

---

# Path Quality

Spring 2012

Software Workshop:  
High-Quality Motion Paths for Robots (and Other Creatures)

---

Dan Halperin  
School of Computer Science  
Tel Aviv University

---

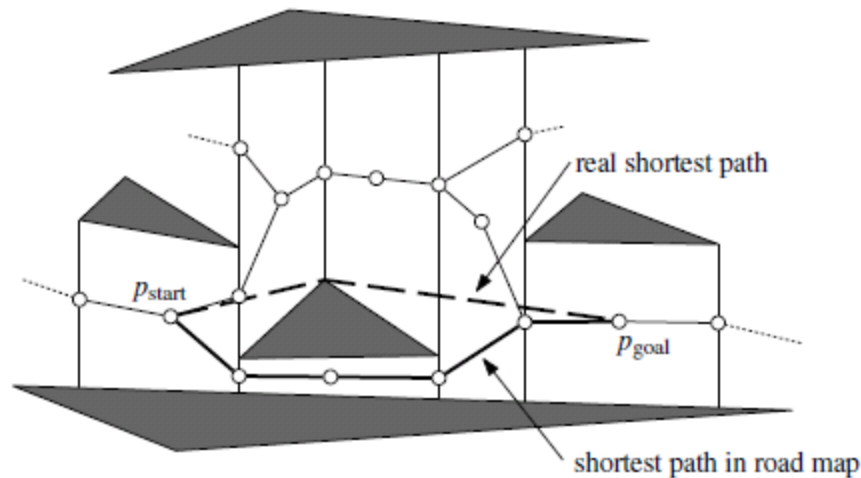
# Today's lesson

- shortest paths
- high clearance paths
- other quality measures
- combined quality criteria and corridor maps
- path quality in sampling-based planners



# Shortest paths among obstacles in the plane [from de Berg et al, Ch. 15]

- first attempt: Dijkstra on the connectivity graph of the trapezoidal map



---

# lesson

- does the graph on which we are searching for the best paths contain the best paths?

---

# properties of the shortest path

a polygonal line whose vertices are the start and goal configurations and vertices of the obstacles

# Computing a shortest path

**Algorithm** SHORTESTPATH( $S, p_{\text{start}}, p_{\text{goal}}$ )

*Input.* A set  $S$  of disjoint polygonal obstacles, and two points  $p_{\text{start}}$  and  $p_{\text{goal}}$  in the free space.

*Output.* The shortest collision-free path connecting  $p_{\text{start}}$  and  $p_{\text{goal}}$ .

1.  $\mathcal{G}_{\text{vis}} \leftarrow \text{VISIBILITYGRAPH}(S \cup \{p_{\text{start}}, p_{\text{goal}}\})$
2. Assign each arc  $(v, w)$  in  $\mathcal{G}_{\text{vis}}$  a weight, which is the Euclidean length of the segment  $\overline{vw}$ .
3. Use Dijkstra's algorithm to compute a shortest path between  $p_{\text{start}}$  and  $p_{\text{goal}}$  in  $\mathcal{G}_{\text{vis}}$ .

# Shortest paths in the plane, complexity

- the visibility-graph algorithm takes  $O(n^2 \log n)$  time where  $n$  is the number of obstacle vertices
- there are output sensitive algorithms (in the size of the visibility graph)
- near-optimal  $O(n \log n)$  algorithm by Hershberger and Suri
- the case of a simple polygon (whose complement is the obstacle) is much simpler



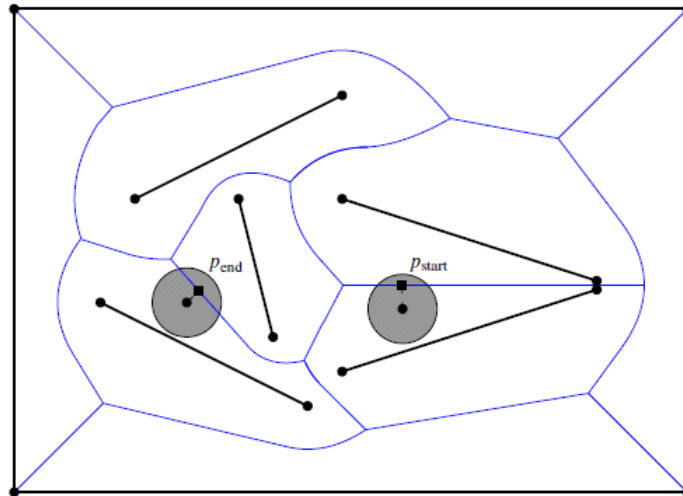
---

# Shortest paths among polyhedra in 3-space

- the setting: point robot moving among polyhedra with a total of  $n$  vertices
- the problem is NP-hard [Canny-Reif]
  - algebraic complexity
  - combinatorial complexity

# High clearance paths

- Voronoi diagrams/the medial axis
- the Voronoi diagram of line segments, and the retraction method for a disc [O'Dunlaing-Yap]



- video [Schirra/Rohnert]

---

# Other quality measures

- other  $L_p$  metrics, e.g., Manhattan ( $L_1$ )
- link number
- number of reverse movements
- low energy
- weighted regions
- many more
  
- multiple criterion optimal paths

---

# Combined quality criteria and corridor maps

- a path is called **Pareto optimal** if no other path has a better value for one criterion without having a worse value for another criterion
- multiple criterion optimization is often hard

---

# Clearance-length combination

- combined measure
- relaxed combination: the visibility Voronoi complex
- corridor maps
- corridor trees in the complement of molecules

# Optimizing a combined measure

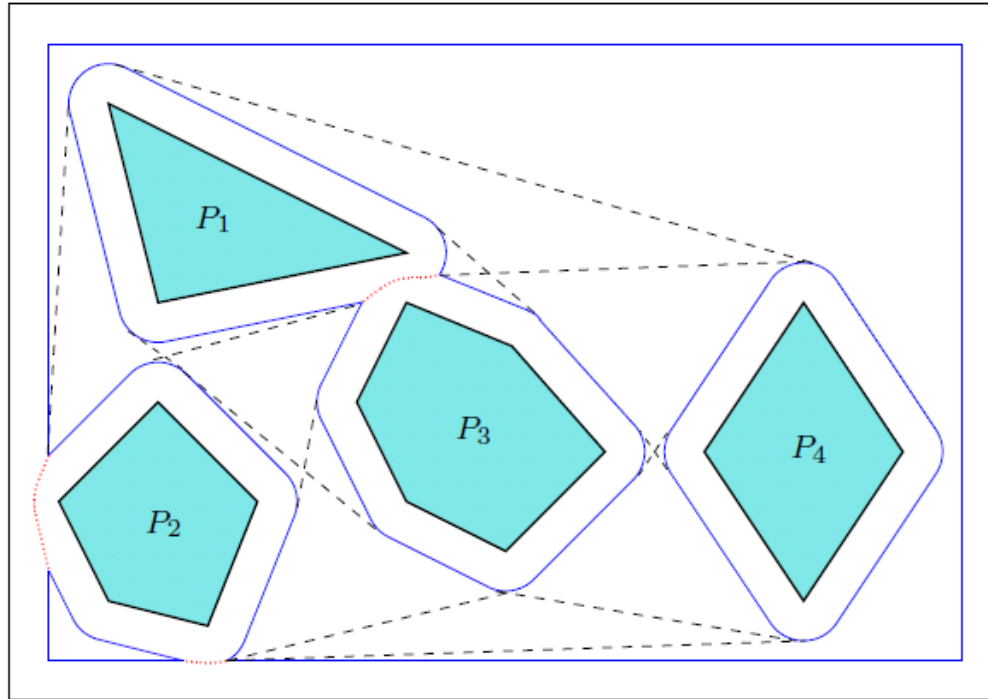
[Wein-van den Berg-H]

$$L^*(C) = \int_{\gamma} \left( \frac{w_{\max}}{w(t)} \right)^{d-1} dt$$

- examples:
  - the optimal path in the presence of a point obstacle is a logarithmic spiral
  - the optimal path in the presence of a segment obstacle is a circular arc
- approximation algorithms for the general polygonal case

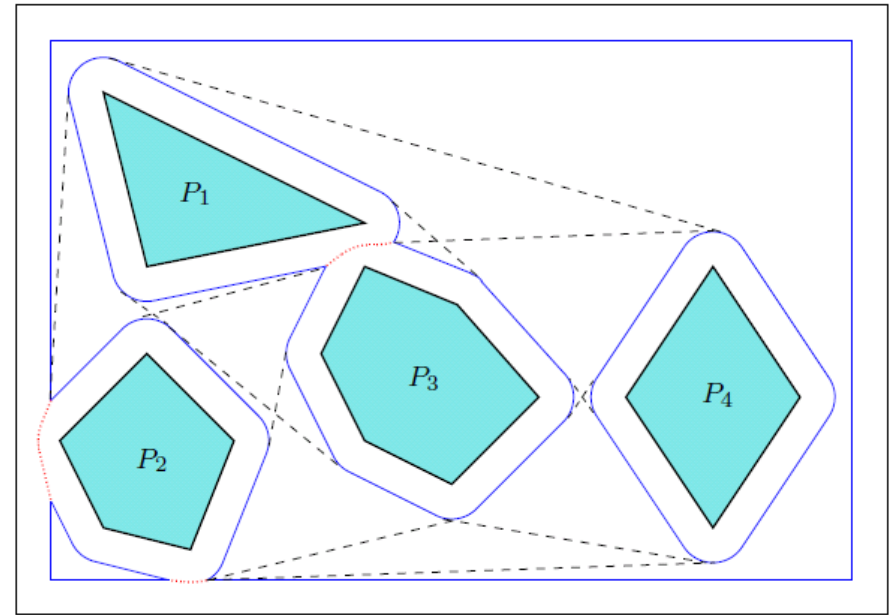
# The visibility Voronoi diagram (VVD)

[Wein-van den Berg-H]



- finding the shortest path with a given clearance  $c$ , while still allowing to make significant shortcuts with lesser clearance on the Voronoi diagram

# The visibility Voronoi complex

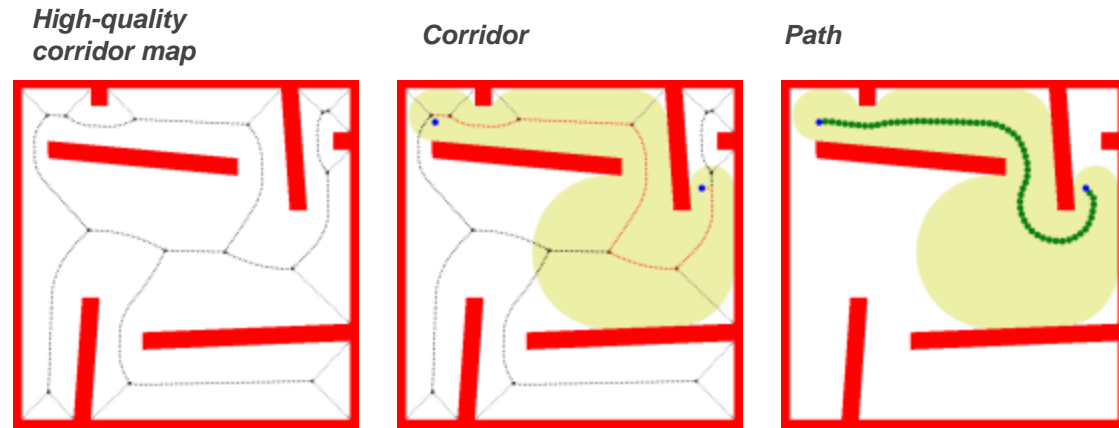


- implicitly encodes the VVD for any clearance  $c$
- interpolates between the visibility diagram ( $c=0$ ) and the Voronoi diagram ( $c=\text{infinity}$ )
- $O(n^2 \log n)$  construction time



# Corridor maps

[Geraerts-Overmars]



- motivated by motion planning in games
- similar to VVD/VVC, augmenting the VD with clearance information
- instead of providing a single solution path, provides a **corridor** among static obstacles, where later one can easily maneuver among dynamic obstacles

---

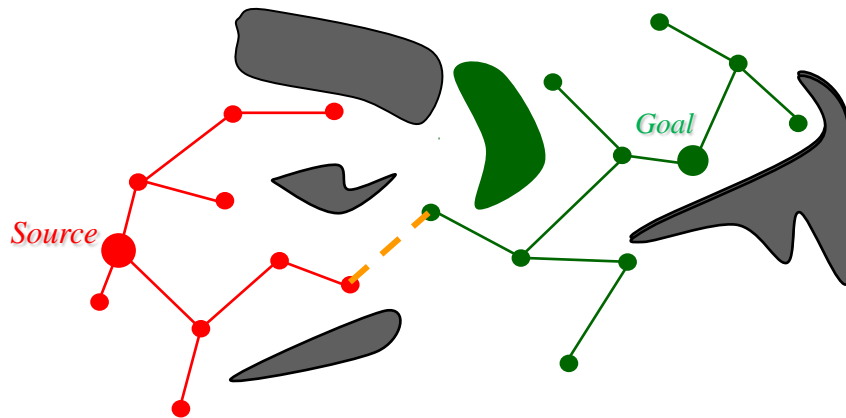
# Path quality in sampling-based planners

- the typical process: building a roadmap graph and running Dijkstra or similar
- recall our earlier test: **does the graph on which we are searching for the best paths contain the best paths?**
- path quality can be very low
- example: path length in BiRRT

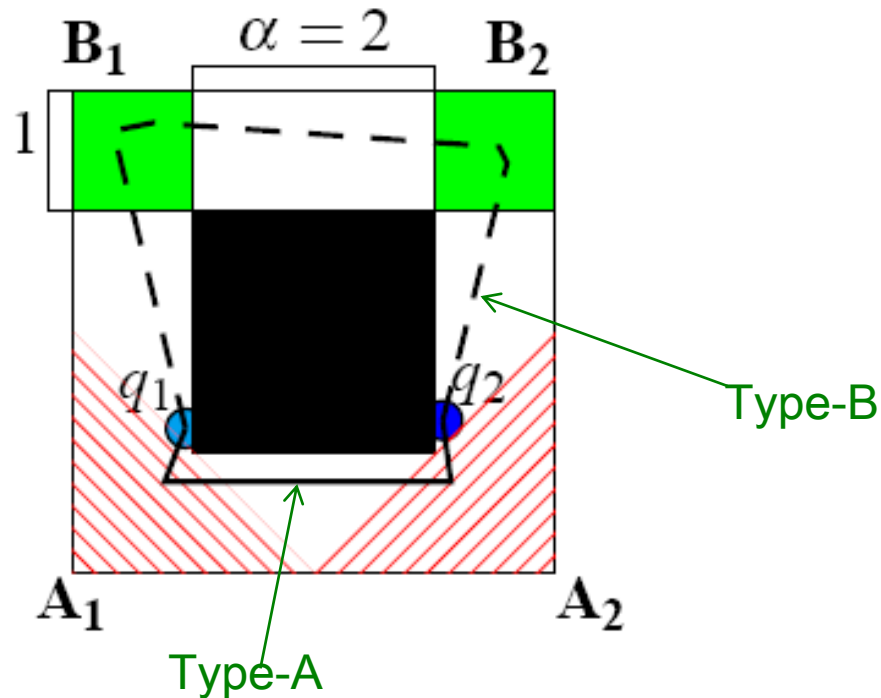
# Bi-RRT reminder: Growing two-trees

[Kuffner and LaValle '00]

- maintain two trees rooted at **source** & **goal**
- construction step –  
sample configurations and expand either tree as in RRT
- merging step –  
connect configurations from both trees

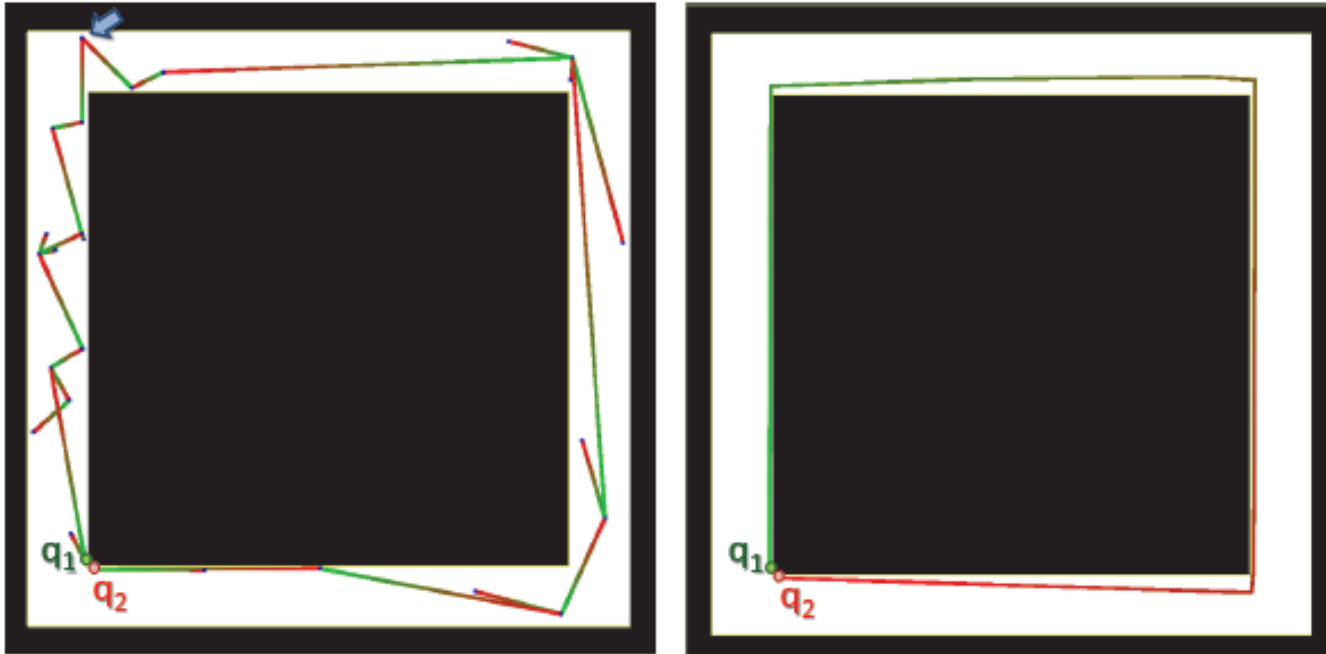


# Example (I) – in OOPSMP



- ❑ 49.4% of paths are over three times worse than optimal (even after smoothing)
- ❑ much larger than the theoretical bound

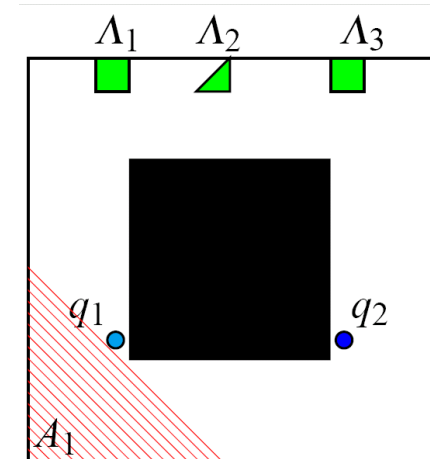
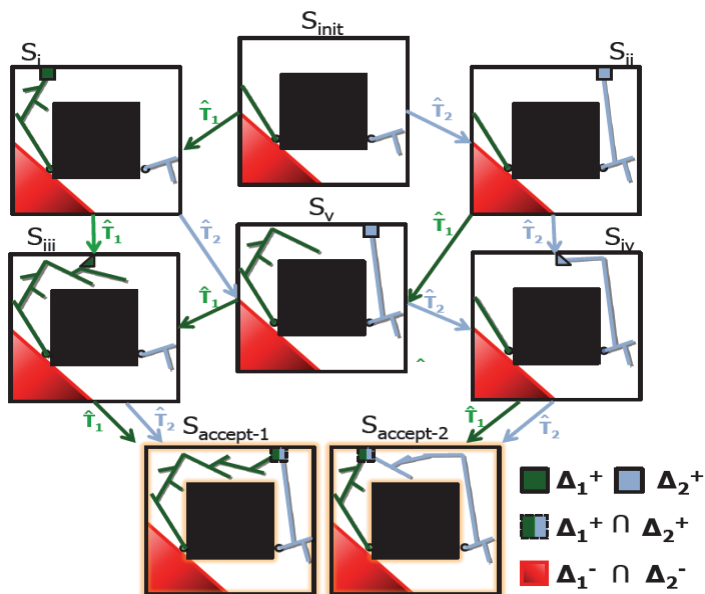
## Example (II) – close-by start and goal configurations



- ❑ 5.9% of paths are over 140 times worse than optimal (even after smoothing)
- ❑ importance of *visibility blocking* – narrow passages not the only king  
(theoretical motivation for Visibility PRM, Laumond *et al.* '00)

# How low can path quality get?

Sampling-Diagram Automata:  
Analysis of path quality in tree planners  
[Nechushtan-Raveh-Halperin, WAFR 2010]

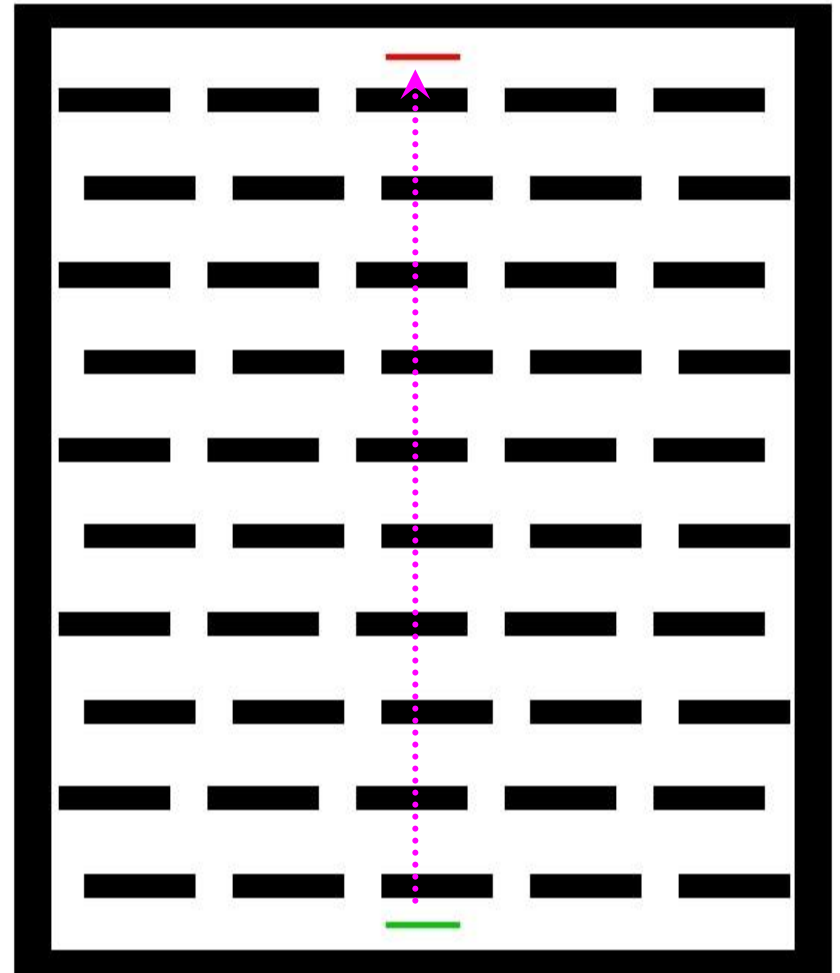


# Improving path quality in sampling-based motion planning, sample work

- Short-cutting heuristics (“path smoothing”)
  - Retraction towards medial axis  
[e.g., Wilmarth et al. '99, Geraerts and Overmars '07]
  - Useful Cycles in PRM [Nieuwenhuisen and Overmars '04]
  - Biasing tree growth by a cost-function  
[e.g., Urmson and Simmons '03, Ettlín and Bleuler '06, Jaillet et al. '08, Raveh et al. '09]
- 
- RRT\* - a modification of RRT [**Karaman and Frazzoli '10**] (for more variants, see paper)
    - the modified RRT\* algorithm converges to an optimal path as running time reaches infinity
    - “Standard”-RRT misses the (precise) optimal path with probability one  
**Still, might be  $\epsilon$ -good, or within same homotopy class as optimal path**

# Improving quality by path hybridization

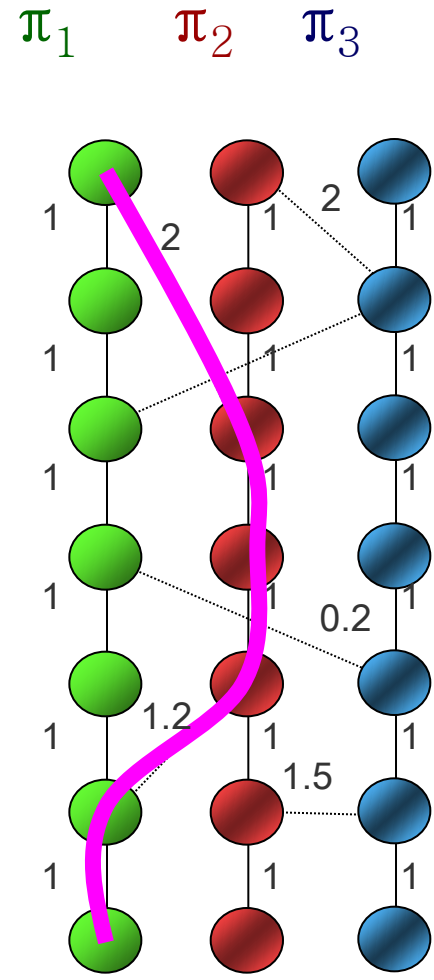
**example:** move the rod from the bottom to the top of a 2D grid  
(*rotation + translation*)



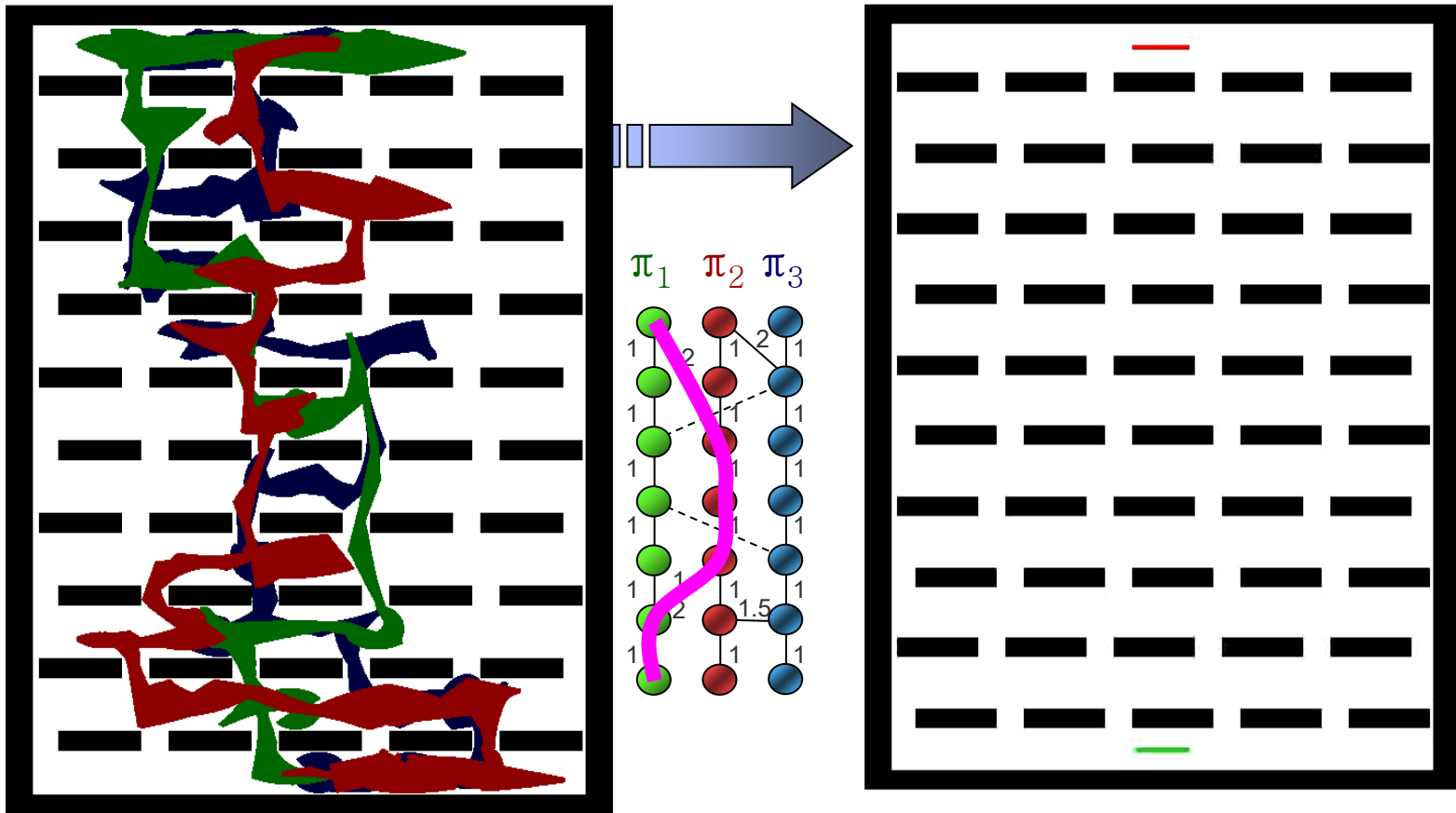




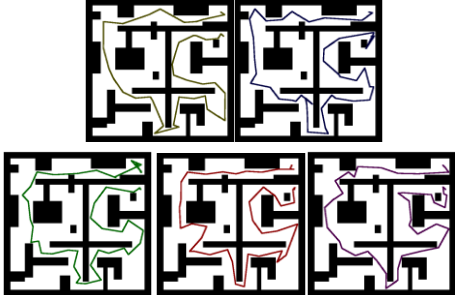
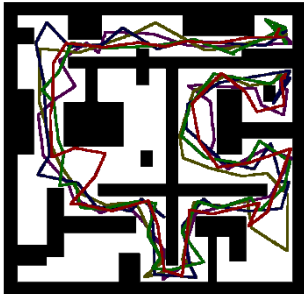
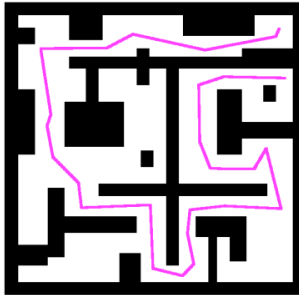
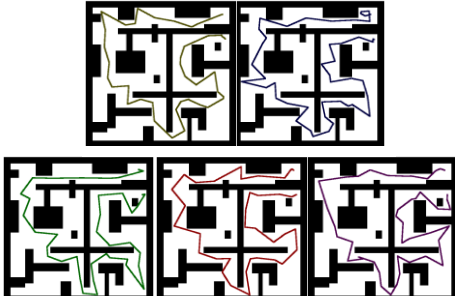
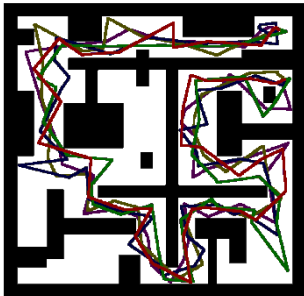
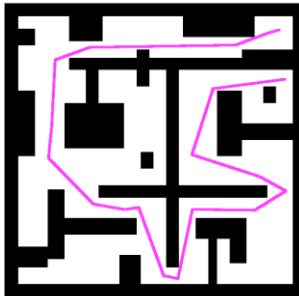
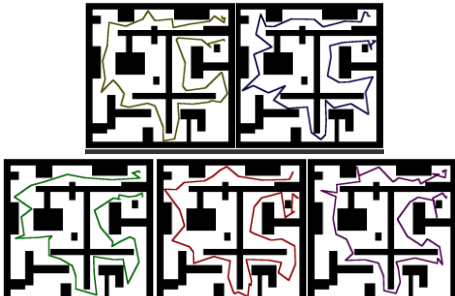
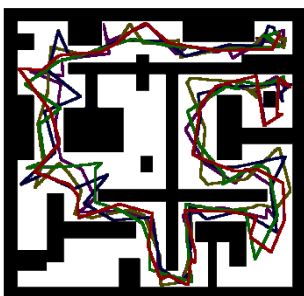

# H-Graphs: Hybridizing multiple motion paths ( = looking for shortcuts)



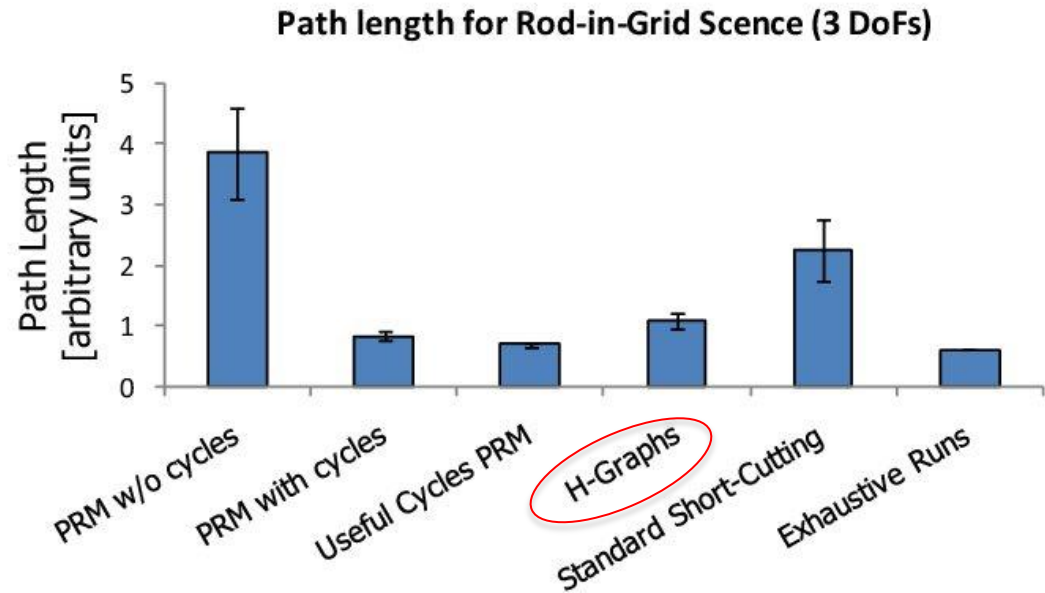
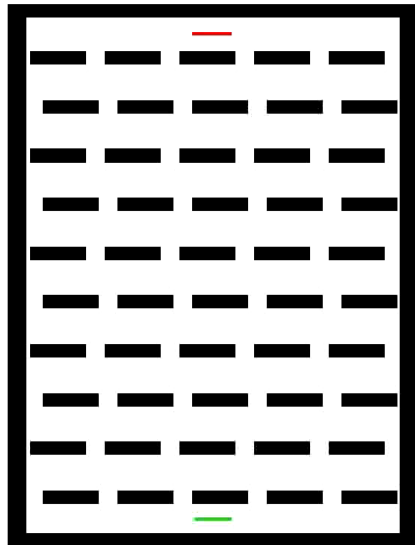
# Hybridizing the paths



# General quality criteria

| Quality Measure   | The Input Paths   | H-Graph   | Output Path   |
|---|---|---|---|
| <i>Clearance and length<br/>(emphasis on clearance)</i> |    |    |    |
| <i>Clearance and length<br/>(emphasis on length)</i>    |   |   |   |
| <i>Path length</i>                                      |  |  |  |

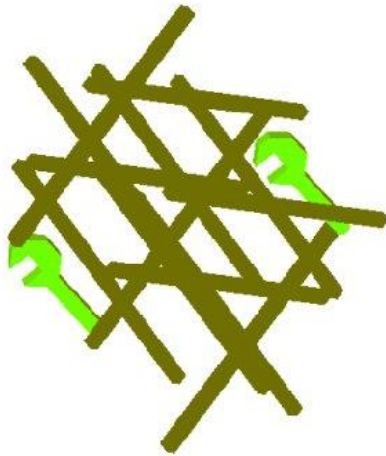
# Rod-in-Grid scene: 3 dof



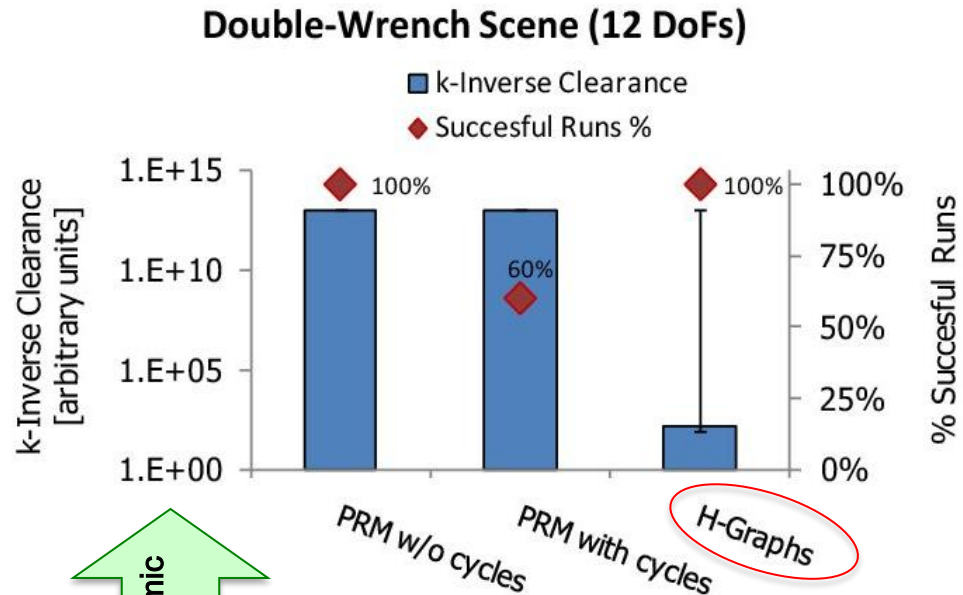
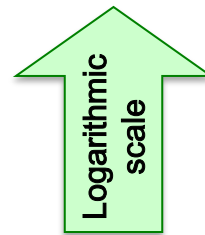
Implemented in the **OOPSMP** package (Plaku, Moll and Kavraki), collision detection – **PQP** (Lin and Manocha)

# Double-Wrench: 12 dof

Switching the two wrenches (rotation + translation x 2)



long runs of PRM  
same time as total time of  
HGraphs

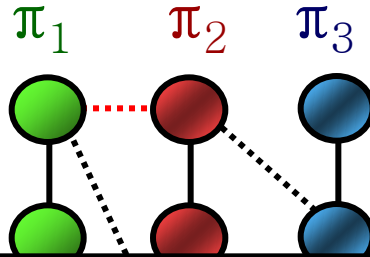


H-Graphs become particularly useful for high-dimensional problems (at least in this example)

# Running-time bottleneck for hybridization:

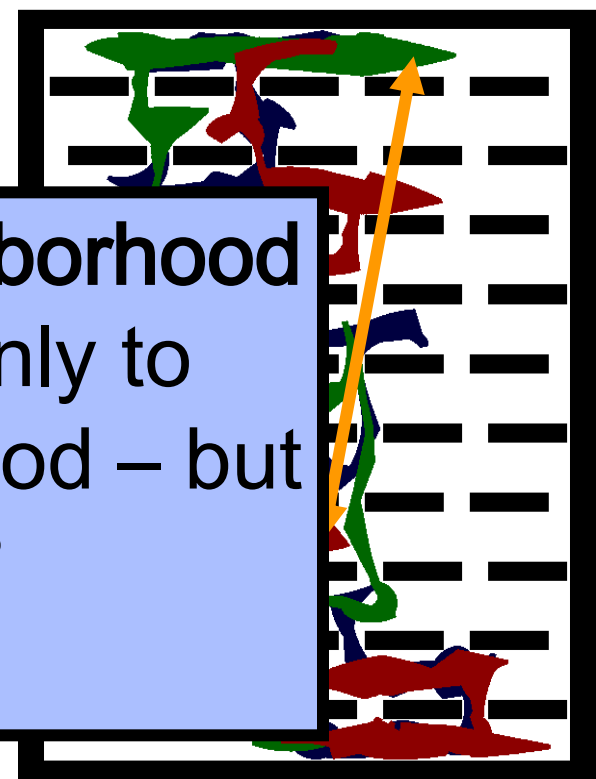
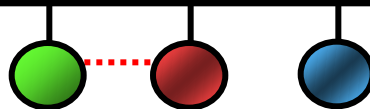
Trying to connect nodes from different paths

in a naïve implementation:  
 $O(n^2)$  potential edges need  
to be tested



we assume  
of paths

**Simple Heuristic – “Neighborhood H-Graphs”:** compare only to nodes in local neighborhood – but can we do better?



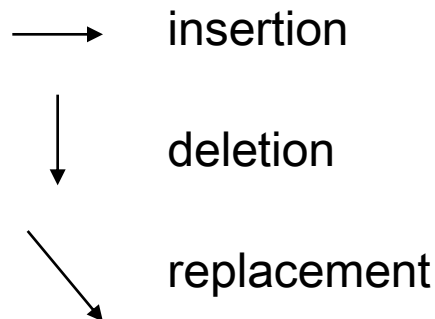
# Edit-distance string matching

## → Linear alignment of motion paths

Comparing “This dog” and “That Dodge” with insertion / deletions / replacement:

THI - S DO - G -  
THAT - DODGE

dynamic-programming algorithm:

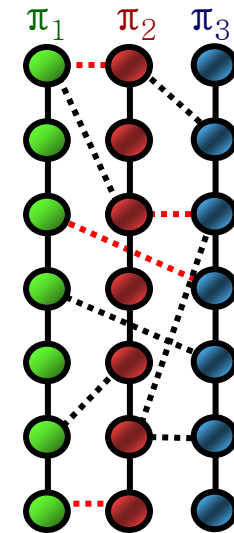
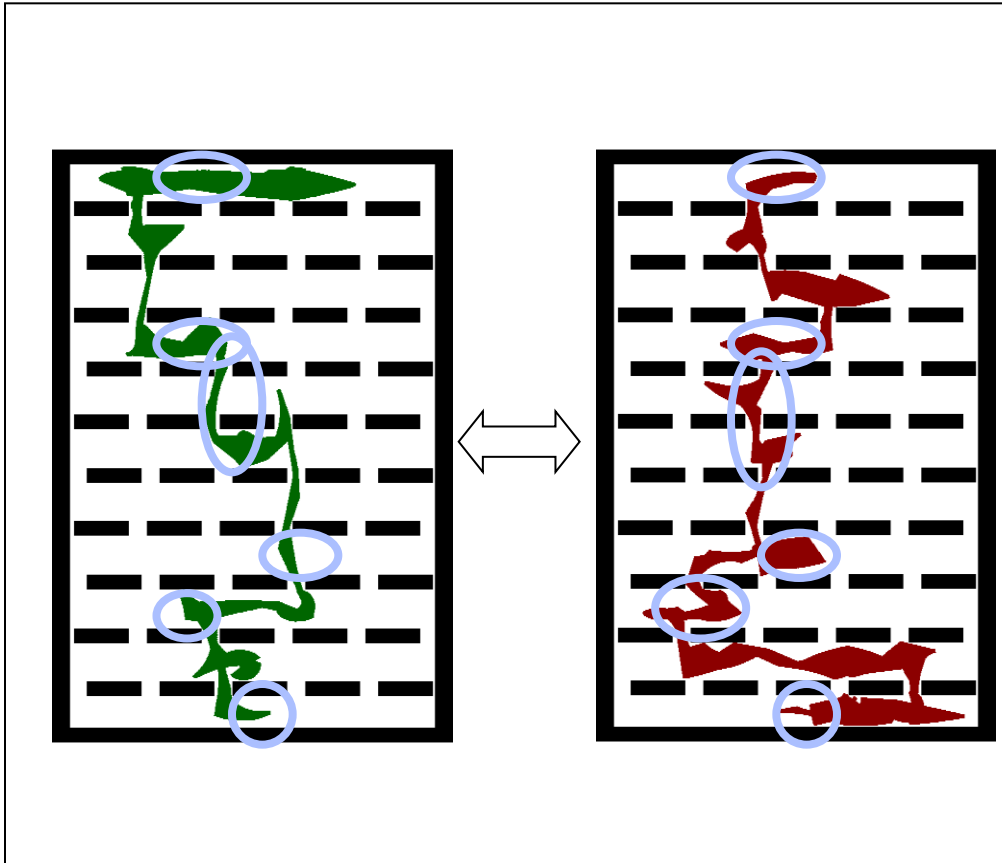


|          |   | <i>t</i> | <i>h</i> | <i>i</i> | <i>s</i> |  |
|----------|---|----------|----------|----------|----------|--|
|          | 0   | → 1      | → 2      | → 3      | → 4      |  |
| <i>h</i> | ↓ ↘ <sup>1</sup> ↓ ↘ <sup>0</sup> ↓ ↘ <sup>1</sup> ↓ ↘ <sup>1</sup> ↓ |          |          |          |          |  |
| <i>a</i> | ↓ ↘ <sup>1</sup> ↓ ↘ <sup>1</sup> ↓ ↘ <sup>1</sup> ↓ ↘ <sup>1</sup> ↓ | 1        | 1        | 2        | 3        |  |
| <i>s</i> | ↓ ↘ <sup>1</sup> ↓ ↘ <sup>1</sup> ↓ ↘ <sup>1</sup> ↓ ↘ <sup>0</sup> ↓ | 2        | 2        | 2        | 3        |  |
|          | 3   | → 3      | → 3      | → 3      | → 2      |  |



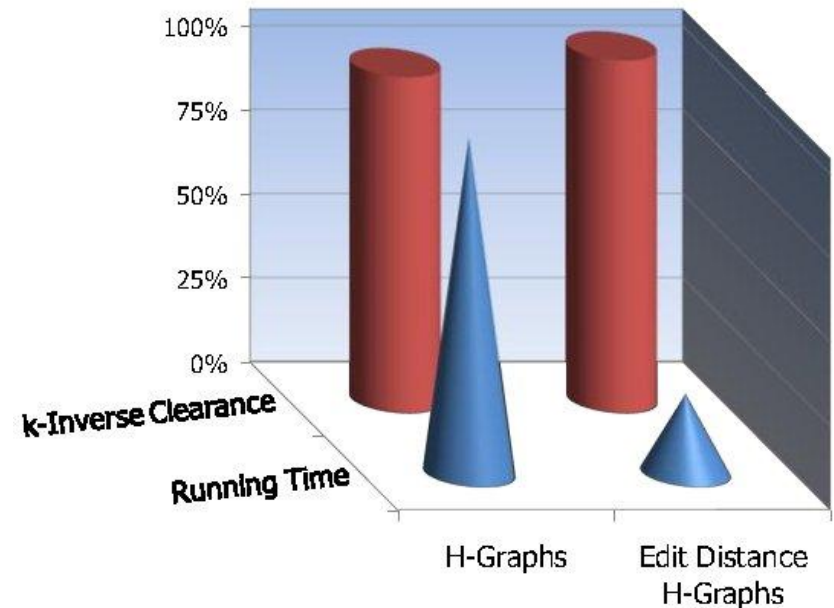
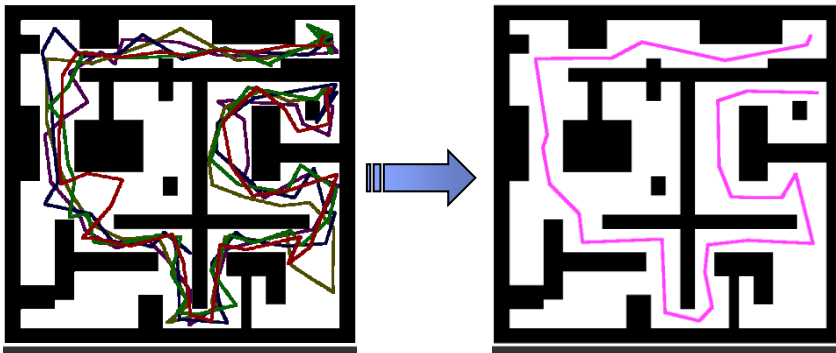
# Alignment length is linear

Now testing only  $O(n)$  edges along the alignment



# Comparison of running times

- hybridizing five motion paths in a 2-D maze:
  - from 3.52 seconds to 0.83 seconds on average (75% decrease), with comparable path quality



# IMPROVING THE QUALITY OF NON-HOLONOMIC MOTION BY HYBRIDIZING C-PRM PATHS

ITAMAR BERGER | BOSMAT ELDAR | GAL ZOHAR | BARAK RAVEH | DAN HALPERIN

School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel

## INTRODUCTION

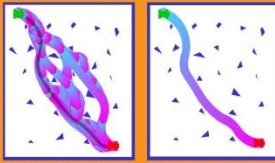
Sampling-based motion planners are an effective means for generating collision-free motion paths. However, the quality of these motion paths, with respect to different quality measures such as path length, clearance, smoothness or energy, is often notoriously low. This problem is accentuated in the case of non-holonomic sampling-based motion planning, in which the space of feasible motion trajectories is restricted. In this study, we combine the C-PRM algorithm by Song and Amato with our recently introduced path-hybridization approach (H-Graphs), for creating high quality non-holonomic motion paths, with combinations of several different quality measures such as path length, smoothness or clearance, as well as the number of reverse car motions.



## H-GRAPHS

We have recently introduced the path-hybridization approach [2, 3], in which an arbitrary number of input motion paths are hybridized to an output path of superior quality, for a range of path-quality criteria. The approach is based on the observation that the quality of certain sub-paths within each solution may be higher than the quality of the entire path. Specifically, we run an arbitrary motion planner  $k$  times (typically  $k=5-6$ ), resulting in  $k$  intermediate solution paths to the motion planning query. From the union of all the edges and vertices in the intermediate paths we create a single weighted graph, with edge weights set according to the desired quality criterion.

We then try to merge the intermediate paths into a single high-quality path, by connecting nodes from different paths with the local planner, and giving the appropriate weights to the new edges. Dijkstra's algorithm is used to find the highest-quality path in the resulting Hybridization-Graph (H-Graph).

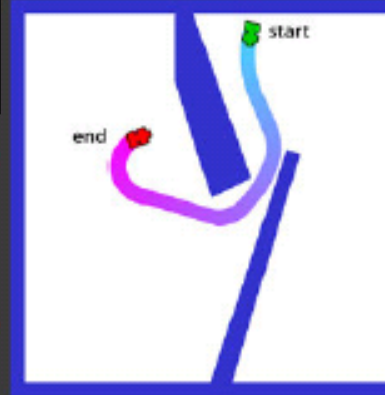
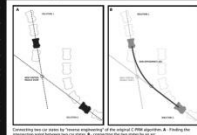


Experimental results of running C-PRM with Path Hybridization. The left panel shows an overlay of six different motion paths that were generated by the original C-PRM algorithm. The right panel shows the hybridization of these paths by the path hybridization algorithm to obtain a shorter path (as described in the text).

## C-PRM WITH PATH HYBRIDIZATION

While the path hybridization approach has been successfully tested over a range of holonomic motion planning problems with many degrees of freedom, its application to non-holonomic motion planning is not trivial.

In particular, whereas it is easy to connect two nearby configurations in the case of holonomic motion, it is in general impossible to linearly interpolate between two states of non-holonomic motion planning, due to the restriction on the set of possible paths. However, we observed that we can simply reverse-engage the original approach



C-PRM + H-GRAPHS

holonomic problems.

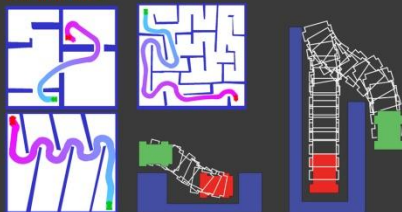


C-PRM



RRT

## EXAMPLES



## IMPLEMENTATION

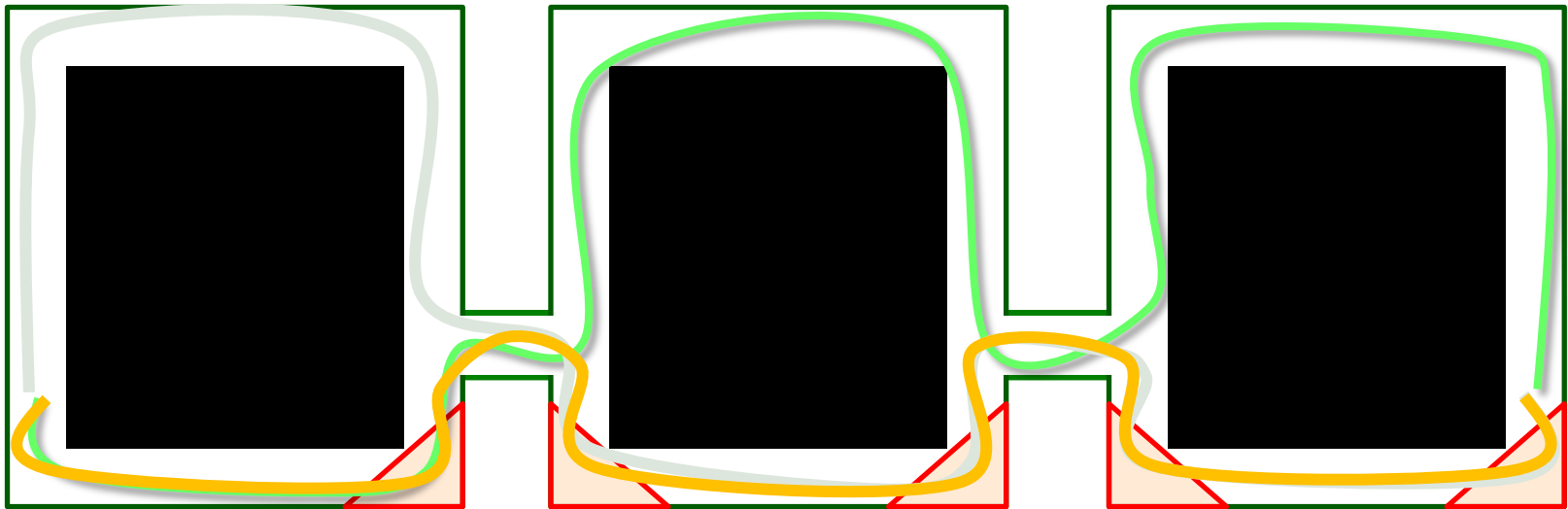
We have implemented the C-PRM algorithm and C-PRM with path hybridization within the framework of the QOOPSMP motion planning package. Our implementation supports the combination of a wide range of path quality criteria (length, smoothness, clearance, number of reverse car motions).

## REFERENCES

- [1] G. Song and N. M. Amato, "Randomized motion planning for car-like robots with C-PRM", in IEEE Int. Conf. on Intelligent Robots and Systems, 2001, pp. 37-42.
- [2] B. Raveh, A. Enoosh, and D. Halperin, "A little more, a lot better: Improving path quality by a simple path merging algorithm", ArXiv e-prints, vol. abs/1001.2391, 2010.
- [3] A. Enoosh, B. Raveh, O. Furman-Schueler, D. Halperin, and N. Ben-Tal, "Generation, comparison and merging of pathways between protein conformations: Gating in k-channels", Biophysical Journal, vol. 95, no. 8, pp. 3850-3860, 2008.
- [4] E. Plaku, K. E. Bekris, and L. E. Kavraki, "OOPS for motion planning: An online open-source programming system," in ICRA 07, 2007, pp. 3711-3716.

applied to car-like motion with various quality criteria: length, smoothness, clearance, number of reverse vehicle motions

# Why do Hgraphs work?



- wrong decision can be taken at every step
- can be solved by **path-hybridization**

# References

- *Shortest Path and Networks*, J.S.B. Mitchell, Chapter 27 of the Handbook on DCG, Goodman-O'Rourke (eds)
- Visibility Graphs, Chapter 15 of the Computational Geometry book by de Berg et al
- more details, more experiments:
  - <http://acg.cs.tau.ac.il/projects>
  - *A Little More, A Lot Better: Improving Path Quality by a Path-Merging Algorithm* [Raveh-Enosh-Halperin] IEEE Trans. on Robotics, 2011

---

THE END