# Point-point plANner

## Algorithm from ROS sbpl package (search based planning lattice) algorithm

### developed by Maxim Likhachev at the University of Pennsylvania in collaboration with Willow Garage

## Search-based ARA\* planning in x, y, theta space

### Library of motion primitives

#### Arcs, lines, turn-in-place

#### Kinematically plausible motions

#### Cost can be assigned to each motion – e.g. make spins and reverses more expensive because they require coming to a halt

#### Natural to pick motions corresponding to path segments

### Fast

#### Nominal runtime .1-.2 sec for planning several meters in relatively clear setting

#### Runtime increases for difficult moves, especially requiring backwards motion or tight squeezes

### Kinematically plausible paths

#### Checks every pose for collision against a 2D costmap, resolution [X] cm^2, against the robot’s full footprint

#### Some pictures showing loops, back up and turn, etc

## Modifications

### Custom motion primitives for Harlie

### Changed output to produce path segments

### Correct for discretization error

# Overall planning

## Designed to allow dynamic replanning without the robot coming to a halt

## Splits path into two sections

### Committed path

#### Short-term plan

#### About 1m, long enough to keep the robot moving for 1-2 seconds

#### Try hardest not to change this - cannot be changed without bringing robot to a halt

#### This is actually handed off to steering

### Uncommitted path

#### Long-term plan

#### Subject to change

## When uncommitted path drops below a length threshold, segments are moved from uncommitted to committed if available

## When committed path runs out (maybe planning is taking a very long time) robot comes to a halt at end of committed path to wait for more segments

## Modifying committed path is done as last resort

### Obstacle detected

### Coming to a halt would be less painful – if target moves behind the robot

## Multithreaded approach – planner thread, feeder thread

### Planner thread

#### Receives new goals

### Tries to ensure feeder thread always has enough path

### Passes

# Future work: separate short-distance planning algorithm

## Robot shows weakness tracking user at short range, turning around

|  |  |  |
| --- | --- | --- |
| State | Action | Caused by |
| Recovery |  | Full replan failure |
| Full replan | * Bring robot to halt | * Partial replan failure * obstacle detected in committed path |
| Partial replan |  | * New goal * obstacle detected in uncommitted path |
| Good |  |  |