A Self-Localization and Path Planning Technique for Mobile Robot Navigation

Jia-Heng Zhou and Huei-Yung Lin
National Chung Cheng University

9th World Congress on Intelligent Control and Automation (WCICA), 2011

Presented by: Dereck Wonnacott
Michigan Technological University
Nov 26, 2012

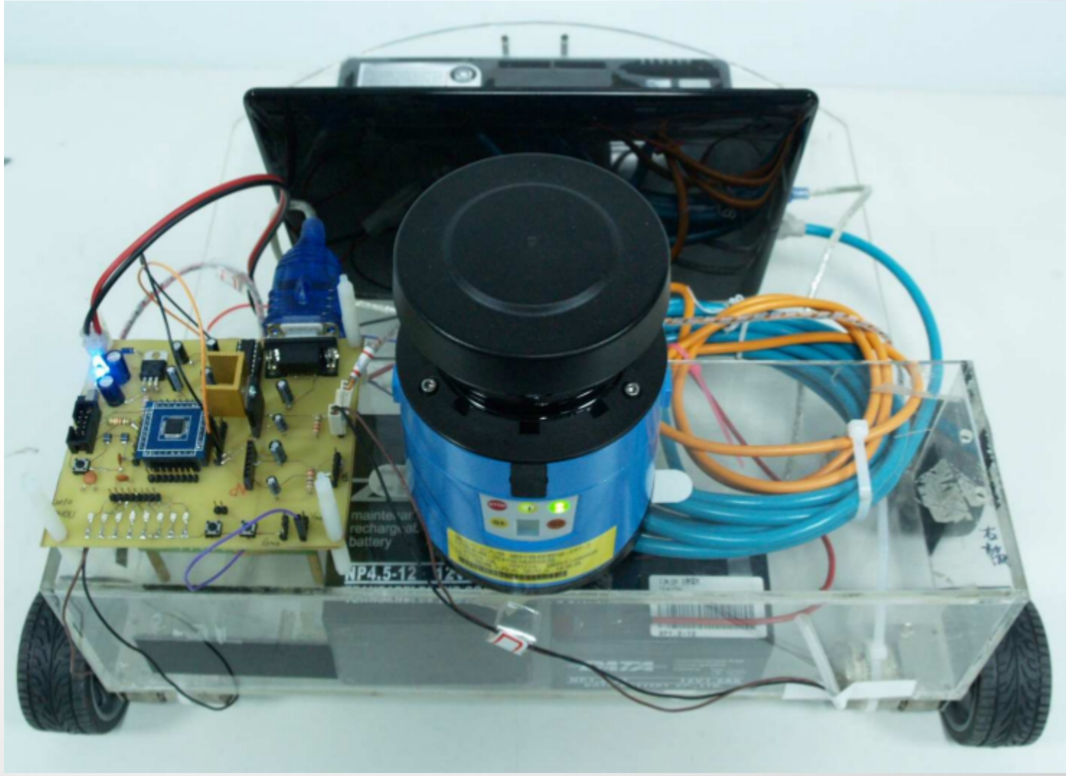
Mobile Robotics - Core Problems

1. Localization
    - Given a map and sensor data, where is the robot?
    - Dead Reckoning
        - Encoders, Gyroscopes, Accelerometers, etc
    - External Sensing
        - LiDar, Cameras, GPS, Sonar
    - How can we integrate these data sources?
2. Path (Global) Planning
    - Given a map, start loc., & goal loc. find a safe path
    - large maps have a huge search space
3. Trajectory (Local) Planning
    - Given a path and current location, produce motor commands

Mobile Robot Platform

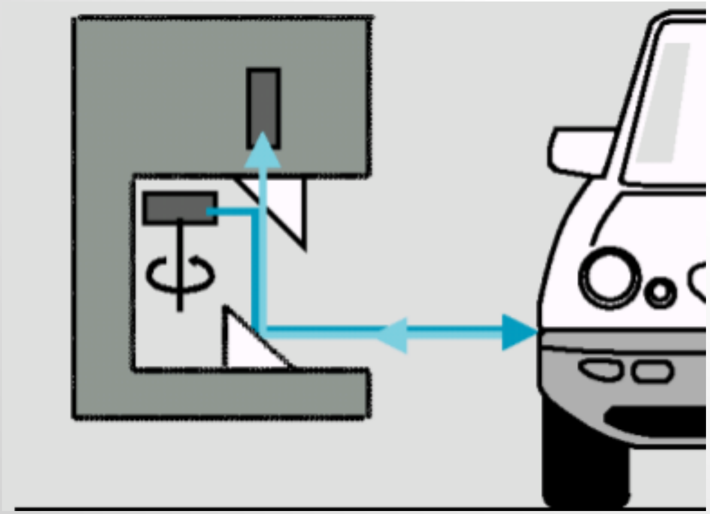
- Differential drive system
- On-board Netbook for computation
- Lidar Sensor

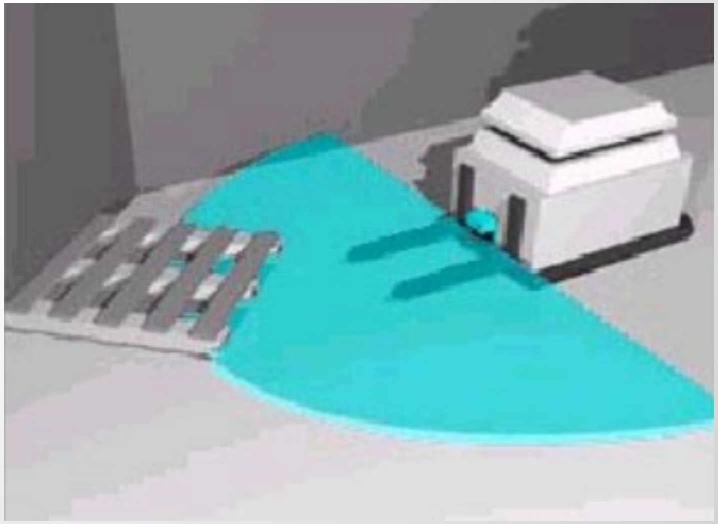
(Very simple!)



Lidar Basics

- Use a laser to measure distance to objects
  - Time of flight measurment
  - Sweep a wide area in fixed angular increments
- The resulting data is a set of points
  - Each point is an X,Y pair relative to the scanner where an object was detected

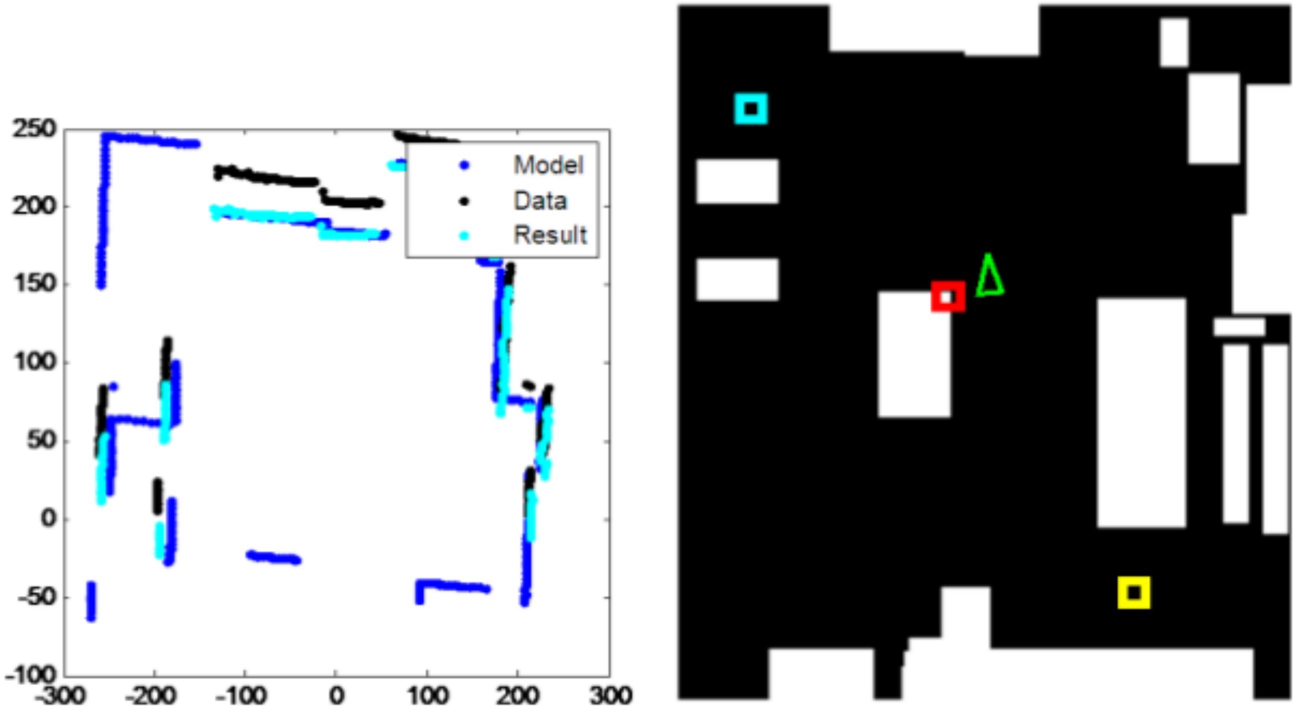




ICP Localization [2]

- ICP - Iterative Closest Point Algorithm
  - Requires a model of the environment (Map)
  - Receives laser sensor data (Data)
- Randomly place the Data around the Map
  - Empirically determine the required sample size
- Iteratively translate and rotate each sample
  - Apply a reasonably complex quartioneon based formula to each element of Data
  - Each iteration minimizes distance of each element in Data to the Map
  - Each sample will converge to a local minimum
- Select the best fitting sample

ICP Localization Results



(a) (b)

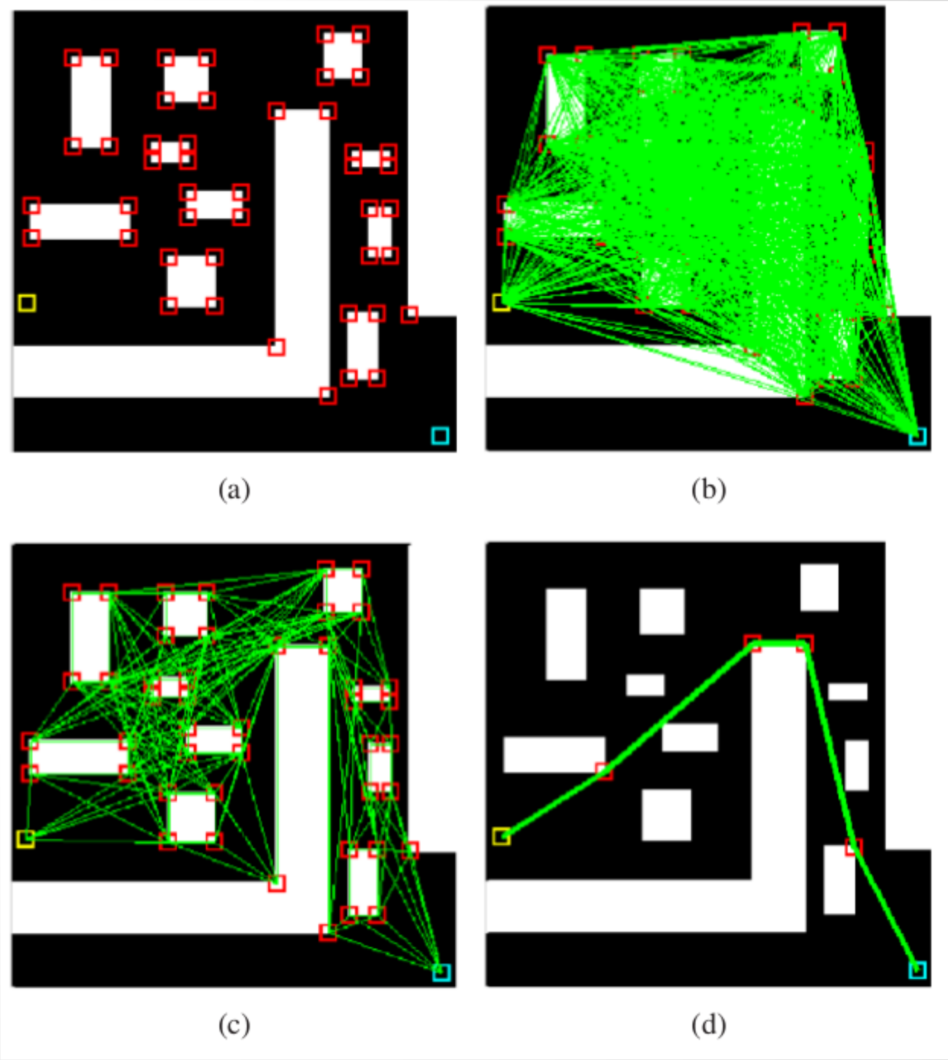
Fig. 3. Self-localization of the robot. (a) The results of ICP registration. (b) The pose of the robot.

Local ICP

- Once the robot is localized globally, save computation by setting initial guess based on previous Localization results and wheel encoder data
- Translate the Data by your best guess on robot motion since last ICP update
- Quickly converges and follows robot on Map

Path (Global) Planning

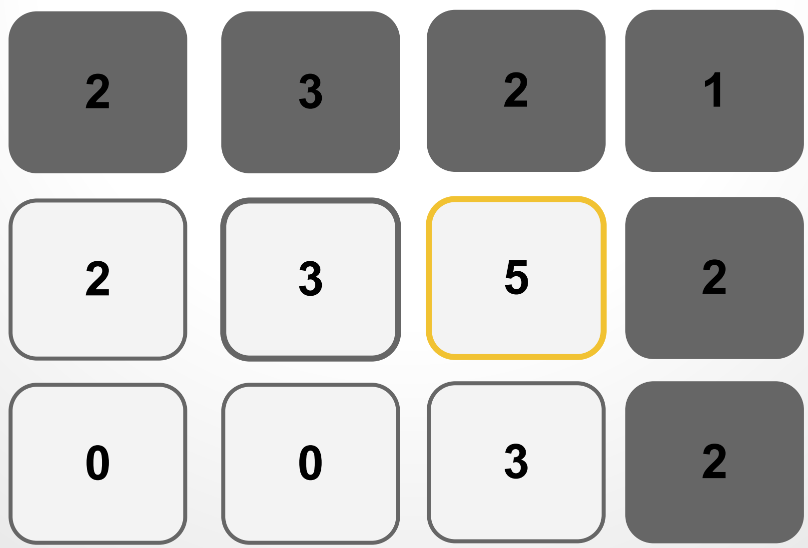
1. Corner Detection
   - Harris Corner Detection
   - (Graph Nodes)
2. Complete Graph
   - Every Possibility
3. Visibility Graph
   - Prune Search Space
4. Optimal Path
   - Global Plan


(a) (b) (c) (d)

Harris Corner Detection [3]

(Very) Generally Speaking:

1. Compare each pixel to it's neighbors
2. Sum the squares of the differences
3. All local maxima represent a corner



Visibility Graph

```
For each edge in CompleteGraph
    If ( edge_has_no_obstructions )
        VisibilityGraph += edge
```

Algorithm 1 VISIBILITY GRAPH

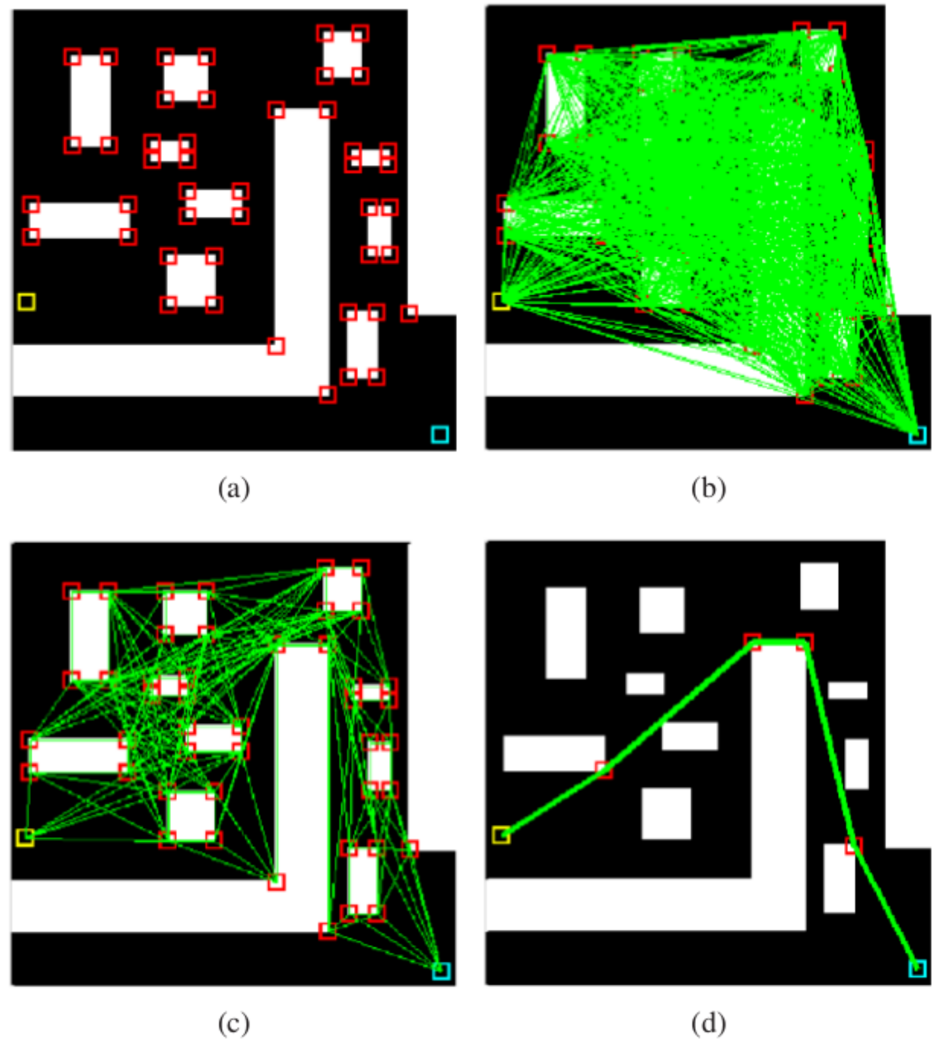
for all vertex $v_n \in V, n = 1, 2, ...m$ **do**
  if v_i has the ability to see a vertex $v_j, v_i \neq v_j$ **then**
    add the (v_i, v_j) to E
  end if
end for
return g

Dijkstra Algorithm

- Famous, and efficient, graph search
- Yes, A* would probably work better
  - Dijkstra doesn't include distance heuristic
- The visibility graph reduces the search space significantly, the search algorithm has little effect on performance in the author's examples.

[Dijkstra_Animation (Wikipedia)]

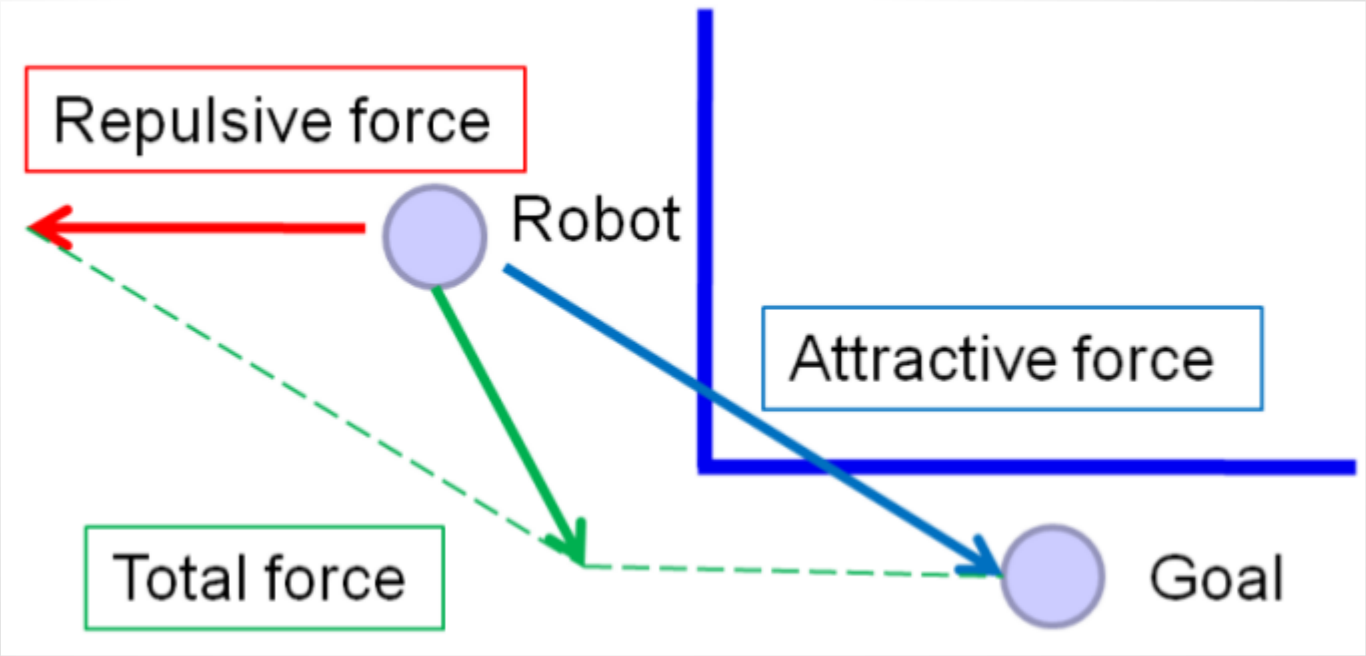
Global Planner Summary



(a) (b) (c) (d)

Trajectory (Local) Planner

- A *Repulsive* force pushes the robot away from obstacles
- An *Attractive* force pulls along the path



Local Planner

$$F_{att} = -k_{att}(x - x_{goal})$$

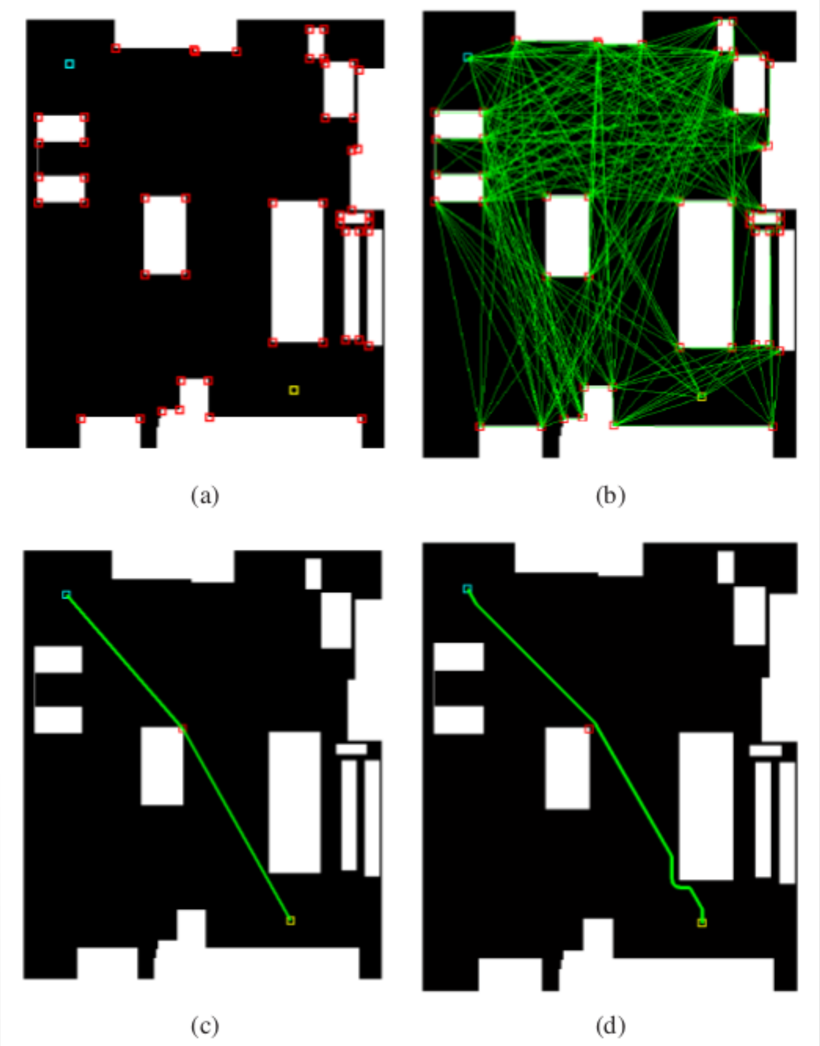
$$F_{rep} = \begin{cases} k_{rep}(\frac{1}{\rho} - \frac{1}{\rho_0})\frac{1}{\rho^2}, & \text{if } \rho \leq \rho_0 \\ 0, & \text{if } \rho > \rho_0 \end{cases}$$

- p : Distance to Obstacle
- p0: Safety Distance

- Many obstacles will produce many forces
- Because of the visibility graph, the robot will not get stuck in convex locations

Motivation for Repulsive forces

- Notice the final path looks good in (c)
- It actually comes too close to the corner of an object however
- The repulsive force maintains reachability



Room for Improvement

1. 3D Lidar
   a. The robot would be able to sense out-of-plane obstacles
2. Localization based on sensed maps
   a. Each map currently needs to be made by hand
   b. Dozens of SLAM techniques exist for this
3. Inflate obstacles by the radius of the robot
   a. The eliminates paths where the robot cannot travel
   b. May also reduce the number of graph nodes
   c. Decreases the chance of needing the repulsive force

Take-Away

- Localization is Hard!
  - Luckily, there are many pre-existing algorithms
- Lidar works really really well
  - But it's expensive, $5000+

- The visibility graph is an awesome tool!
  - This it the paper's primary contribution to robotics

- Separate the Global and Local planners
  - Significant reduction in complexity in both

References

[1] Jia-Heng Zhou; Huei-Yung Lin; , "A self-localization and path planning technique for mobile robot navigation," *Intelligent Control and Automation (WCICA), 2011 9th World Congress on* , vol., no., pp.694-699, 21-25 June 2011

[2] Besl, P.J.; McKay, H.D.; , "A method for registration of 3-D shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* , vol.14, no.2, pp.239-256, Feb 1992

[3] C. Harris and M. Stephens (1988). "A combined corner and edge detector". *Proceedings of the 4th Alvey Vision Conference*. pp. 147–151.