Motion Planning Lecture 13

Multi-Robot Motion Planning

Wolfgang Hönig (TU Berlin) and Andreas Orthey (Realtime Robotics)

July 17th, 2024

Recap Last Week

Last Week

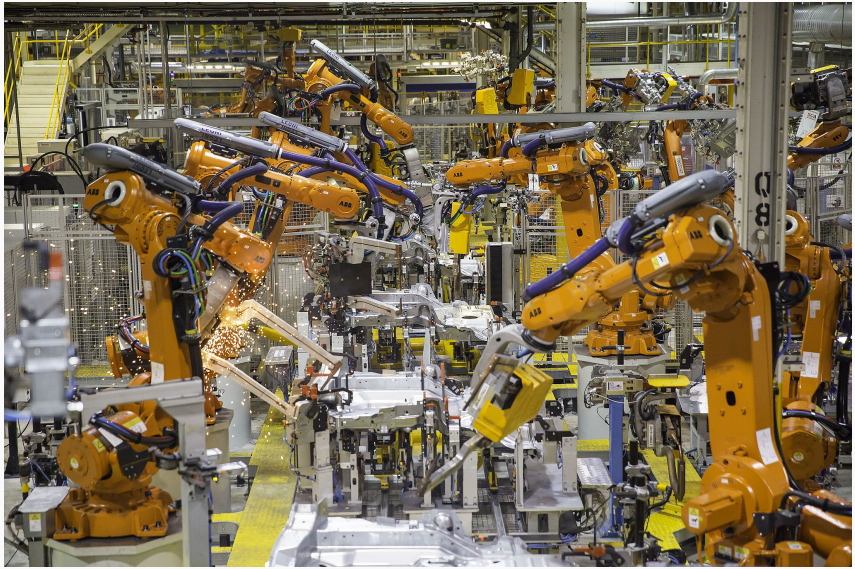
- Optimization-based planning (KOMO)
- Comparison of Search, Optimization, and Sampling
- Hybrid approaches

Today

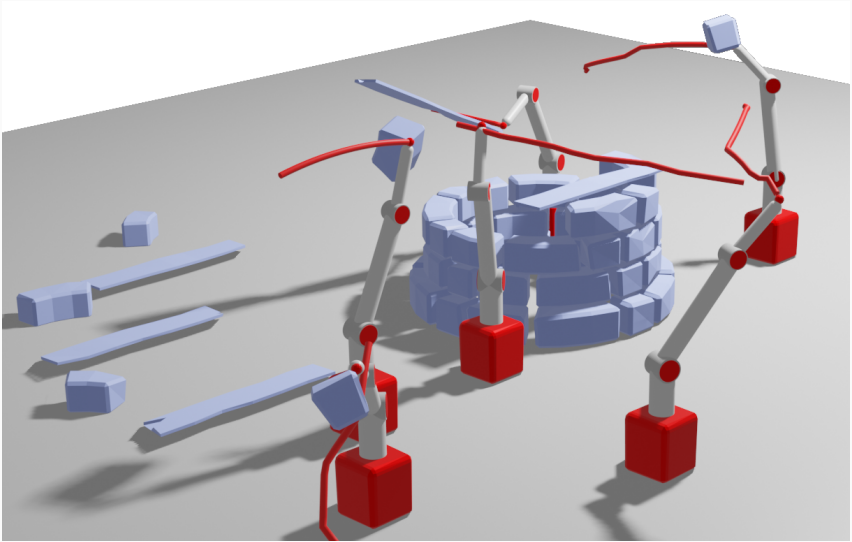
- Multi-robot motion planning
- Planning with manifold constraints

Multi-robot navigation

Multi-robot welding



Multi-robot assembly



Coordination of drones

Multi-robot planning

Multi-robot planning: Coordinate motion of multiple robots acting in the same environment.

Multi-robot planning

Central idea: Multi-robot planning is just planning in the *composite configuration space*.
Robot state spaces X_1, X_2, X_3, Composite space $X = X_1 \times X_2 \times X_3$.

How to adjust primitive methods?

- Sampling
- Interpolation
- Propagation/Steering
- Collision-Checking

Multi-robot planning

Computational complexity is exponential in the number of dimensions.

John Canny, "The complexity of robot motion planning", 1988 [1]

Multi-robot planning

Goal

Find decompositions of the composite configuration space to make multi-robot planning more efficient (while keeping completeness/optimality).

Multi-robot planning: Taxonomy

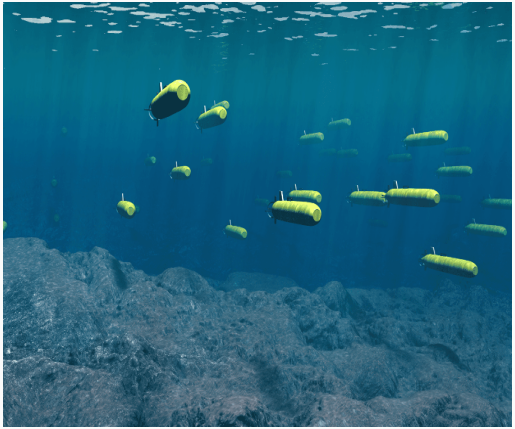
- Are your robots identical? (Homogeneous vs Non-Homogeneous)
- If homogeneous, are the goals interchangeable? (Labeled vs Unlabeled)
- Who controls your robots? (Centralized vs Decentralized)
- What kind of cost do you want to minimize?
  - Makespan (last arrival time)
  - Flowtime (total arrival time)

Multi-robot planning: Taxonomy

- Homogeneous planning
  - Pebbles on a graph
  - Conflict-based search
- Non-Homogeneous planning
  - Prioritized planning (vertical)
  - Decomposed planning (horizontal)
  - M*
  - dRRT/dRRT*

Homogeneous Planning

Homogeneous planning



Homogeneous planning

Also called Multi-agent path finding (MAPF)

- All robots are identical
- All robot state spaces are identical (modulo robot-robot collisions)

Main Idea

Problem can be reduced to pebbles-on-a-graph [2]

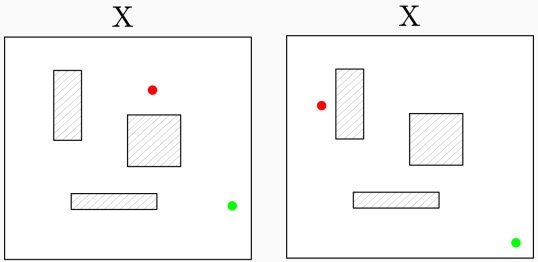
Homogeneous planning: Pebbles on a graph reduction

- Assume we have a homogeneous team of M robots
- Let $X_C = X \times X \times \cdots \times X$ (M times) be the composite state space

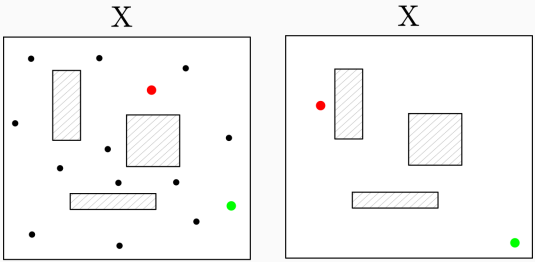
Homogeneous Planning

Example of pebbles on a graph in 2D

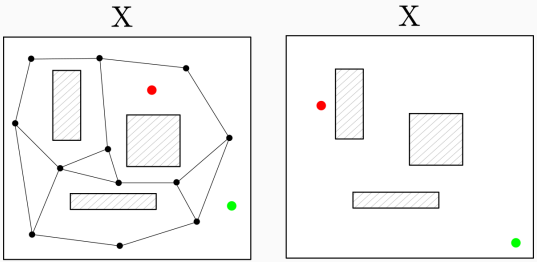
Homogeneous planning: Pebbles on a graph reduction



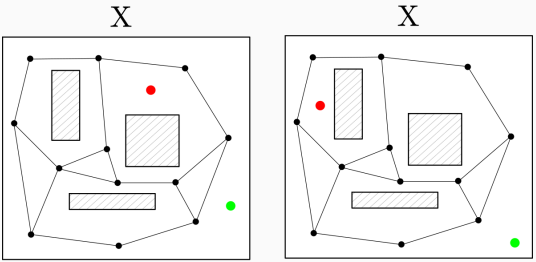
Homogeneous planning: Pebbles on a graph reduction



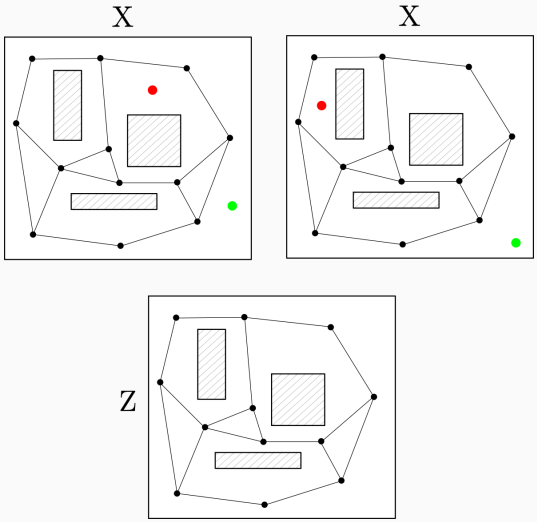
Homogeneous planning: Pebbles on a graph reduction



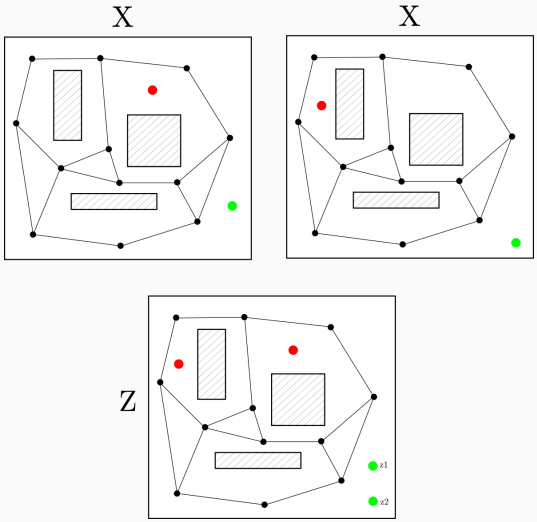
Homogeneous planning: Pebbles on a graph reduction



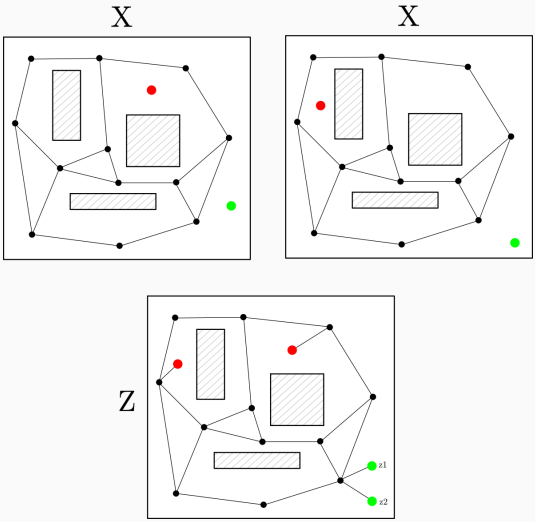
Homogeneous planning: Pebbles on a graph reduction



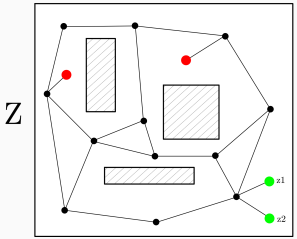
Homogeneous planning: Pebbles on a graph reduction



Homogeneous planning: Pebbles on a graph reduction



Homogeneous planning: Pebbles on a graph reduction



Pebbles-on-a-graph Problem

Move M pebbles on a graph from a start arrangement to a goal arrangement. Pebbles are not allowed to collide.

see e.g. Kornhauser, "Coordinating Pebble Motion on Graphs" (1984) [3]

Homogeneous Planning

Theory of pebbles on a graph

Homogeneous planning: Pebbles on a graph reduction

Let $X_C = X \times \ldots \times X$ be a homogeneous planning problem for M robots with M start/goal pairs $(x_I^1, x_G^1), \ldots, (x_I^M, x_G^M)$.

Pebbles-on-a-graph reduction

- Create a *single* roadmap G on X.
- Add start/goal pair vertices to X and connect to G
- Solve this simultaneously while taking robot-robot collisions into account.

Homogeneous planning: Pebbles on a graph reduction

Pebbles-on-a-graph

Let $G = (V, E)$ be a graph on X. Let us assume that there are M pebbles (or agents), which start at vertices $s_1, \ldots, s_M$ (start arrangement) and need to go to vertices $g_1, \ldots, g_M$ (goal arrangement). [3]

Approaches to pebbles-on-a-graph

- Push and Swap [4]
- Integer Linear Programming [5]
- Conflict-based search [6]

Homogeneous planning: Pebbles on a graph reduction

Pebbles-on-a-graph

Let $G = (V, E)$ be a graph on X. Let us assume that there are M pebbles (or agents), which start at vertices $s_1, \ldots, s_M$ (start arrangement) and need to go to vertices $g_1, \ldots, g_M$ (goal arrangement). [3]

Approaches to pebbles-on-a-graph

- Push and Swap [4]
- Integer Linear Programming [5]
- Conflict-based search

Homogeneous Planning

Conflict-based search

Homogeneous planning: Solving pebbles on a graph

Conflict-based search I

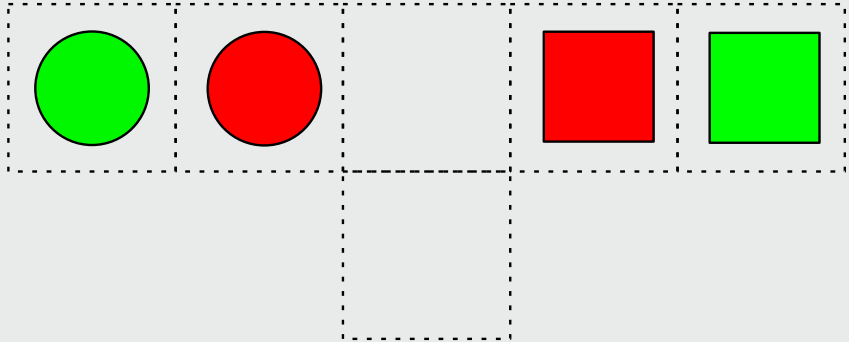
- Assume each pebble has two moves
- Wait at current vertex
- Move to adjacent vertex
- Goal: Find set of moves to reach goal arrangement.

Conflict-based search II

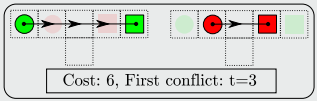
- Key idea: Resolve collisions one-by-one
- First, plan for each pebble individually
- Second, resolve collisions
  - Assume pebbles A,B collide at time T at vertex V
  - Add collision constraints to problem
  - Either (1) A should not be V at time T
  - Or (2) B should not be V at time T
  - This creates a *constraint tree*
  - Pick next entry based on "Best cost" and "First collision first" tie-breaker.

Homogeneous planning: Solving pebbles on a graph

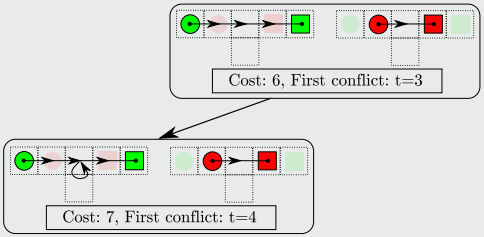
Constraint Tree Example



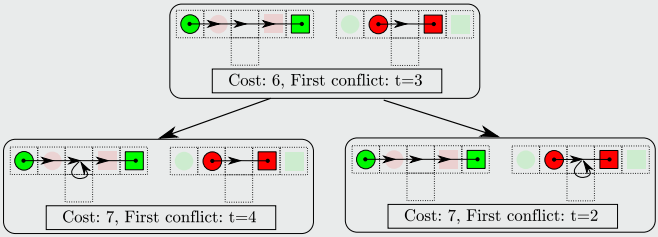
Constraint Tree Example



Constraint Tree Example

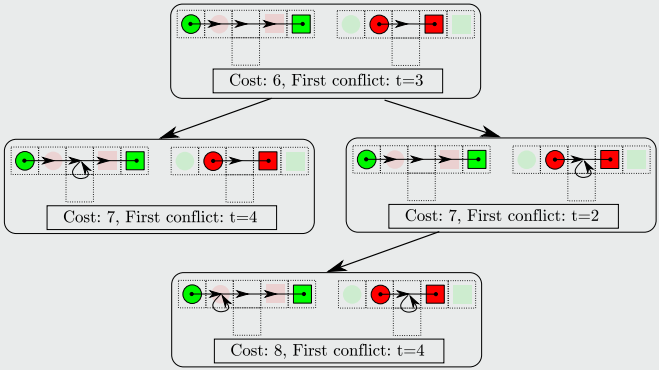


Constraint Tree Example



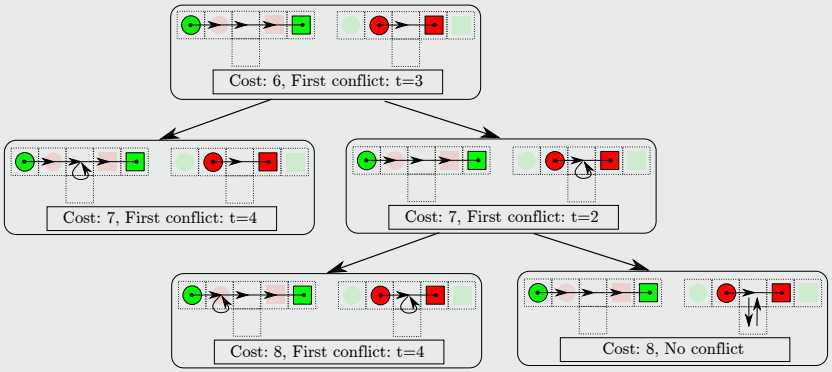
Homogeneous planning: Solving pebbles on a graph

Constraint Tree Example



Homogeneous planning: Solving pebbles on a graph

Constraint Tree Example



Optimality

Theorem

Conflict-based search returns the optimal solution.

Proof

Part 1: Proof that no valid paths are removed by conflict resolution.

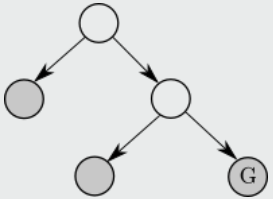
Part 2: Proof that a goal node path has the lowest cost.

Optimality

Theorem

Conflict-based search returns the optimal solution.

Proof



Best-first search with open nodes (grey), closed nodes (white), and goal nodes (G).

Optimality

Theorem

Conflict-based search returns the optimal solution.

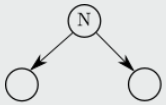
Proof I: No valid paths are removed by conflict resolution.

Let N be a node in the constraint tree, and $CV(N)$ be all valid and consistent paths at N.

(1) If we add a constraint at N, we split into nodes N_1, N_2.

(2) A valid path needs to be either in N_1 or in N_2 or it is invalid.

(3) Therefore, $CV(N)$ is split into $CV(N_1)$ and $CV(N_2)$ (a valid path is either in N_1 or in N_2).



Optimality

Theorem

Conflict-based search returns the optimal solution.

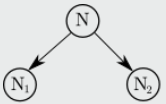
Proof I: No valid paths are removed by conflict resolution.

Let N be a node in the constraint tree, and $CV(N)$ be all valid and consistent paths at N.

(1) If we add a constraint at N, we split into nodes N_1, N_2.

(2) A valid path needs to be either in N_1 or in N_2 or it is invalid.

(3) Therefore, $CV(N)$ is split into $CV(N_1)$ and $CV(N_2)$ (a valid path is either in N_1 or in N_2).



Optimality

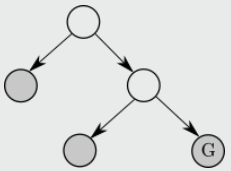
Theorem

Conflict-based search returns the optimal solution.

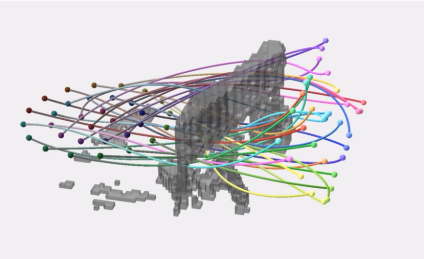
Proof II: A goal node path has the lowest cost.

(1) Let us assume that we reached a goal node G with cost $c(G)$.
(2) Let p be an arbitrary valid path. Then p must be in an open node N. Then it is lower bounded by the best cost at N, i.e. $c(N(p)) \leq c(p)$.
(3) Since we used best-first search, the cost at G is the lowest of all open nodes. Therefore

$$c(G) \leq c(N(p)) \leq c(p)$$

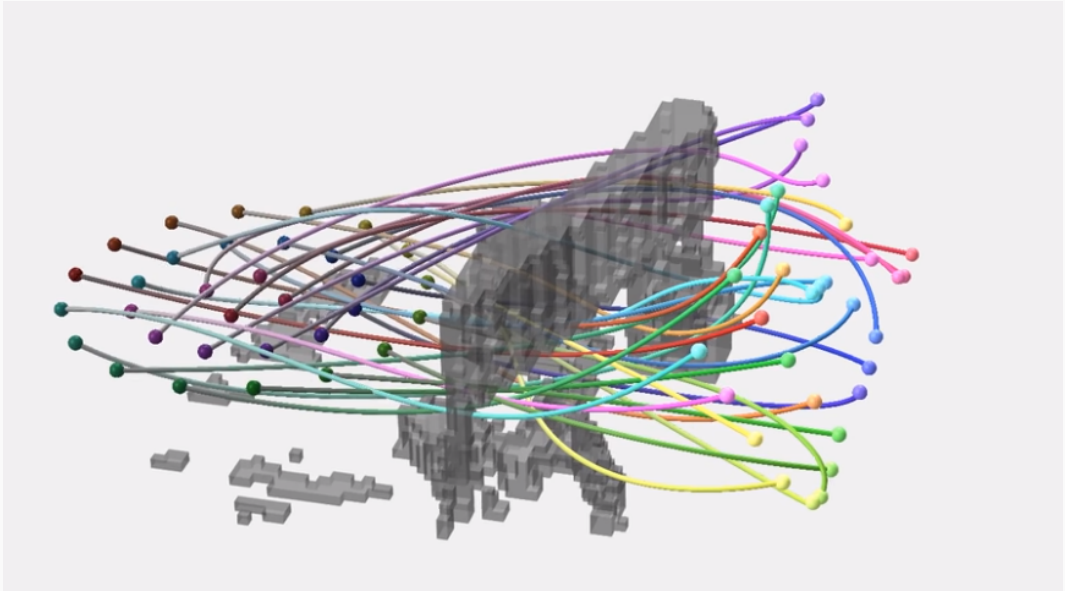


Homogeneous multi-robot planning



Example of 32 drones on composite state space $\mathbb{R}^{192}$.
Using homogeneous multi-robot planning: Computation time is 50s (roadmap generation on $\mathbb{R}^6$) plus 0.8s (conflict annotation) plus 0.5s (conflict-based search) plus 6.5s (optimization) [7]

Homogeneous multi-robot planning



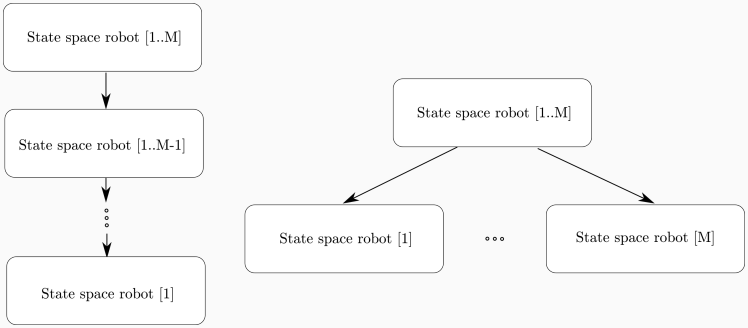
Non-Homogeneous Multi-robot Motion Planning

Non-homogeneous planning

Non-Homogeneous Multi-Robot Planning

- All state spaces differ
- No easy reduction possible
- Can usually not efficiently be solved in composite state space

Non-Homogeneous Multi-Robot Planning

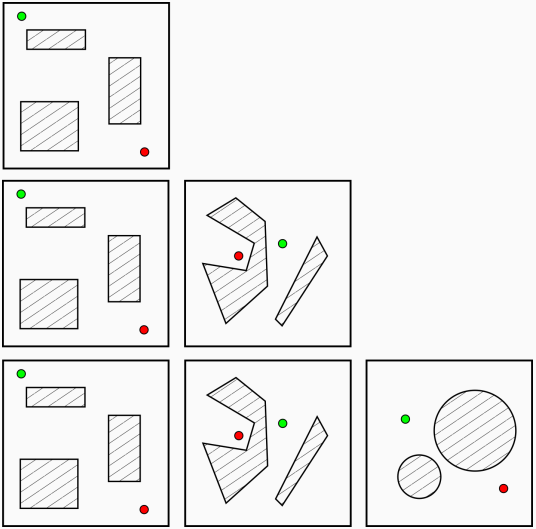


Main approaches

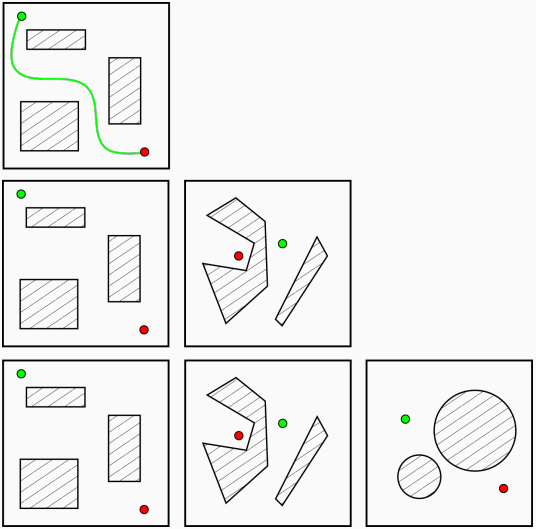
- Prioritized (also greedy, or "vertical")
- Decomposition (also "horizontal")

Prioritized Multi-Robot Motion Planning

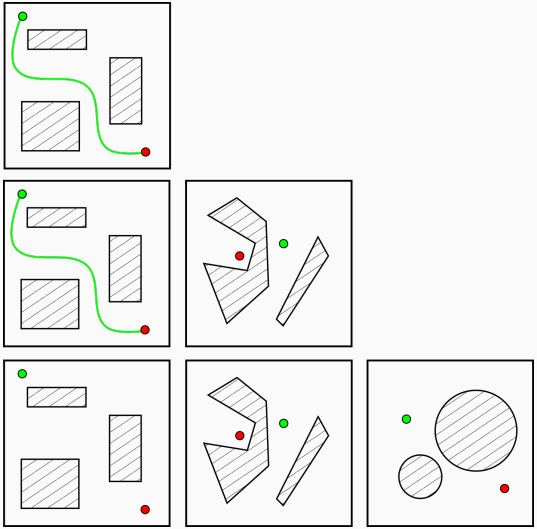
Non-Homogeneous Multi-Robot Planning: Prioritized



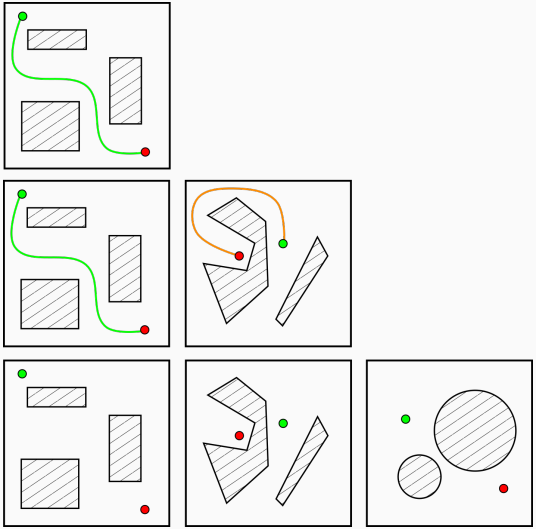
Non-Homogeneous Multi-Robot Planning: Prioritized



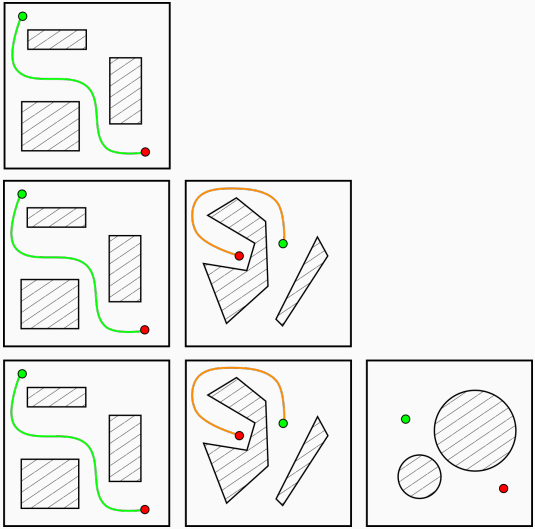
Non-Homogeneous Multi-Robot Planning: Prioritized



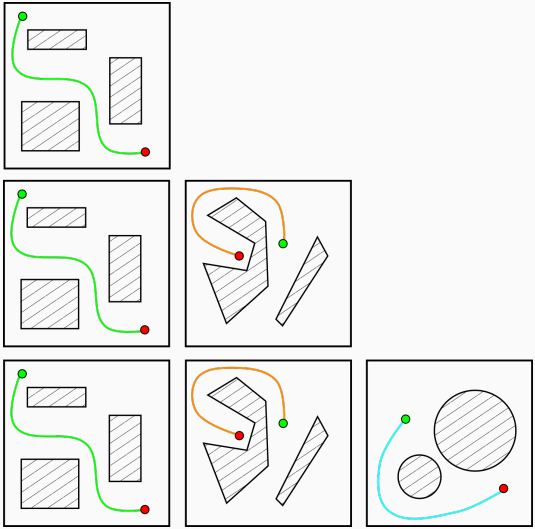
Non-Homogeneous Multi-Robot Planning: Prioritized



Non-Homogeneous Multi-Robot Planning: Prioritized



Non-Homogeneous Multi-Robot Planning: Prioritized

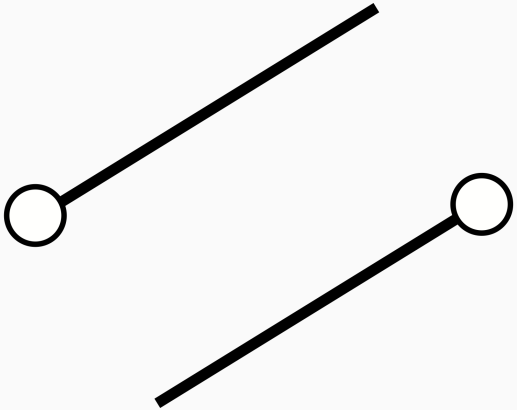


Non-Homogeneous Multi-Robot Planning

- Let $X = Y \times Z$ be the composite state space.
- Prioritized planning: Find path on Y, then use this as a constraint on X
- Constraint can be modelled as a *path restrictions*.

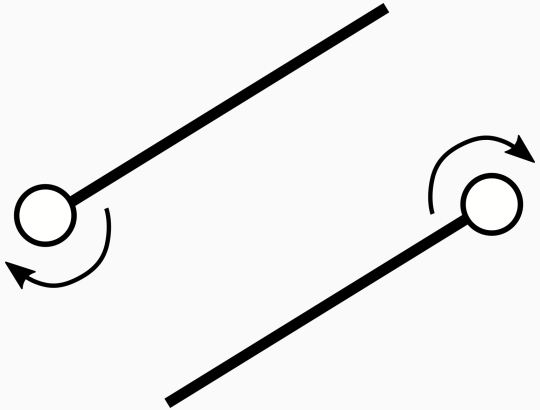
Non-Homogeneous Multi-Robot Planning

Consider two 1-d robots with state space S^1 (the circle).

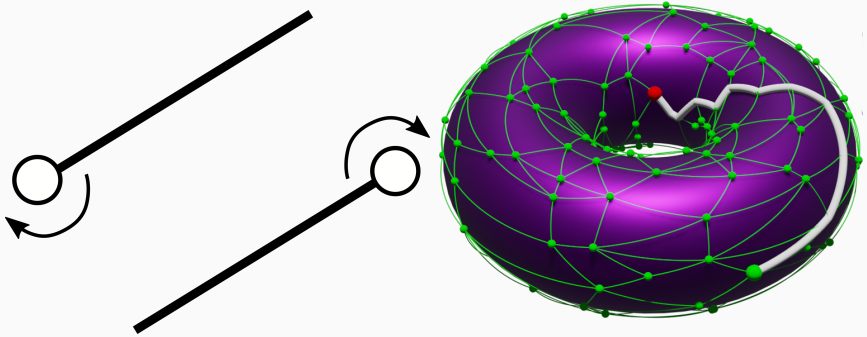


Non-Homogeneous Multi-Robot Planning

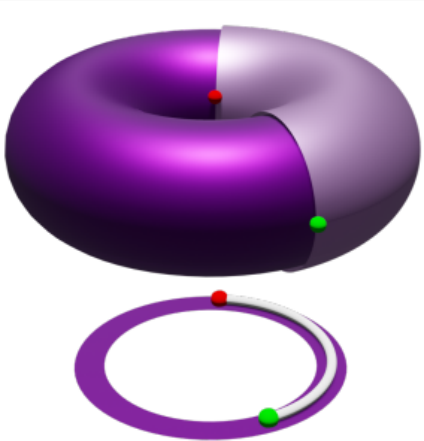
Consider two 1-d robots with state space S^1 (the circle).



Composite state space is the torus T^2.



Prioritized planning can be seen as projection T^2 down to S^1.
A path on S^1 induces a path restriction on T^2.



Composite State Space

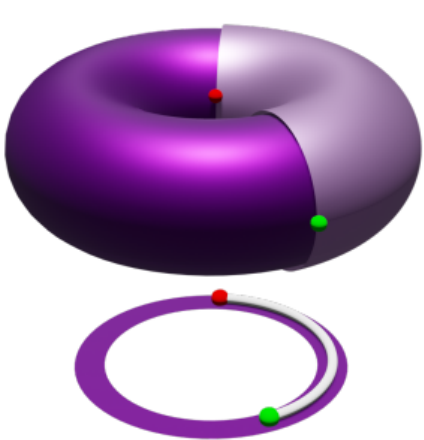
Let $X = Y \times Z$ be the (composite) state space.

Projection

A projection $\pi : X \to Y$ is a mapping from X to Y. Example: $\pi_{T^2} : (\theta_1, \theta_2) \to (\theta_1)$

Base space

Given a projection $\pi : X \to Y$, we call Y the *base space*.



Restriction

Let $X = Y \times Z$ and $\pi : X \to Y$ be a projection.

Given a subset U of Y, we call

$$\pi^{-1}(U) = \{x \in X \mid \pi(x) \in U\} \qquad (1)$$

a restriction.

Path Restriction

Let $p : [0,1] \to Y$ be a path with image $U = p([0,1]) \subset Y$.

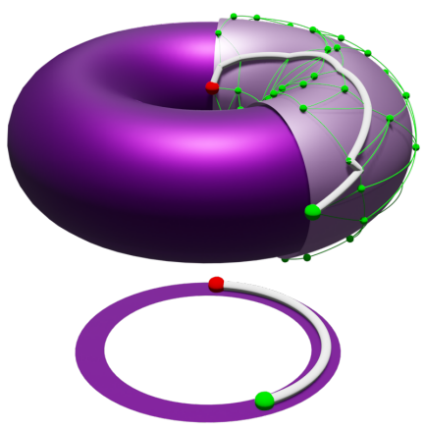
Then $\pi^{-1}(U)$ is called the *path restriction*.

Restriction sampling

Let $\pi_{T^2} : (\theta_1, \theta_2) \to (\theta_1)$ be the projection $\pi : T^2 \to S^1$, and p be a path on S^1.

Restriction sampling:

1. Sample an element y in $p([0, 1])$.
2. Sample an element of $\pi^{-1}(y)$

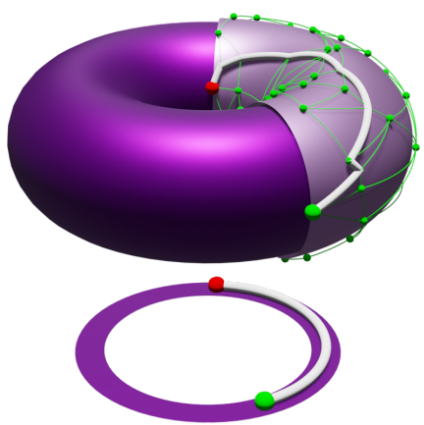


Non-Homogeneous Multi-Robot Planning

Guarantees

Probabilistic completeness?

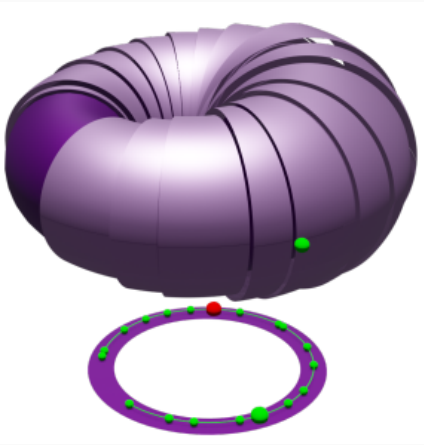
Asymptotic optimality?



Non-Homogeneous Multi-Robot Planning

Probabilistic completeness and asymptotic optimality is possible. Requires two changes.

- Replace path restriction sampling with *graph restriction sampling*
- Continue sampling on base space(s).



A Orthey, S Akbar, M Toussaint, "Multilevel Motion Planning: A Fiber Bundle Formulation", IJRR, 2023 [8]

Non-Homogeneous Multi-Robot Planning

GreedyPrioritizedPlanner(x_I, X_G, $X_1, \ldots, X_K$)

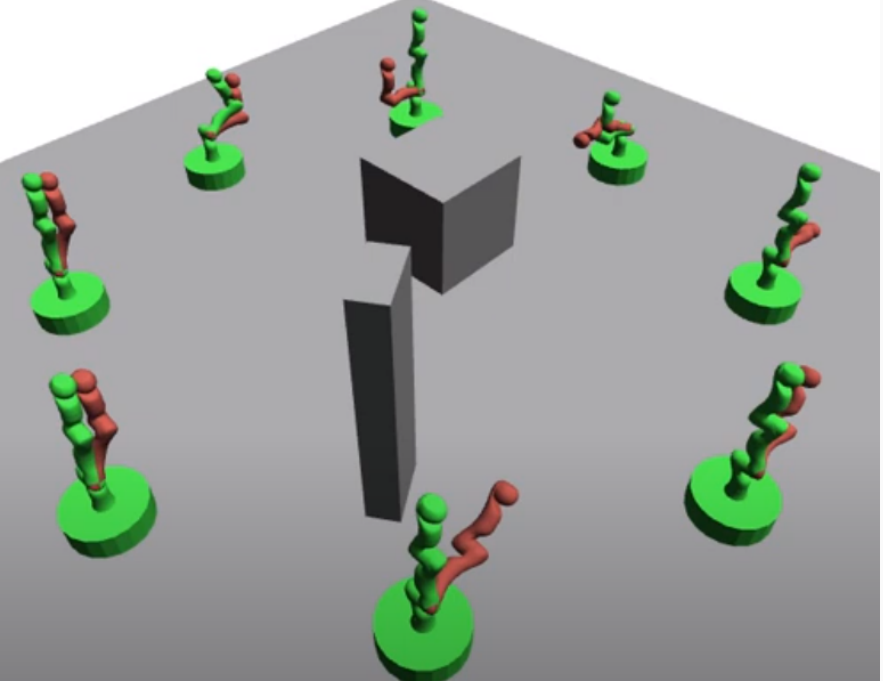
1. For $k = 1$ to K do
2. While not terminated (X_k)
3. Grow(X_k)
4. p_k = GetPath(X_k)
5. SetPathRestriction(p_k, X_{k+1})

Non-Homogeneous Multi-Robot Planning

PrioritizedPlanner(x_I, X_G, $X_1, \ldots, X_K$)

1. $Q \leftarrow \emptyset$ (priority queue)
2. For $k = 1$ to K do
3. $\quad Q$.push(X_k)
4. $\quad$ While not terminated (X_k)
5. $\quad\quad X_{select} \leftarrow Q.pop()$
6. $\quad\quad$ Grow(X_{select})
7. $\quad\quad Q$.push(X_{select})

A Orthey, M Toussaint, "Rapidly-Exploring Quotient-Space Trees: Motion Planning using Sequential Simplifications", ISRR, 2019 [**Orthey2019ISRR**]



Advantages of Prioritized Planning

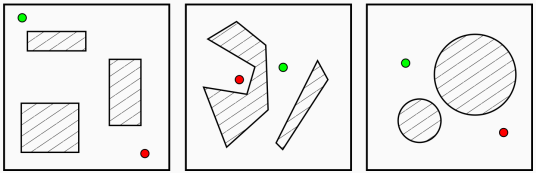
- Paths/graphs on base space as necessary condition on solution (same principle as admissible heuristics/A*)
- Fast if robots are near-decomposable (robot-robot collisions are rare)

Disadvantages of Prioritized Planning

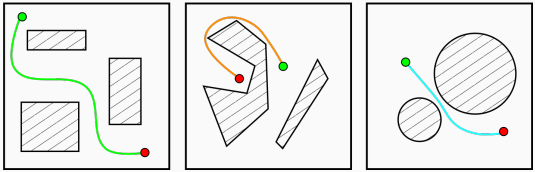
- Ordering of robots needs to be provided
- Unclear which spaces to grow first

Decomposed Multi-Robot Motion Planning

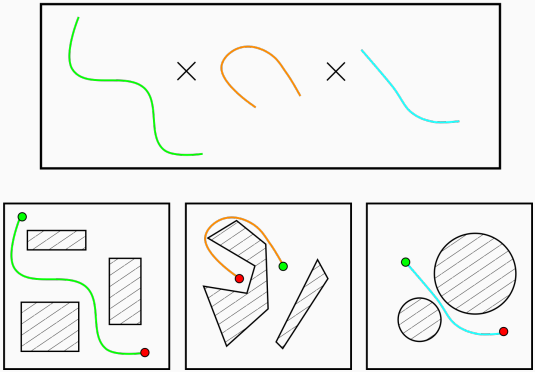
Non-Homogeneous Multi-Robot Planning: Decomposed



Non-Homogeneous Multi-Robot Planning: Decomposed



Non-Homogeneous Multi-Robot Planning: Decomposed



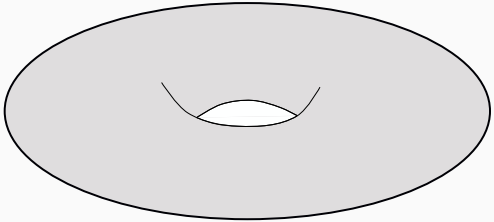
Non-Homogeneous Multi-Robot Planning: Decomposed

Intersection of path restrictions

- Let $X = Y \times Z$ be the composite state space.
- Decomposed planning: Use projections $\pi_Y : X \to Y$ and $\pi_Z : X \to Z$.
- Find path p_1 on Y, and find path p_2 on Z.
- Compute path restrictions $R_1 = \pi_Y^{-1}(p_1)$ and $R_1 = \pi_Z^{-1}(p_2)$.
- Define motion planning problem in intersection $X_R = R_1 \cap R_2$.

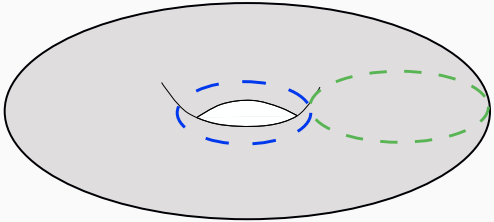
Non-Homogeneous Multi-Robot Planning: Decomposed

Example: Path restrictions on $T^2 = S_1^1 \times S_2^1$ with $\pi_1 : T^2 \to S_1^1$, and $\pi_2 : T^2 \to S_2^1$.



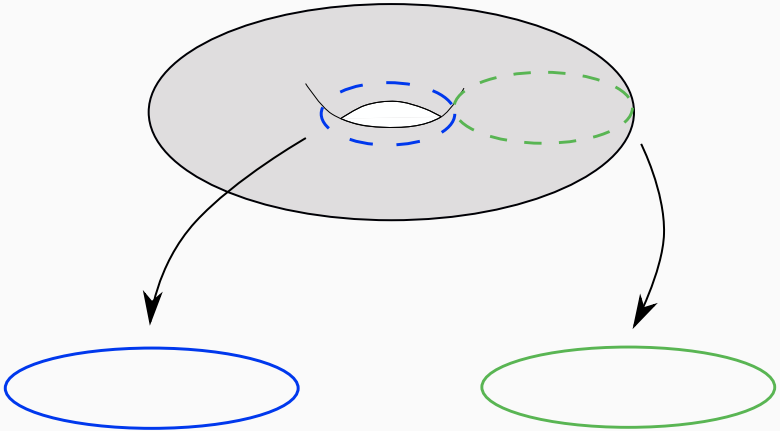
Non-Homogeneous Multi-Robot Planning: Decomposed

Example: Path restrictions on $T^2 = S_1^1 \times S_2^1$ with $\pi_1 : T^2 \to S_1^1$, and $\pi_2 : T^2 \to S_2^1$.



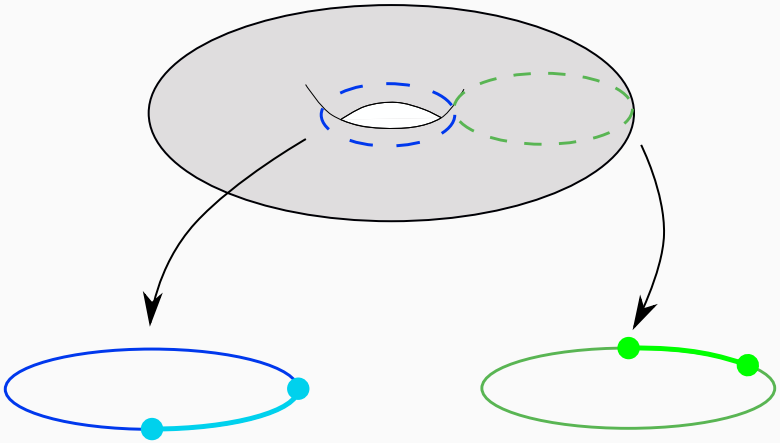
Non-Homogeneous Multi-Robot Planning: Decomposed

Example: Path restrictions on $T^2 = S_1^1 \times S_2^1$ with $\pi_1 : T^2 \to S_1^1$, and $\pi_2 : T^2 \to S_2^1$.



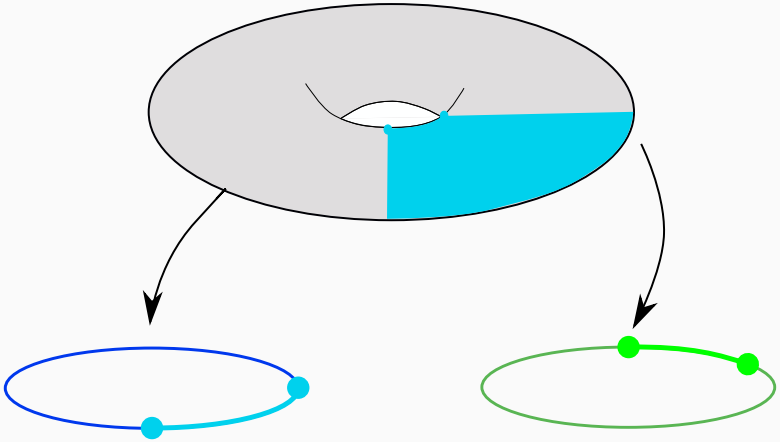
Non-Homogeneous Multi-Robot Planning: Decomposed

Example: Path restrictions on $T^2 = S_1^1 \times S_2^1$ with $\pi_1 : T^2 \to S_1^1$, and $\pi_2 : T^2 \to S_2^1$.



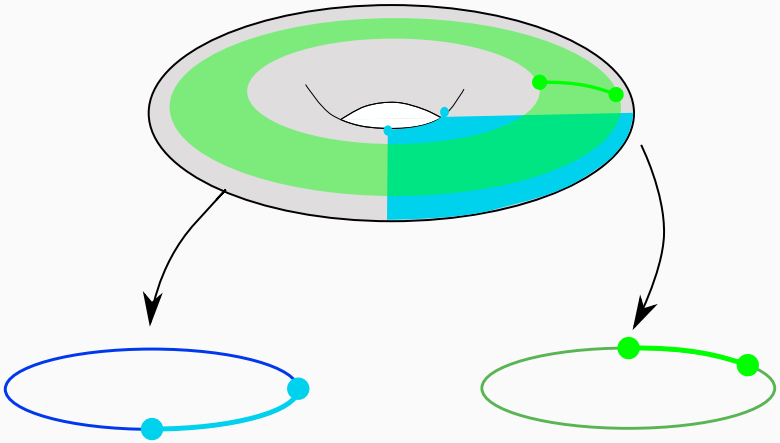
Non-Homogeneous Multi-Robot Planning: Decomposed

Example: Path restrictions on $T^2 = S_1^1 \times S_2^1$ with $\pi_1 : T^2 \to S_1^1$, and $\pi_2 : T^2 \to S_2^1$.



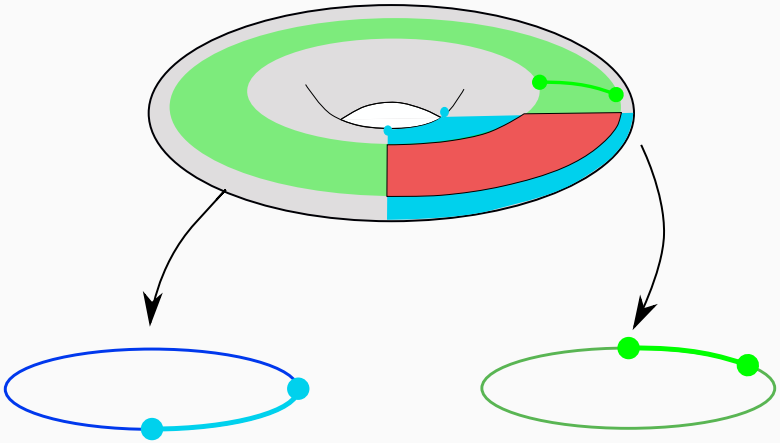
Non-Homogeneous Multi-Robot Planning: Decomposed

Example: Path restrictions on $T^2 = S_1^1 \times S_2^1$ with $\pi_1 : T^2 \to S_1^1$, and $\pi_2 : T^2 \to S_2^1$.



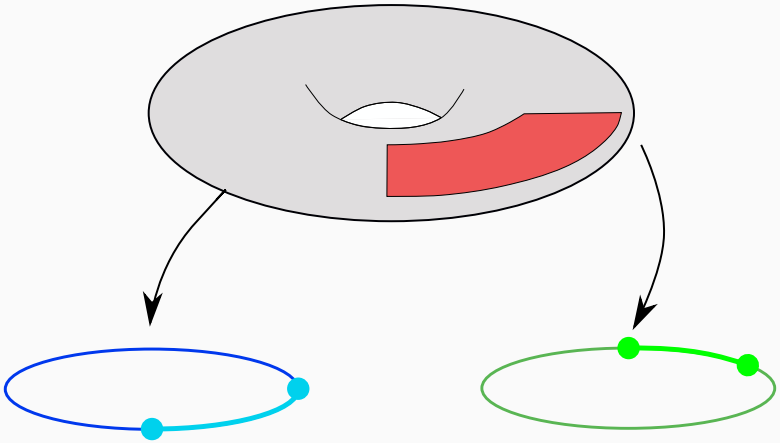
Non-Homogeneous Multi-Robot Planning: Decomposed

Example: Path restrictions on $T^2 = S_1^1 \times S_2^1$ with $\pi_1 : T^2 \to S_1^1$, and $\pi_2 : T^2 \to S_2^1$.



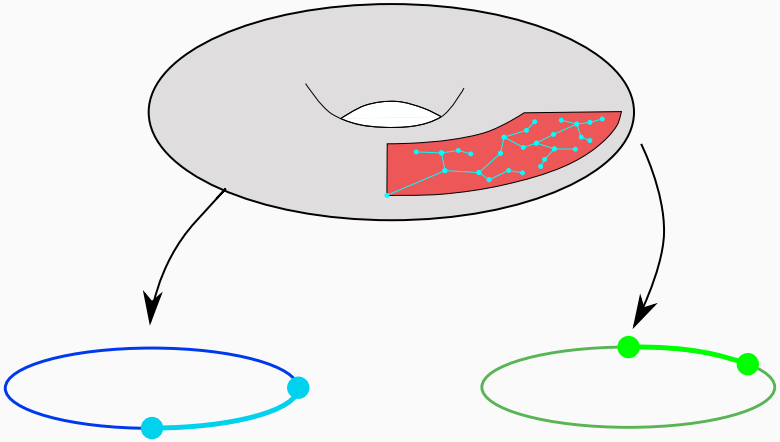
Non-Homogeneous Multi-Robot Planning: Decomposed

Example: Path restrictions on $T^2 = S_1^1 \times S_2^1$ with $\pi_1 : T^2 \to S_1^1$, and $\pi_2 : T^2 \to S_2^1$.



Non-Homogeneous Multi-Robot Planning: Decomposed

Example: Path restrictions on $T^2 = S_1^1 \times S_2^1$ with $\pi_1 : T^2 \to S_1^1$, and $\pi_2 : T^2 \to S_2^1$.



Non-Homogeneous Multi-Robot Planning: Decomposed

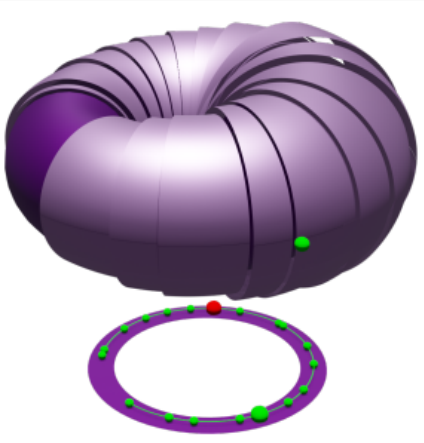
Intersection of path restrictions

- Intersection of path restrictions is equivalent to space of path reparameterizations ($s : [0, T] \rightarrow [0, 1]$).
- Finding a path over this intersection is called *path coordination* [9]

Is this optimal?

Non-Homogeneous Multi-Robot Planning: Decomposed

Adding completeness:
Replace path restriction with graph restriction.
$\Rightarrow$ *Graph coordination*



Non-Homogeneous Multi-Robot Planning: Decomposed

Decomposed planning

- Step 1: Compute individual graphs on component state spaces
- Step 2: Consider the (implicit) product of graphs on the composite state space
- Step 3: Expand edges by optimistically follow shortest paths on the component state spaces
- Step 4: If conflicts arise, backtrack around conflict areas
  - M*: Using cost-to-go estimate [10]
  - dRRT/dRRT*: Using directional oracle [11]

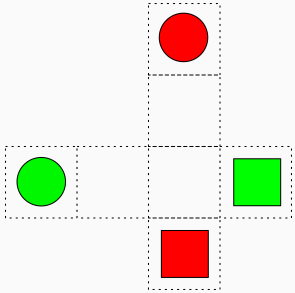
M*

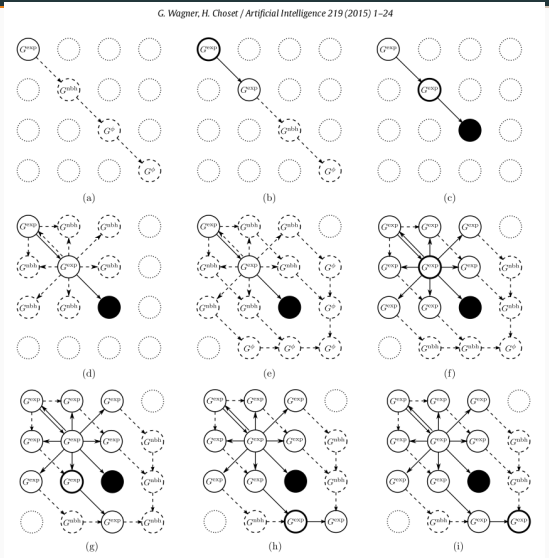
Non-Homogeneous Multi-Robot Planning: Decomposed

Conflict-resolution in graph coordination

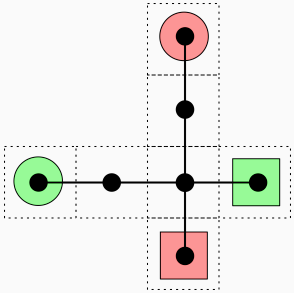
M* works like A*, using as admissible heuristic the cost-to-go of individual spaces

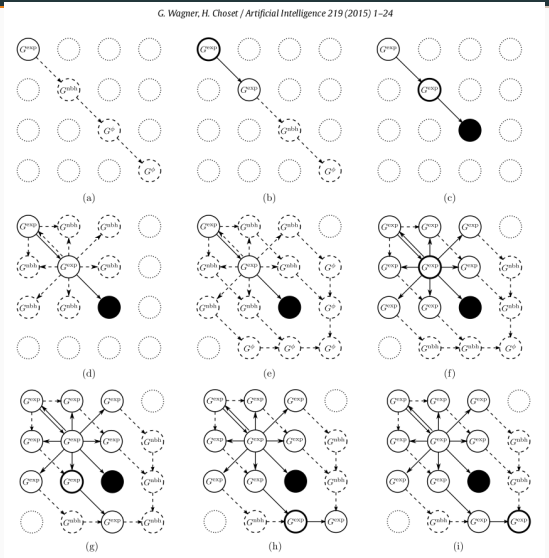
Non-Homogeneous Multi-Robot Planning: Decomposed





Non-Homogeneous Multi-Robot Planning: Decomposed





Non-Homogeneous Multi-Robot Planning: Decomposed

M*

M* is efficient, because it optimistically exploits implicit graph.

- Individual shortest paths are admissible heuristics from individual robots.
- M* combines those admissible heuristics in an optimal way

Drawback: Inefficient for higher dimensions (number of neighbors grows exponentially)

Discrete RRT (dRRT)

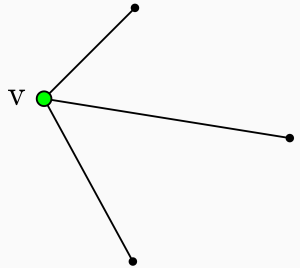
Non-Homogeneous Multi-Robot Planning: Decomposed

dRRT

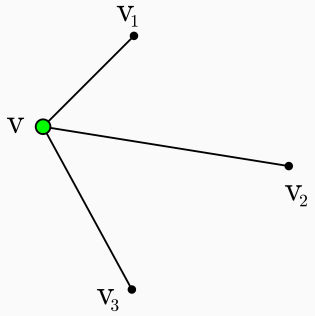
Discrete-RRT (dRRT) works like M*, but uses an oracle function to pick best neighbor

- Oracle works for euclidean spaces, i.e. planning in $\mathbb{R}^n$.
- Oracle gives you a faster (approximate) ordering of neighbors

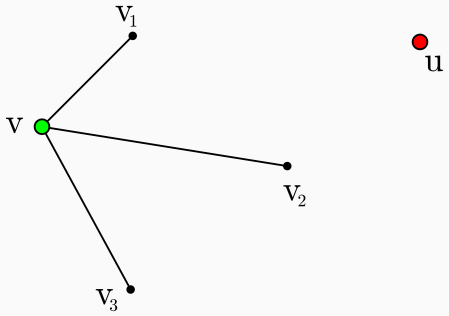
Non-Homogeneous Multi-Robot Planning: Decomposed



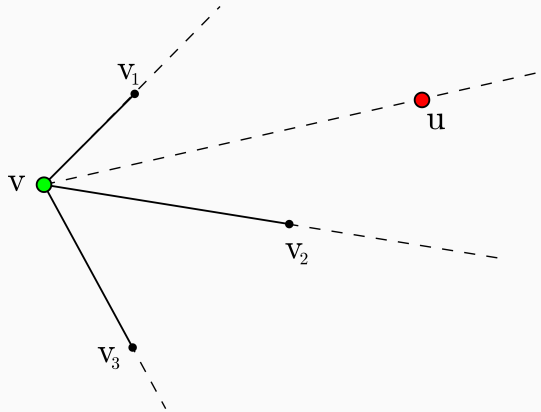
Non-Homogeneous Multi-Robot Planning: Decomposed



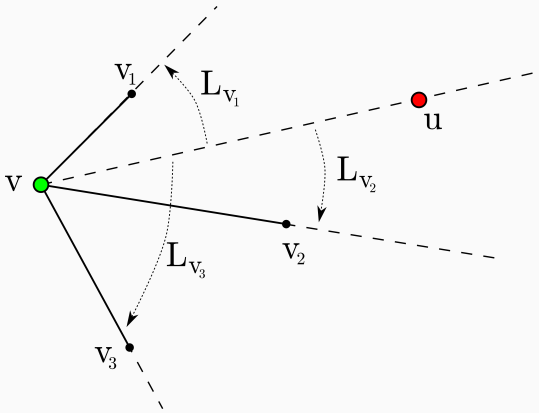
Non-Homogeneous Multi-Robot Planning: Decomposed



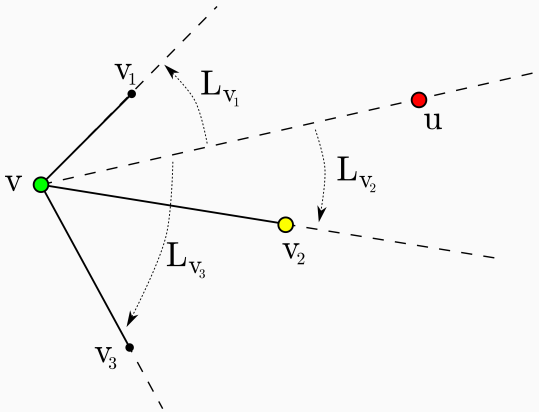
Non-Homogeneous Multi-Robot Planning: Decomposed



Non-Homogeneous Multi-Robot Planning: Decomposed



Non-Homogeneous Multi-Robot Planning: Decomposed



Non-Homogeneous Multi-Robot Planning: Decomposed

Directional oracle in dRRT

Given vertex v, neighbor edges $(v, v') \in E$, and random point u in $\mathbb{R}^n$

$$O(v, u) = \underset{v'}{\operatorname{argmin}}\{\theta_v(u, v') \mid (v, v') \in E\}$$

Summary multi-robot approaches

Multi-Robot Planning: Homogeneous planning

Homogeneous planning: Advantages

- Superior reduction if all robots are equivalent
- Conflict-based search (optimal)
- Scales well with number of robots

Homogeneous planning: Disadvantages

- Cannot be applied to non-homogeneous teams

Multi-Robot Planning: Prioritized non-homogeneous planning

Prioritized multi-robot planning

- Replaces original problems with a sequence of simpler problems
- Solutions to simpler problems provide admissible heuristics

Prioritized multi-robot planning: Disadvantages

- Requires ordering

Multi-Robot Planning: Decomposed non-homogeneous planning

Decomposed multi-robot planning

- Planning for each robot individually (could be done in parallel)
- Combine solutions into one tensor graph on composite state space
- If environment is static, graphs could be precomputed

Decomposed multi-robot planning: Disadvantages

- Individual graphs need to be dense to give good solutions

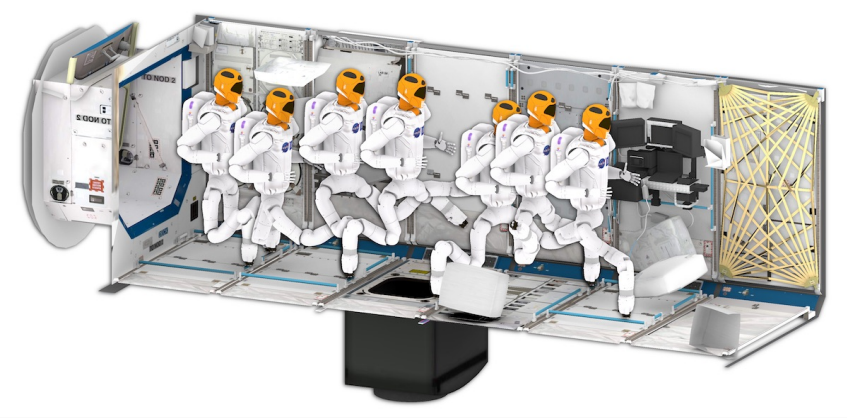
Planning with Manifold Constraints

Planning with Manifold Constraints

Manifold constraints

Constraints on a state space which remove effective degrees of freedom

Planning with Manifold Constraints: Contacts



Planning with Manifold Constraints: Grasping

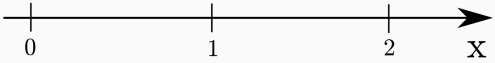
Planning with Manifold Constraints: Surface welding

Zero-measure sets

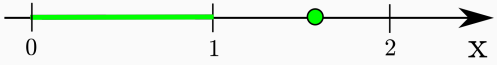
Lebesgue-measure

- Measure as generalization of volume
- Lebesgue-measure (or box measure): n-dimensional volume, corresponds to length (1D), area (2D), volume (3D).

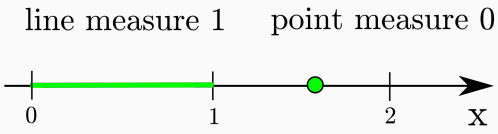
Planning with Manifold Constraints



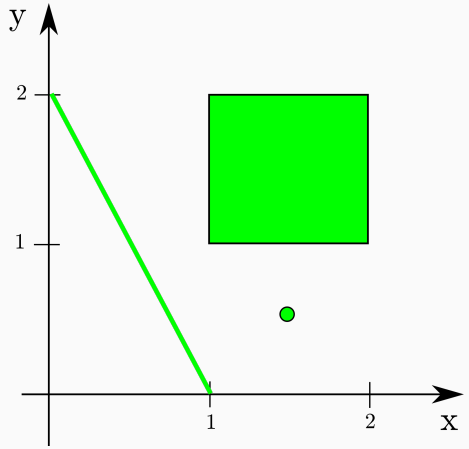
Planning with Manifold Constraints



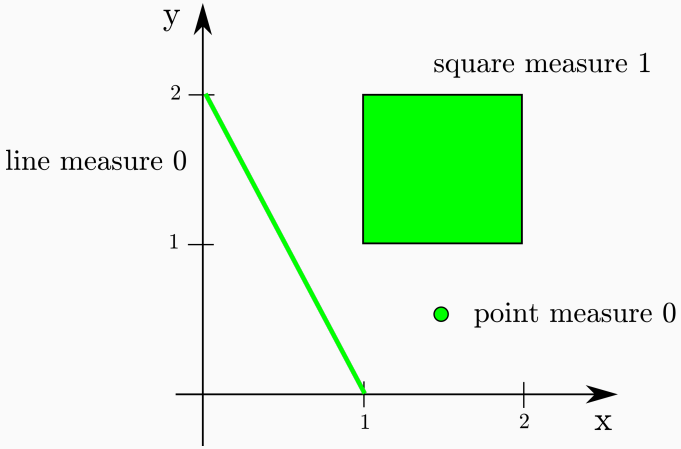
Planning with Manifold Constraints



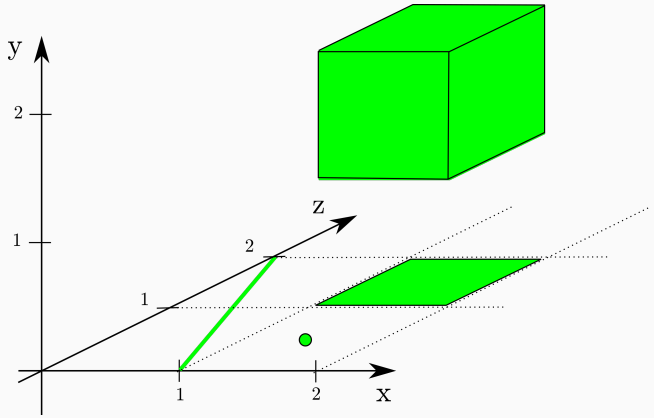
Planning with Manifold Constraints



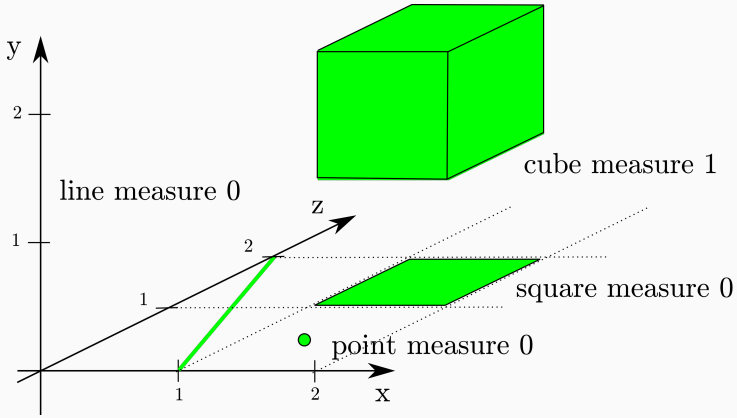
Planning with Manifold Constraints



Planning with Manifold Constraints



Planning with Manifold Constraints

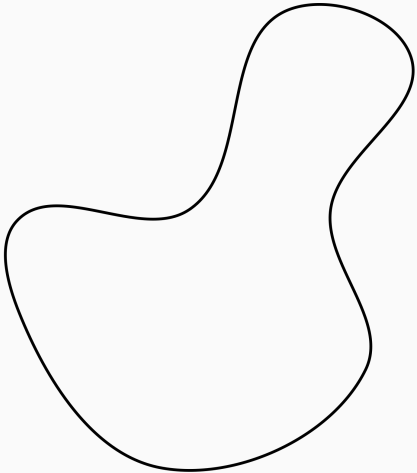


Construction Lebesgue-measure for n-dimensional box

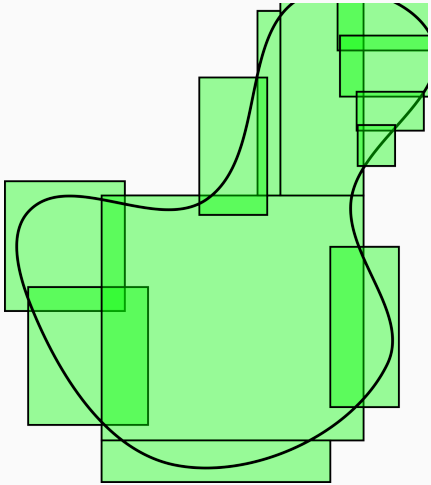
- Define an n-dimensional box $B = \prod_{i=1}^{n}[a_i, b_i]$ (this is the Cartesian product of intervals $[a_i, b_i]$, such that $a_i < b_i$)
- Define Lebesgue-measure as

$$\mu(B) = vol(B) = \prod_{i=1}^{n}(b_i - a_i). \tag{2}$$

Planning with Manifold Constraints



Planning with Manifold Constraints



Construction Lebesgue-measure

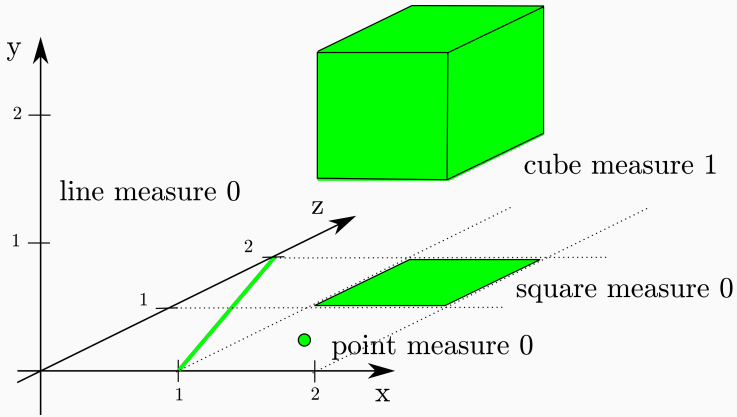
- Let $U \subset \mathbb{R}^n$ be any set.
- Define Lebesgue-measure as the minimal volume over all sets of boxes which cover U.

$$\mu(U) = vol(U) = \inf_{C} \left\{ \sum_{B \in \mathbf{C}} vol(B) \right\},$$

with $\mathbf{C}$ being a collection of boxes covering U.

Zero-measure sets

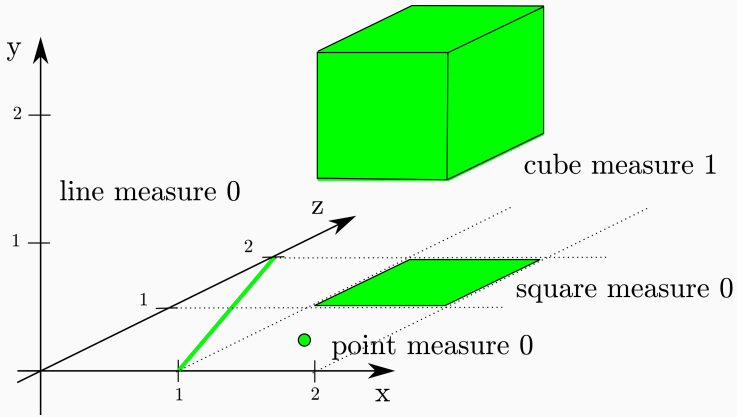
- Lebesgue measure μ is defined relative to dimensionality of space.
- Let n be the dimensionality of the state space.
- A set has zero measure if $d \leq n - 1$, whereby d is the dimension of the set (minimal number of parameters to describe it).



Zero-measure sets

- Probability of sampling in a zero-measure set is zero.

Planning with Manifold Constraints



Positive-measure sets

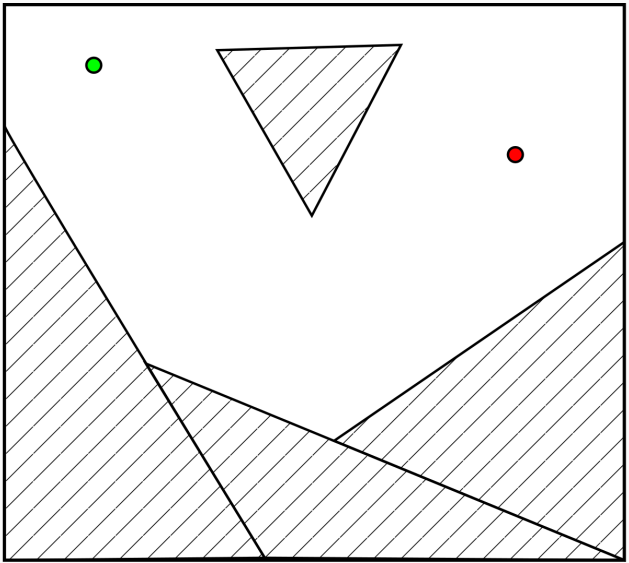
- If $U \subseteq X$ is a set with measure $\mu(U) > 0$, then the probability of sampling U with uniform sampling of X is one.

References

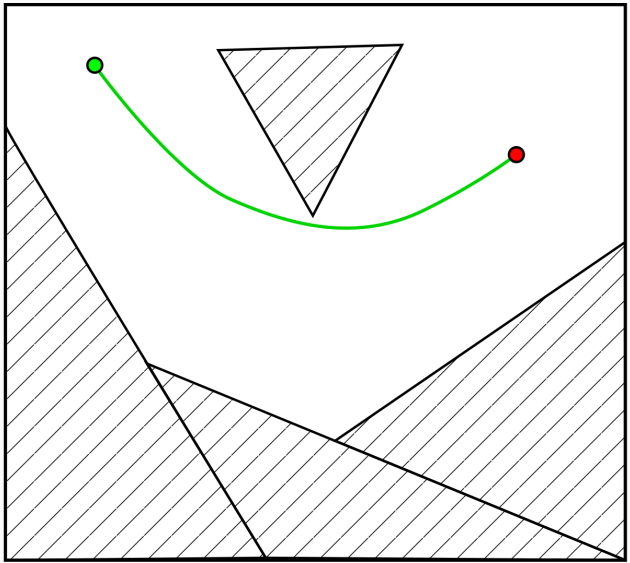
- LaValle, Planning Algorithms, 2006 (Sec. 5.1.3)
  `http://lavalle.pl/planning/node190.html`
- Measure theory `https://en.wikipedia.org/wiki/Measure_(mathematics)`

Planning with zero-measure sets

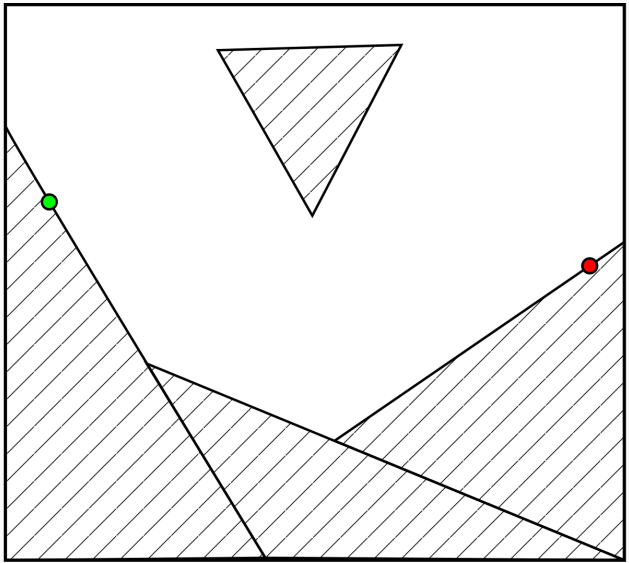
Planning with Manifold Constraints



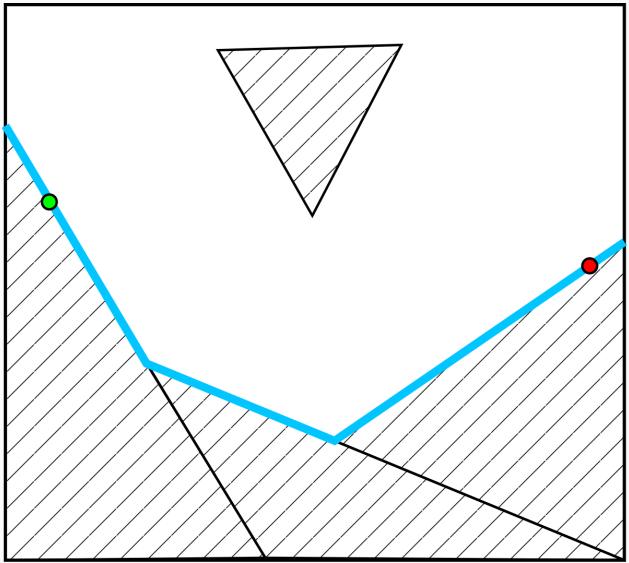
Planning with Manifold Constraints



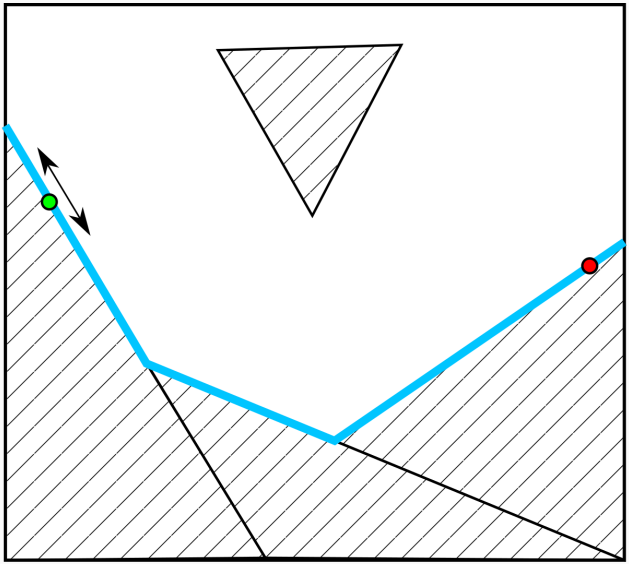
Planning with Manifold Constraints



Planning with Manifold Constraints



Planning with Manifold Constraints



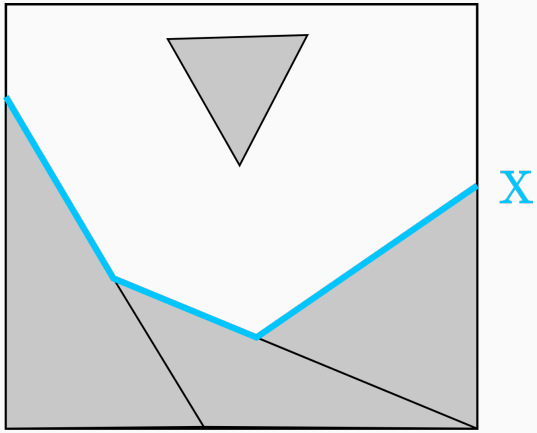
Motion Planning with Manifold Constraints

- Define constraint function $F(q) : Q \to \mathbb{R}^k$ such that $F(q) = 0$ when k constraints are fulfilled.
- This implies that there are $m = n - k$ effective degrees of freedom.
- Constraint function thus defines an m-dimensional constrained configuration space

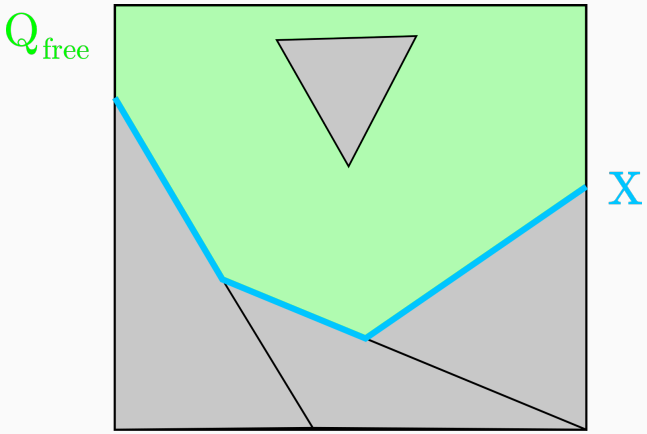
$$X = \{q \in Q \mid F(q) = 0\}$$

- State space Q is called the *ambient space* of X.

Planning with Manifold Constraints



Planning with Manifold Constraints



Planning with Manifold Constraints

Zero-measure constraints

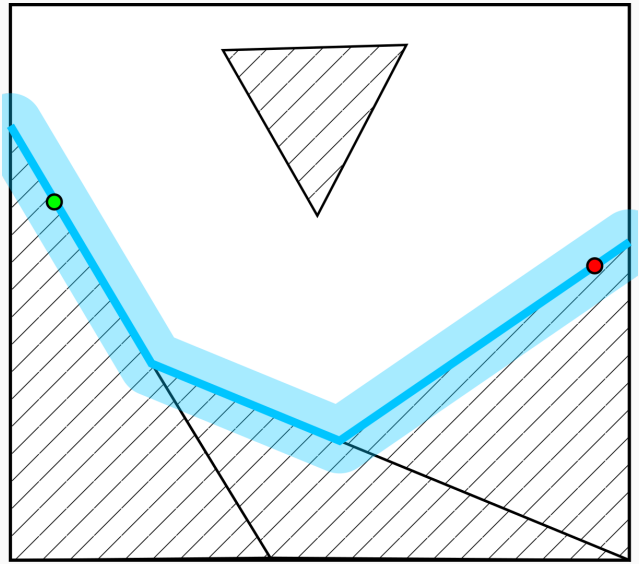
- Sampling not directly possible (zero chance to hit constraints)
- Interpolation not directly possible (will almost always move into ambient space)
- Planning with Manifold Constraints needs to address those two issues.

Planning with Manifold Constraints

Constraint planning approaches

- Relaxation-based
- Projection-based
- Continuation-based

Relaxation-based constraint planning



Relaxation-based constraint planning

- Given some $\epsilon > 0$, define relaxed configuration space

$$X_\epsilon = \{q \in Q \mid \|F(q)\| \leq \epsilon\}$$

- Rejection sampling: Uniform sampling and reject everything outside X_ϵ.

Relaxation-based constraint planning

- Given some $\epsilon > 0$, define relaxed configuration space

$$X_\epsilon = \{q \in Q \mid \|F(q)\| \leq \epsilon\}$$

- Rejection sampling: Uniform sampling and reject everything outside X_ϵ.

Interpolation?

Relaxation-based constraint planning

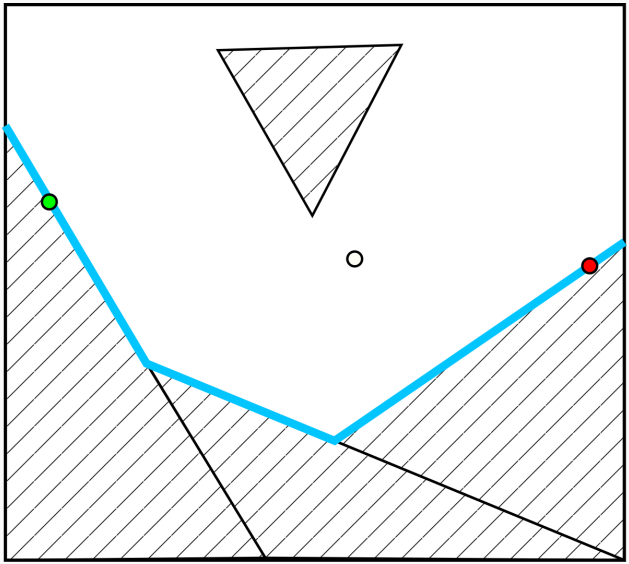
- Given some $\epsilon > 0$, define relaxed configuration space

$$X_\epsilon = \{q \in Q \mid \|F(q)\| \leq \epsilon\}$$

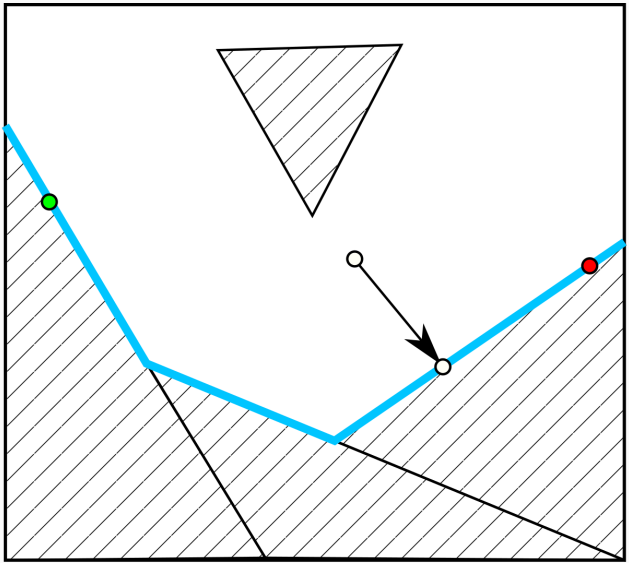
- Rejection sampling: Uniform sampling and reject everything outside X_ϵ.

Probabilistic Completeness?

Projection-based constraint planning



Projection-based constraint planning

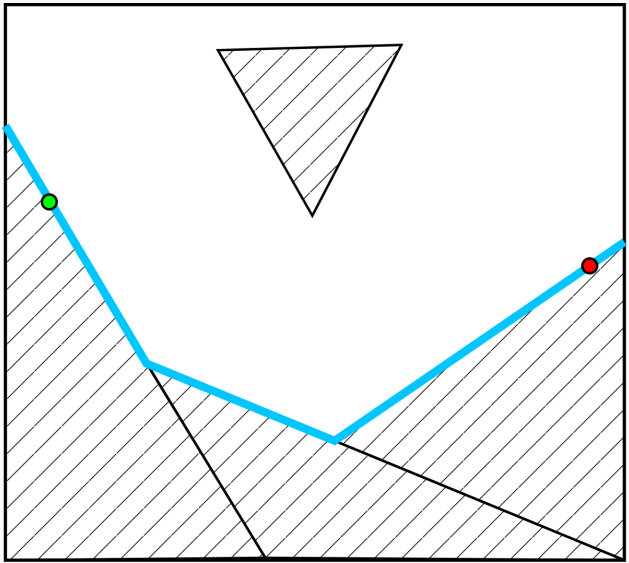


Projection-based constraint planning

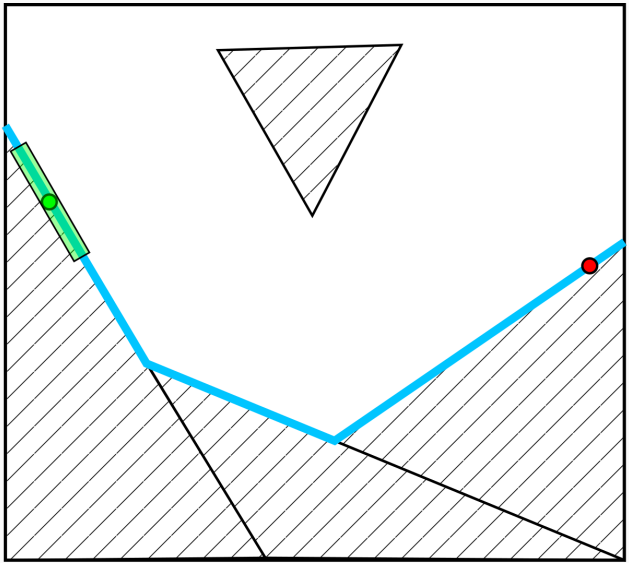
- Given constraint configuration space X and ambient space Q, define projection $\pi : Q \to X$.
- Projection sampling: Uniform sampling and projecting onto X

Interpolation?
Probabilistic Completeness? [**berenson2009manipulation**]

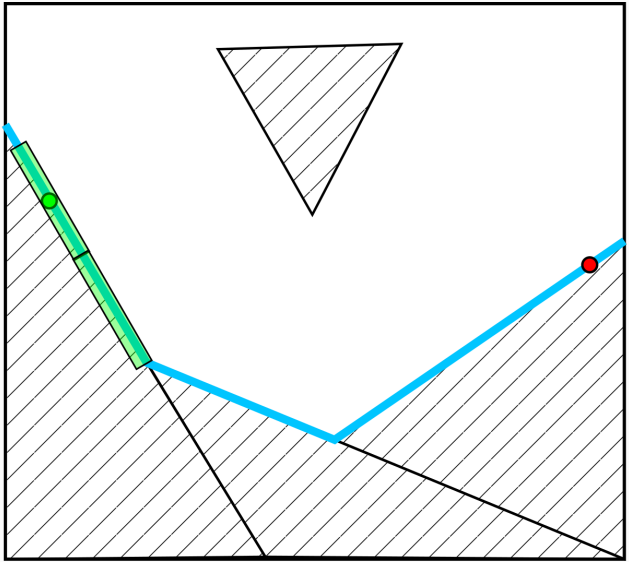
Continuation-based constraint planning



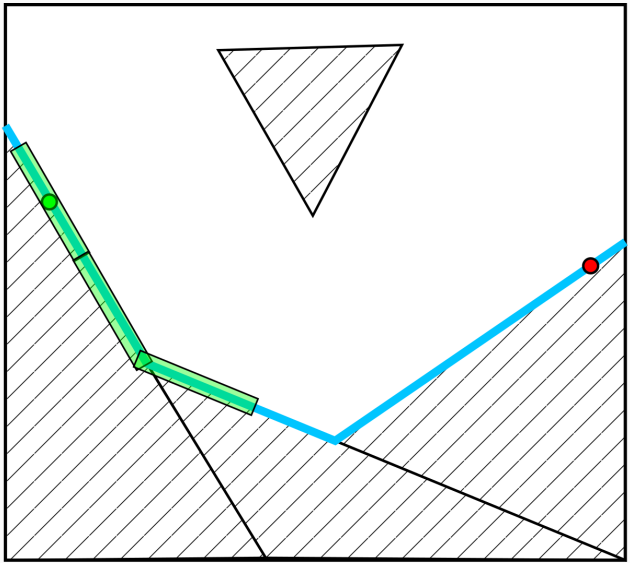
Continuation-based constraint planning



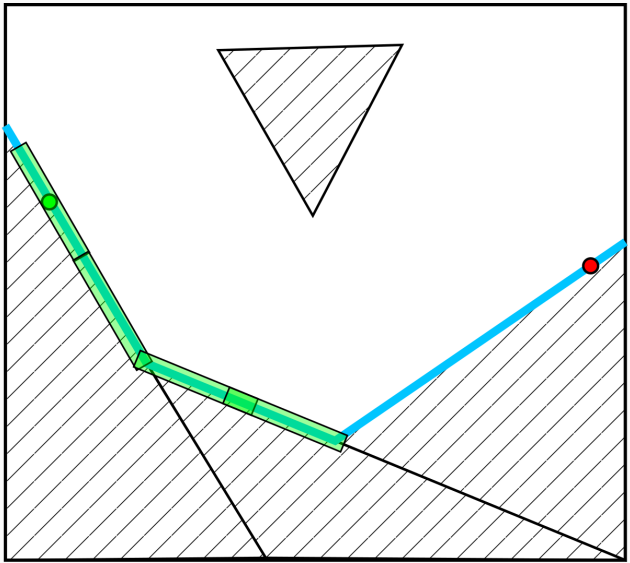
Continuation-based constraint planning



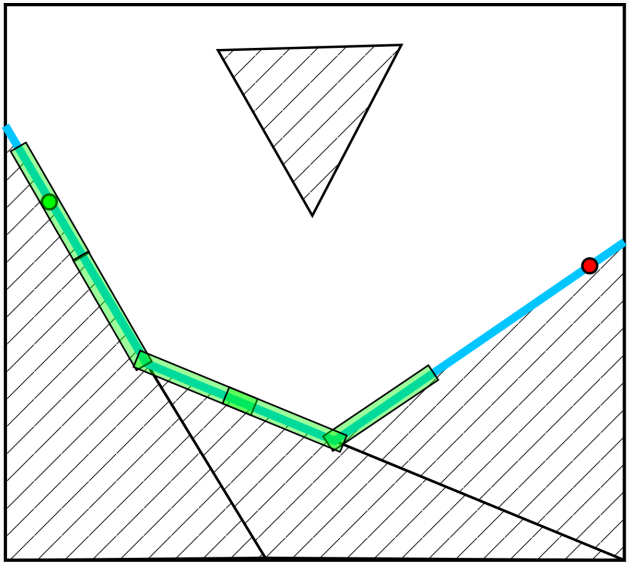
Continuation-based constraint planning



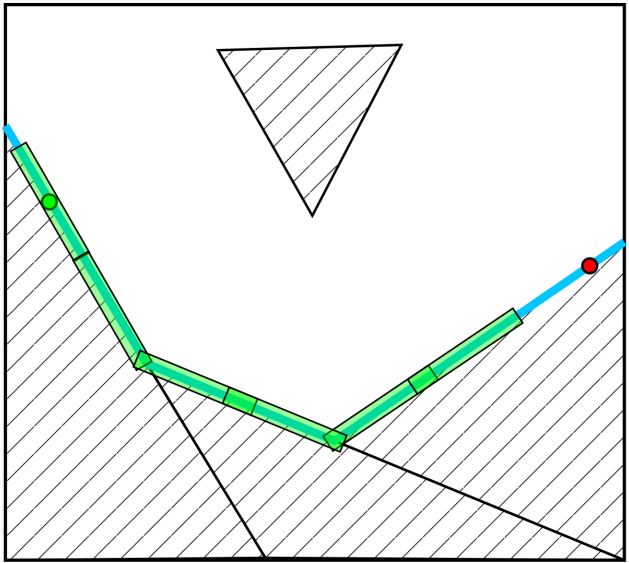
Continuation-based constraint planning



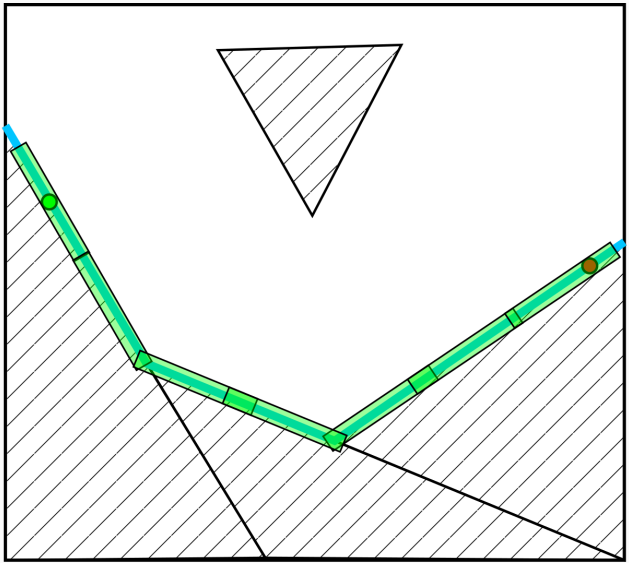
Continuation-based constraint planning



Continuation-based constraint planning



Continuation-based constraint planning

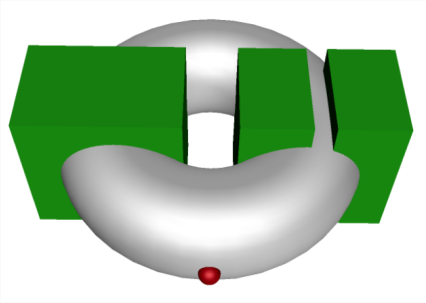


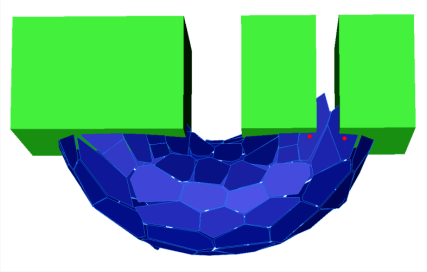
Continuation-based constraint planning

- Compute piecewise-linear approximations (a *chart*) to compute a cover (an *atlas*) on the constraint configuration space Q
- Atlas sampling

Interpolation?
Probabilistic Completeness? [**jaillet2012path**]

Continuation-based constraint planning





BSc/MSc Theses Opportunities

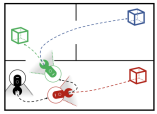
BSc/MSc Theses Opportunities

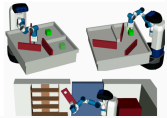
If you found this course interesting, we invite you to write a thesis with us.

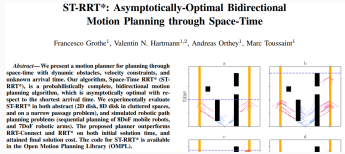
BSc/MSc Theses Opportunities

- Most important requirements: C++ or Python / Linear Algebra
- Wide variety of Topics on Robotics/Motion Planning/Machine Learning
  - Construction Assembly
  - Quadrocopter Flight and Transport
  - Solving Physical Puzzles with Robots
  - Theory of Motion Planning
- Weekly supervision
- Involved in cutting edge research
- Possibility of writing a paper

BSc/MSc Theses Opportunities (Examples papers)







- Janis Eric Freund, et al., Asymptotically Optimal Belief Space Planning in Discrete Partially-Observable Domains, IROS, 2024
- Bora Bayraktar, et al., Solving Rearrangement Puzzles using Path Defragmentation in Factored State Spaces, RAL, 2023
- Francesco Grothe, et al., ST-RRT*: Asymptotically-Optimal Bidirectional Motion Planning through Space-Time, ICRA, 2022
- Marie-Therese Khoury, et al., Efficient Sampling of Transition Constraints for Motion Planning under Sliding Contacts, CASE, 2021

Conclusion

Today

- Multi-robot planning in composite state space
- Projections as requirement for efficient planning
- Homogeneous planning: Pebbles on a graph and conflict-based search
- Non-homogeneous planning: Prioritized vs. Decomposed
- Manifold constraints and zero-measure sets

Exam next week

- Please post questions on the ISIS forum

References i

[1] John Canny. *The complexity of robot motion planning*. MIT press, 1988.

[2] Wenying Wu, Subhrajit Bhattacharya, and Amanda Prorok. "Multi-Robot Path Deconfliction through Prioritization by Path Prospects". In: *arXiv preprint arXiv:1908.02361* (2019).

[3] Daniel Kornhauser, Gary L. Miller, and Paul Spirakis. "Coordinating Pebble Motion on Graphs, The Diameter of Permutation Groups, and Applications". In: *Symposium on the Foundations of Computer Science*. IEEE. Oct. 1984, pp. 241–250.

References ii

[4] Ryan Luna and Kostas E Bekris. "Efficient and complete centralized multi-robot path planning". In: *IEEE International Conference on Intelligent Robots and Systems*. 2011, pp. 3268–3275.

[5] Jingjin Yu and Steven M LaValle. "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics". In: *IEEE Transactions on Robotics* 32.5 (2016), pp. 1163–1177.

[6] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. "Conflict-based search for optimal multi-agent pathfinding". In: *Artificial Intelligence* 219 (2015), pp. 40–66.

References iii

[7] Wolfgang Hönig, James A. Preiss, T. K. Satish Kumar, Gaurav S. Sukhatme, and Nora Ayanian. "Trajectory Planning for Quadrotor Swarms". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 856–869.

[8] Andreas Orthey, Sohaib Akbar, and Marc Toussaint. "Multilevel Motion Planning: A Fiber Bundle Formulation". In: *International Journal of Robotics Research* (2023).

[9] Thierry Siméon, Stéphane Leroy, and Jean P Laumond. "Path coordination for multiple mobile robots: A resolution-complete algorithm". In: *IEEE Transactions on Robotics and Automation* 18.1 (2002), pp. 42–49.

[10] Glenn Wagner and Howie Choset. "Subdimensional expansion for multirobot path planning". In: *Artificial Intelligence* 219 (2015), pp. 1–24.

[11] Kiril Solovey, Oren Salzman, and Dan Halperin. “Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning”. In: *International Journal of Robotics Research* 35.5 (2016), pp. 501–513.