Motion Planning via Manifold Samples (MMS)

Oren Salzman

Outline

- Background
- Hybrid Motion Planners
- MMS
- Experimental results
- Technical Details

Outline

- Background
- Hybrid Motion Planners
- MMS
- Experimental results
- Technical Details

Motion Planning - Definitions

- **Workspace** – A description of the (2D or 3D) world consisting of a **robot** and **obstacles**
- **Configuration Space-** The space of parameters that define the robot's location and orientation in the workspace
- **Degrees of Freedom-** The minimal number of parameters required to uniquely define a position of the robot.
- **Free Space** (C_{free})- Set of valid configurations
- **Forbidden Space** (C_{forb})**-** Set of invalid configurations

Probabilistic Planning

- Capture the connectivity of the configuration space by **sampling** configurations.
- Connect samples by **local planning**

✓ Applicable for **high dimensions**
✗ Sensitive to **tight passages**

PRM – Probabilistic Roadmap

- Multi query planner
- Preprocesses configuration space into a graph (roadmap)
  - Randomly sample n valid robot configurations ("milestones")
  - Connect close-by configurations by dense sampling ("local-planning")
  - Discard invalid edges

Combinatorial Motion Planning

- Critical Curves
  - Consider the movement of a robot while a feature of the robot is in contact with a feature of an obstacle
  - The movement defines a curve in the configuration space
  - The curve is a potential transitions between C_{free} and C_{forb}
  - Inserting the curves into an arrangement yields a subdivision of cells, each completely in C_{free} or C_{forb}
- Minkowski Sums
  - Allow representation of the configuration space of a translating robot

Related Work

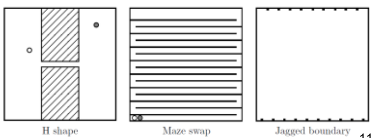
- **Probabilistic Planning**
  - Kavraki, L.E., Svestka, P., Latombe, J.C., Overmars, M.: *Probabilistic roadmaps for path planning in high dimensional configuration spaces*
  - Lavalle, S.M.: Rapidly-exploring random trees: *A new tool for path planning.*
  - Hsu, D., Latombe, J.C., Motwani, R.: *Path planning in expansive configuration spaces*
- **Combinatorial Motion Planning**
  - Reif, J.H.: *Complexity of the mover's problem and generalizations.*
  - Schwartz, J.T., Sharir, M.: *On the "piano movers" problem. II. general techniques for computing topological properties of real algebraic manifolds.*

Outline

- Background
- Hybrid Motion Planners
- MMS
- Experimental results
- Technical Details

Coordinating Two disks

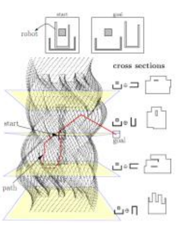
- S. Hirsch and D. Halperin. Hybrid motion planning: Coordinating two discs moving among polygonal obstacles in the plane. WAFR 2002 [HH02]
- Compute the configuration space explicitly for each disk
- Distinguish cell pairs as colliding with obstacles, inter-colliding and free
- Add PRM sampling within inter-colliding cells



*Figures taken from [HH02]

Hybrid Motion Planning Using Minkowski Sums

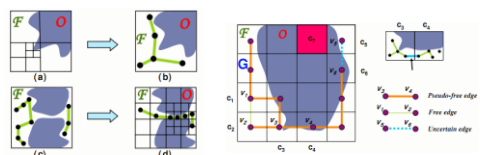
- Ming Lien, J.: *Hybrid motion planning using Minkowski sums. RSS 2008 [Lie08]*
- Connect a dense set of near-by configuration space slices.
- Each slice is decomposed to free and forbidden cells
- Adjacent slices are connected in an inexact manner by heuristic interpolation and local-planning



*Figures taken from [Lie08]

ACD & PRM

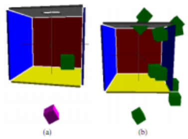
- Liangjun Zhang, Young J. Kim, and Dinesh Manocha. A hybrid approach for complete motion planning. IROS 2007 [ZKM07]
- Divide space to free, forbidden and mixed cells using PRM in conjunction with approximate cell decomposition



*Figures taken from [ZKM07]

RRT with 3DoF Local Planner

- Jade Yang and Elisha Sacks. RRT path planner with 3 DOF local planner. ICRA, 2006 [YS06]
- 6 dof RRT planner is presented with a 3 dof local planner hybridizing probabilistic, heuristic and deterministic methods



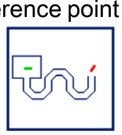
*Figures taken from [YS06]

Outline

- Background
- Hybrid Motion Planners
- MMS
- Experimental results
- Technical Details

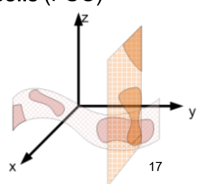
The Setting

- Two dimensional polygonal robot R translating and rotating in the plane amidst polygonal obstacles.
- Both the robot and the obstacles are not restricted to be convex polygons.
- The robot is represented by a reference point located at its center of mass



Overview

- Multi-query planner
- Preprocessing stage constructs a graph capturing the connectivity of C using **manifolds** as **samples.**
  - The manifolds are decomposed into cells in C_{free} and C_{forb}
  - A cell in C_{free} is a **free space cells** (FSC)
  - FSCs are **nodes** in the connectivity graph.
  - Intersecting FSCs induce **edges** in the connectivity graph.

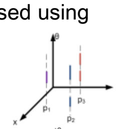


Overview (cont.)

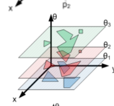
- What types of manifolds do we consider?
- How do we choose a manifold sample?
- How do we compute a path?

Constraints and Manifolds

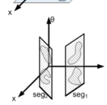
- We consider manifolds defined by **constraints** on the general position of the robot
- Manifolds are constructed and decomposed using **primitives**
- Constraints:
  - **Point constraint** - The robot is free to rotate about a fixed reference point in the workspace



  - **Rotation constraint** - The robot is free to translate anywhere within the workspace, but its rotation angle remains fixed

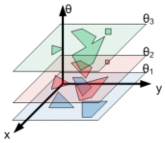


  - **Segment constraint** - The robot is free to rotate, but it can only translate along a given workspace segment



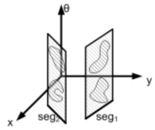
Primitives – Rotation Primitive

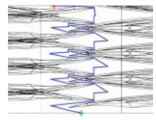
- P_θ -the rotation primitive for the constraining angle θ
- Constructed by computing the Minkowski sum of $-R_\theta$ (R rotated by θ and reflected on the origin) with the obstacles
- "Layer" – the computation of P_θ at a fixed angle θ



Primitives – Segment Primitive

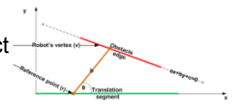
- P_s -the segment primitive for the constraining segment s
- Constructed by computing an **arrangement** of **critical curves**
  - Each critical curves is a potential transition between C_{free} and C_{forb}
  - Inserting the curves into an arrangement yields a subdivision of cells, each completely in C_{free} or C_{forb}



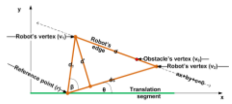


Primitives – Segment Primitive

- Critical curves Types:
  - Robot's vertex in contact with an obstacle's edge



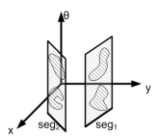
  - Robot's edge in contact with an obstacle's vertex



- Critical curves are rational functions
  - $\alpha = \frac{x_r - x_s}{x_t - x_s}$
  - $T = \tan(\frac{\theta}{2})$ $\alpha = \frac{p_2 T^2 + p_1 + p_0}{q_2 T^2 + q_1 + q_0}$

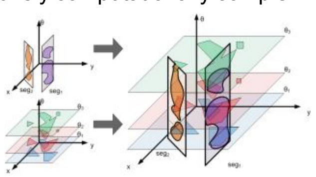
Primitives – Point Primitive

- P_p -the point primitive for the constraining point p
- Identical to the segment primitive where the source and target defining the segment are identical



Primitives – Summary

- Rotation primitive
  - Using Minkowski sums,
  - Relatively "cheap" to compute
- Segment primitive
  - Using arrangement of rational functions
  - Relatively computationally complex



Updating The Connectivity Graph

- Definitions and Notations
  - Ψ - a family of **constraints** (point, segment, angle)
  - m_ψ – a **manifold** defined by a constraint ψ in Ψ
  - P_ψ – a **Ψ-primitive** applied to an element ψ in Ψ is the **construction** of the manifold m_ψ and its **decomposition** into FSCs

Algorithm 1 Update Connectivity Graph ($\psi_m \in \Psi$,V,E)

1: $FSC_m \leftarrow P_\Psi(\psi_m)$
2: $E \leftarrow E \cup \{fsc_1, fsc_2 \mid fsc_1 \in V, fsc_2 \in FSC_m, fsc_1 \cap fsc_2 \neq \emptyset\}$
3: $V \leftarrow V \cup \{fsc \mid fsc \in FSC_m\}$

Generating Manifolds - Choosing Constraints

- Recap – Applying angle primitive is less time consuming than applying segment primitives
- Algorithm has two parameters –
  - n_θ – number of angles to be generated
  - n_s – number of segments to be generated
- Rotation angles (naïve)
  - Subdivide [-π..π] to angles every 360 / n_θ degrees
  - For each angle, snap to the closest rational angle
  - Construct the rotation primitive
- Segment primitive
  - Generate a segment
    - Randomly or use heuristic
  - Filter
  - Construct the segment primitive

27

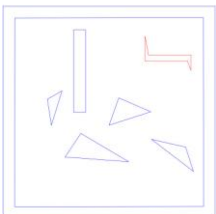
Generating Segments

- Region of Interest - RoI
  - Considering the segment primitive in a subset of the range of rotations
  - results in a somewhat “weaker" yet faster primitive than considering the whole range
- Motivation
  - Cells are “similar” in close layers
  - “Large” cells tend to overlap in distant layers
  - “Small” cells may “shift” in direction between close layers

28

Generating Segments (cont)

- Demonstrating motivation:
  - Scene:



  - Layers at consecutive rotations: Minkowski_slices.avi

29

Generating Segments (cont)

- Small cell heuristic
  - Choose an FSC
  - Identify the appropriate FSC in the next layer (or previous one)
  - The segment is chosen by connecting random points from each FSC
  - Used with a “small” RoI
- Large cell heuristic
  - Choose an FSC
  - Randomly choose a segment in the FSC
  - Used with a “large” RoI

Algorithm 4 Generating segment

```
p1 ← random_num(0...1)
if p1 ≥ random_segment_probability then
  return random_segment()
else
  fsc ← random(cell)
  α ← size(fsc)
  p2 ← random_num(0...1)
  if p2 ≤ α then
    return small_cell_heuristic(fsc)
  else
    return large_cell_heuristic(fsc)
  end if
end if
```

30

Connected Components Filtering

- **Motivation -** constructing and decomposing **only** manifolds that **may** connect different connected components of the connectivity graph
- A “candidate" segment s is first checked against all existing FSCs that may intersect it

Algorithm 3 Filter segment primitive (s,RoI)

```
cc_ids ← ∅
for all θ ∈ RoI do
  for all fsc ∈ P_θ do
    if fsc ∩ s ≠ ∅ then
      cc_ids ← cc_ids ∪ connected_component_id(fsc)
    end if
    if |cc_ids| > 1 then
      return do not filter
    end if
  end for
end for
return filter
```

- If all FSCs are in the same connected component in the roadmap, s can be discarded

31

Putting it all together

- Exploration phase (nθ times)
  - Generate angle
  - Construct layer
- Connection Phase (ns times)
  - Generate segment
  - If not filtered
    - Apply predicate
- Query Phase
  - Add layers containing source & target configurations
  - Search connectivity graph
  - Constructing a path requires path planning in each FSC (easy?)

32

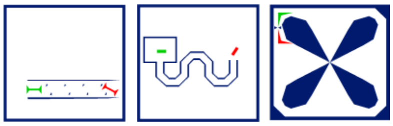
Outline

- Background
- Hybrid Motion Planners
- MMS
- Experimental results
- Technical Details

Experimental Results

- Scenarios

Tunnel Snake Flower



Experimental Results

- Optimization efficiency (flower scenario)

Scenario	Choosing segments	Filtering	n_θ	n_s	t
Flower	random	not used	20	8192	1418
		used	20	8192	112
	heuristic	not used	40	512	103
		used	20	1024	20

- Parameter Sensitivity (flower scenario)

		n_θ			
		10	20	40	90
n_s	256	(6,×)	(11,×)	(12,×)	(16,×)
	512	(7,×)	(13,×)	(14,×)	(25,×)
	1024	(16,×)	(20,✓)	(23,✓)	(35,✓)
	2048	(30,×)	(35,✓)	(38,✓)	(51,✓)

Experimental Results

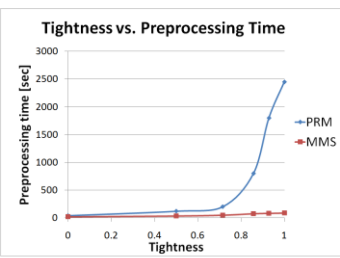
- Comparison with PRM
- OOPSMP PRM Implementation
  - http://www.kavrakilab.org/OOPSMP/index.html

	PRM	Our Scheme
Sample type	Point	Manifold
Node type	Point	FSCs
Node connection method	Local Planner	Intersecting FSCs
Data structure	Connected paths	Connected cells

Scenario	SP			PRM			Speedup
	n_θ	n_s	t	k	% sampling time	t	
Tunnel	20	512	100	20	0.0125	180	1.8
Snake	40	256	22	20	0.025	140	6.3
Flower	20	1024	20	24	0.0125	40	2

Experimental Results

- Tightening the configuration space



Outline

- Background
- Hybrid Motion Planners
- MMS
- Experimental Results
- Technical Details – Critical Curves

Notation

- Robot R – simple polygon with vertices $\{v_1, \ldots, v_n\}$

Where $v_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$ and edges $\{(v_1, v_2), \ldots (v_n, v_1)\}$

- The reference point of R is located at the origin
- A configuration $q = (r_q, \theta_q)$ where q maps a vertex v_i by the following mapping:

$$v_i(q) = M(\theta_q) v_i + r_q$$

Where $M(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ is the rotation matrix.

Parametrization

- Given a segment $seg = [s, t]$ where $s = \begin{pmatrix} x_s \\ y_s \end{pmatrix}$

and $t = \begin{pmatrix} x_t \\ y_t \end{pmatrix}$ define the parameterization

$(\alpha, T) \in [0, 1] \times \mathbb{R}$ as

$$r_q = (1-\alpha)s + \alpha t,$$
$$\theta_q = 2 \arctan T.$$

Hence $M(T) = \frac{1}{1+T^2} \begin{bmatrix} 1-T^2 & -2T \\ 2T & 1-T^2 \end{bmatrix}$

Robot's Vertex – Obstacle's edge

- Robot vertex: $v_i(\alpha, T) = M(T)v_i + (1-\alpha)s + \alpha t.$
- Obstacle edge: $e = (v_{o_1}, v_{o_2})$ supported by $l : ax + by + c = 0$
- Critical curve: $v_i(q) \in l$
- Hence,

$$a[(1-T^2)x_i - 2Ty_i] + a(1+T^2)(1-\alpha)x_s + a(1+T^2)\alpha x_t +$$
$$b[(2T)x_i + (1-T^2)y_i] + b(1+T^2)(1-\alpha)y_s + b(1+T^2)\alpha y_t +$$
$$c(1+T^2) = 0,$$

- Finally:

$$\alpha = \frac{p_2 T^2 + p_1 T + p_0}{q_2 T^2 + q_0}$$

$p_2 = a(x_i - x_s) + b(y_i - y_s) - c\ ,$ $q_2 = a(x_t - x_s) + b(y_t - y_s)\ ,$ $a = y_{o_1} - y_{o_2}\ ,$

$p_1 = 2(ay_i - bx_i)\ ,$ $b = x_{o_2} - x_{o_1}\ ,$

$p_0 = -a(x_i + x_s) - b(y_i + y_s) - c\ ,$ $q_0 = q_2\ ,$ $c = x_{o_1} y_{o_2} - x_{o_2} y_{o_1}$

Outline

- Background
- Hybrid Motion Planners
- MMS
- Experimental Results
- Technical Details – Rational Function Traits

General

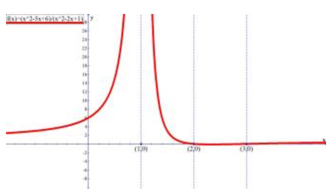
- Class supplying curve-dedicated predicates to the generic arrangement algorithm
- Predicates supplied include:

Predicate	Description
`Intersect_2`	Computes the intersections of two curves. The intersections are either an intersection point or curve, representing an overlapping subcurve of the two curves.
`Compare_x_2`	Compares two points according to their x-coordinate
`Compare_y_at_x_2`	Compares the y-coordinates of a point and a curve at the point's x-coordinate

- Existing traits for rational functions
  - Based on CORE
  - Too slow

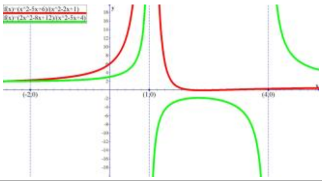
Rational Function

- Fundamental data structure
- Computes and stores a subdivision of the x-axis into intervals that do not change sign.



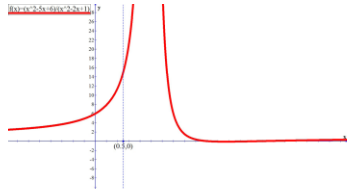
Rational Function Pair

- Fundamental data structure
- Computes and stores a subdivision of the x-axis into intervals where the functions does not relative position.



Point

- Y-coordinate is never needed explicitly
- Point is represented as an x-coordinate and a rational function



Code Optimizations

- Handles
- Caching
- Canonicalization of pairs

Experimental results

n, m	#(V,E,F)	rational	**algebraic**		μs per vertex
2,10	(98, 220, 123)	0.01	0.12	0.16	1275
2,20	(360, 766, 407)	0.07	0.35	0.22	977
2,30	(900, 1862, 963)	0.2	0.75	0.26	837
2,40	(1836, 3766, 1931)	0.4	1.47	0.27	805
2,50	(2306, 4724, 2419)	0.55	1.92	0.28	836
2,60	(3676, 7486, 3811)	0.87	2.9	0.3	790
2,70	(5066, 10298, 5233)	1.21	4.05	0.29	800
2,80	(6110, 12394, 6285)	1.5	4.93	0.3	807
2,90	(7828, 15868, 8041)	1.93	6.37	0.3	814
2,100	(8918, 18056, 9139)	2.28	7.19	0.31	806
6,10	(114, 254, 141)	0.05	0.24	0.22	2148
6,20	(566, 1188, 623)	0.28	0.97	0.28	1724
6,30	(1222, 2526, 1305)	0.63	2.09	0.3	1710
6,40	(2074, 4266, 2193)	1.14	3.63	0.31	1751
6,50	(3268, 6670, 3403)	1.89	5.7	0.33	1744
6,60	(4800, 9764, 4965)	2.8	8.24	0.33	1718
6,70	(7020, 14238, 7219)	4.06	11.78	0.34	1678
6,80	(8636, 17482, 8847)	5.21	14.59	0.35	1690
6,90	(11266, 22790, 11525)	6.87	19.15	0.35	1700
6,100	(13124, 26534, 13411)	8.2	22.85	0.35	1741

Questions?