1 Configuration Space Path Planning

- Reference: 1) A Simple Motion Planning Algorithm for General Purpose Manipulators by T. Lozano-Perez, 2) Siegwart, section 6.2.1
- Fast, simple to implement
- Can handle multiple DOF Robots, complicated non-convex obstacles
- Can plan motion in cluttered environments
- A key component of task level programming “Plan Path (collision free) from start to goal”
- Use an approximate, discrete, quantized approach to reduce mathematical complexity. No attempt to model object surfaces
- Configuration: Set of parameters that completely specify the position of an object. We can use joint space for N-DOF Robot as its configuration. Cartesian not unique
- C-Space: Set of all possible configurations
- Key idea for Path Planning: Map obstacles into the robot’s C-Space. Creates regions of C-Space that contain obstacles and free-space. Then plan path in C-Space.
- Problem: Joint Space can still be high dimensional.
- Solution: Use “slices” of C-Space for path planning. Projections of N-DOF space into set of N-1 Dimensional Projections.
- 2-D C-Space creation and Path Planning is simple. Represent 3-D as 2-D slices and plan in these spaces.
- Basic Idea: Determine range of legal joint values for a joint parameter given ranges of the previous joints.
- Advantages: Planning becomes simple search in 1 and 2-D space. Disadvantages: Loss of accuracy (granularity of projection), increased storage as DOF increases.
- Tradeoff: Simple vs. intersecting high DOF manifolds of objects and manipulators.

![Configuration Space for Two Link Manipulator with obstacles](image)

Figure 1: Configuration Space for Two Link Manipulator with obstacles
2 Configuration Space Path Planning Examples

Below are some simple examples of how Configuration Space (CSPACE) path planning works. We will be using a two-link, planar, pick-and-place manipulator as in figure 1. This design allows us to avoid computing arm collisions with the obstacles - we just have to worry about the gripper (assumed to be a point) moving into an obstacle.

![Figure 2: Two Link Manipulator used in examples](image)

We give the system Cartesian start and goal positions. Each Cartesian position has 2 inverse kinematic solutions, so the path planner can plan 4 distinct paths from each distinct start and goal position. A set of known obstacles is given as part of the environment. Here is the algorithm:

1. Partition each of the robot’s joints into discrete regions, say every 5 degrees. This creates a $72 \times 72$ discrete angular grid.

2. Iterate over all combinations of joint 1 and joint 2 (we use the center of each 5 degree range as the test point). If the forward kinematics for these joint angles lies inside an obstacle, we classify this part of the $72 \times 72$ grid as forbidden, otherwise it is a legal configuration.

3. Given a Cartesian start and end position, do the following:
   (a) Using Inverse Kinematics, find the solutions to $\theta_1$ and $\theta_2$ for the start and goal position. There will be 2 solutions for each position.
   (b) Choosing 1 of the 4 possible solutions, perform a breadth-first search in the joint-space grid from $(\theta_{1\text{start}}, \theta_{2\text{start}})$ to $(\theta_{1\text{goal}}, \theta_{2\text{goal}})$. Mark the path.

We will use a simple Breadth First Search in Configuration Space of the manipulator to find a path in free space between start and goal positions represented as joint configurations. Pseudo-code below.

```
FINDPATH(\theta_{1\text{start}}, \theta_{2\text{start}}, \theta_{1\text{goal}}, \theta_{2\text{goal}})
if((\theta_{1\text{start}}, \theta_{2\text{start}}) \equiv (\theta_{1\text{goal}}, \theta_{2\text{goal}})) then stop- GOAL=FOUND
ADD((\theta_{1\text{start}}, \theta_{2\text{start}})) to QUEUE and mark as VISITED
While QUEUE NOT EMPTY and GOAL NOT FOUND do
   POP top item in QUEUE and assign to (\theta_1, \theta_2)
   if((\theta_1, \theta_2) \equiv (\theta_{1\text{goal}}, \theta_{2\text{goal}})) then GOAL=FOUND
   else ADD to QUEUE each Free Space UNVISITED 4-neighbor of (\theta_1, \theta_2)
   Mark each neighbor added to QUEUE as VISITED
   and Remember which node Opened this node (its predecessor, (\theta_1, \theta_2))
If GOAL==FOUND
Recreate path by following predecessor chain from (\theta_{1\text{start}}, \theta_{2\text{start}}) to (\theta_{1\text{goal}}, \theta_{2\text{goal}})
```

There are 4 examples that follow showing the joint-space and Cartesian-space paths for the same Cartesian start and goal positions using each of the 4 possible inverse kinematic solutions. In each example, the manipulator has link lengths $\text{L}_1 = 5$ and $\text{L}_2 = 3$.

In the joint-space diagrams, the path from start to goal is shown with asterisks. In the Cartesian-space plots, the path and obstacles are shown along with the manipulator links superimposed at every 5th point on the path.
Cartesian Start: (0, 4.1) Goal: (4.9, -5): (q1start, q2start) = (53, 127) (q1goal, q2goal) = (294, 58)
Cartesian Start:(0.4, 1) Goal:(4.9, -5): (q1start, q2start) = (127, 233) (q1goal, q2goal) = 336, 302
Cartesian Start: (0, 4.1) Goal: (4.9, -5): (q1 start, q2 start) = (53, 127) (q1 goal, q2 goal) = 336, 302

Planned Path 3
Cartesian Start: (0, 4.1) Goal: (4.9, -5): (q1start, q2start) = (127, 233) (q1goal, q2goal) = 294, 57

Planned Path 3