# Motion Planning

---

## collision detections

- using triangle faces
- model the mesh with spherical meshes
- Convex Decomposition
- use signed distance function (SDF)

## Probabilistic Roadmap Method (PRM)

### algorithm

1. Map construction phase:
    - Randomly sample states in $C_{free}$
        - $C = [\theta 1_{min}, \theta 1_{max}] \times [\theta 2_{min}, \theta 2_{max}] \times ... \times [\theta n_{min}, \theta n_{max}]$
        - How to sample states in $C_{free}$ : Rejection Sampling
            - sample uniformly over $C$
            - Reject the sample not in the $C_{free}$, that is, collide with environment or obstacles.
    - Connect every sampled state to its neighbors
        - choose $k$ closest neighbors to each sampled state
        - check if the line between the two states intersects with any obstacles or environment
            - linearly interpolate between the two states to find if any collision exists
2. Query phase:
    - Run path finding algorithims like Dijkstra to find a path from start state to target state
    - if start state and target state are not in the graph, first find the nearest state to start state and the nearest state to target state and find a path between these two states, then connnect this path the start and target.

### Limitations: Narrow Passages

- issue: sampling in narrow passages are sparse and highly possible to be discarded after collision detection
- solution:
    i. Guassion sampling:
        - Generate one sample $q_1$ uniformly in $C$
        - Generate another sample $q_2$ from a guassion distribution centered at $q_1$ with variance $\sigma$
        - if $q_1 \in C_{free} \wedge q_2 \notin C_{free}$, add $q_1$ to the vertex set.
          attribute: samples are near the boundary, but samples in the narrow passage are still too sparse.
    ii. Bridge Sampling:
        - Generate one sample $q_1$ uniformly in $C$
        - Generate another sample $q_2$ from a guassion distribution centered at $q_1$ with variance $\sigma$
        - $q_3 = (q_1 + q_2)/2$
        - if $q_1 \notin C_{free} \wedge q_2 \notin C_{free} \wedge q_3 \in C_{free}$, add $q_3$ to the vertex set.
          attribute: samples are dense in the narrow passage, but sparse in other areas.
    iii. Hybrid Sampling:
        - use Bridge Sampling to sample $N_1$ verts and Uniform Sampling to sample $N_2$ verts. Then find path among these $N_1 + N_2$ verts.

### suitable application case

- static scenes: if scene is dynamically changing, PRM needs to construct a new map every time

---

## Rapidly-Exploring Random Tree (RRT)

### algorithm

1. start with $q_{start}$
2. decide a destination state for this step.
   - strategy 1: random exploration: randomly sample a state in $C_{free}$
   - strategy 2: greedy exploitation: go along $q_{now}$ and $q_{goal}$
   - set hyperparameter $\beta$ to control the exploration/exploitation trade-off
3. find the nearest neighbor to the destination state on the current tree, denote it as $q_{now}$
4. connect $q_{now}$ to $q_{dest}$ with a straight line path, and go along this line with a step size $\alpha$ to $q_{new}$
5. check if the path collides with any obstacles or environment and if $q_{new} \in C_{free}$ and if $q_{new}$ has already been visited

### refinement

1. RRT-Connect: Grow two trees from $q_{start}$ and $q_{goal}$
2. Shortcutting: refine jerky, unnatural paths
   - Sample two points along the path and check if the line connecting them intersects with any obstacles or environment
   - If the line does not intersect, connect the two points with a straight line path and delete the original path between these two points.
3. Shortcutting with Random Restarts: do RRT multiply times, and apply shortcutting accordingly. Pick the path with the smallest cost.

---

# Control System

---

In the following part, we have $\theta_e(t) = \theta_{dest} - \theta_{now}$

1. Proportional Control
   - $u = K_p(\theta_{dest} - \theta_{now})$
   - if control signal changes derivative of state, we have

     $$\dot{\theta}_{now} = K_p(\theta_{dest} - \theta_{now})$$

     consider that $\theta_e = \theta_{dest} - \theta_{now}$, then we have

     $$\dot{\theta}_{dest} - \dot{\theta}_e = K_p\theta_e(t)$$

     Assume $\dot{\theta}_{dest} = c$, the solution to this ODE is

     $$\theta_e(t) = \frac{c}{K_p} + (\theta_e(0) - \frac{c}{K_p})e^{-K_p t}$$

   - if control signal changes the second derivative of the state, we have

     $$\ddot{\theta}_e(t) + K_p\theta_e(t) = 0$$

     which is a Simple Harmonic Motion
   - For the first case, where control signal changes the first derivative of the state, the final state will have a steady-state error as long as $\dot{\theta}_{dest} \neq 0$; and in the second case, the final state will never stop oscillating.
2. Integral Control
   - $u = K_i\int_0^t(\theta_e t)dt$
   - if control signal changes the first derivative of the state, we have

     $$\ddot{\theta}_e(t) + \theta_e(t) = 0$$

     which is a Simple Harmonic Motion
   - useful when the system has a steady-state error and we want to minimize it.
3. Derivative Control
   - $u = K_d\dot{\theta}_e(t)$
   - Accounts for "future behavior" or "trend"
   - Attempts to reduce overshooting
4. PI Control

- $u = K_p\theta_e(t) + K_i\int_0^t(\theta_e t)dt$
- if the control signal changes the first derivative of the state, we have

$$K_p\ddot{\theta}_e(t) + K_i\dot{\theta}_e(t) + \theta_e(t) = 0$$

  which will have three kind of solutions:
  - over damped: there exist a overshooting
  - under damped: the state converge slowly to the destination
  - critically damped: there is no overshooting, and the state converge fast to the destination

5. PD control
   - $u = K_p\theta_e(t) + K_d\dot{\theta}_e(t)$
   - if the control signal changes the second derivative of the state, we have

$$\ddot{\theta}_e(t) + K_d\dot{\theta}_e(t) + K_p\theta_e(t) = 0$$

   which will also have three kind of solutions like PI control.
6. PID control
   - $u = K_p\theta_e(t) + K_i\int_0^t(\theta_e(t))dt + K_d\dot{\theta}_e(t)$
   - if the control signal changes the first derivative of the state, or the state itself, we will have three solutions like PI control.

## compare between different control systems

Effects of *increasing* a parameter independently

| **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 |