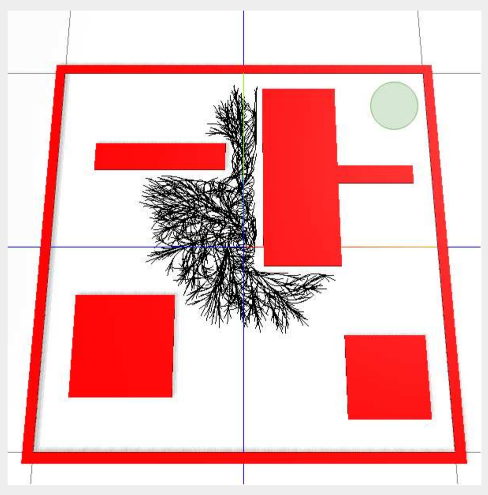
Learning Efficient Maneuver Sets for Kinodynamic Motion Planning

Aravind Sivaramakrishnan, Zakary Littlefield and Kostas E. Bekris
Department of Computer Science, Rutgers, the State University of New Jersey

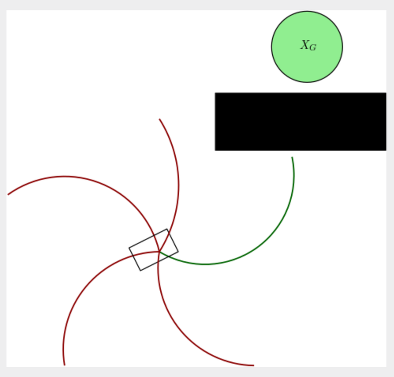
Motivation

- **Objective**: Improve the per iteration performance of a kinodynamic planner.
- **Key idea**: Instead of random propagation, compute local maneuvers that balance an exploitation-exploration trade-off
- Exploitation maneuvers guide the system towards the goal given local heuristic information
- Exploration maneuvers move the system in different directions to deal with situations where heuristic does not provide good guidance.



Informed maneuvers navigate the robot (center) to the goal (green) while exploring the workspace.

Motion planning with curated maneuvers



From a large set of random maneuvers, the exploitative (green) maneuver minimizes heuristic to the goal (green). Explorative (red) maneuvers are obtained iteratively by maximizing the dispersion to previously curated maneuvers **[2]**.

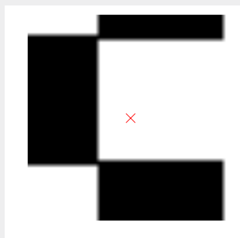
- First solution statistics between DIRT using random and curated maneuvers:

	Iteration	Time	Path Cost
Random	1471	0.2	50.47
Curated	686	12.15	48.13

- Very effective in finding a high-quality solution in fewer number of iterations
- Prohibitive to compute online
- **Goal:** Develop a data-driven approach that achieves the same objective as the curation but can generate the maneuvers fast.

Input to the learning process

- A regular set of points $\mathbb{X}_{local}$ in the vicinity of x_0 are collision checked to generate a binary 2D map o_{local} indicating the presence of obstacles in the workspace.
- The heuristic $h(x)$ is also evaluated at each $x \in \mathbb{X}_{local}$, resulting in a 2D matrix h_{local}.



o_{local}

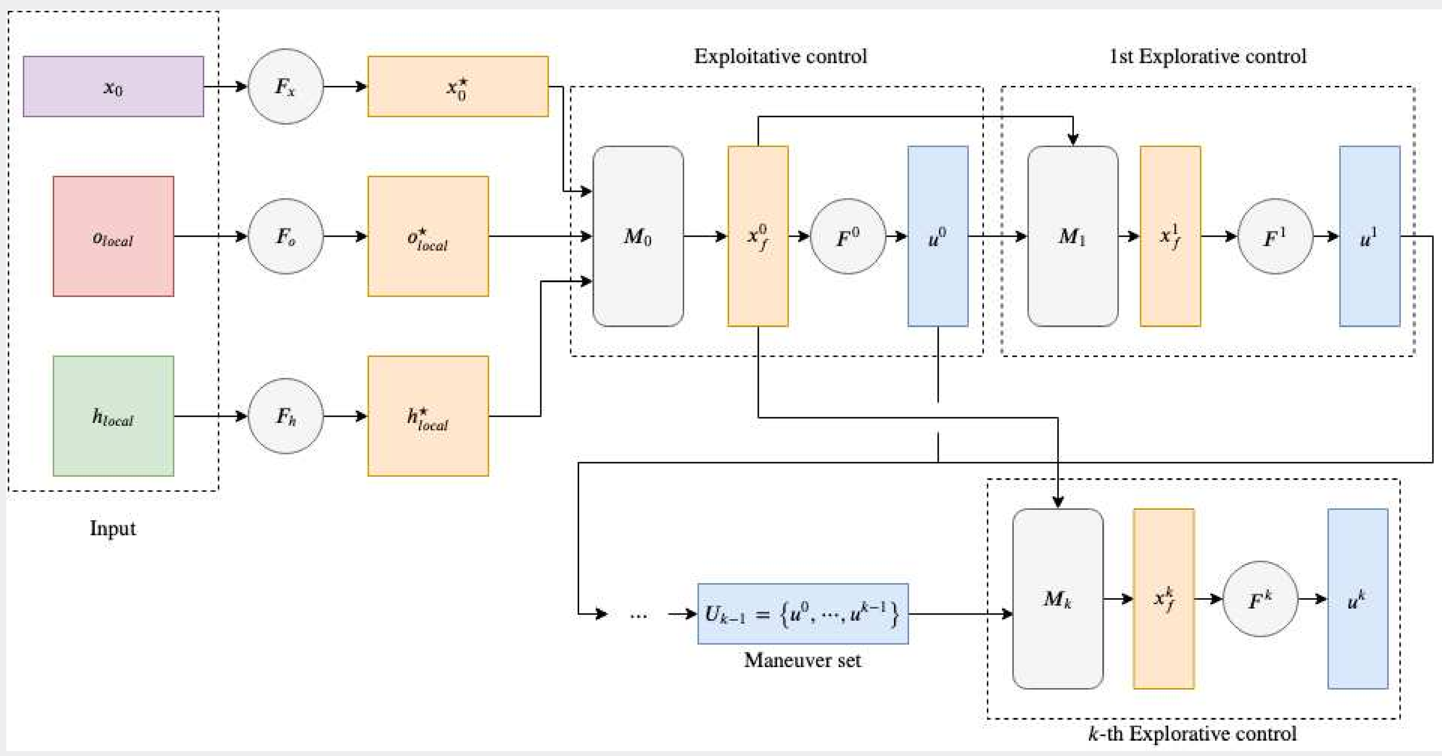
h_{local}

Proposed architecture

- Multi-layered neural networks F_x, F_o, F_h act on the inputs to produce $x_0^*, o_{local}^*, h_{local}^*$.
- An operator $M_0(x_0^*, o_{local}^*, h_{local}^*)$ produces feature vector x_f^0.
- Exploitative control u^0 is obtained as $u^0 = F^0(x_f^0)$, where F^0 is also a neural network.
- Remaining N exploratory controls are obtained as follows.

$$x_f^k = M_k(x_f^0, U_{k-1})$$
$$u^k = F^k(x_f^k)$$

where for all $k \geq 1$, $U_k = \{u^0, u^1, .., u^{k-1}\}$. For the exploitative control ($k = 0$), U_{k-1} is the empty set.

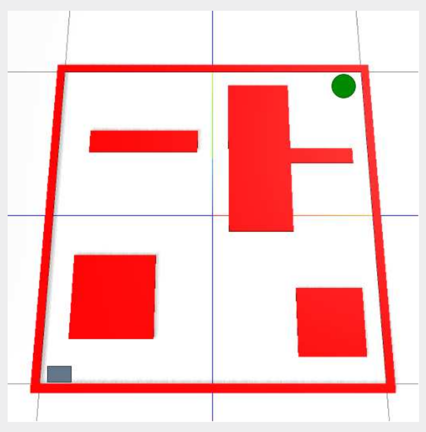


Computation graph of $U_k = \hat{f}(x_0, o_{local}, h_{local})$. For $k = N, U_k = \hat{U} = \{u^0, \cdots, u^N\}$.

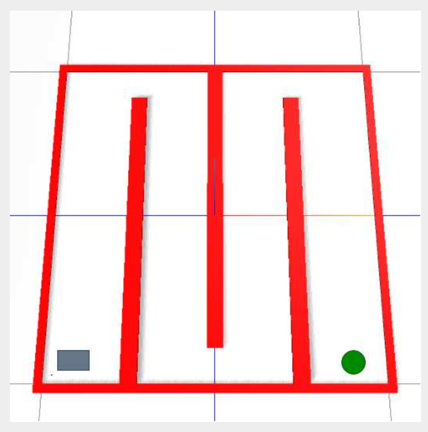
Experimental Setup

Treaded vehicle with 5 dim. state space (SE(2) augmented by steering angle and forward velocity) and 2 dim. control space (acceleration of left and right treads) used in our experiments.

- Randomly place obstacles in the workspace so they cover one-third of the reachable workspace.
- Execute the DIRT planner **[1]** with the online curation procedure on multiple problem instances in such workspaces.
- Euclidean distance to the goal in the workspace is used as the heuristic function.
- For each node x_0 the planner selects to propagate, store o_{local} and h_{local} maps, and maneuver set $\hat{U}$ of size 5 curated from 1000 randomly sampled maneuvers.
- Train two networks - fully connected (`FC`) and convolutional (`Conv`)



`Greedy`



`Explore`

Environments: The grey rectangle is the starting pose of the robot (facing right) and the green circle is the goal region. The robot must avoid the red obstacles.

Solution statistics

We compare the performance of DIRT (after 50k iterations) for the following maneuvers:

- random (DIRT - Random)
- only exploitative control predicted by the network (DIRT - FC (Exploit), DIRT - Conv (Exploit))
- both exploitative and explorative controls are predicted by the networks ((DIRT - FC (All), DIRT - Conv (All))

Algorithm	NumSolns	FirstSolnIters	FirstSolnCost	FinalSolnIters	FinalSolnCost
DIRT - Random	30	3446.67	59.64	23277.57	49.44
DIRT - FC (Exploit)	30	2246.67	56.54	17050.37	49.89
DIRT - FC (All)	30	**620**	**47.58**	**16921.5**	**45.47**
DIRT - Conv (Exploit)	30	3366.67	65.03	27774.67	48.38
DIRT - Conv (All)	30	2006.67	54.8	25671.07	48.16

Solution statistics for `Greedy`. All values are averaged over `NumSolns`. Best values highlighted in bold.

Algorithm	NumSolns	FirstSolnIters	FirstSolnCost	FinalSolnIters	FinalSolnCost
DIRT - Random	30	15666.67	163.60	33254.13	149.47
DIRT - FC (Exploit)	29	**12000**	155	31794.86	140.06
DIRT - FC (All)	30	18766.67	**133.83**	**28119.66**	**130.92**
DIRT - Conv (Exploit)	29	27666.67	182.16	39924.96	172.14
DIRT - Conv (All)	30	14066.67	143.71	28194.83	139.43

Solution statistics for `Explore`. All values are averaged over `NumSolns`. Best values highlighted in bold.

Challenges

- **Quality:** How to increase the rate of collision-free maneuvers?
- **Cost:** How to make network inference cost comparable to returning a random maneuver?
- **Data efficiency:** How to deal with higher dimensional systems and more complex environments?
- **Uncertainty:** How to deal with more realistic sensing input?

References

1. Littlefield, Z., and Bekris, K. E. 2018. Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions. In *IROS*.
2. Green, C.J., and Kelly, A. 2007. Toward optimal sampling in the space of paths. In *ISRR*.

{as2578, zwl2, kb572}@rutgers.edu