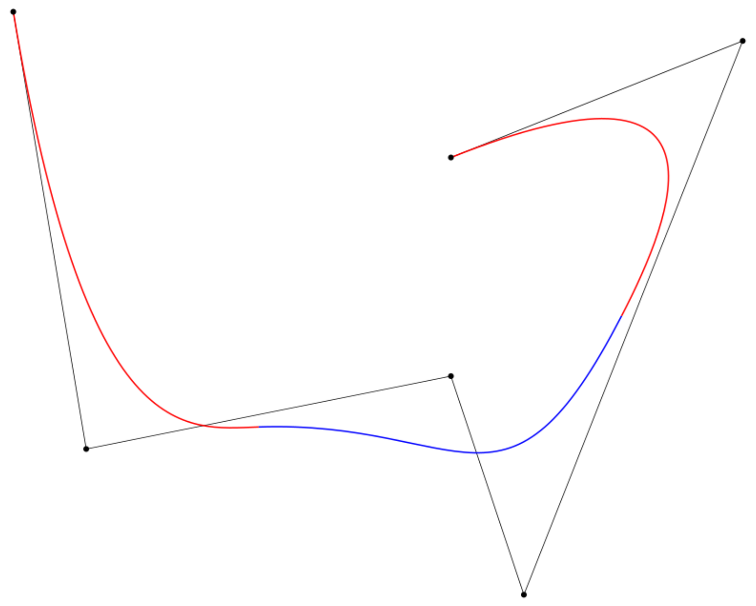


- **Bezier curves**
- *Continuous curvature planning with obstacle avoidance capabilities in urban scenarios*

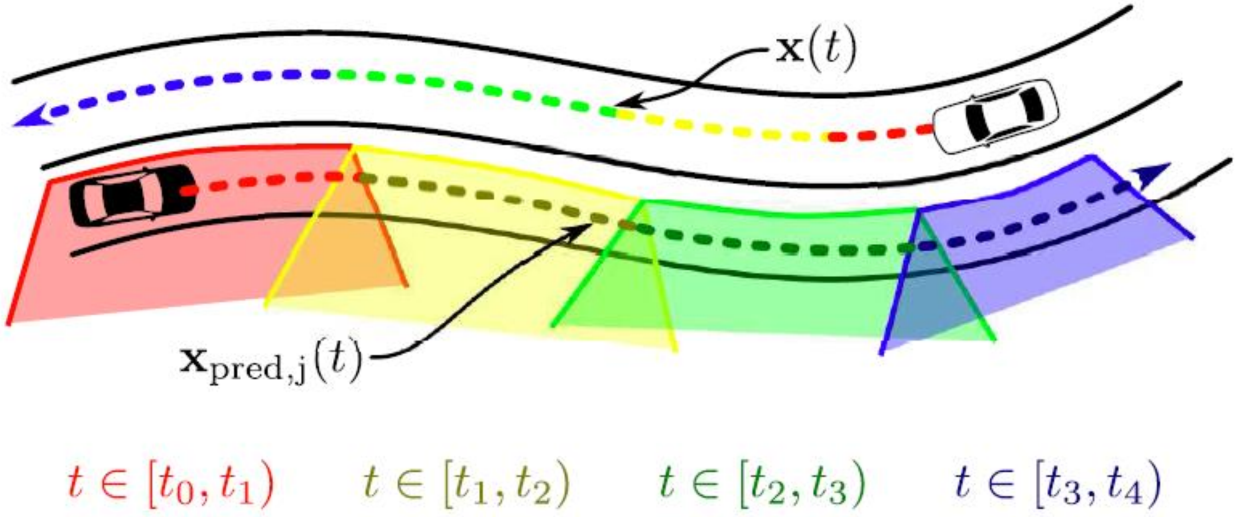


3. Interpolating Curve Planner

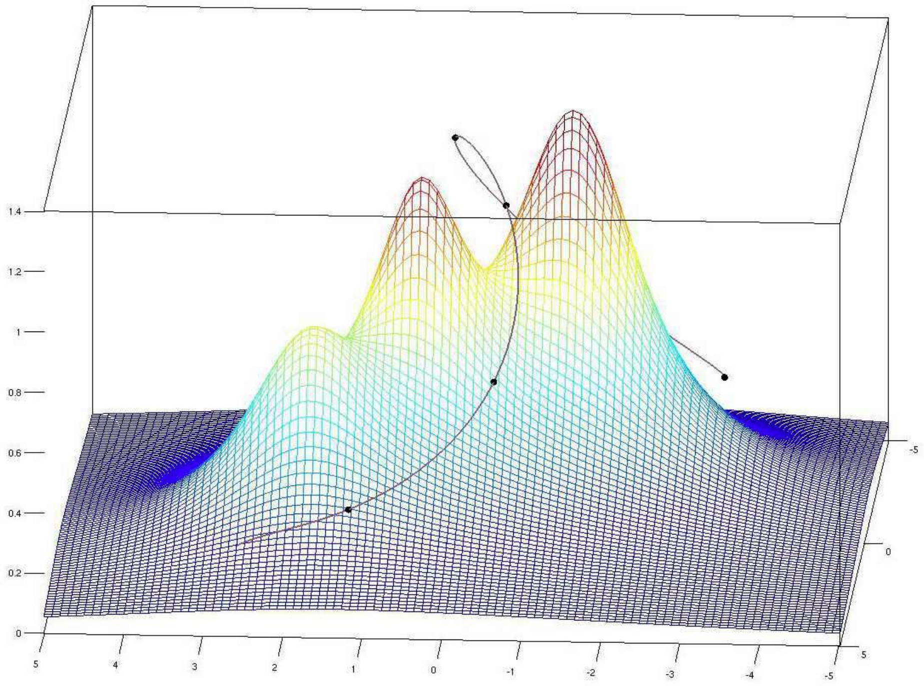
- **Spline curves**
- *Kat-5: Robust systems for autonomous vehicle navigation in challenging and unknown terrain*



- **Function optimization**
- *Trajectory Planning for BERTHA - a Local, Continuous Method*

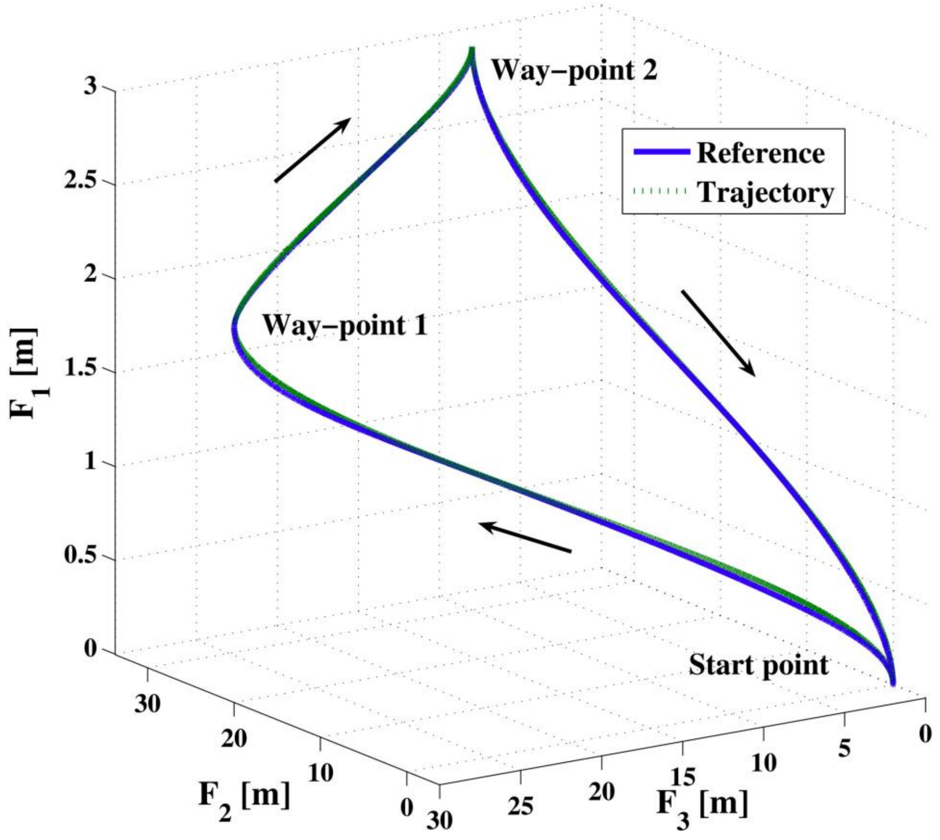


- **Binary linear programming**
- *3D Path Planning in a Threat Environment*

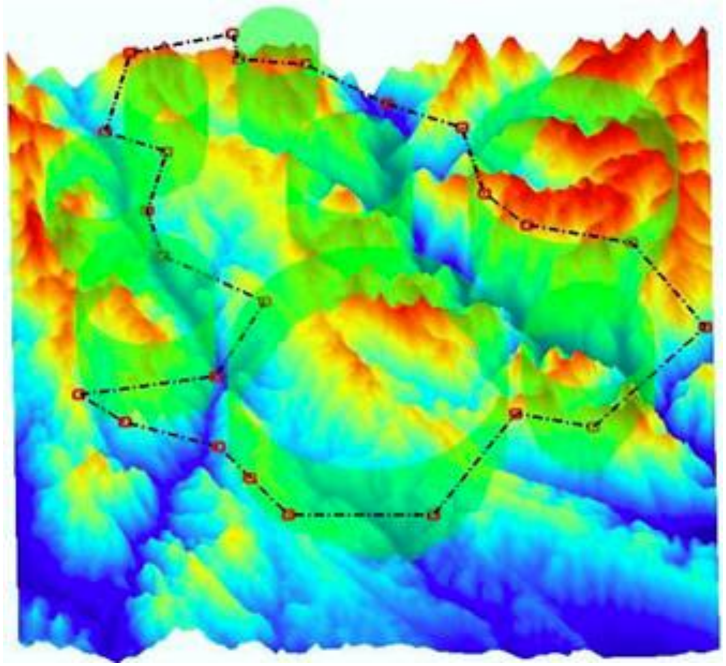


4. Numerical Optimization Approaches

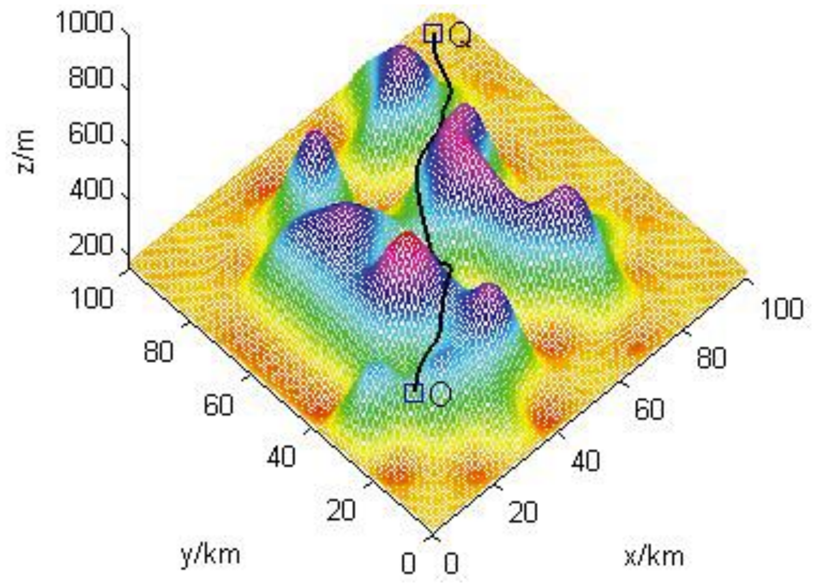
- **Non-linear programming**
- *Flatness-Based Trajectory Planning/Replanning for a Quadrotor Unmanned Aerial Vehicle*



- **Genetic algorithm(GA)**
- *Optimal Path Planning for UAVs Using Genetic Algorithm*



- **Particle swarm optimization (PSO)**
- *Three-Dimensional Path Planning for UAV Based on Improved PSO Algorithm*



- **Ant colony optimization (ACO)**
- *Path Planning for Indoor UAV Based on Ant Colony Optimization*

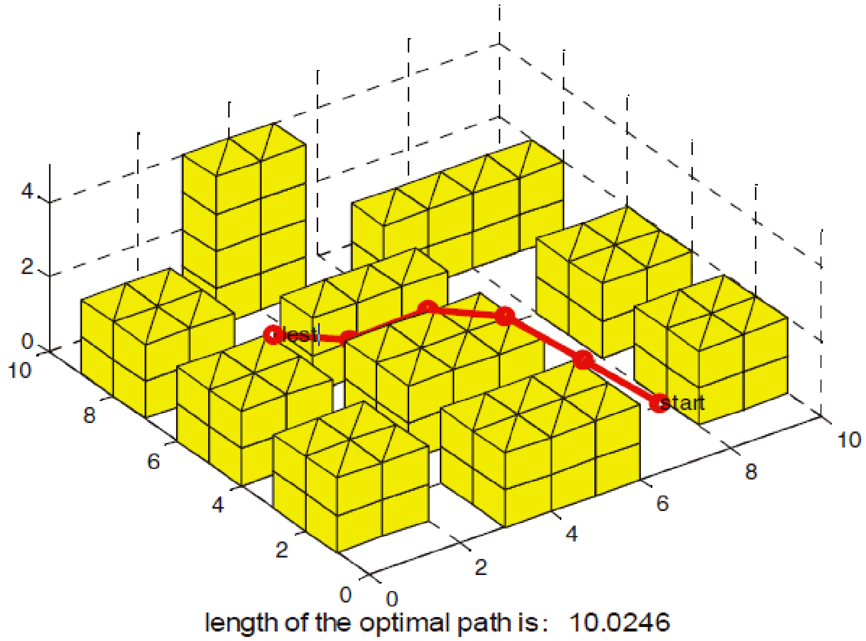


Fig 4. The optimal path in case 1

THANKS YOU!