Motion Planning Lecture 12

Optimization Wrap-Up and Method Comparison

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

July 10, 2024

Recap: Optimization-Based Methods

- Gradients to find local minima efficiently (e.g., CHOMP)
- Convex Programming: global minimum, some variants (QP, LP) can be solved in polynomial time
- Splines for geometric motion planning
- Differential Flatness: Find optimal solutions to kinodynamic motion planning with geometric reasoning efficiently
- Sequential Convex Programming (SCP): Use the benefits of convex programming for non-convex problems by convexifying constraints and/or objective

More Optimization

Kth Order Markov Optimization (KOMO) [1] (1)

- Assumption: Cost/constraints depend only on the last k configurations
- Use configurations as decision variables, only
  - Implicit Euler integration to includes derivatives like velocity or accelerations
- Define a nonlinear program (NLP) to solve

kth order assumption makes matrices sparse and nonlinear optimization efficient

Kth Order Markov Optimization (KOMO) [1] (2)

KOMO

Discretize configuration/state: $\mathbf{x}_0, \ldots, \mathbf{x}_T$

$$\begin{aligned} \text{minimize } & \sum_{t=0}^{T} \mathbf{f}_t(\mathbf{x}_{t-k:t}) \\ \text{subject to } & \mathbf{g}_t(\mathbf{x}_{t-k:t}) \leq 0 \\ & \mathbf{h}_t(\mathbf{x}_{t-k:t}) = 0 \end{aligned}$$

with $\mathbf{x}_{t-k:t} = (\mathbf{x}_{t-k}, \ldots, \mathbf{x}_t)$ being tuples of $k+1$ consecutive states.

2D Single Integrator

State: $\mathbf{x}_t = (x_t, y_t)$

Velocity can be approximated as $(\frac{x_t - x_{t-1}}{\Delta t}, \frac{y_t - y_{t-1}}{\Delta t})$ (i.e., $k = 1$)

$$\begin{aligned} \text{minimize } & \sum_{t=0}^{T} \|\mathbf{x}_t - \mathbf{x}_{t-1}\|_2^2 \\ \text{s.t.} & \|(\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t\|_2 - v_{max} \leq 0 \,\forall t \\ & \mathbf{x}_0 = \mathbf{x}_{start} \\ & \mathbf{x}_T = \mathbf{x}_{goal} \end{aligned}$$

How to encode collision constraints?

Kth Order Markov Optimization (KOMO) [1] (3)

K-th order markov optimization (KOMO)

This can be solved using the Lagrangian method

$$\text{minimize } \sum_{t=0}^{T} \mathbf{f}_t(\mathbf{x}_{t-k:t}) + \nu \sum_{t=0}^{T} \mathbf{g}_t(\mathbf{x}_{t-k:t}) + \mu \sum_{t=0}^{T} \mathbf{h}_t(\mathbf{x}_{t-k:t})$$

This is usually done iteratively. First, find a solution with low ν,μ, then use found solution and optimize with larger ν,μ.

Kth Order Markov Optimization (KOMO) [1] (4)

Objectives

Objectives $\mathbf{f}_t$ usually include quadratic terms (because they are easier to handle during optimization) like

- Position constraints $\mathbf{f}(\mathbf{x}_t) = (\mathbf{x}_t - \mathbf{q}_t)^T(\mathbf{x}_t - \mathbf{q}_t)$
- Minimize velocity: $\mathbf{f}(\mathbf{x}_{t-1:t}) = \|\mathbf{x}_t - \mathbf{x}_{t-1}\|^2$.
- Minimize accelerations: $\mathbf{f}(\mathbf{x}_{t-2:t}) = \|\mathbf{x}_t + \mathbf{x}_{t-2} - 2\mathbf{x}_{t-1}\|^2$.

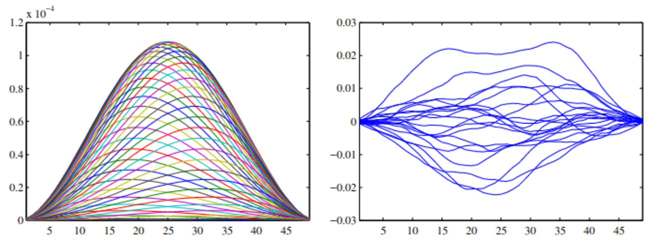
Code base / library: RAI `https://github.com/MarcToussaint/rai`

Stochastic Trajectory Optimization for Motion Planning (STOMP) [2]

Key Idea

Gradients may be noisy or discontinous. Use numerical method to approximate gradient first, then apply gradient descent.

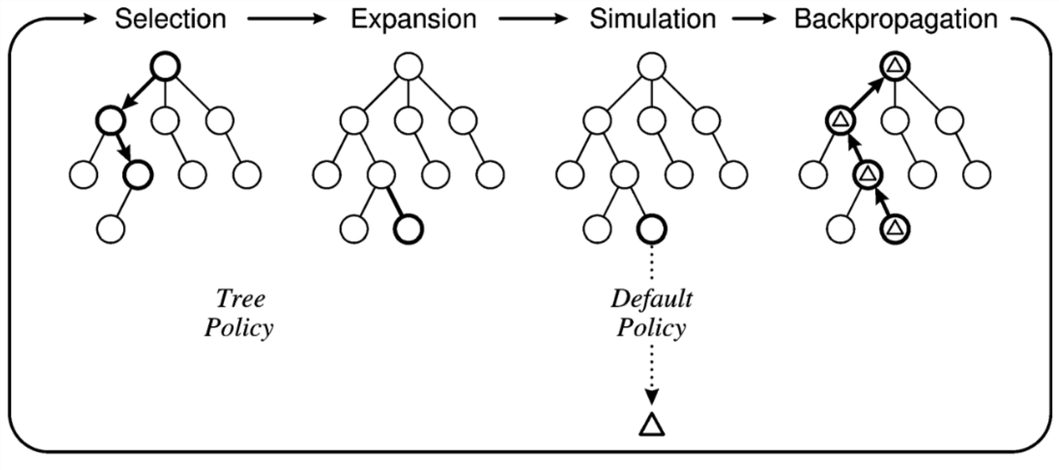
1. Create K trajectories by adding Gaussian noise to an initial guess



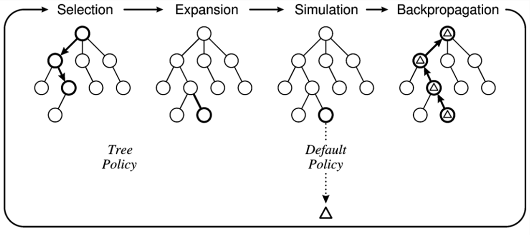
Left: Visualization of co-variance matrix for noise
Right: 20 random trajectory disturbances

2. For each trajectory and timestep compute the cost and then probability (softmax)
3. Gradient descent step: move towards the expectation of the noise value

Monte Carlo Tree Search (MCTS) (1) [3]



Monte Carlo Tree Search (MCTS) (2) [3]



- Use reward function to guide search (high reward if at goal)
- Each node represents a state, a path from the root to a leaf a trajectory
- Can plan under uncertainty (dynamics, observations, environment) with unknown distributions
- Extensions for continuous action spaces
- High computational effort

Taxonomy Optimization-based Approaches (1)

By Decision Variables

- Discretized configuration sequence $\mathbf{q}_k$
  - KOMO [1]
  - TrajOpt (SCP variant) [4]
  - CHOMP [5]
  - STOMP [2]
- Discretized action and configuration sequences $\mathbf{u}_k$ and $\mathbf{q}_k$
  - GuSTO [6], SCvx [7] (SCP variants)
  - MCTS [3]
- Other parameters
  - Spline optimization (polynomial parameters or Bézier control points)

Taxonomy Optimization-based Approaches (2)

By Solver Type

- Convex optimization
  - TrajOpt [4], GuSTO [6], SCvx [7] (SCP variants)
  - Spline optimization (polynomial parameters or Bézier control points)
- Non-convex, gradient-based optimization
  - CHOMP [5]
  - KOMO [1]
- Gradient-free / Blackbox
  - STOMP [2]
  - MCTS [3]

What is the “best” optimization-based motion planner? (1)

Geometric Motion Planning

Splines (if applicable) or KOMO

Why?

Spline optimization is a QP (global solution, very fast computation), but not all constraints can be encoded.
KOMO is fast in practice, relatively easy to tune, and can encode arbitrary costs/constraints

What is the “best” optimization-based motion planner? (2)

Kinodynamic Motion Planning

Splines (if applicable), SCP, or control-based methods (not covered here)

Why?

Spline optimization is a QP (global solution, very fast computation), but not all constraints can be encoded.

SCP can encode all constraints (with linearization) nicely, but is slow in practice (see [8])

Comparing Search-, Sampling-, and Optimization-based Approaches

Recap

Foundations

2 Weeks (problem formulation, terminology, collision checking)

Search-based

2 Weeks (A* and variants; state-lattice-based planning)

Sampling-based

5 Weeks (RRT, PRM, OMPL, Sampling Theory)

Optimization-based

2.5 Weeks (SCP, KOMO)

Current and Advanced Topics

2.5 Weeks (Comparative Analysis, Machine Learning and Motion Planning, Hybrid- and Multi-Robot approaches)

Algorithms and Software Packages

Search-based

A^*

wA^*, A^*_ϵ

State Lattices + X

SBPL

Sampling-based

PRM, LazyPRM

EST, RRT, RRT-Connect

RRT*, PRM*, FMT*, Informed RRT*

kinodynamic RRT, SST, AO-RRT

OMPL

Optimization-based

CHOMP, STOMP

KOMO

SCP (TrajOpt, GuSTO, SCvx)

Splines

---

cvxpy, SCPToolbox.jl, RAI

Which algorithms directly support kinodynamic motion planning?

Properties (1)

What are the strongest properties for each category?

	Search	Sampling	Optimization
Completeness	resolution-complete (A*)	probabilistically complete (RRT*)	Yes (CP), No (SCP, KOMO)
Optimality	optimal w.r.t. resolution (A*)	asymptotically optimal (RRT*)	Yes (CP), Locally (SCP, KOMO, ...)
Complexity / Convergence	polynomial time to find optimal solution (A*)	exponential convergence for any solution; sublinear to optimal solution (PRM*)	polynomial time to find optimal solution (LP, some QP)

Properties (2)

What are the strongest properties for each category?

	Search	Sampling	Optimization
Scalability with state/action dimension	exponential	exponential	polynomial
Scalability with duration/path length	almost linear (depends on h)	roughly linear	polynomial

What is the “best” approach? (1)

Geometric Motion Planning

1. If space can be easily discretized $\Rightarrow$ A*, potentially followed by optimization-based smoothing
2. else $\Rightarrow$ RRT-Connect, followed by optimization-based smoothing (KOMO, Splines)

Why?

A* has the strongest theoretical guarantees (optimality, convergence), great runtime and memory usage, and is easy to tune (many variants; choice of heuristic function). RRT can handle implicitly defined environments, but requires the workspace to be bounded for sampling.

What is the “best” approach? (2)

Kinodynamic Motion Planning

1. If (almost) free space $\Rightarrow$ optimization (Splines, KOMO, SCP)
2. Else if T is “small” $\Rightarrow$ optimization (KOMO or SCP)
3. Else if d is “low” $\Rightarrow$ sampling (SST*, AO-RRT), followed by optimization-based smoothing
4. Else (obstacles, T is “high” and d is high) $\Rightarrow$ hierarchical/hybrid

Why?

Optimization has low runtime, good scalability with d, and produces nice results. However it does not work well for maze-like environments or large T.

What is the “best” approach? (3)

Practical Consideration

OMPL is by far the simplest to use/prototype, so start with that as baseline in any case.

No “best” approach!

Results can vary widely, depending on robot type, scenario, and objectives.

Testing (and benchmarking) is still needed for all applications.

Ideas here are biased and based on experience, not statistically validated science.

Hybrid Approaches

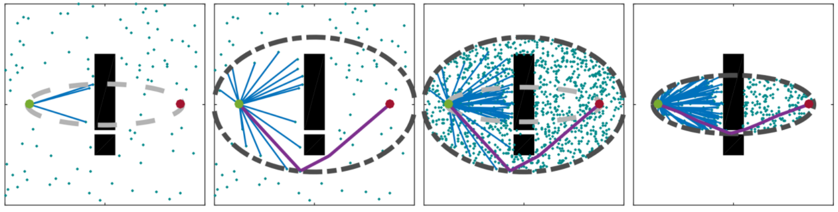
Search + Sampling

PRM*

- Sampling to construct a graph
- Search (A*) to find lowest-cost solution

Informed RRT*, BIT*

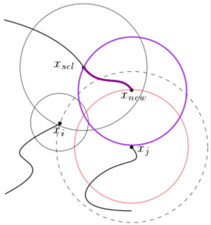
- Sampling to construct a tree
- Use search-inspired heuristic to focus sampler



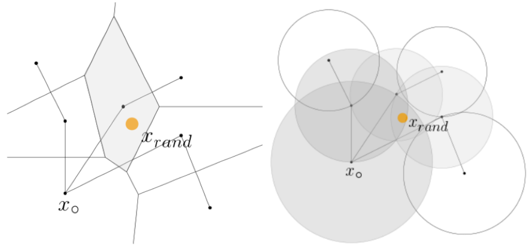
Search + Sampling: Dominance-Informed Region Tree (DIRT) [9]

Dominance-Informed Region

- A ball around each node in the tree (which represents $\mathbf{q}$)
- Locally low $f = g + h$ value $\Rightarrow$ larger radius
- Radii are updated as the tree grows
- Balls represent volume of "influence"

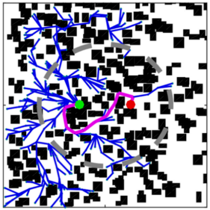


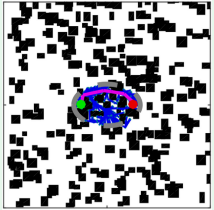
- Similar to RRT
- Rather than selecting $\mathbf{q}_{near}$ via Voronoi bias, randomly pick one configuration whose Dominance-Informed Region covers $\mathbf{q}_{rand}$



Search + Sampling: Dominance-Informed Region Tree (DIRT) [9]

Informed RRT* [10] Insight





After a solution is found (and thus a cost bound is known), refinement should focus on the “relevant region”.

What are similarities and differences between DIRT and Informed RRT*?

Both use a heuristic function

Informed RRT* changes the asymptotic behavior; DIRT the search itself

Video

https://doi.org/10.1109/IROS.2018.8593672

Search + Optimization

State-Lattice A*

- Optimization to generate motion primitives
- Search to find solution in implicitly defined graph

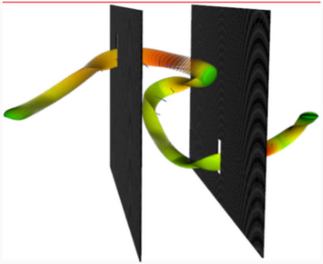
A* followed by Smoothing

- Search to generate waypoints
- Optimization to find smooth trajectory

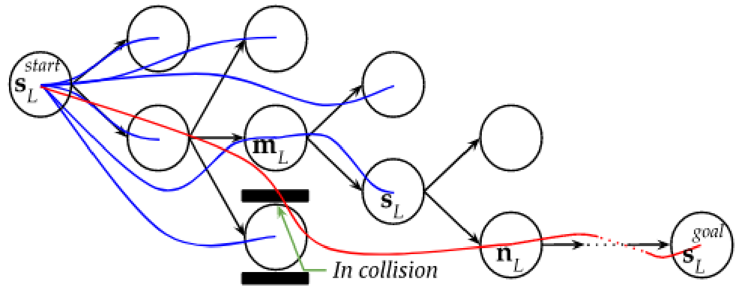
Search + Optimization: Interleaving Search and Trajectory Optimization (IN-SAT) [11]

Key Idea

- Hierarchical Planning
- Use search (wA*) to plan in low dimensions (6D: position and orientation)
- Use optimization (polynomial spline QP) to refine cost-to-come and cost-to-go in 12D (warm-started with low-dim solution)

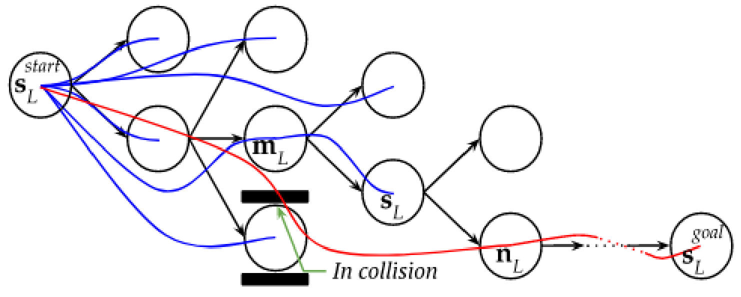


Search + Optimization: Interleaving Search and Trajectory Optimization (IN-SAT) [11]



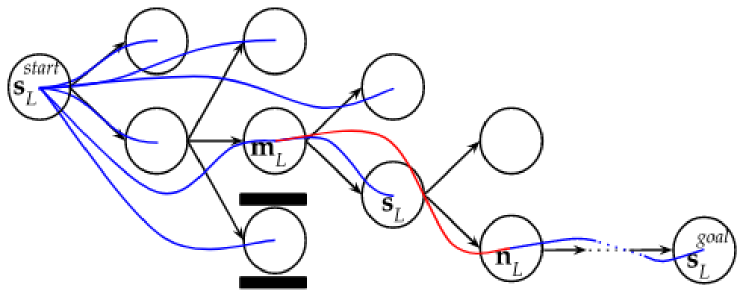
- Shown
  - low-level search graph: nodes = circles; black arrows = edges
  - Smooth trajectories (cost-to-come): blue edges
- Step 1: Expand node in OPEN; here: expand s_L, which has successor n_L (low 6D space)

Search + Optimization: Interleaving Search and Trajectory Optimization (IN-SAT) [11]



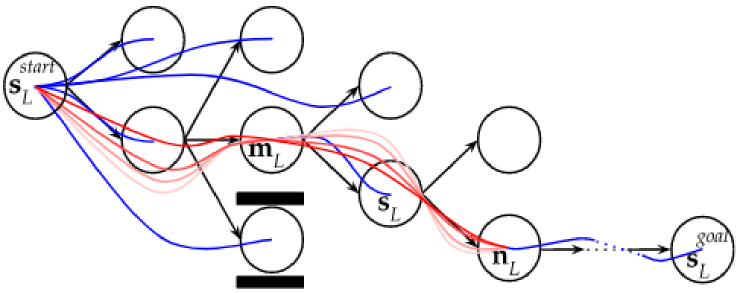
- Step 2: Attempt to optimize trajectory from start to goal via intermediate state n_L (red line)
- If successful $\Rightarrow$ Step 5
- Else $\Rightarrow$ Repair (Step 3)

Search + Optimization: Interleaving Search and Trajectory Optimization (IN-SAT) [11]



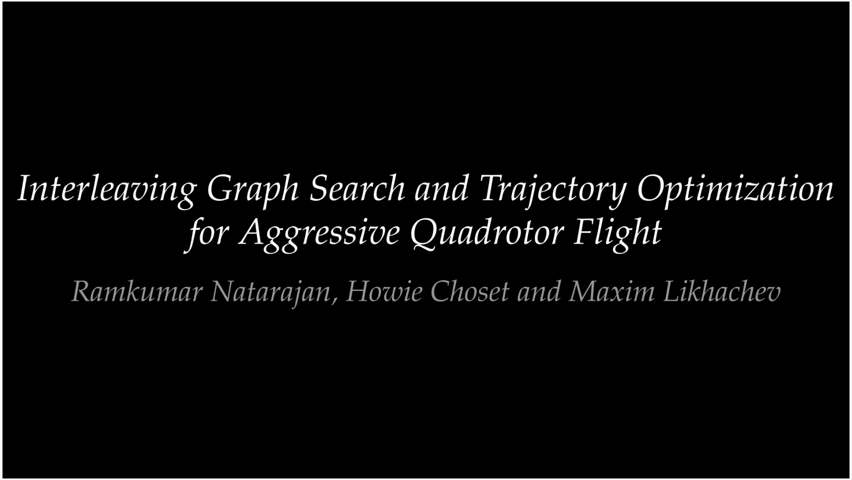
- Step 3: Attempt to optimize trajectory from intermediate states between start and s_L towards the goal via intermediate state n_L (red line)
- Iterative process, starting from successor of start state

Search + Optimization: Interleaving Search and Trajectory Optimization (IN-SAT) [11]



- Step 5: Re-optimize without intermediate constraints

Video



https://doi.org/10.1109/LRA.2021.3067298

Sampling + Optimization

RRT(*) followed by Smoothing

- Sampling to find initial solution (homotopy class)
- Optimization to find optimal trajectory within that homotopy class

STOMP

- Sampling to estimate gradient robustly
- Gradient descent to optimize

Sampling + Optimization: RABIT*

RRT*-RBO [12]

- Add edges by solving small optimization problems
- Sampler: RRT*
- Optimizer: BSplines

Enables to use RRT* for kinodynamic planning.

RABIT* [13]

- Add edges by solving small optimization problems
- Sampler: BIT*
- Optimizer: CHOMP

Better convergence than BIT* for geometric problems.

Search + Sampling + Optimization

DIRT/Informed RRT* followed by Smoothing

- Search + Sampling to find initial solution (homotopy class)
- Optimization to find optimal trajectory within that homotopy class

STOMP initialized by A*

- A* to find initial waypoints
- Sampling to estimate gradient robustly
- Gradient descent to optimize

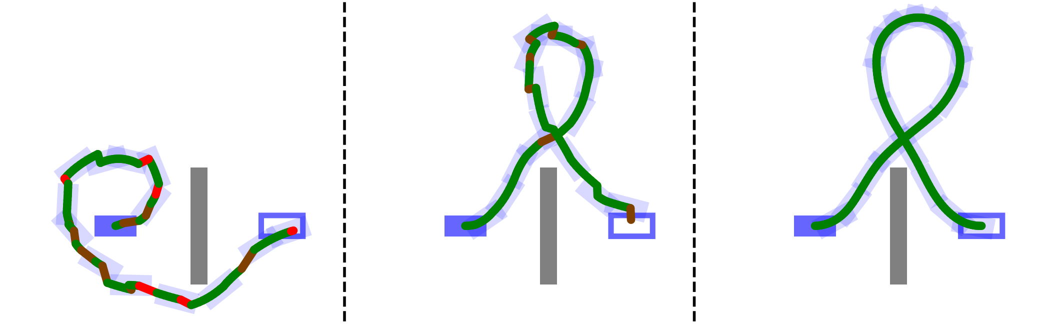
RABIT*

Since BIT* borrows some ideas from search.

Key Idea

Allow "jumps" in the solution and iteratively reduce their magnitude.

Example: Unicycle that can only turn left



A) $\delta = 0.41 \quad T\Delta t = 13.3s$ B) $\delta = 0.22 \quad T\Delta t = 16.3s$ C) $\delta = 0.00 \quad T\Delta t = 19.5s$

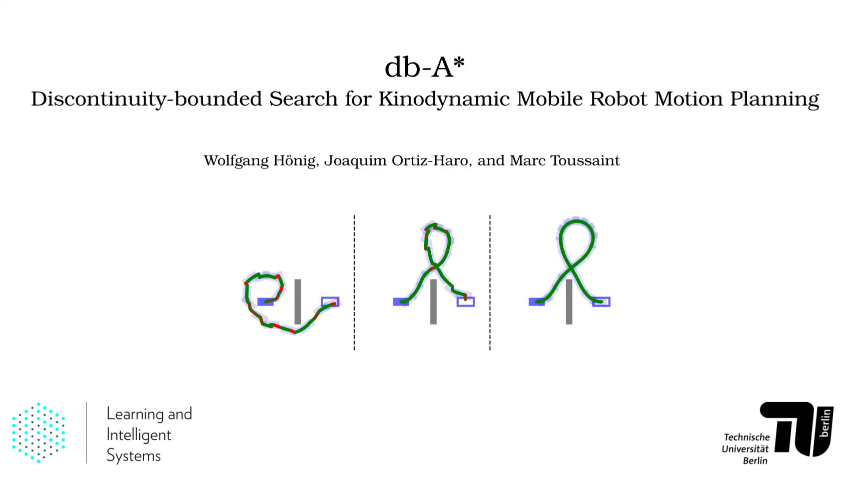
Search + Sampling + Optimization: db-A* [14]

Algorithm 1: kMP-db-A*: Kinodynamic Motion Planning with db-A*

$\mathcal{M} \leftarrow \emptyset$ ▷ *Set of motion primitives*
$c_{\max} \leftarrow \infty$ ▷ *Solution cost bound*
for $n = 1, 2, \ldots$ **do**
  $\mathcal{M} \leftarrow \mathcal{M} \cup \texttt{AddPrimitives}()$
  $\delta \leftarrow \texttt{ComputeDelta}(\mathcal{M})$
  $\mathbf{X}_d, \mathbf{U}_d, T_d \leftarrow \texttt{db-A*}(\mathbf{x}_s, \mathbf{x}_f, \mathcal{X}_{\text{free}}, \mathcal{M}, \delta, c_{\max})$
  if $\mathbf{X}_d, \mathbf{U}_d$ *successfully computed* **then**
    $\mathbf{X}, \mathbf{U}, T \leftarrow \texttt{Optimization}(\mathbf{X}_d, \mathbf{U}_d, T_d, c_{\max})$
    if $\mathbf{X}, \mathbf{U}$ *successfully computed* **then**
      $\texttt{Report}(\mathbf{X}, \mathbf{U}, T)$ ▷ *New solution found*
      $c_{\max} \leftarrow \min(c_{\max}, J(\mathbf{X}, \mathbf{U}, T))$ ▷ *cost bound*
    $\mathcal{M} \leftarrow \mathcal{M} \cup \texttt{ExtractPrimitives}(\mathbf{X}, \mathbf{U})$

db-A*: Combination of A* and kd-Trees, to compute solutions with bounded discontinuities δ

Video



https://youtu.be/dLNheLa5wAc

Conclusion

- KOMO as efficient nonlinear motion planning formulation
- Search-, Sampling-, Optimization-based approaches have different strength and weaknesses
- A* and RRT* are good initial choices for the geometric case; Splines, SCP, and SST*/AO-RRT good initial choices for the kinodynamic case
- Hybrid solutions can combine advantages of search-, sampling-, and optimization-based approaches

Next Time

- Advanced Topics: Multi-Robot Motion Planning

Suggested Reading

1. John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. **"Motion Planning with Sequential Convex Optimization and Convex Collision Checking".** In: *The International Journal of Robotics Research* 33.9 (Aug. 1, 2014), pp. 1251–1270. ISSN: 0278-3649. DOI: 10.1177/0278364914528132, Section 4.1
2. References in the slides

[1] Marc Toussaint. **“A Tutorial on Newton Methods for Constrained Trajectory Optimization and Relations to SLAM, Gaussian Process Smoothing, Optimal Control, and Probabilistic Inference”.** In: *Geometric and Numerical Foundations of Movements*. Ed. by Jean-Paul Laumond, Nicolas Mansard, and Jean-Bernard Lasserre. Vol. 117. Cham: Springer International Publishing, 2017, pp. 361–392. ISBN: 978-3-319-51546-5 978-3-319-51547-2. URL: `http://link.springer.com/10.1007/978-3-319-51547-2_15` (visited on 08/16/2021).

[2] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. **“STOMP: Stochastic Trajectory Optimization for Motion Planning”.** In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 4569–4574. DOI: 10.1109/ICRA.2011.5980280.

[3] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. **“A Survey of Monte Carlo Tree Search Methods”.** In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1 (Mar. 2012), pp. 1–43. ISSN: 1943-0698. DOI: 10.1109/TCIAIG.2012.2186810.

[4] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. **"Motion Planning with Sequential Convex Optimization and Convex Collision Checking".** In: *The International Journal of Robotics Research* 33.9 (Aug. 1, 2014), pp. 1251–1270. ISSN: 0278-3649. DOI: 10.1177/0278364914528132.

[5] Matthew Zucker, Nathan D. Ratliff, Anca D. Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M. Dellin, J. Andrew Bagnell, and Siddhartha S. Srinivasa. **"CHOMP: Covariant Hamiltonian Optimization for Motion Planning".** In: *I. J. Robotics Res.* 32.9-10 (2013), pp. 1164–1193. DOI: 10.1177/0278364913488805.

[6] Riccardo Bonalli, Abhishek Cauligi, Andrew Bylard, and Marco Pavone. **"GuSTO: Guaranteed Sequential Trajectory Optimization via Sequential Convex Programming".** In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019 International Conference on Robotics and Automation (ICRA). May 2019, pp. 6741–6747. DOI: 10.1109/ICRA.2019.8794205.

[7] Danylo Malyuta, Taylor P. Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behçet Açıkmeşe. **"Convex Optimization for Trajectory Generation: A Tutorial on Generating Dynamically Feasible Trajectories Reliably and Efficiently".** In: *IEEE Control Systems Magazine* 42.5 (2022), pp. 40–113. DOI: 10.1109/MCS.2022.3187542.

[8] Welf Rehberg, Joaquim Ortiz de Haro, Marc Toussaint, and Wolfgang Hönig. **“Comparison of Optimization-Based Methods for Energy-Optimal Quadrotor Motion Planning”.** In: *CoRR* abs/2304.14062 (2023). DOI: 10.48550/arXiv.2304.14062. arXiv: 2304.14062.

[9] Zakary Littlefield and Kostas E. Bekris. **“Efficient and Asymptotically Optimal Kinodynamic Motion Planning via Dominance-Informed Regions”.** In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2018, pp. 1–9. DOI: 10.1109/IROS.2018.8593672.

[10] Jonathan D. Gammell, Timothy D. Barfoot, and Siddhartha S. Srinivasa. **"Informed Sampling for Asymptotically Optimal Path Planning".** In: *IEEE Transactions on Robotics* 34.4 (Aug. 2018), pp. 966–984. ISSN: 1941-0468. DOI: 10.1109/TRO.2018.2830331.

[11] Ramkumar Natarajan, Howie Choset, and Maxim Likhachev. **"Interleaving Graph Search and Trajectory Optimization for Aggressive Quadrotor Flight".** In: *IEEE Robotics and Automation Letters* 6.3 (July 2021), pp. 5357–5364. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2021.3067298. arXiv: 2101.12548.

[12] Samantha Stoneman and Roberto Lampariello. **"Embedding Nonlinear Optimization in RRT for Optimal Kinodynamic Planning".** In: *53rd IEEE Conference on Decision and Control*. 53rd IEEE Conference on Decision and Control. Dec. 2014, pp. 3737–3744. DOI: 10.1109/CDC.2014.7039971.

[13] Sanjiban Choudhury, Jonathan D. Gammell, Timothy D. Barfoot, Siddhartha S. Srinivasa, and Sebastian Scherer. **"Regionally Accelerated Batch Informed Trees (RABIT*): A Framework to Integrate Local Information into Optimal Path Planning".** In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4207–4214. DOI: 10.1109/ICRA.2016.7487615.

[14] Wolfgang Hönig, Joaquim Ortiz de Haro, and Marc Toussaint. **"db-A*: Discontinuity-bounded Search for Kinodynamic Mobile Robot Motion Planning".** In: *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*. IEEE, 2022, pp. 13540–13547. DOI: 10.1109/IROS47612.2022.9981577.