Algorithmic Robotics and Motion Planning

Path quality

Dan Halperin
School of Computer Science
Tel Aviv University

Fall 2019-2020

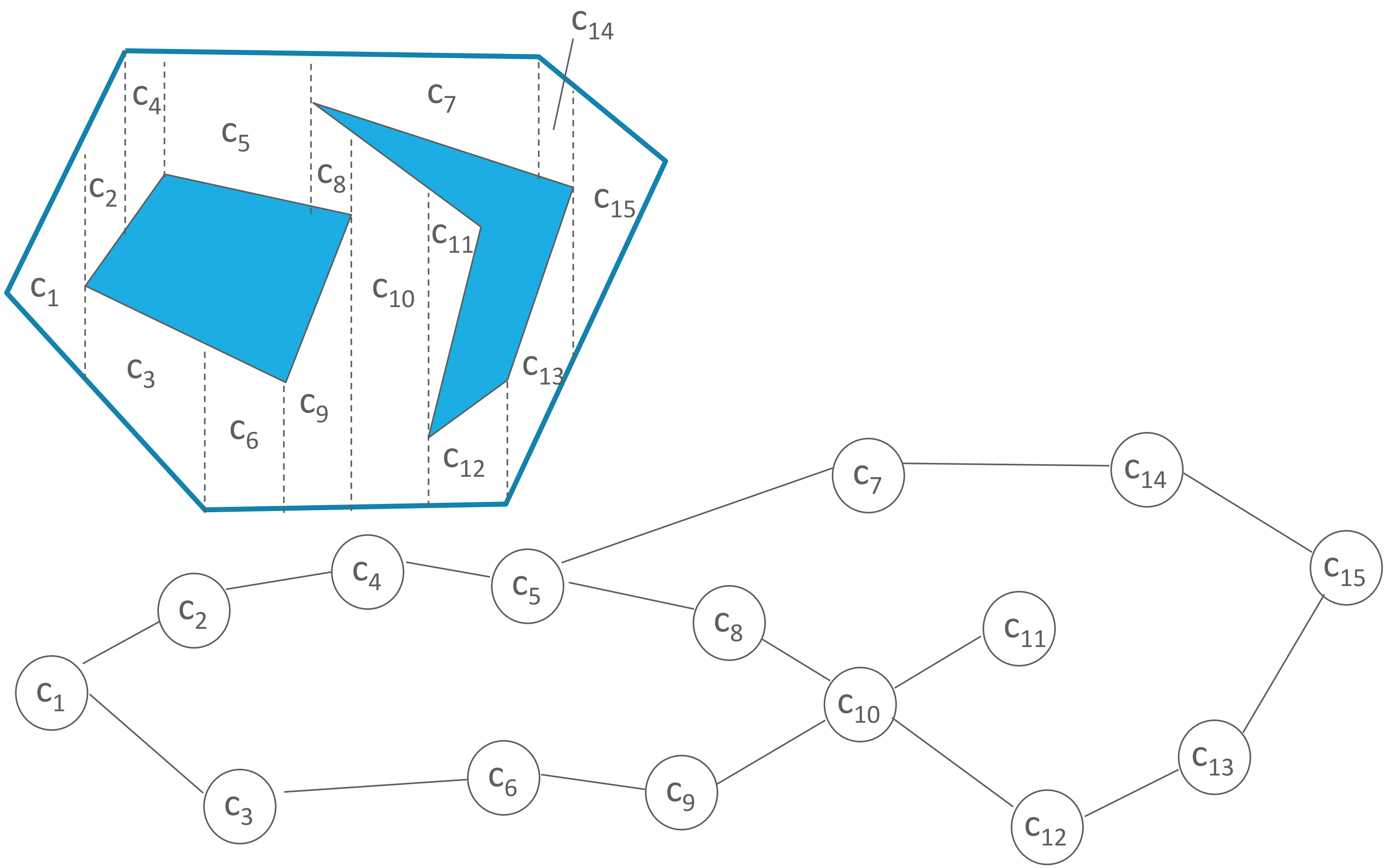
Overview

- basic quality measures
- multi-objective optimization and corridor maps
- path hybridization
- more on optimality of SB planners

Notice: abbreviated bib references, like [CR87], refer to the bibliography in the chapter *Algorithmic Motion Planning* by Halperin, Salzman and Sharir, CRC 2018

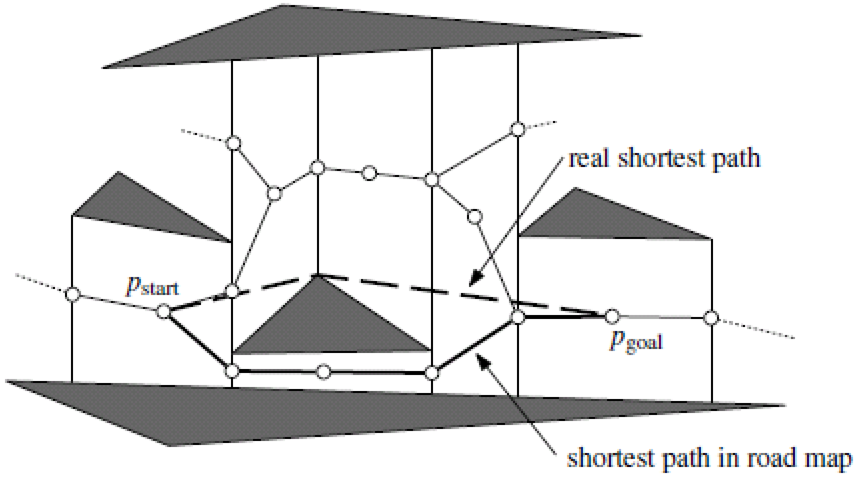
Basic quality measures

Shortest paths among obstacles in the plane



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

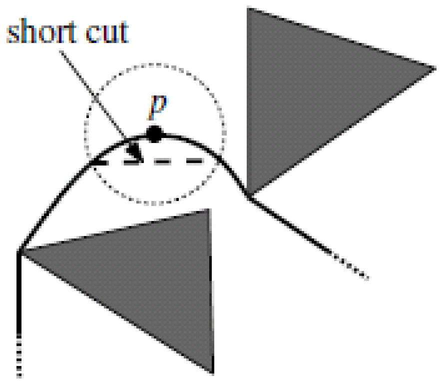


Important test

- does the graph on which we are searching for the best paths contain the (almost) 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_{start} and p_{goal} in the free space.
Output. The shortest collision-free path connecting p_{start} and p_{goal}.
1. $\mathcal{G}_{\text{vis}} \leftarrow$ 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_{start} and p_{goal} in $\mathcal{G}_{\text{vis}}$.

Computing the visibility graph

Algorithm VISIBILITYGRAPH(S)
Input. A set S of disjoint polygonal obstacles.
Output. The visibility graph $\mathcal{G}_{vis}(S)$.

1. Initialize a graph $\mathcal{G} = (V, E)$ where V is the set of all vertices of the polygons in S and $E = \emptyset$.
2. **for** all vertices $v \in V$
3.   **do** $W \leftarrow$ VISIBLEVERTICES(v, S)
4.   For every vertex $w \in W$, add the arc (v, w) to E.
5. **return** $\mathcal{G}$

Computing shortest paths in the plane, complexity

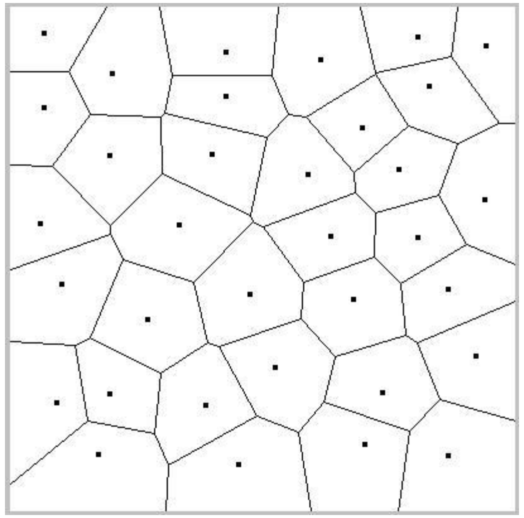
- the visibility-graph algorithm takes $O(n^2 \log n)$ time where n is the number of obstacle vertices
- there are output sensitive algorithms (in the size of the visibility graph)
- near-optimal $O(n \log n)$ algorithm by [HS99]
- the case of a simple polygon (whose complement is the obstacle) is much simpler, $O(n)$

Shortest paths among polyhedra in 3-space

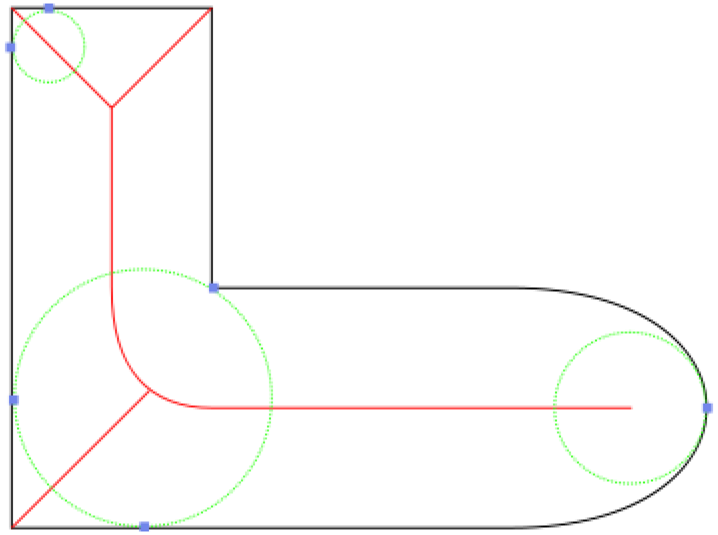
- the setting: point robot moving among polyhedra with a total of n vertices
- the problem is NP-hard [CR87]
  - algebraic complexity
  - combinatorial complexity

High clearance paths

- Voronoi diagrams/the medial axis



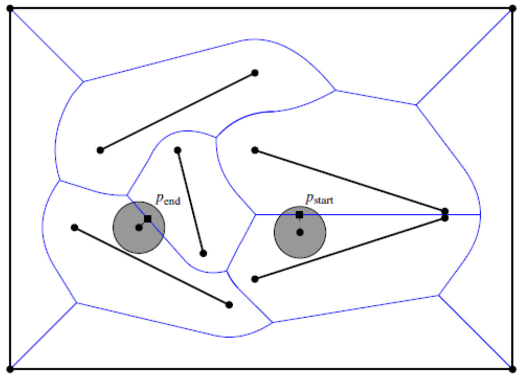
[www.cs.wustl.edu]



[commons.wikimedia.org]

High clearance paths, cont’d

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



- short paths along the diagram

[Rohnert-Schirra]

- Voronoi diagrams in higher dimensions are non-trivial to compute in practice but various approximations exist

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 objective optimal paths

Multi-objective optimization

Corridor maps

Multi-objective optimization

- scalarization
  - linear
  - non linear
- Pareto optimal solutions

  A solution (path) is called Pareto optimal if no other solution (path) has a better value for one criterion without having a worse value for another criterion

Clearance-length combination

- non-linear scalariztion and corridors
- Pareto optimal solutions (partial set): the visibility Voronoi complex
- corridor maps

Optimizing a combined measure
non-linear scalarization [WBH08]

- Weighing length and clearance

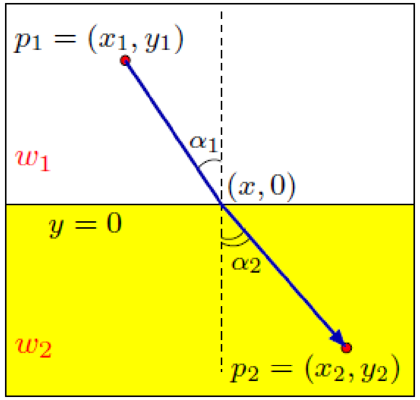
$$L_\delta^*(\gamma) = \int_\gamma \left(\frac{1}{c(\gamma(t))} \right)^\delta dt$$

- In the plane we let $\delta = 1$, then we integrate over the inverse of the clearance

- examples:
  - the optimal path in the presence of a point obstacle

Optimal path for a point robot in the presence of a point obstacle for the combined measure

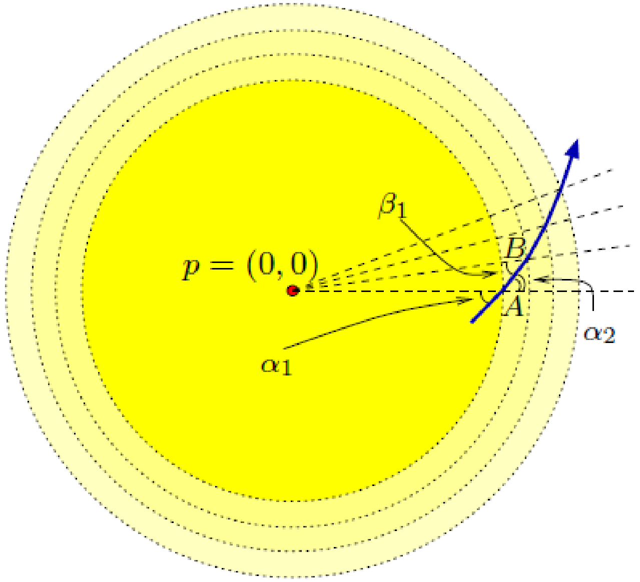
- input: s starting point, g goal
- Lemma 1: the optimal path is smooth
- Lemma 2: obeys Snell's law of refraction



$$w_2 \sin \alpha_1 = w_1 \sin \alpha_2$$

Optimal path for … combined measure, cont'd

- assume the obstacle is in the origin



$$\sin \alpha_2 = \frac{r_2}{r_1} \sin \alpha_1 .$$

$$\frac{r_2}{\sin(\pi - \alpha_2)} = \frac{r_1}{\sin \beta_1} ,$$

$$\sin \beta_1 = \frac{r_1}{r_2} \sin(\pi - \alpha_2) = \frac{r_1}{r_2} \sin \alpha_2 = \sin \alpha_1 .$$

- the angle between the line from the origin to the point ϒ(t) on the curve and the tangent at ϒ(t) is fixed: ϒ(t) is a logarithmic spiral

Logarithmic spiral

- first described by Descartes,

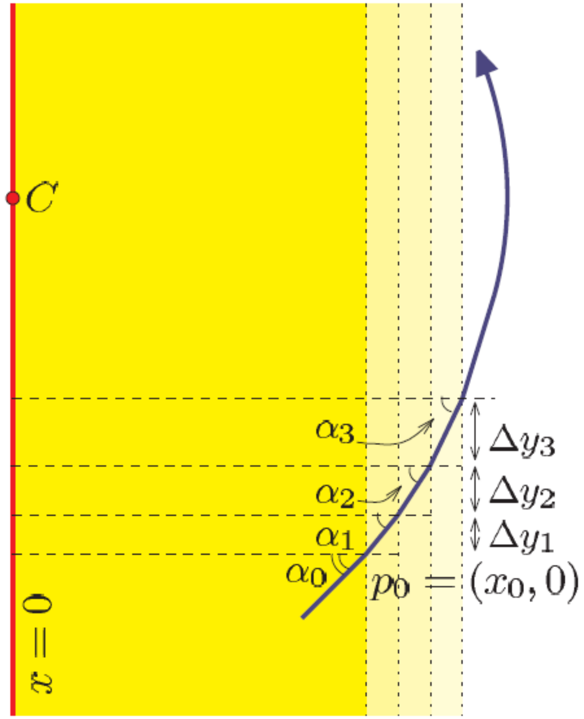
studied by Jacob Bernoulli

- appears in relation to motion in nature

[positivelyparkinsons.blogspot.com]

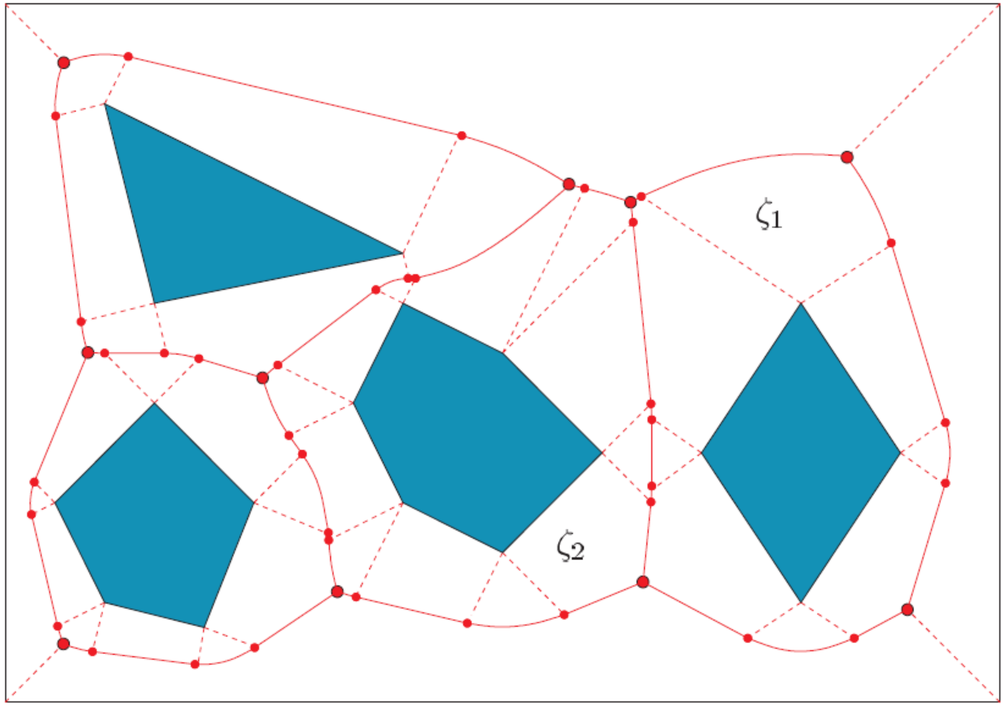
Optimal path for a point robot in the presence of a line obstacle for the combined measure

- a circular arc



Optimal path among polygonal obstacles for the combined measure

- comprises of logarithmic spirals circular arcs and portions of the Voronoi diagram: line segments and parabolic arcs
- at most 12n segments, where n is the number of vertices of the polygons
- approximation algorithms:
  - [WBH 08]
  - [AFS 16]



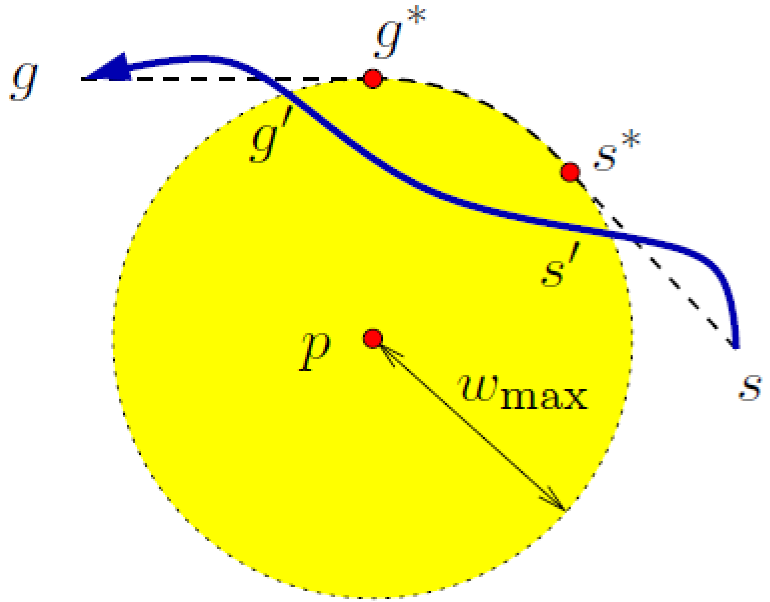
Corridors

A *corridor* $C = \langle \gamma(t), w(t), w_{\max} \rangle$ in a d-dimensional workspace (typically $d = 2$ or 3) is defined as the union of a set of d-dimensional balls whose center points lie along the *backbone path* of the corridor, which is given by the continuous function $\gamma : [0, L] \longrightarrow \mathbb{R}^d$. The radii of the balls along the backbone path are given by the function $w : [0, L] \longrightarrow (0, w_{\max}]$. Both γ and w are parameterized by the length of the backbone path. In the following, we refer to $w(t)$ as the *width* of the corridor at point t. The width is positive at any point along the corridor, and does not exceed $w_{\max}$, a prescribed *desired width* of the corridor.

- the interior of the corridor should be disjoint from the interior of the obstacles

Computing optimal corridors

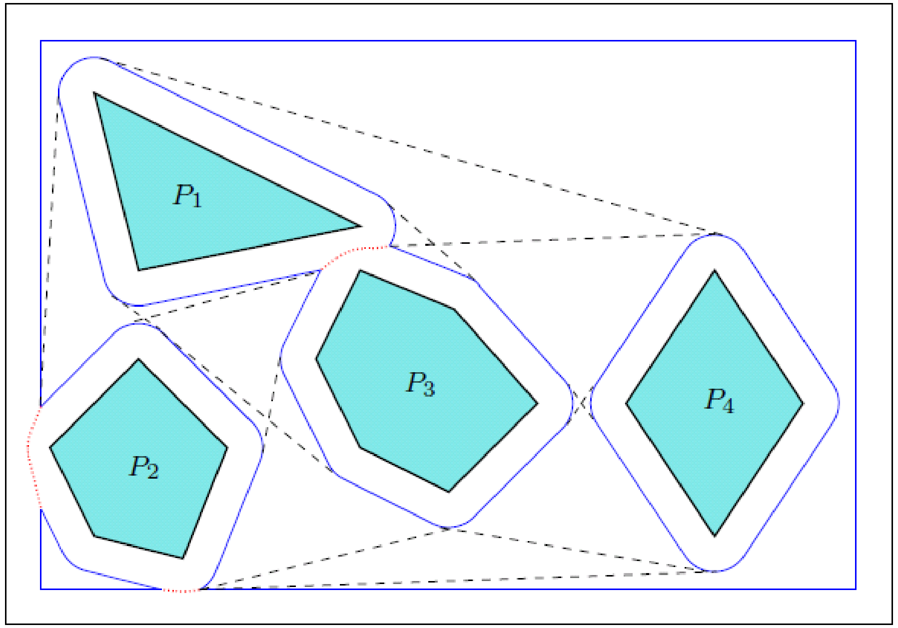
- It is a variant of optimal paths in the combined measure, with bounded clearance w_{max}
- Example: the case of start and goal far from a point obstacle



and now to something completely different:

Pareto optimal solutions and the length-clearance optimization

The visibility Voronoi diagram (VVD)
[WBH07]



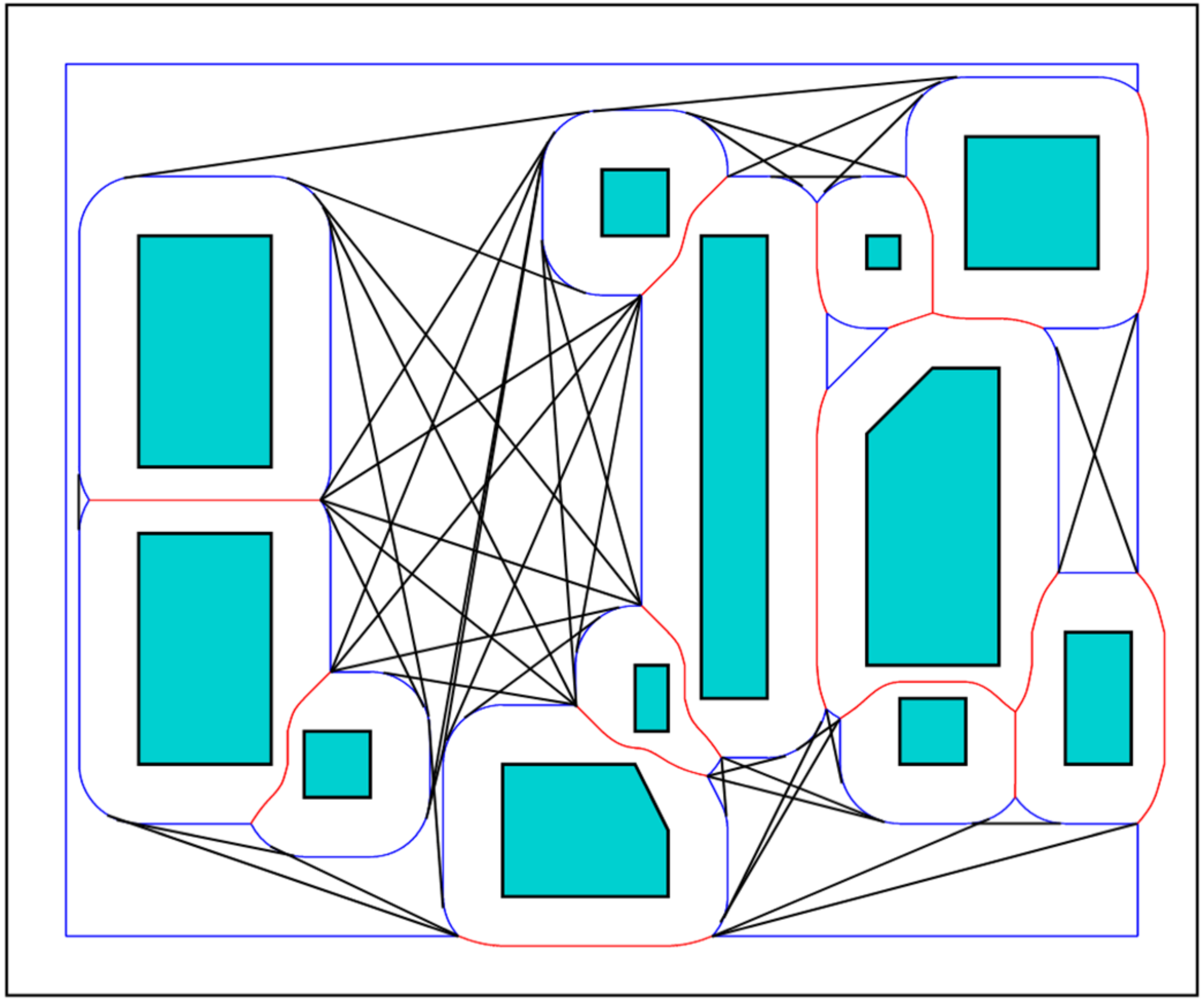
- finding the shortest path with a given clearance c, while still allowing to make significant shortcuts with lesser clearance on the Voronoi diagram

VVD$^{(c)}$, c=

blue edges: boundary of expanded obstacles

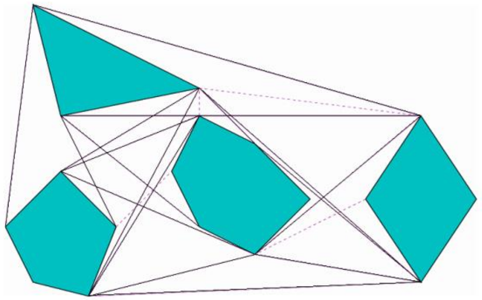
black edges: visibility diagram

red edges: Voronoi diagram

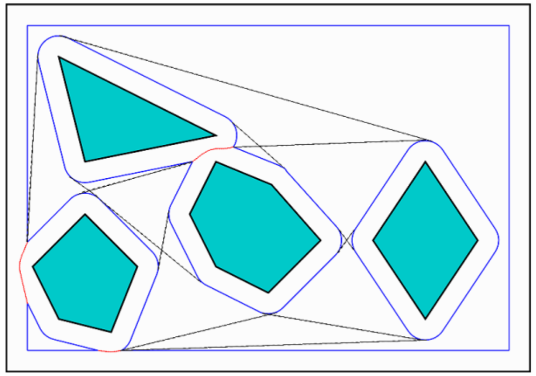


The visibility-Voronoi complex

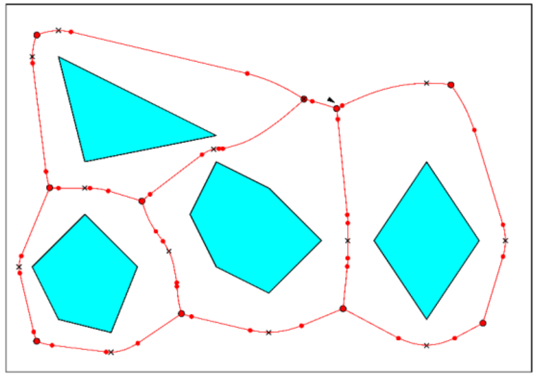
- the $VV^{(c)}$-diagram interpolates between the visibility graph and the Voronoi diagram:



$c = 0$



$c > 0$

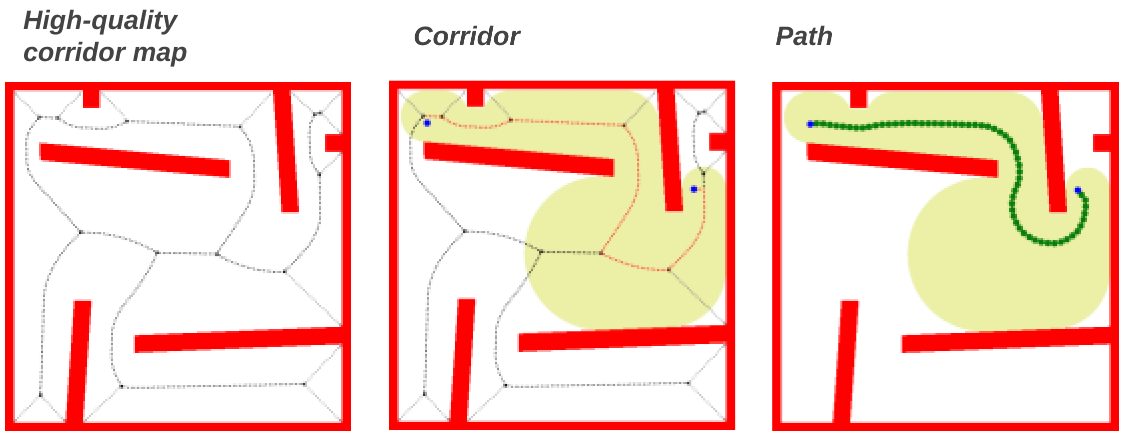


$c = \infty$

- the VV-complex encapsulates $VV^{(c)}$-diagrams for all c-values
- $O(n^2 \log n)$ construction time

Corridor maps

[Geraerts-Overmars]

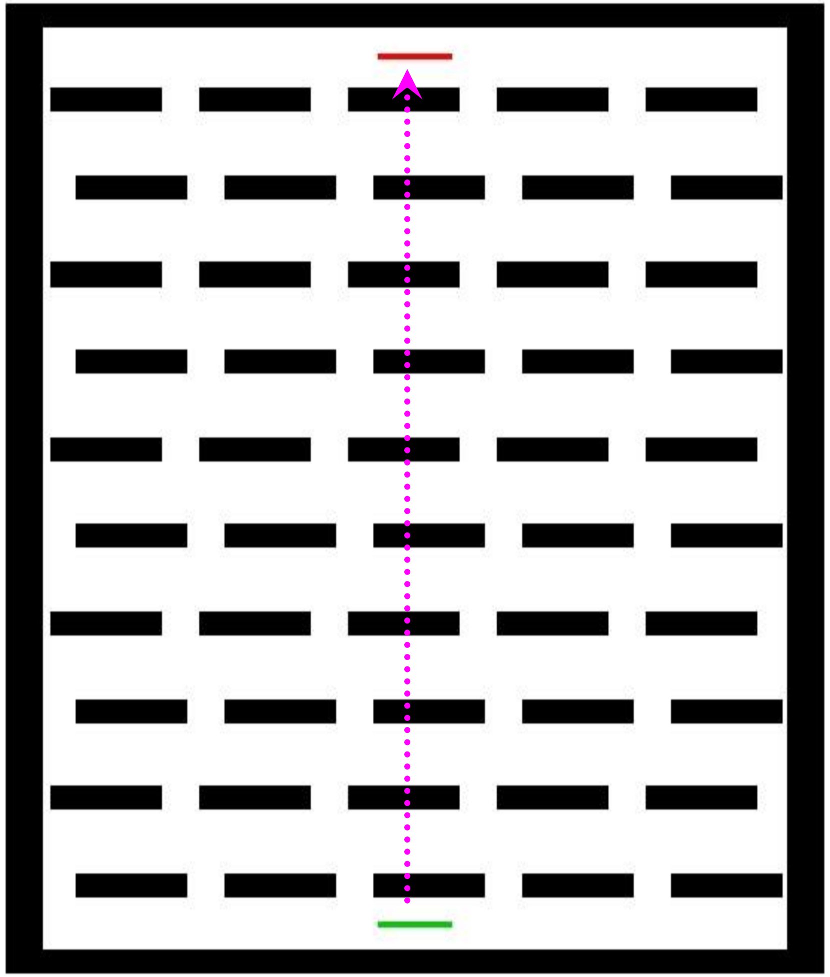


- motivated by motion planning in games
- similar to VVD/VVC, augmenting the VD with clearance information
- instead of providing a single solution path, provides a corridor among static obstacles, where later one can easily maneuver among dynamic obstacles

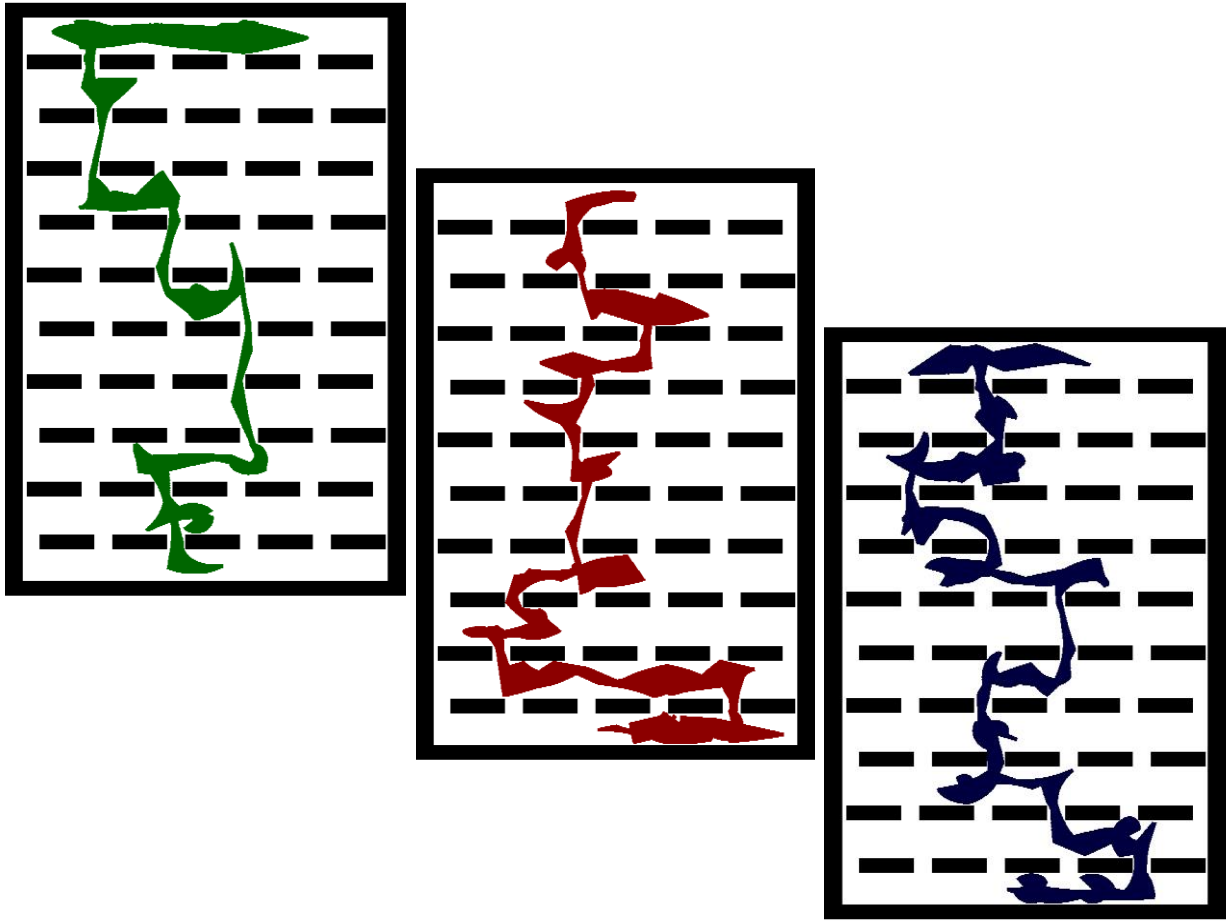
Path hybridization

Improving quality by path hybridization
[REH 11]

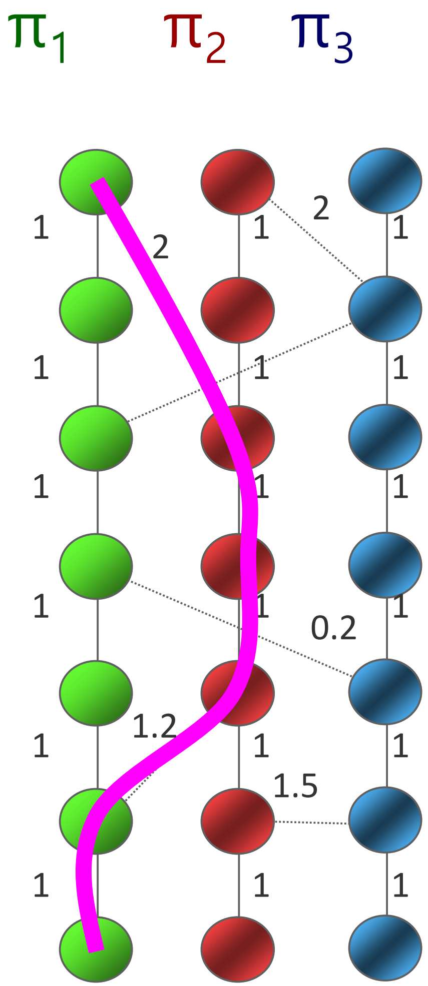
example: move the rod from the bottom to the top of a 2D grid *(rotation + translation)*



3 randomly generated motion paths



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



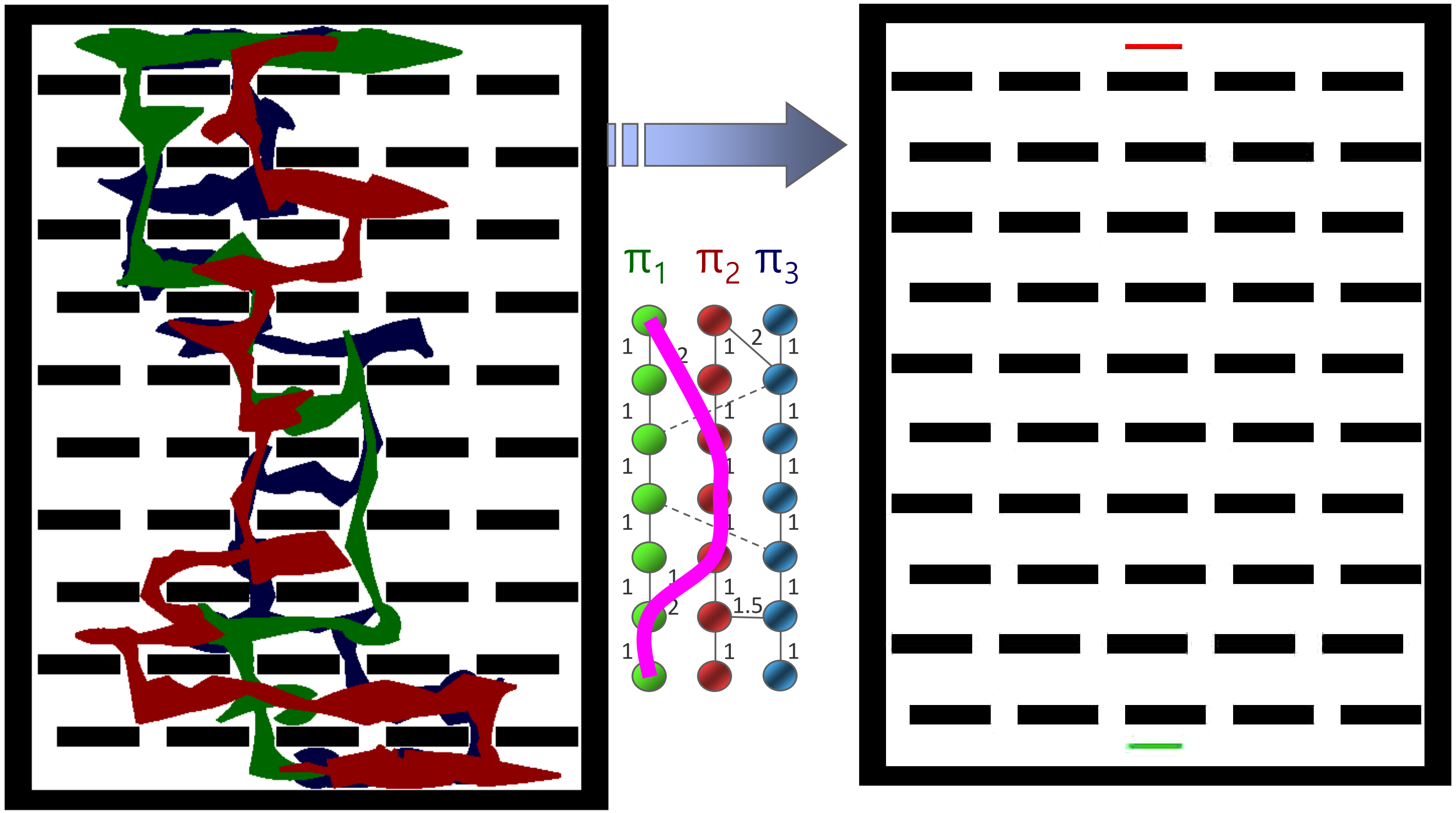
Algorithm 1 Building an Hybridization–Graph

Build-H-Graph(`PathsList`)

`PathsList`: a set of l input solution paths from initial to goal configuration
G : an output H-Graph

```
initialize-H-Graph(PathsList)
for all π1, π2 ∈ PathsList do
  potentialBridgeEdges = a list of potential bridging edges
  between π1 and π2
  for all e ∈potentialBridgeEdges do
    π_local = localPlanner(e.from → e.to)
    if valid(π_local) then
      G.addWeightedEdge(e)
    end if
  end for
end for
return G
```

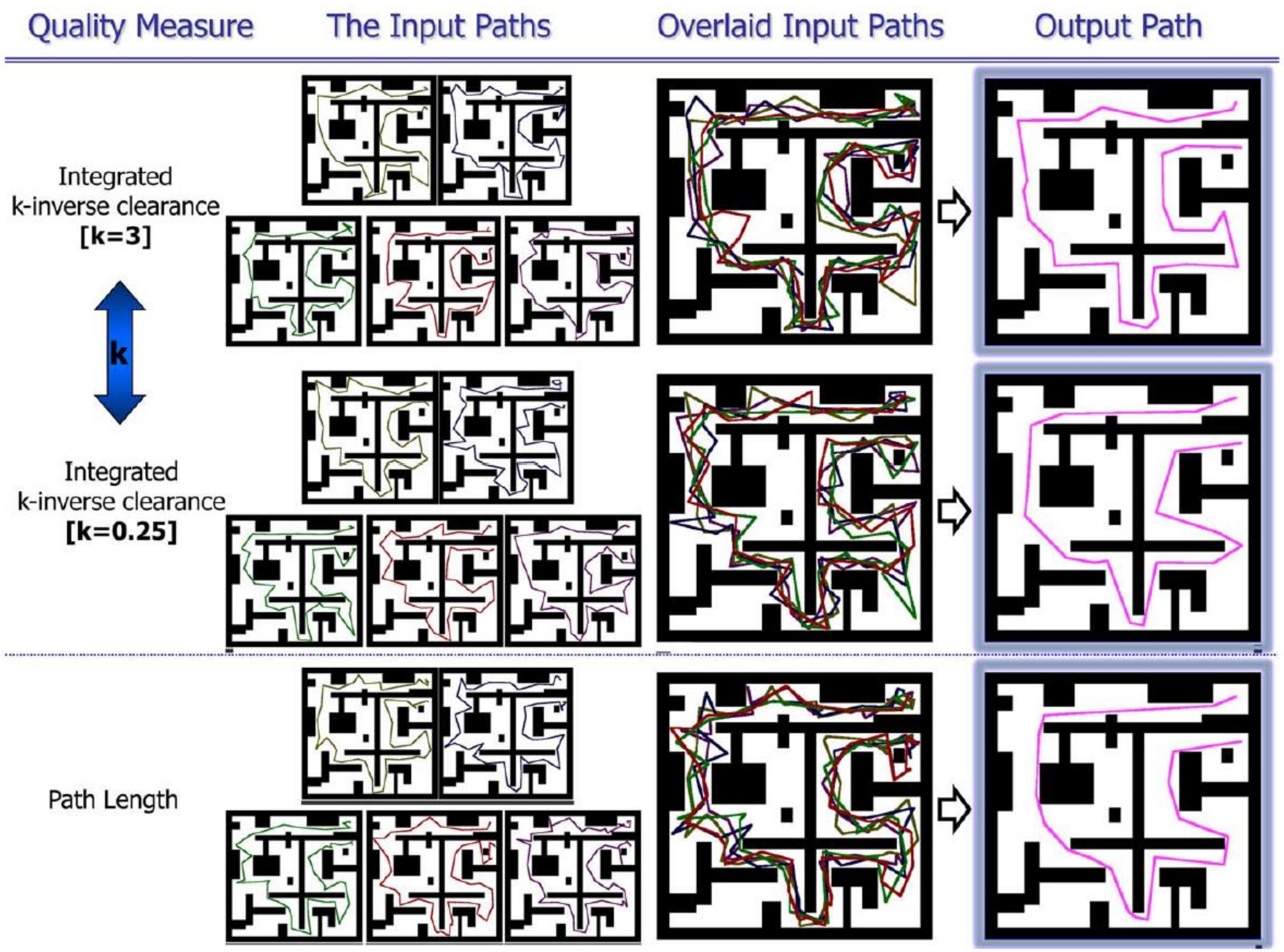
Hybridizing the paths



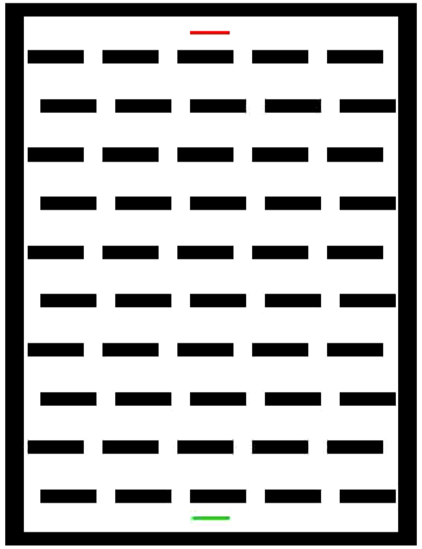
Length-clearance optimization

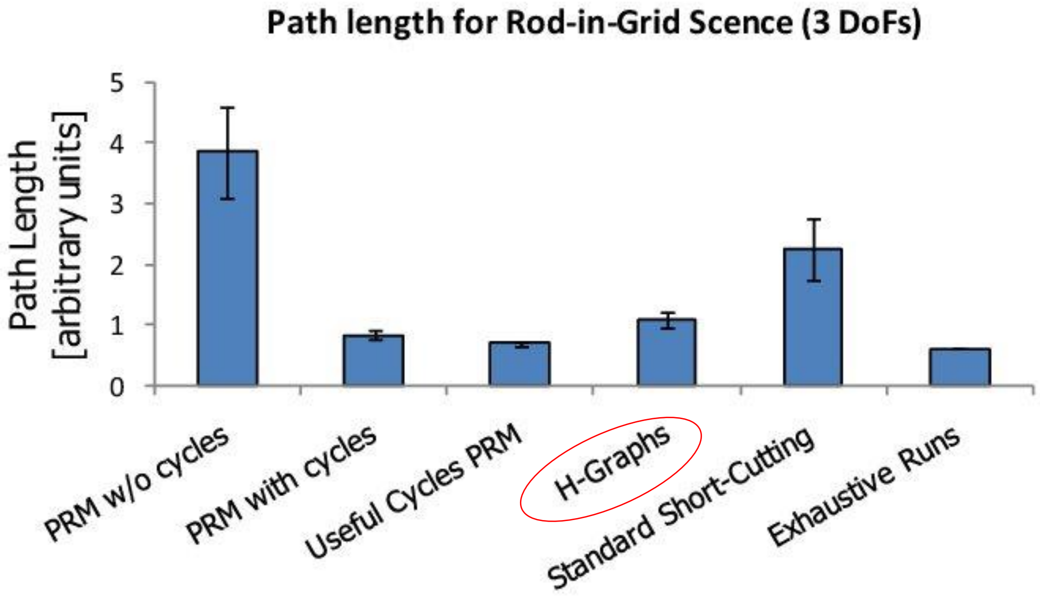
- Scalarization, weighted length
- integrated k-inverse clearance, path length weighted by C^{-k}
- large k: more emphasis on clearance

Uniform treatment of general quality criteria



Rod-in-Grid scene: 3 dof

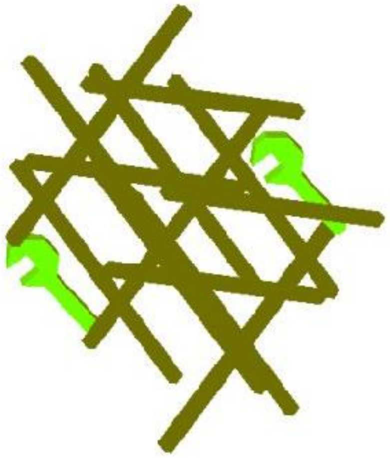


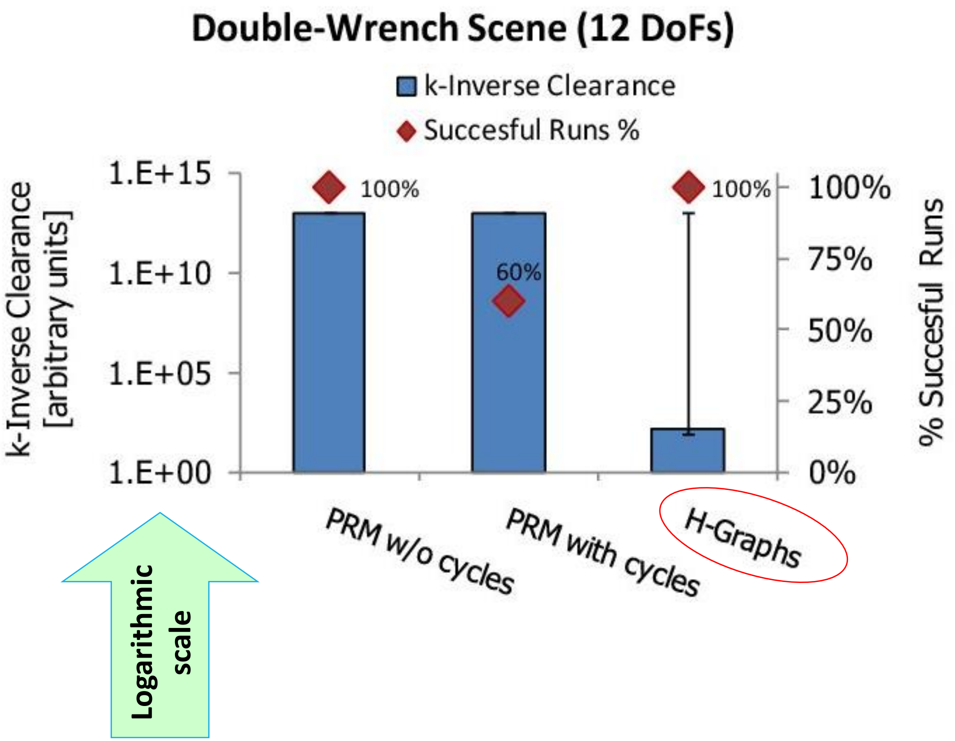


Implemented in the **OOPSMP package** (Plaku, Moll and Kavraki), collision detection – **PQP** (Lin and Manocha)

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)

Scene adapted from Nieuwenhuisen et al., ICRA 04’

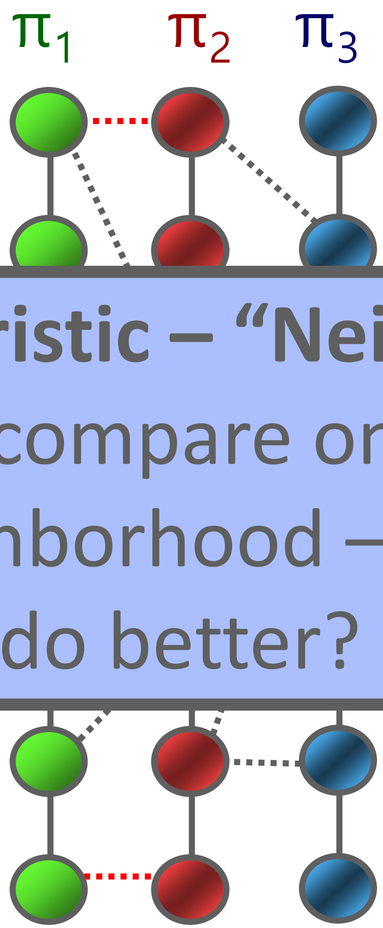
Running-time bottleneck for hybridization:

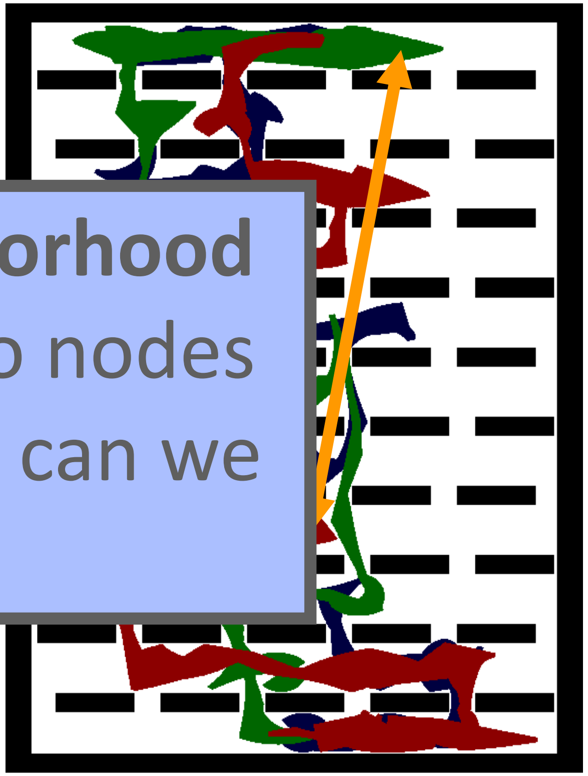
Trying to connect nodes from different paths

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

we assu

paths is





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

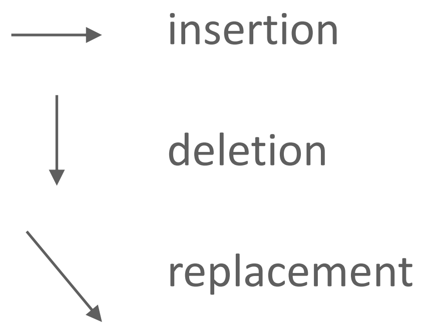
Comparing “This dog” and “That Dodge” with insertion / deletions / replacement:

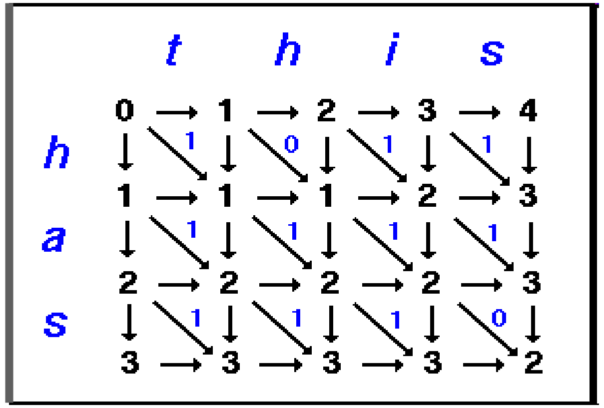
T H I – S D O – G –

T H A T – D O D G E

---

dynamic-programming algorithm:





Algorithm 2 Dynamic–Program for Matching Two Paths

MatchPaths(p,q)

C: a cost matrix $\in \Re^{m \times n}$

$\{For\ i = 0\ or\ j = 0,\ we\ define\ C_{i,j} = \infty\}$

TB: a symbolic trace-back matrix

for i=1 to m **do**

for j=1 to n **do**

`Match` $\Leftarrow C_{i-1,j-1} + \Delta(p_i, q_j)$

`Up` $\Leftarrow C_{i-1,j} + \mathrm{GAP_{ext}}$

`Left` $\Leftarrow C_{i,j-1} + \mathrm{GAP_{ext}}$

$C_{i,j} \Leftarrow \min(\texttt{Match}, \texttt{Up}, \texttt{Left}, 0)$

$$TB_{i,j} \Leftarrow \begin{cases} \text{"}\nwarrow\text{"} & \text{for Match} \\ \text{"}\uparrow\text{"} & \text{for Up} \\ \text{"}\leftarrow\text{"} & \text{for Left} \end{cases}$$

end for

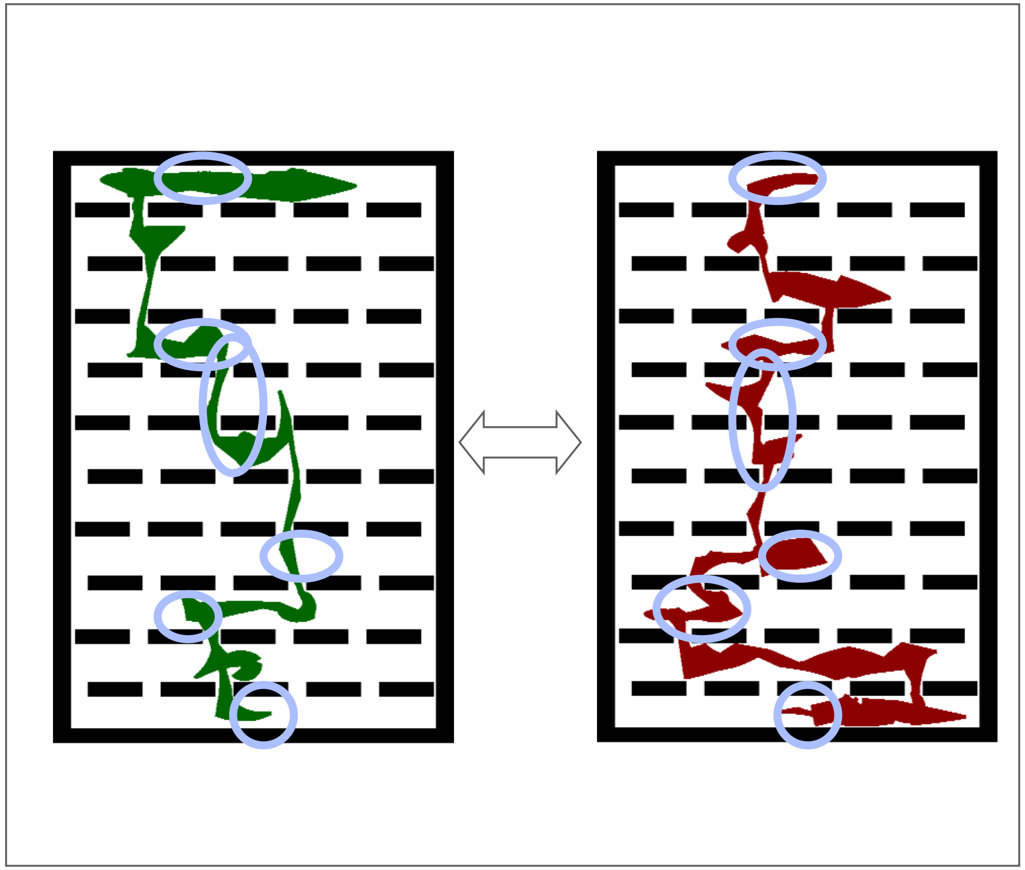
end for

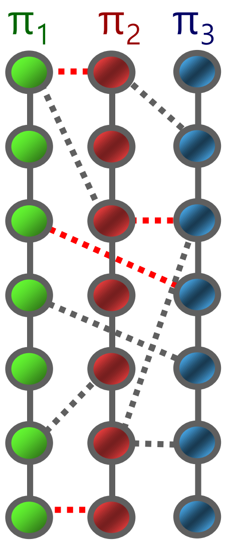
return matrices C and TB

- $\mathrm{GAP_{init}}$ (omitted above) vs $\mathrm{GAP_{ext}}$

Alignment length is linear

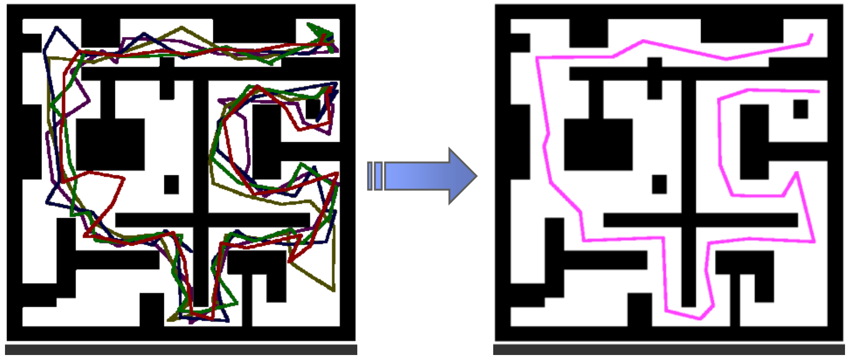
Now testing only O(n) edges along the alignment

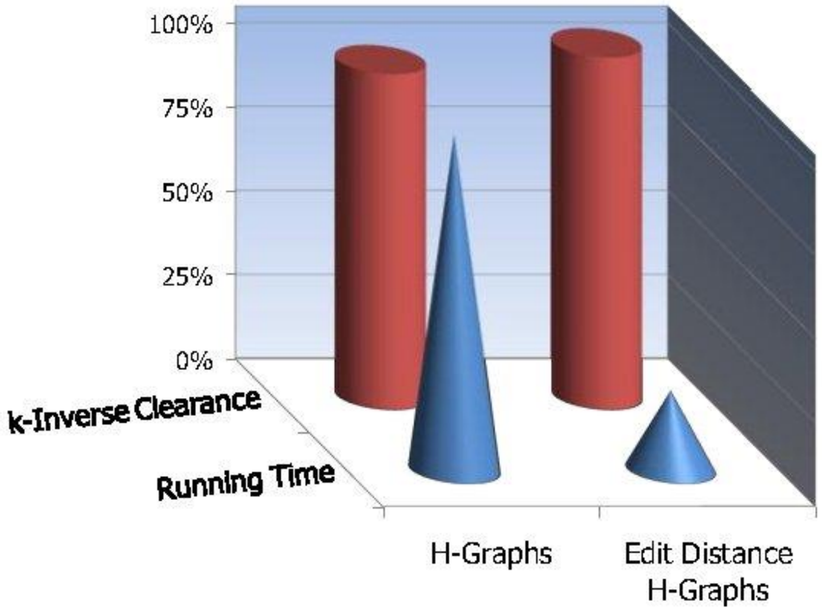


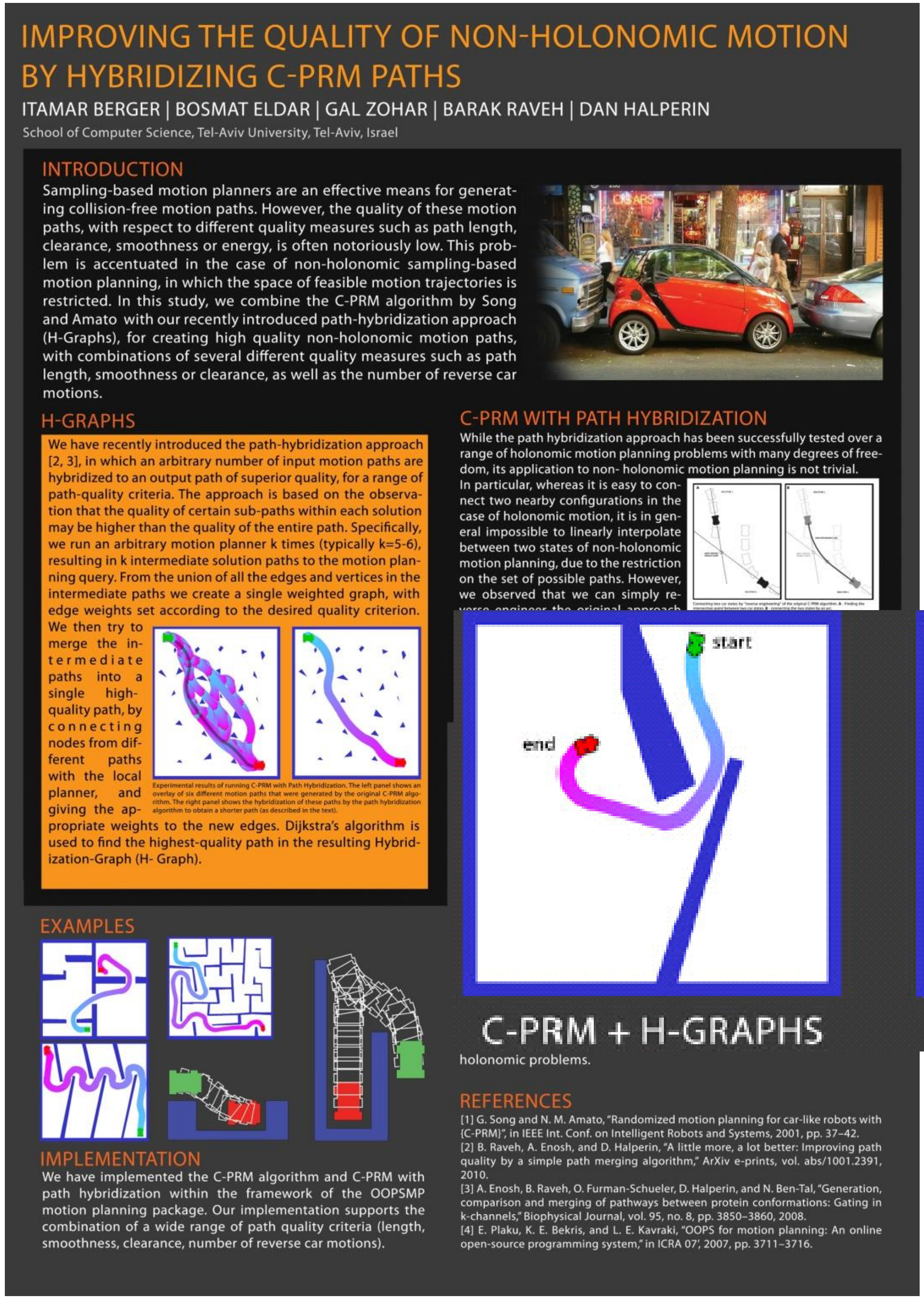


Comparison of running times

- 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 re-
...
holonomic problems.

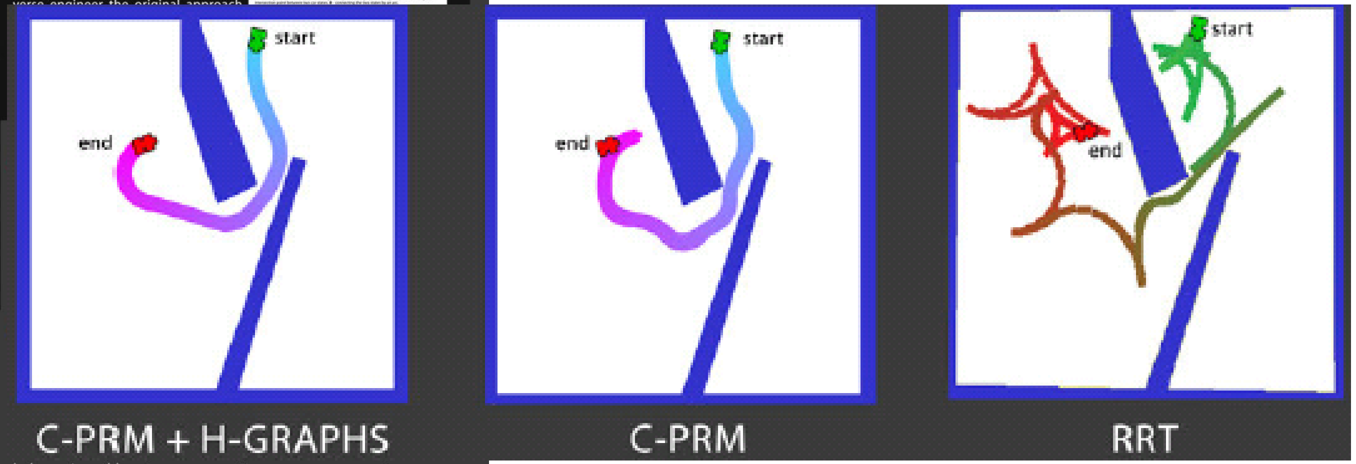
EXAMPLES

IMPLEMENTATION

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

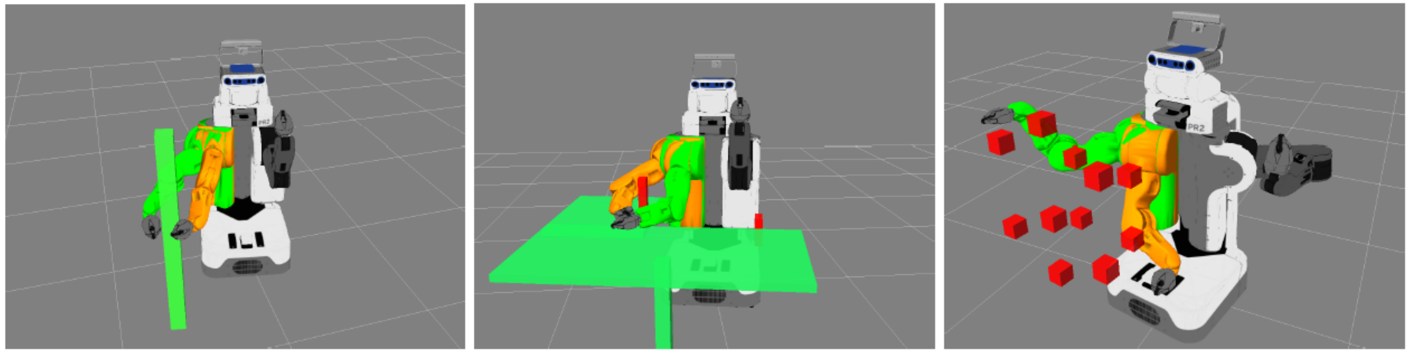
REFERENCES

[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. Enosh, 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. Enosh, 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.



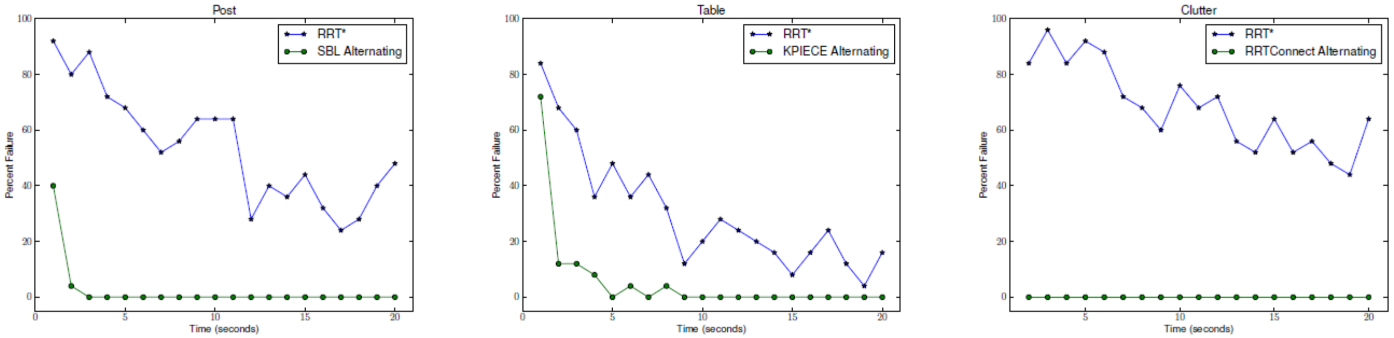
applied to car-like motion with various quality criteria: length, smoothness, clearance, number of reverse vehicle motions

Luna et al, ICRA13



(a) Post (b) Table (c) Clutter

Fig. 5: Manipulator scenes for the 7-DOF arm of the PR2. The start pose is shown in green, and the goal pose is orange.

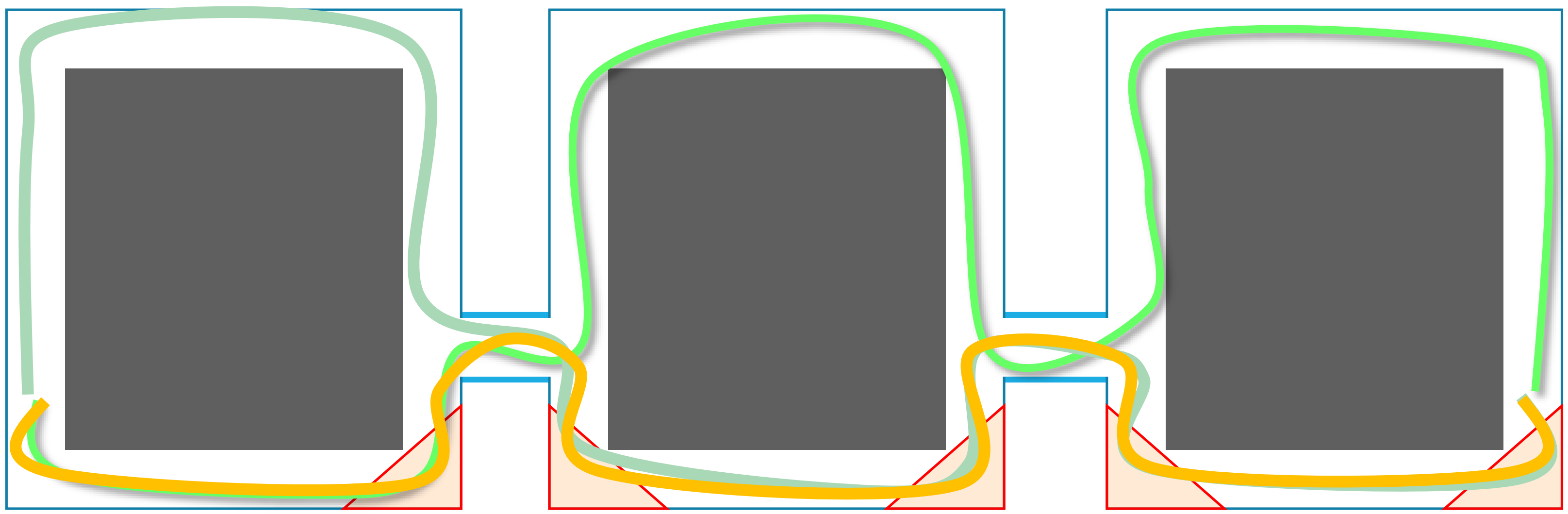


(a) Post: SBL/RRT* (b) Table: KPIECE/RRT* (c) Clutter: RRT-Connect/RRT*

Fig. 7: The percentage of experiments that failed to find any solution to the given query within the given time budget. Values are out of 25 possible attempts.

In experiments in higher dimensions alternating Hgraphs+shortcut beat RRT* given the same amount of running time

Why do Hgraphs work?



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

More on optimization in SB planners

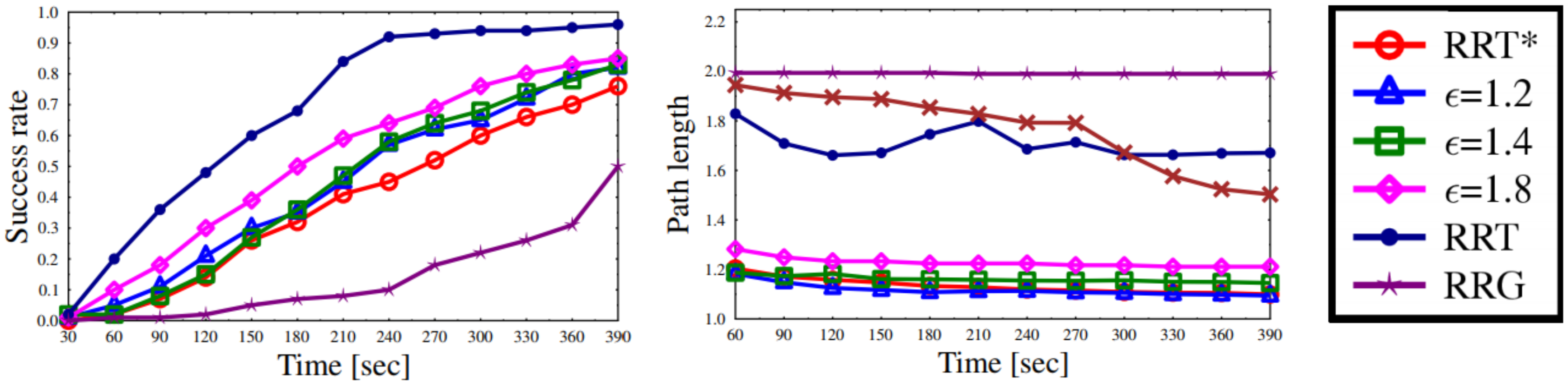
Remark: RRG [KF11]

- like the RRT* algorithm, but we simply make all the valid (collision free) connections between x_{new} and the nodes in X_{near} , with two edges in opposite direction for each node in X_{near}
- the RRT* tree is a subgraph of RRG
- RRG requires more storage space and is practically more time consuming than RRT*

Tradeoff (speed vs. quality): LBT-RRT

[SH14,16]

- Lower-bound RRT:
  - Guarantees convergence to (1+ ε)OPT
    - When ε=0 behaves like RRG
    - When ε=∞ behaves like RRT
  - An edge (v,v') is collision checked only if it can potentially improve the cost of any vertex on the shortest-path tree rooted in v' by at least 1+ ε



Oren Salzman, Dan Halperin:
Asymptotically Near-Optimal RRT for Fast, High-Quality Motion Planning. IEEE Trans. Robotics 32(3): 473-483 (2016)

References

References

- Shortest Path and Networks, J.S.B. Mitchell, Chapter 31 of the Handbook on DCG, Goodman-O'Rourke-Toth (eds)

- Visibility Graphs, Chapter 15 of the Computational Geometry book by de Berg et al

- Ron Wein, Jur P. van den Berg, Dan Halperin: The visibility-Voronoi complex and its applications. Comput. Geom. 36(1): 66-87 (2007)
- Ron Wein, Jur P. van den Berg, Dan Halperin: Planning High-quality Paths and Corridors Amidst Obstacles. I. J. Robotics Res. 27(11-12): 1213-1231 (2008)

References, cont'd

(suitable for the final project)

- Barak Raveh, Angela Enosh, Dan Halperin:
  A Little More, a Lot Better: Improving Path Quality by a Path-Merging Algorithm. IEEE Trans. Robotics 27(2): 365-371 (2011)

- Ryan Luna, Ioan Alexandru Sucan, Mark Moll, Lydia E. Kavraki, Anytime solution optimization for sampling-based motion planning. ICRA 2013: 5068-5074

- Oren Salzman, Dan Halperin:
  Asymptotically Near-Optimal RRT for Fast, High-Quality Motion Planning. IEEE Trans. Robotics 32(3): 473-483 (2016)

THE END