Decision Making for Robots and Autonomous Systems

Fall 2015

George Konidaris
gdk@cs.duke.edu

Useful Robots

Robots should be *general purpose machines*.

That means:

- Programmable
- End-user programmable

How do we do this?

Learning from Demonstration

Procedural knowledge is hard for robots but easy for humans.

Rather than program the robot using code, how about we program it by *showing it what to do*?

This is known as **learning from demonstration** (LfD).

Inverse RL

LfD largely assumes we *know* the MDP, and want to find a policy, or perhaps we don't know just the goal.

Is this always realistic?

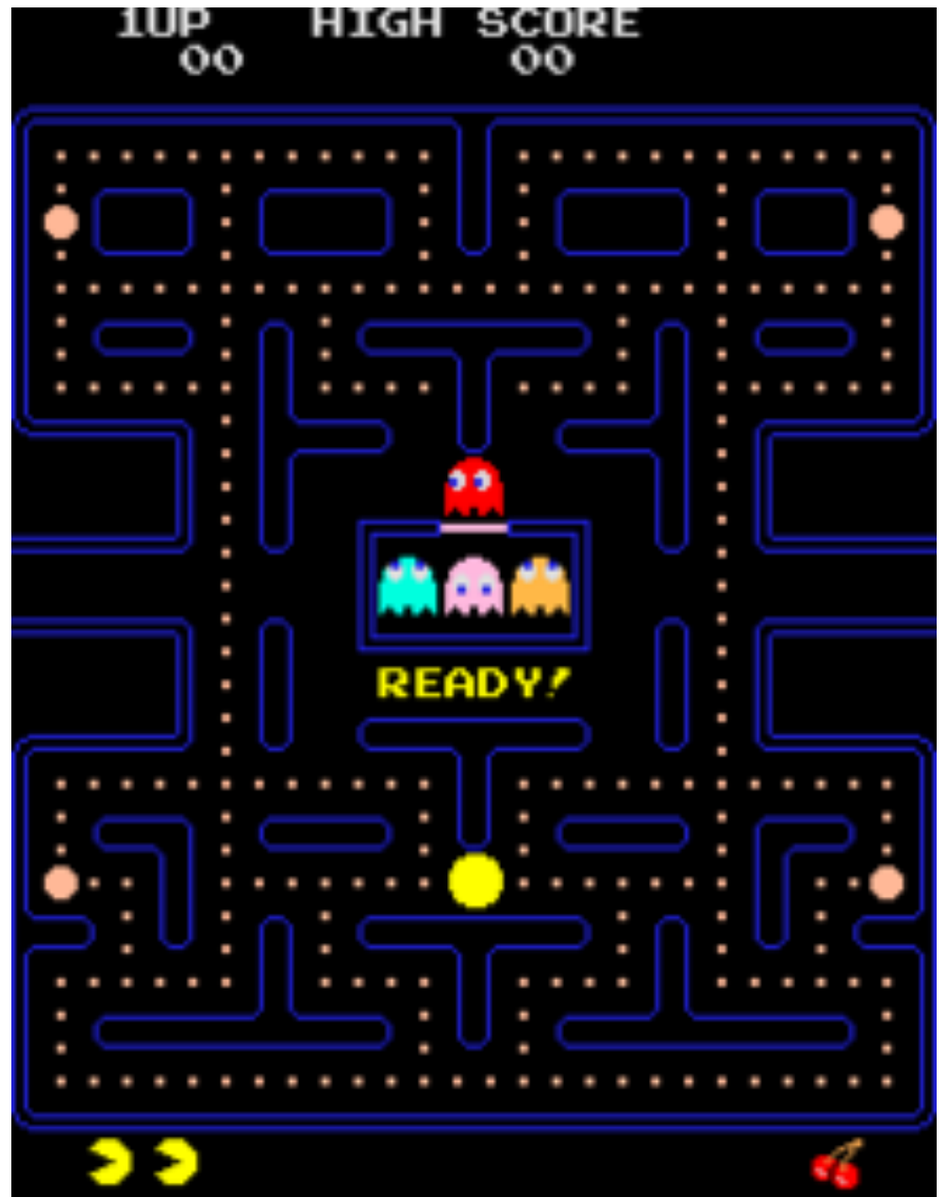
Inverse RL

Inverse RL:

- Assume we don't know reward function
- Watch expert demonstrations
- Infer reward function from demonstrations

Central question: *what reward function would have caused an optimizing agent to generate this behavior?*

Examples



Examples

Inverse RL

More formally, given an MDP/R:

$$M = \{S, A, T, \gamma\}$$

… and some trajectories:

$$\tau_i = \{(s_1^i, a_1^i), (s_2^i, a_2^i), ..., (s_n^i, a_n^i)\}$$

find reward function *R* such that the actions taken by π^* (**obtained by optimizing for *R***) match those in τ.

Inverse RL

Note!

R is not unique.

In particular … recall from utility theory:

- Positive affine modifications to *R* retain optimal policy.

That's OK, any of them will do!

Abbeel's Algorithm

Assume a linear basis Φ for reward function:

- $R(s_t) = w \cdot \Phi(s_t)$

Expected return:

$$\mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i R(s_i)\right]$$

$$= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i w \cdot \Phi(s_i)\right]$$

$$= w \cdot \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \Phi(s_i)\right]$$

[Abbeel & Ng, ICML 2004]

Abbeel's Algorithm

So let:

$$\mu(\pi) = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i \Phi(s_i)\right]$$

… and define μ_E be the *empirically observed expected features* over our sample data τ.

If we can match feature expectations, then we are getting the same return (regardless of *w*).

Abbeel's Algorithm

max margin

1. Randomly pick some policy $\pi^{(0)}$, compute (or approximate via Monte Carlo) $\mu^{(0)} = \mu(\pi^{(0)})$, and set $i = 1$.
2. Compute $t^{(i)} = \max_{w:\|w\|_2 \leq 1} \min_{j \in \{0..(i-1)\}} w^T(\mu_E - \mu^{(j)})$, and let $w^{(i)}$ be the value of w that attains this maximum.
3. If $t^{(i)} \leq \epsilon$, then terminate.
4. Using the RL algorithm, compute the optimal policy $\pi^{(i)}$ for the MDP using rewards $R = (w^{(i)})^T \phi$.
5. Compute (or estimate) $\mu^{(i)} = \mu(\pi^{(i)})$.
6. Set $i = i + 1$, and go back to step 2.

Abbeel's Algorithm

"margin"

$$\begin{aligned} \max_{t,w} \quad & t \\ \text{s.t.} \quad & w^T \mu_E \geq w^T \mu^{(j)} + t, \; j = 0, \ldots, i-1 \\ & ||w||_2 \leq 1 \end{aligned}$$

step 2

choice of reward function
that maximizes improvement
under observed feature
frequencies

Can prove that this converges to a good match, etc.

Experiments

Gridworld

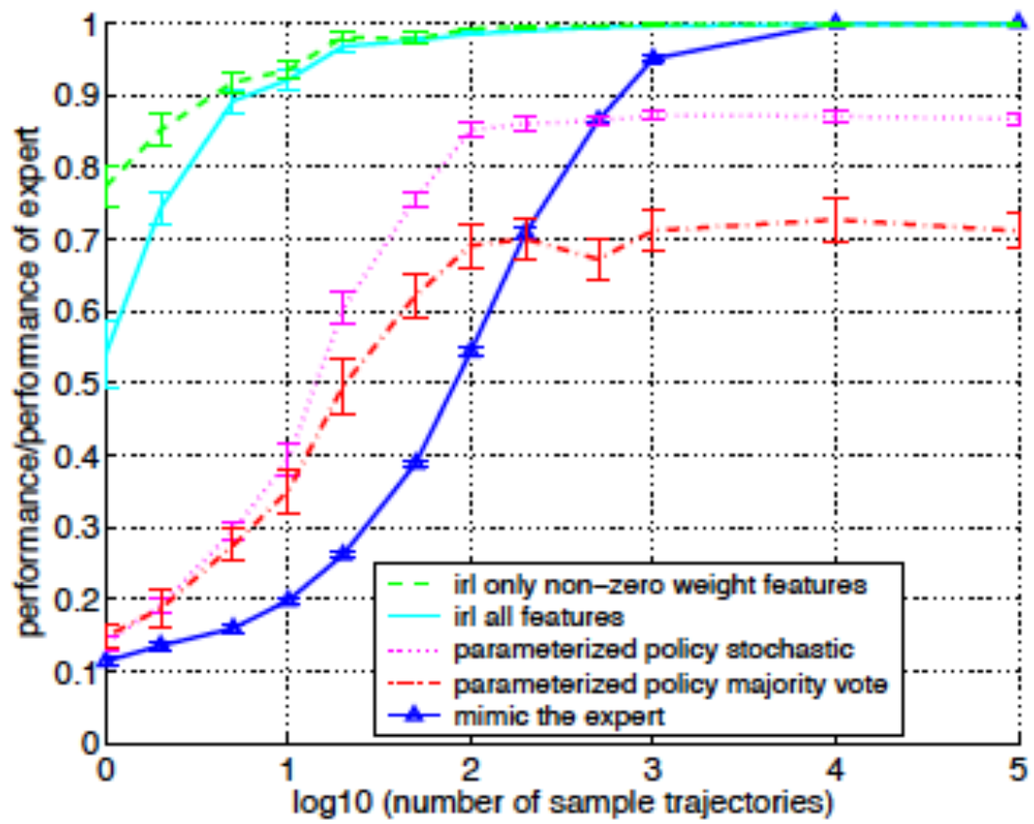


Figure 4. Plot of performance vs. number of sampled trajectories from the expert. (Shown in color, where available.) Averages over 20 instances are plotted, with 1 s.e. errorbars. Note the base-10 logarithm scale on the x-axis.

Experiments

Driving Sim

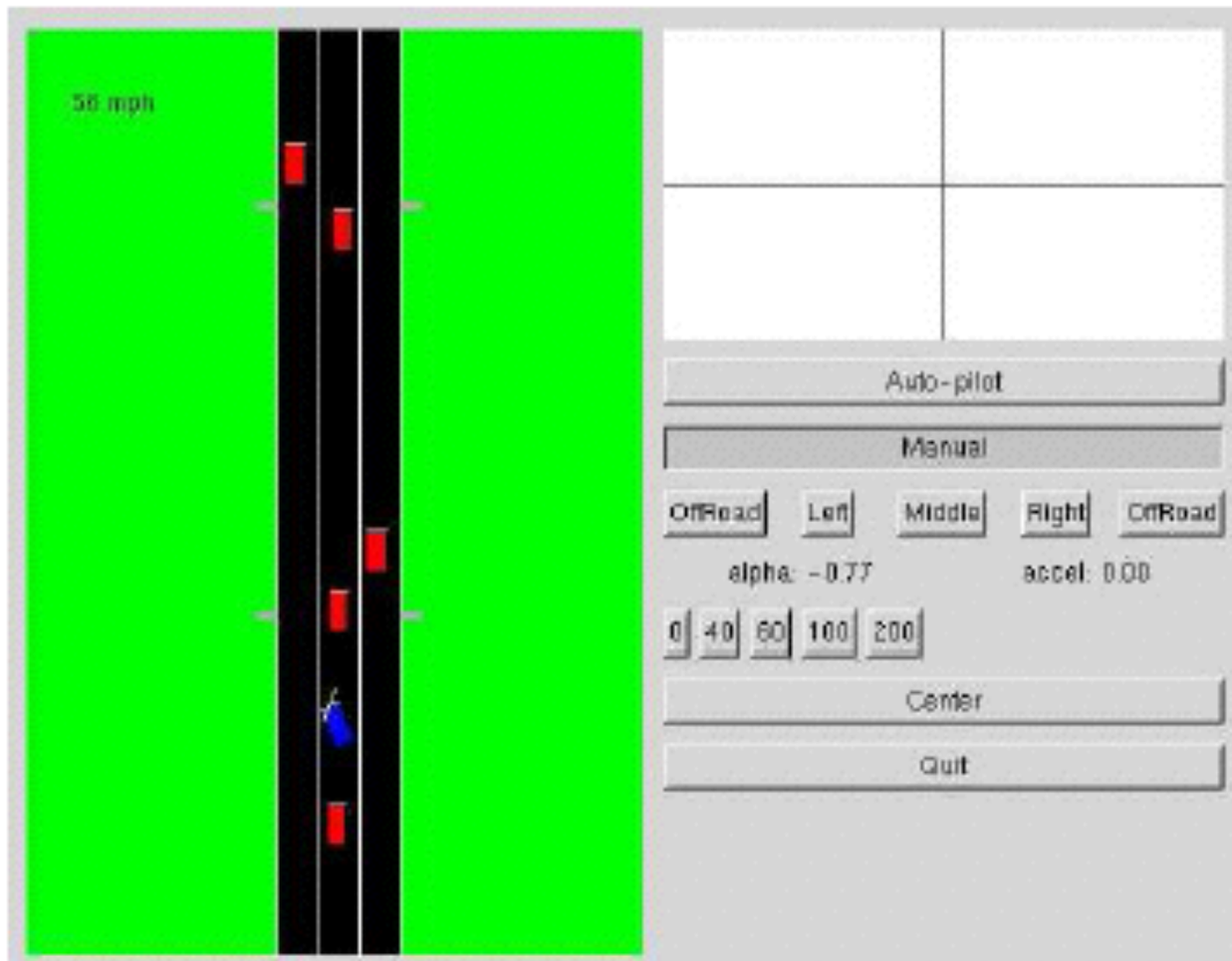


Figure 5. Screenshot of driving simulator.

Experiments

Driving Sim

Table 1. Feature expectations of teacher $\hat{\mu}_E$ and of selected/learned policy $\mu(\tilde{\pi})$ (as estimated by Monte Carlo). and weights w corresponding to the reward function that had been used to generate the policy shown. (Note for compactness, only 6 of the more interesting features, out of a total of 15 features, are shown here.)

		Collision	Offroad Left	LeftLane	MiddleLane	RightLane	Offroad Right
1	$\hat{\mu}_E$	0.0000	0.0000	0.1325	0.2033	0.5983	0.0658
	$\mu(\tilde{\pi})$	0.0001	0.0004	0.0904	0.2287	0.6041	0.0764
	$\tilde{w}$	-0.0767	-0.0439	0.0077	0.0078	0.0318	-0.0035
2	$\hat{\mu}_E$	0.1167	0.0000	0.0633	0.4667	0.4700	0.0000
	$\mu(\tilde{\pi})$	0.1332	0.0000	0.1045	0.3196	0.5759	0.0000
	$\tilde{w}$	0.2340	-0.1098	0.0092	0.0487	0.0576	-0.0056
3	$\hat{\mu}_E$	0.0000	0.0000	0.0000	0.0033	0.7058	0.2908
	$\mu(\tilde{\pi})$	0.0000	0.0000	0.0000	0.0000	0.7447	0.2554
	$\tilde{w}$	-0.1056	-0.0051	-0.0573	-0.0386	0.0929	0.0081
4	$\hat{\mu}_E$	0.0600	0.0000	0.0000	0.0033	0.2908	0.7058
	$\mu(\tilde{\pi})$	0.0569	0.0000	0.0000	0.0000	0.2666	0.7334
	$\tilde{w}$	0.1079	-0.0001	-0.0487	-0.0666	0.0590	0.0564
5	$\hat{\mu}_E$	0.0600	0.0000	0.0000	1.0000	0.0000	0.0000
	$\mu(\tilde{\pi})$	0.0542	0.0000	0.0000	1.0000	0.0000	0.0000
	$\tilde{w}$	0.0094	-0.0108	-0.2765	0.8126	-0.5099	-0.0154

Unsegmented IRL

What if the agent is deploying different options with different reward functions?

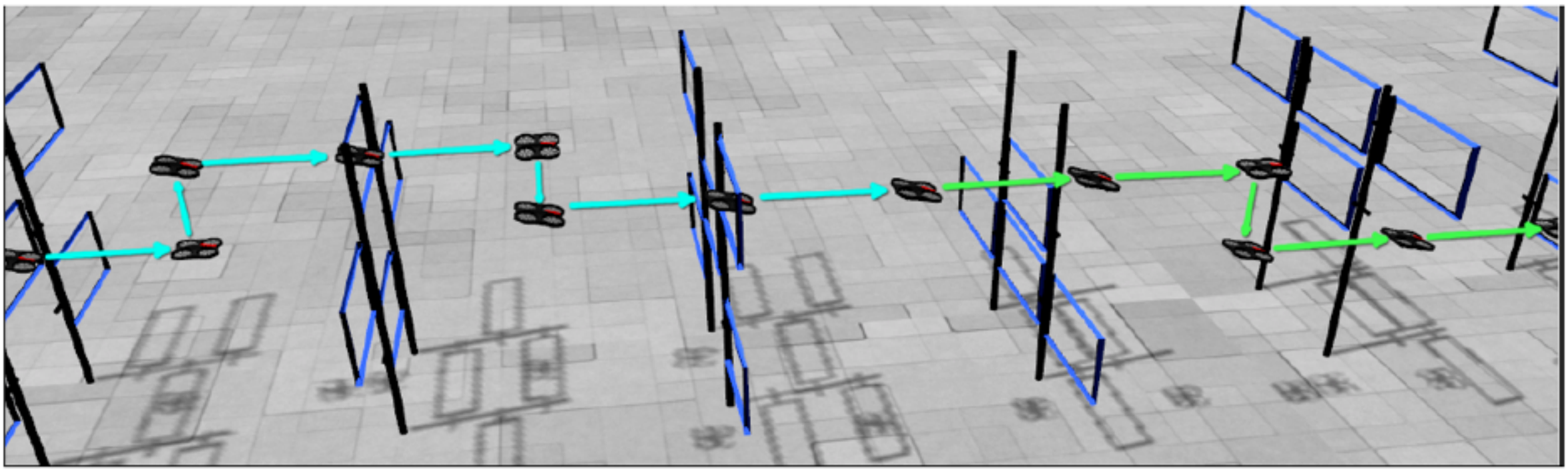


Fig. 6. The quadcopter gate domain. The agent is initially flying through as many hoops as possible (cyan) but switches to avoiding the hoops (green).

Unsegmented IRL

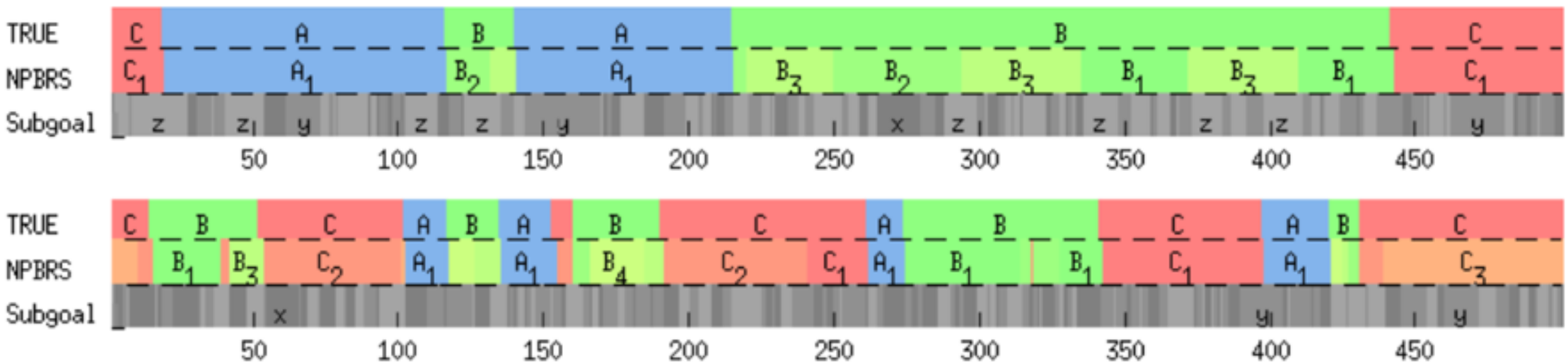


Fig. 9. Segmentations for the quadcopter gate domain for trajectory length 500 showing using segmentation based on NPBRS and subgoal states.

[Ranchod et al., IROS 2015]