A Data Structure for Rapidly Testing Reachability for Robotic Motion Planning

Caelan Garrett and Tomás Lozano-Pérez

MIT Computer Science and Artificial Intelligence Laboratory

Introduction

Problem

- Robot plans movements and actions to accomplish goals in a known environment
- Needs a method of determining reachability: places the robot can move without colliding with obstacles
- Current approach implicitly tests if one point is reachable but:
  - Complexity $O(n^3 \log(n))$ for 3 degrees of freedom
  - Inefficient for testing many points and testing regions

Approach

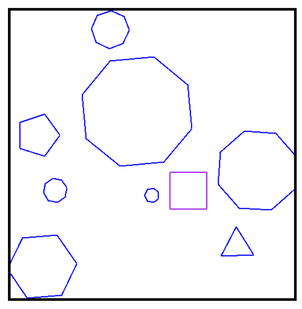
- Represent robot configuration space as a data structure with spatial partitioning tree and graph properties
  - $O(1)$ to explicitly test region and point reachability
  - Quick updates to changes in environment
  - New applications for smart object placement

PR2 Robot Planning and Moving Objects

Configuration Space

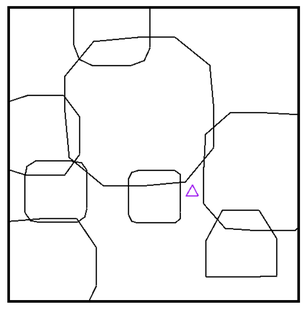
Definition

- Configuration space users the robot's and obstacles' geometries to represent points that the robot could be at without collisions by growing the obstacles
- Simplifies motion planning by reducing it to moving a point through a space
- Robot locations without collisions are points outside the grown objects
- Configuration space has dimensions of the degrees of freedom of the robot



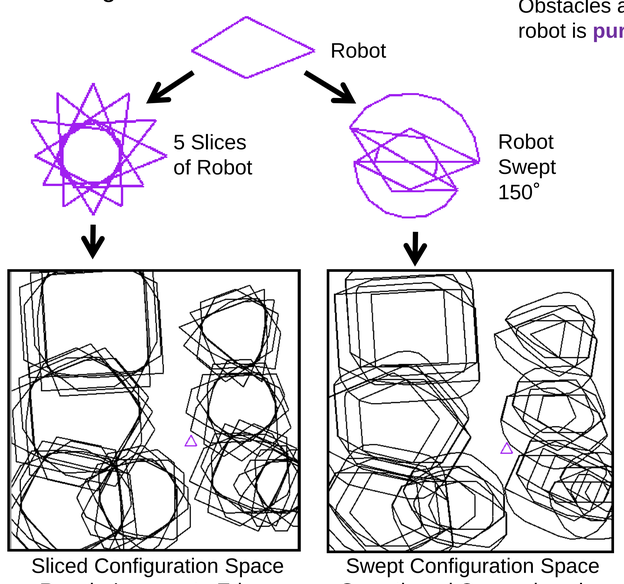
Robot Environment (Real)

Obstacles are **blue** and the robot is **purple**



Configuration Space (Abstract)

Grown obstacles are **black** and the robot 's reference point is a **purple** triangle



Sliced Configuration Space
Rough, Inaccurate Edges

Swept Configuration Space
Smooth and Comprehensive

Swept Volumes

- Robot moves in a 3 dimensional configuration space of its x, y location and θ orientation
- Easy to grow under one orientation
- Hard to continuously grow under many orientations (swept volume)
  - Usually approximated by slicing several orientations and growing each separately
  - Because our system tests regions of the configuration space at once:
    - Can improve performance by sweeping robot before growing obstacles under angle ranges
    - No misclassification of bad points

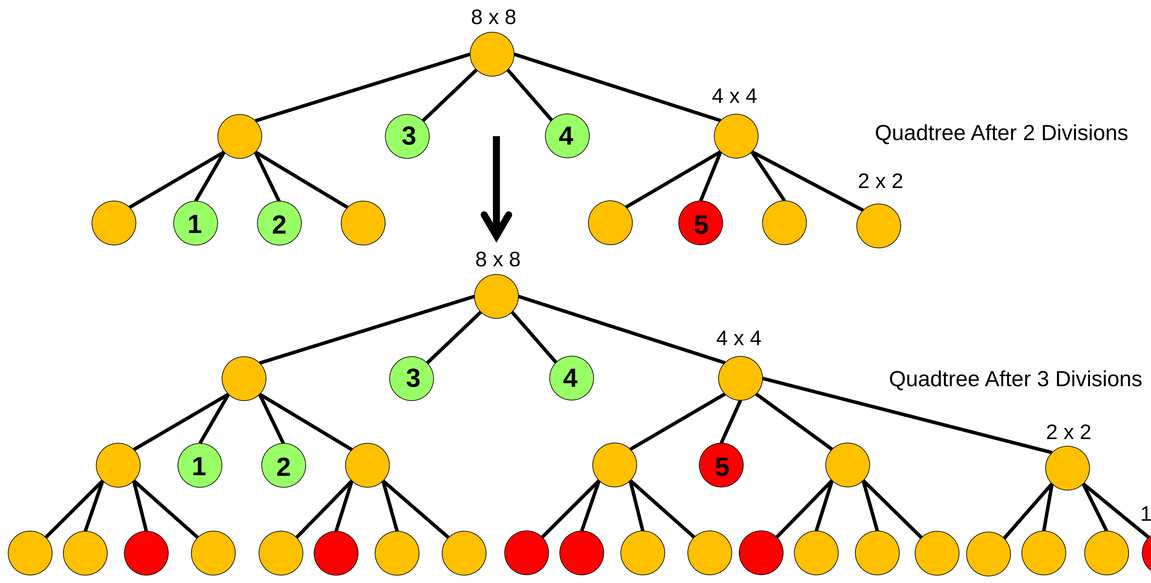
Future Work

- Map representation of robot base pose configuration space to hand pose configuration space to create new space of hand reachable locations
- Include selection of different robot hand grasps in this new space
- Account for regions unseen by the robot's visibility space as special obstacles in the world
- Integrate data structure with PR2 robot planning modules to replace existing implicit functions

Data Structure

Spatial Partitioning Tree

- Configuration spaces in R^d are represented as hierarchical trees of d-hypercubes
- Nodes represent cubic regions in the where all points have the same property
- If not all points are alike, the node splits into 2^d children who combined compose the node
- Characterize nodes by if all, none, or some of their points are free

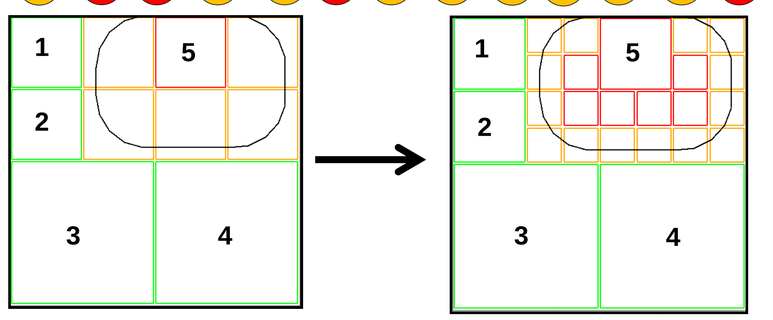


Quadtree representing a 2 dimensional configuration space

Shows split between 2nd and 3rd divisions

Satisfied nodes in the 2nd division are labeled

Green nodes are free from collisions
Red nodes are all colliding
Orange nodes are mixed

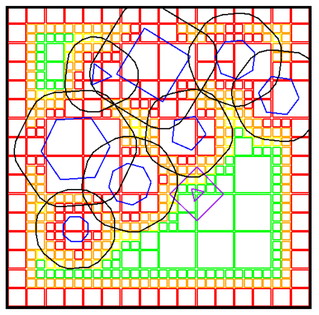


Configuration Space After 2 Divisions

Configuration Space After 3 Divisions

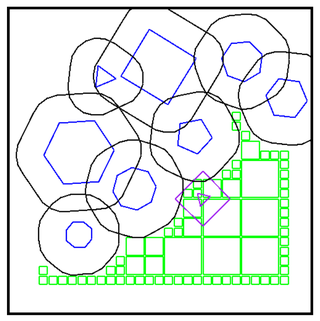
Octree

- An octree represents the robot's x, y, θ configuration space ($d = 3$)
- Tree stops splitting at some resolution
- The robot avoids nodes that are only partially free at the lowest resolution
- Approximation isn't harmful because when moving under error, moving close to objects will likely cause collisions
- $O(1)$ Depth implies quick point testing

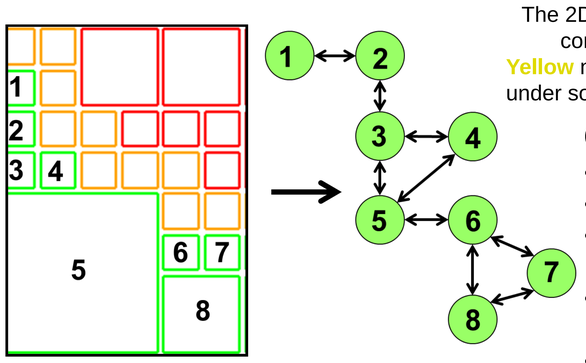


The 2D projection of the full configuration space

Yellow nodes are not colliding under some angle



The 2D projection of the reachable region

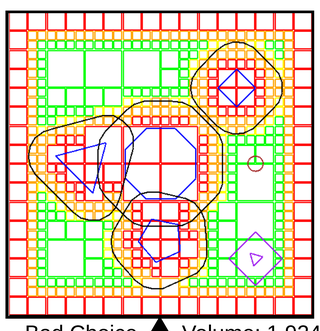


Quadtree leaves and corresponding graph

Graph

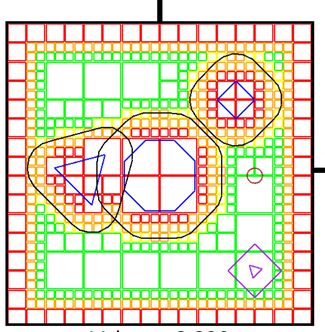
- Leaves of the tree cover the space
- Connect leaves to form graph of free regions
- Neighbors easily determined by searching the tree for nodes intersecting adjacent areas
- DFS to determine connectedness and find arbitrary paths
- Modified Dijkstra can find shortest paths

Object Manipulation

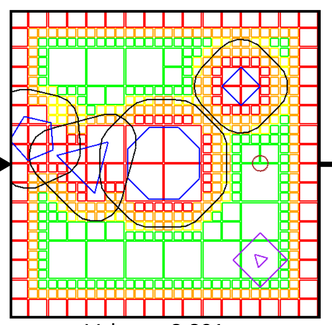


Bad placement from choosing a vertex of the cut to place the object

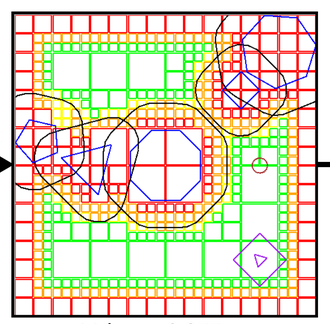
Bad Choice Volume: 1.924



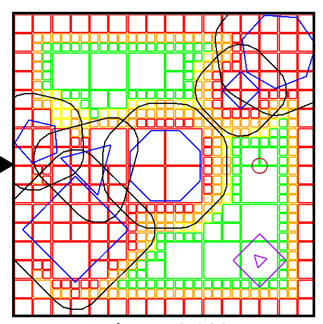
Start Volume: 3.896



1st Object Volume: 3.891



2nd Object Volume: 3.877



3rd Object Volume: 2.632

Goals

- Want smart, reachable placements
- Placements should maximize the available placement and movement space without blocking paths to goals
- Optimal packing is NP-Hard; want an efficient approximate placement
- Cost function is the amount of connected, free configuration space volume
- Data structure complements problem by providing quick estimates of the volume displaced by a placement and new connectedness of region

Approach

- Create candidate list from object's configuration space
- Filter the list to eliminate not reachable placements
- Temporarily add the object at each remaining points by growing a fake subtree rooted in the free leaves that intersect the object
- Modify the neighbors of these temporary subtrees and DFS on the robot region to determine the new volume
- Select and fully add the object at the point with the maximum volume by expanding the tree

Robot placing 3 objects in sequence. It successfully reduces the area affected while avoiding placing objects in locations that would disconnect it from its goal (the **brown** circle)

PR2 robot placing a chair
Because chairs are comparable in size to the robot, bad placements can affect the robot's movement