2d Navigation

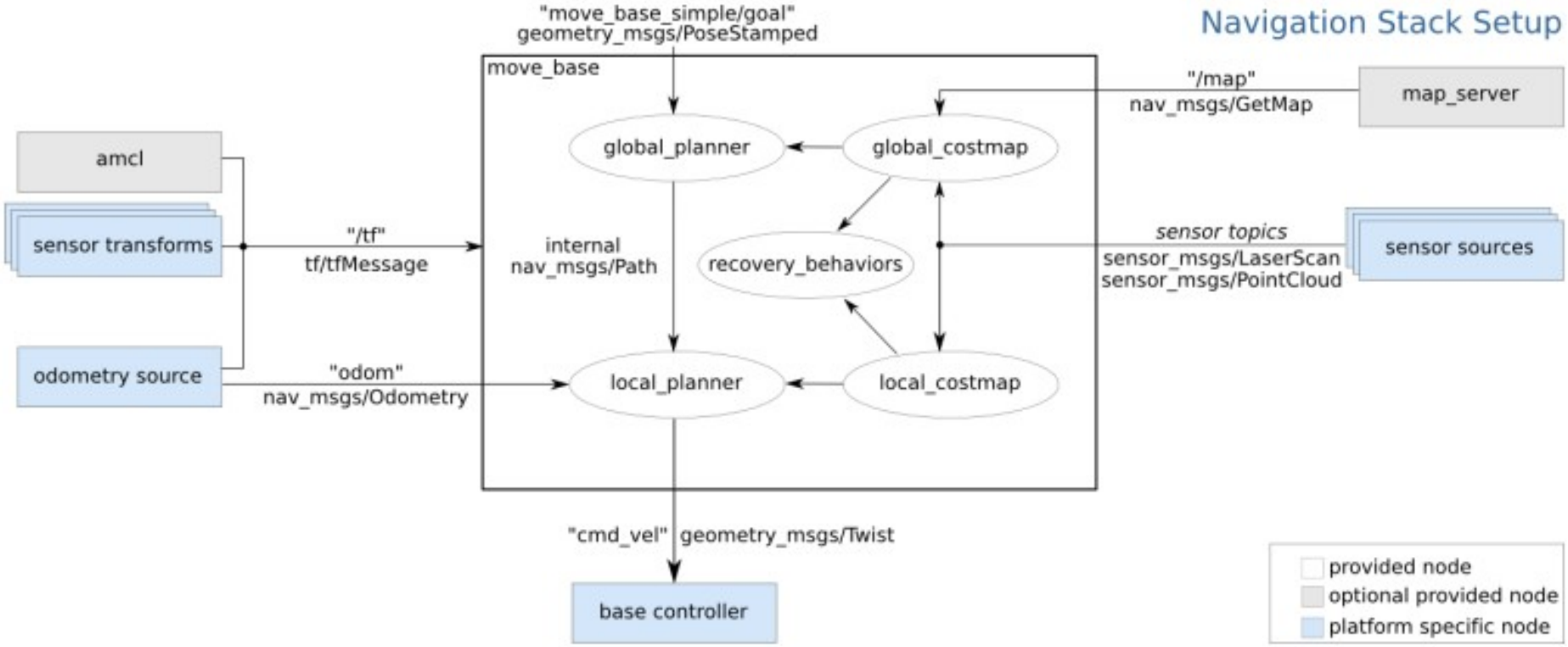
Dr. Rainer Hessmer

June 2011

Navigation in a Nutshell

- Map
- Start pose, goal pose
- Global plan
- Monte Carlo Localization for pose estimation (where am I?)
  - Odometry
  - Laser scan data
- Local plan → drive commands

ROS Navigation Stack



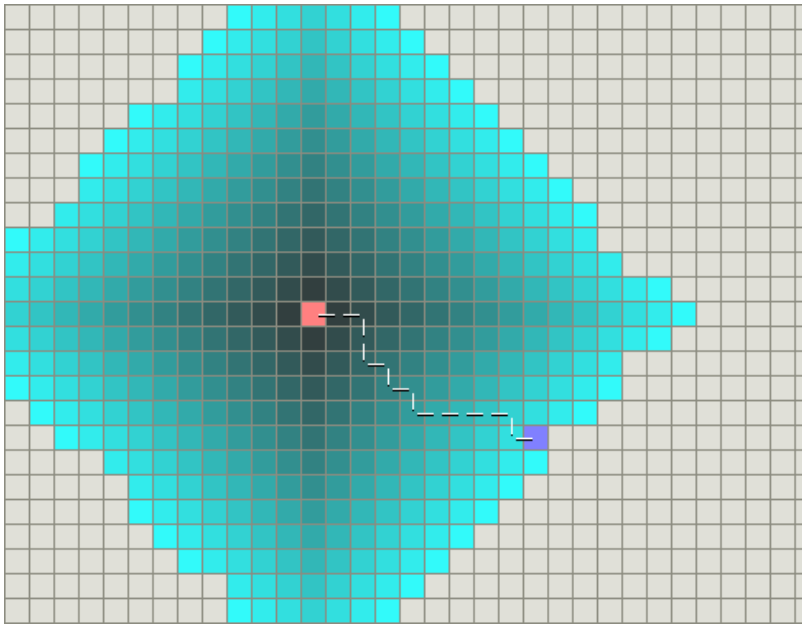
from
http://www.ros.org/wiki/navigation/Tutorials/RobotSetup

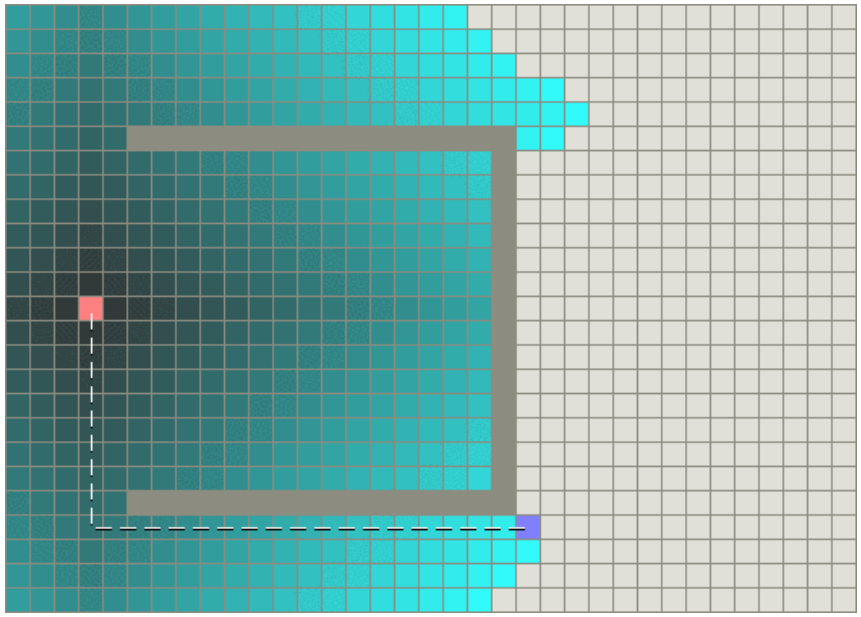
Global Path Planning

- Ingredients:
  - Map
  - Start pose, goal pose
  - Robot footprint
  - Path finding algorithm
    - Dijkstra
    - Best-First-Search
    - A*

Dijkstra's Algorithm

- Easy to understand
- Finds a shortest path
- Fairly inefficient



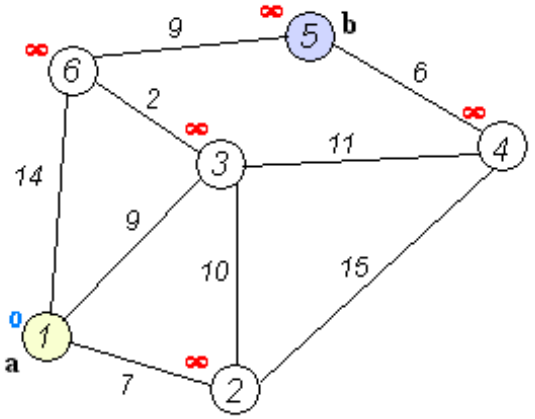


Images from http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Dijkstra

Generalized distance = cost

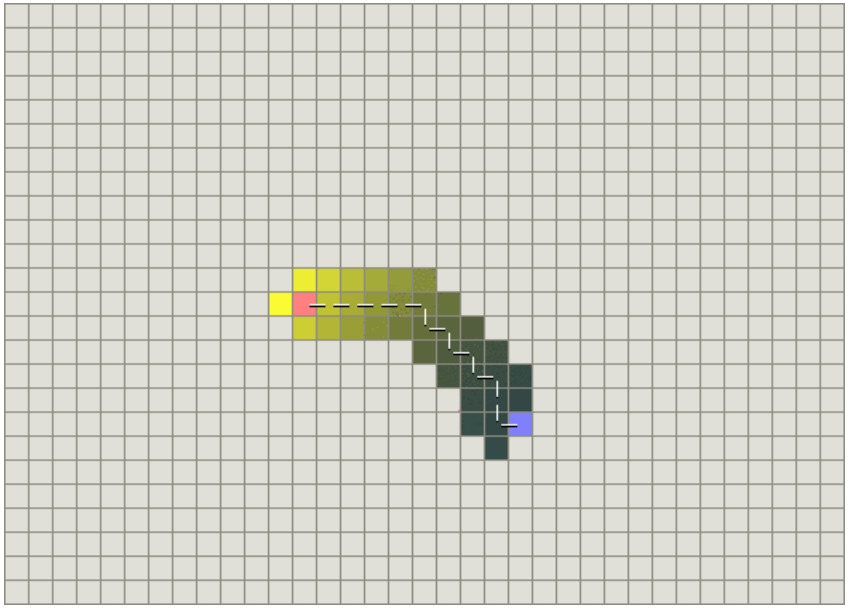
http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

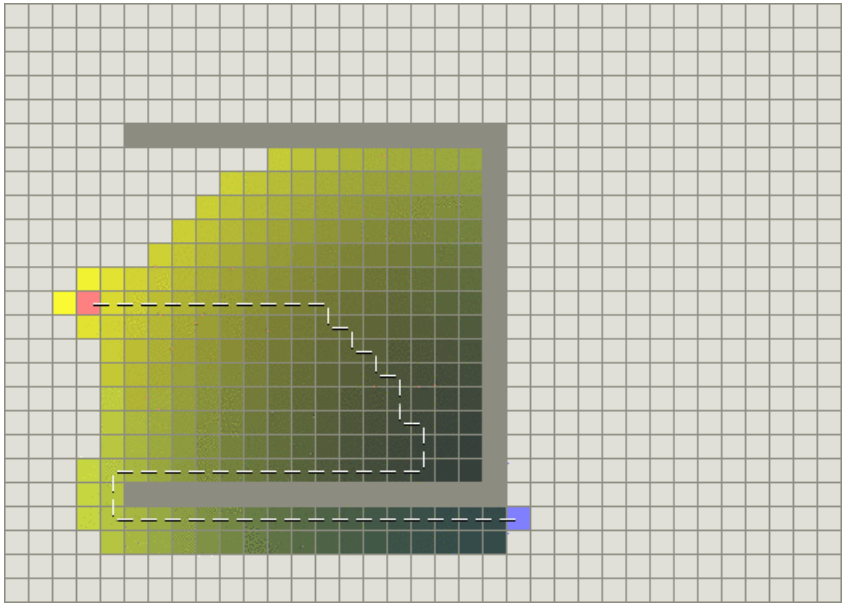


- Setup
  - Starting point = initial node
  - Cost of node Y: Cost to get from initial node to Y
  - Every node gets a cost value (e.g. distance): Initial node = 0; others infinity (∞)
  - Mark all nodes as unvisited. Set initial node as current.
- Lather, rinse, repeat
  - For current node, iterate through unvisited neighbors and calculate tentative cost. If this distance is less than previously recorded cost, overwrite the cost.
  - When all neighbors of the current node are considered, mark current node as visited. No need to check again. Recorded cost is final and minimal.
  - If all nodes visited, finish. Otherwise, unvisited node with smallest distance becomes next current node. Repeat.

Greedy Best-First-Search

- Uses estimate (heuristic) for how far from the goal a cell (vertex) is
- Fast but does not guarantee shortest path

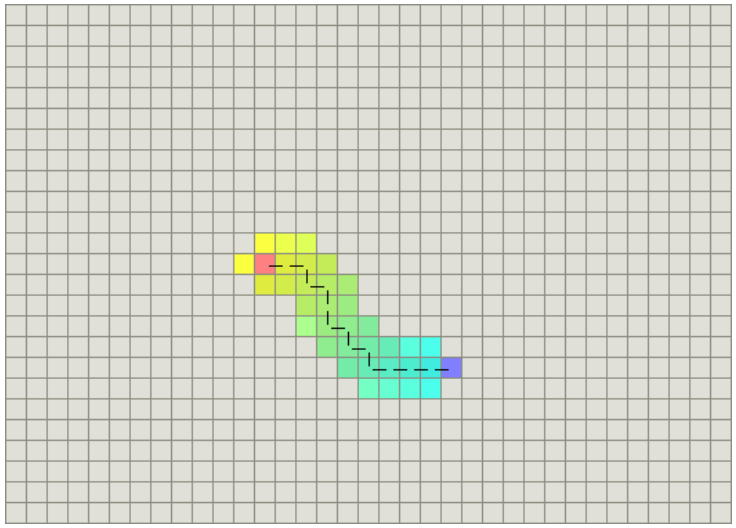


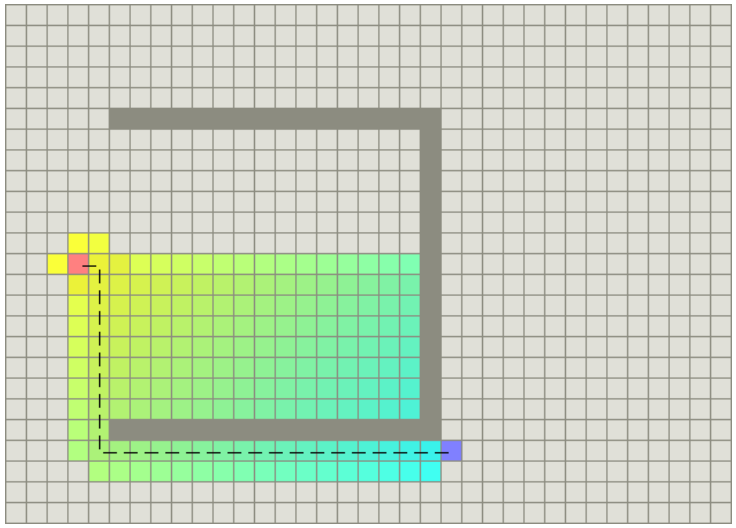


Images from http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

A* Algorithm

- Combines Dijkstra with Best-First-Search
- Finds a shortest path
- Uses heuristic to perform directed search → much faster than Dijkstra
- Uses vertex n that has the lowest
  f(n) = g(n) + h(n)
  - g(n): cost of the path from starting to vertex n
  - h(n): optimistic heuristic for cost from vertex n to the goal.





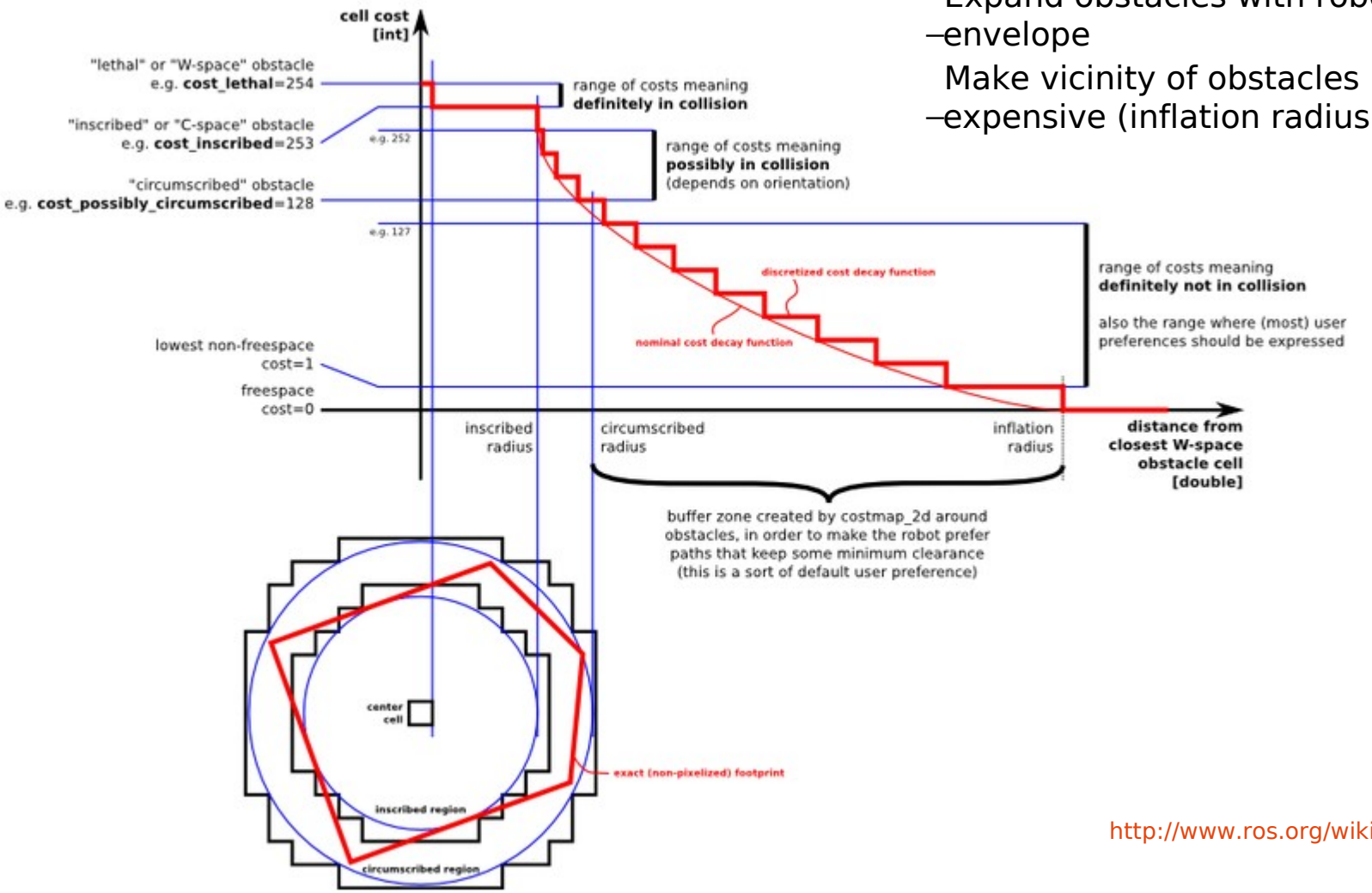
Images from http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

ROS Global Path Planner

- Generates a high level plan for the navigation stack to follow (Dijkstra's algorithm)
- Create a series of waypoints for the local planner to achieve.
- Grid-based, assumes circular robot.
- Produces waypoints that are optimistic for the actual robot footprint → path may be infeasible to follow
- Does not take robot dynamics into account → potentially dynamically infeasible path
- Links:
  - http://www.ros.org/wiki/navigation/GlobalPlanner
  - http://www.ros.org/wiki/navfn

Cost Function

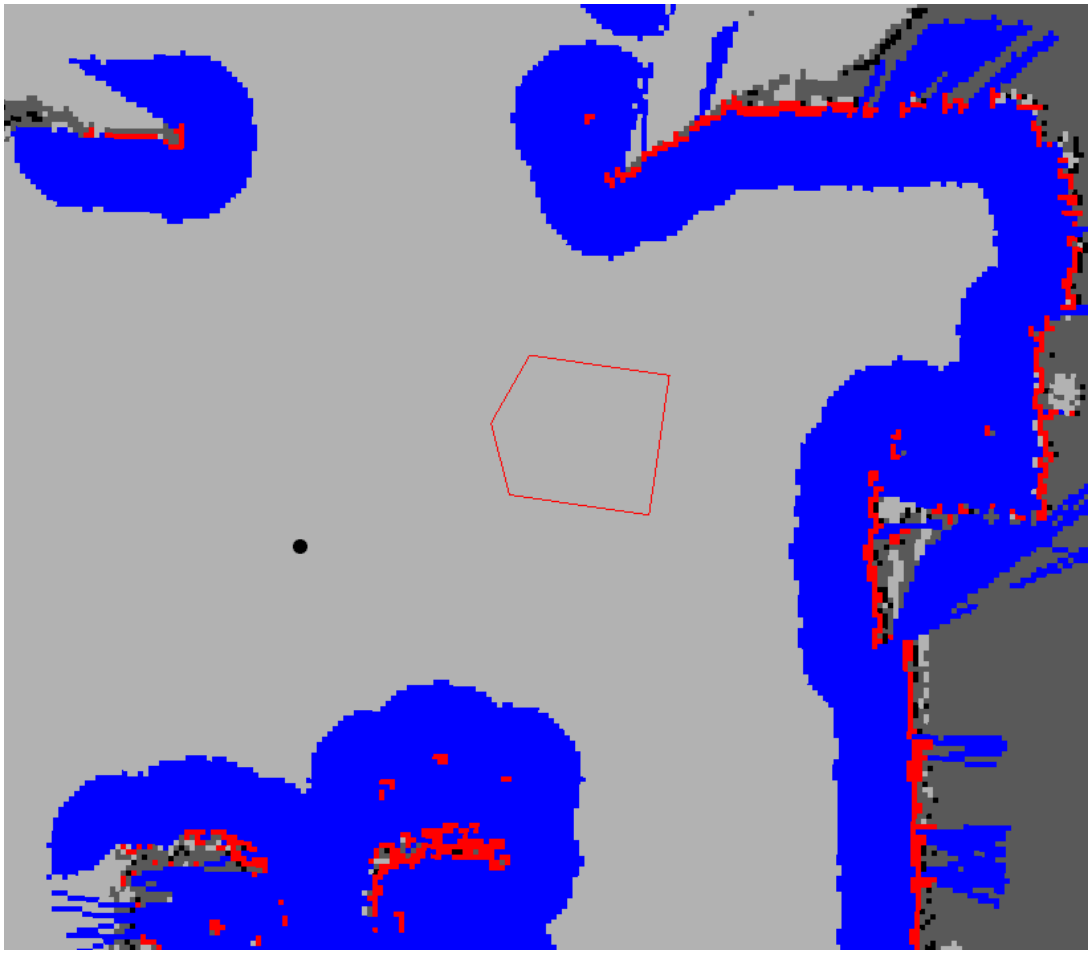
- Obstacle locations have infinite cost
- Expand obstacles with robot footprint – envelope
- Make vicinity of obstacles more expensive (inflation radius)



http://www.ros.org/wiki/costmap_2d

Cost Map Example

- Red cells: Obstacles
- Blue cells: Obstacles inflated by inscribed radius of the robot
- Red polygon: Robot footprint



http://www.ros.org/wiki/costmap_2d

Recap

- Map + Costs = Global Cost Map
- Global Cost Map + Start + Target = Global Plan (way points)

- Now we are ready to actively navigate

Where am I

- AMCL: Adaptive Monte Carlo Localization (http://www.ros.org/wiki/amcl)
  - Odometry
  - Laser scan data
  - Global (static) map
- For animation see my web page: http://www.hessmer.org/robotics/monte-carlo-location-for-robots

Where do I go from here

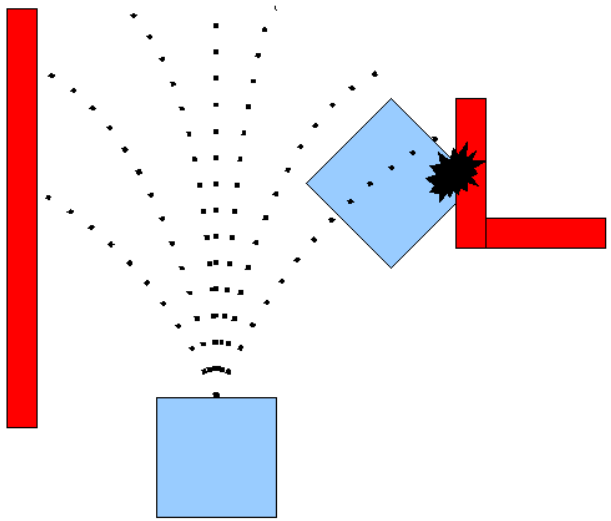
- Local path planning
- Local cost map
- Stay close to the global path
- Adhere to dynamics of the robot (min/max speed and acceleration)
- Generate drive commands

```
TrajectoryPlannerROS:
  max_vel_x: 0.2
  min_vel_x: 0.05
  max_rotational_vel: 0.5
  min_in_place_rotational_vel: 0.1

  acc_lim_th: 3.2
  acc_lim_x: 2.5
  acc_lim_y: 0

  holonomic_robot: false

  # goal tolerance parameters
  yaw_goal_tolerance: 0.1
  xy_goal_tolerance: 0.2
  latch_xy_goal_tolerance: true
```



For more details see:
http://www.ros.org/wiki/base_local_planner

Local Path Planning (2)

- Discretely sample in the robot's control space (dx,dy,dtheta)
- For each sampled velocity, perform forward simulation from the robot's current state to predict what would happen if the sampled velocity were applied for some (short) period of time.
- Evaluate (score) each trajectory resulting from the forward simulation, using a metric that incorporates characteristics such as: proximity to obstacles, proximity to the goal, proximity to the global path, and speed. Discard illegal trajectories (those that collide with obstacles).
- Pick the highest-scoring trajectory and send the associated velocity to the mobile base.
- Rinse and repeat.

From: http://www.ros.org/wiki/base_local_planner

Cost Function

cost =

path_distance_bias * (distance to path from the endpoint of the trajectory in map cells or meters depending on the meter_scoring parameter)

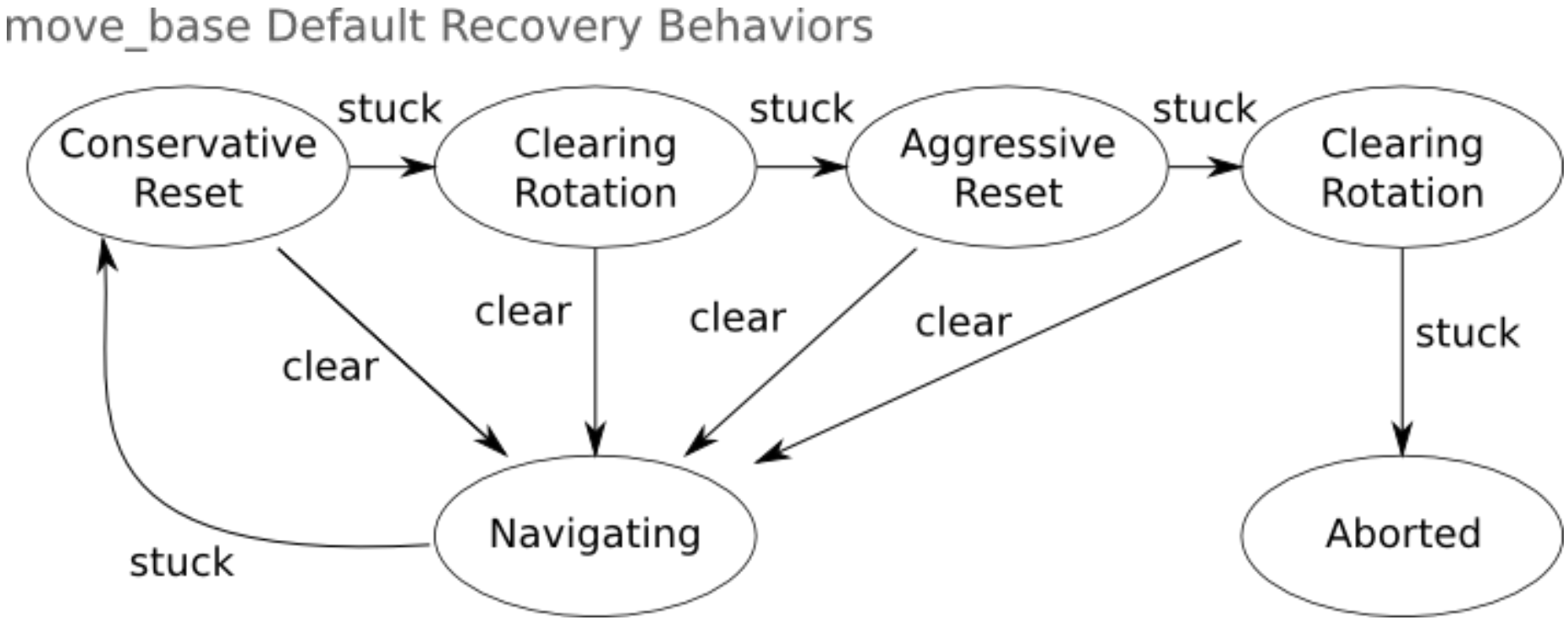
+ goal_distance_bias * (distance to local goal from the endpoint of the trajectory in map cells or meters depending on the meter_scoring parameter)

+ occdist_scale * (maximum obstacle cost along the trajectory in obstacle cost (0-254))

Why a Local Cost Map?

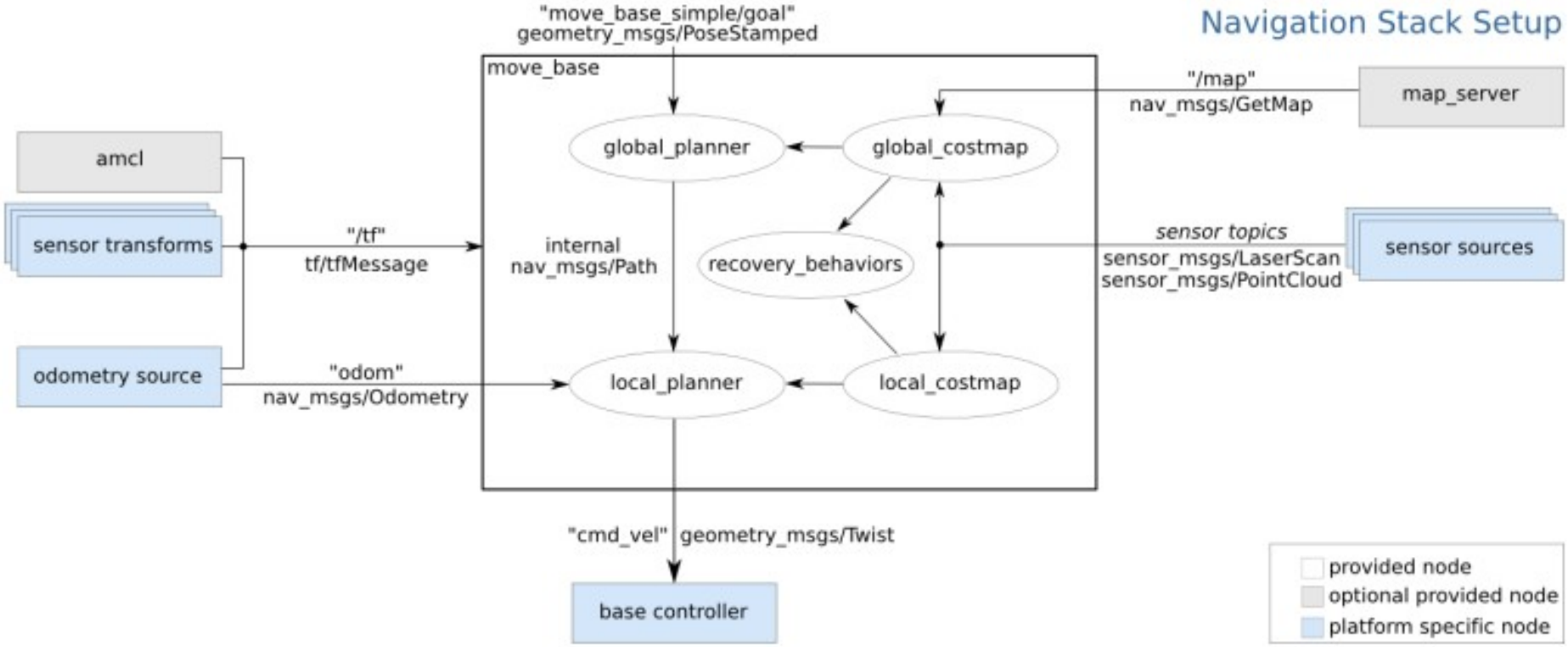
- Adhere to the actual obstacles rather than the situation that existed when the global (static) map was created.
- Avoid dynamic obstacles like humans walking around
- Issues:
  - How long to keep obstacles in memory (e.g. walking human)
  - Potential to get blocked completely (e.g. human circling the robot)
  - Need for escape (recovery) behavior

Recovery Behaviors



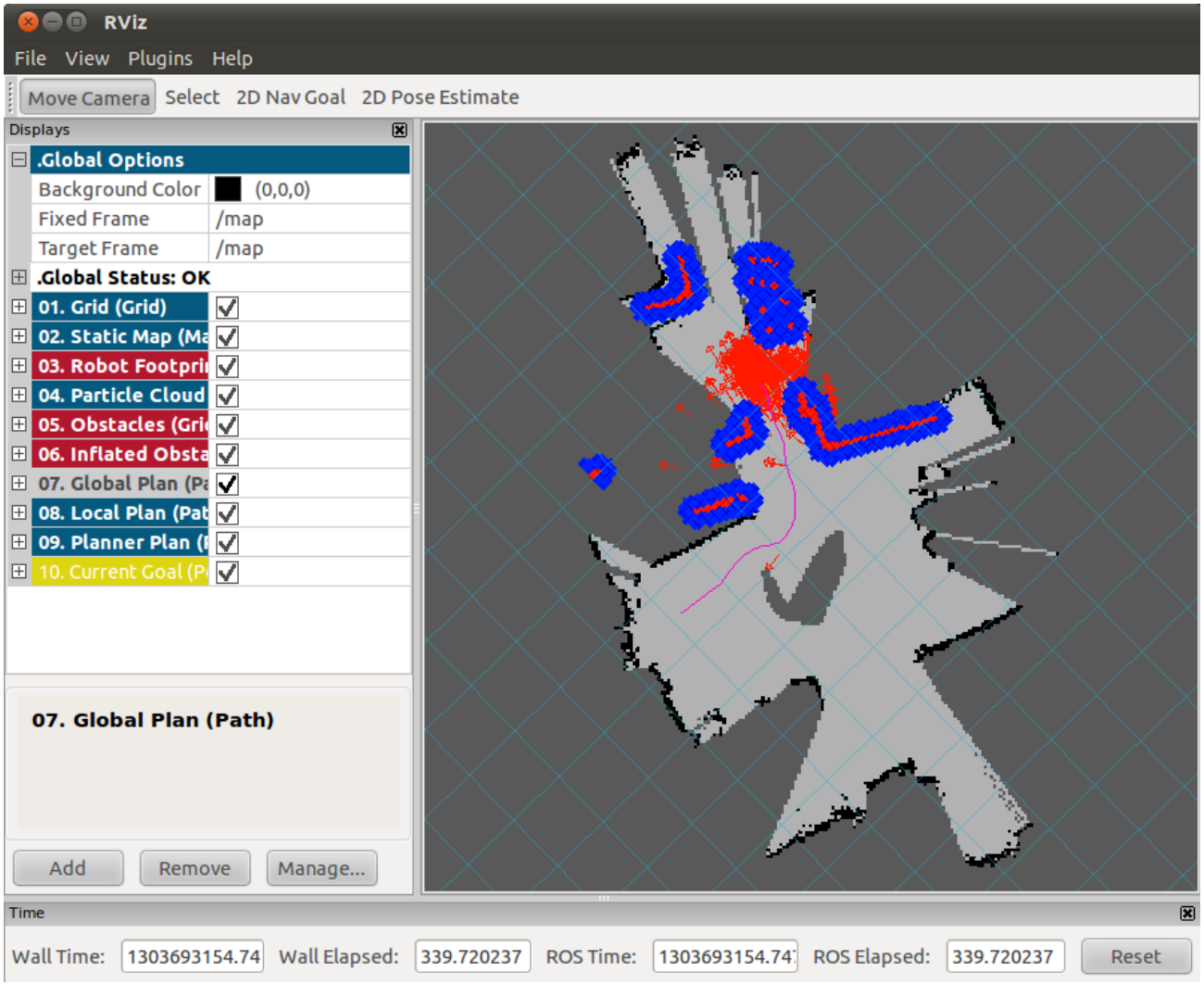
From: http://www.ros.org/wiki/move_base

ROS Navigation Stack



from
http://www.ros.org/wiki/navigation/Tutorials/RobotSetup

Navigation in Action



http://www.youtube.com/watch?v=j-5iaRHzZlM&feature=player_detailpage

Outlook

- Navigation without a global map
- Autonomously exploring an area

References

- ROS Navigation
  http://www.ros.org/wiki/navigation
- Navigation Tuning Guide
  http://www.ros.org/wiki/navigation/Tutorials/Navigation%20Tuning%20Guide
- The Office Marathon: Robust Navigation in an Indoor Office Environment
  http://www.willowgarage.com/sites/default/files/icra2010_marder-eppstein.pdf

Questions?