Planning in Dynamic Environments with Conditional Autoregressive Models

Introduction

Planning agents find actions at each decision point by considering future scenarios from their current state against a model of their world. Though typically slower at decision-time than model-free agents, agents which use planning have several appealing qualities:

- Can be configured and tuned with constraints
- Directly test long-term consequences
- Do not explicitly require training prior to decision time
- Adapt to new environments without training

To perform well, planning-based agents need an accurate future model of their environment for evaluating actions, however, this future model isn't usually available in the real world. In this paper, we demonstrate how to leverage recent improvements in generative modeling [2, 3] to create powerful dynamics models that are fast enough to be used for forward planning in agent-independent environments.

Methodology

We utilize a two-phase training procedure on a $48 \times 48 \times 1$ grid environment with dynamic obstacles and a moving goal.

1. Learn a compact, discrete representation (Z) of pixel-space frames with a VQ-VAE model [3]
2. A conditional gated PixelCNN [2] is trained to predict one-step ahead Z representations of sequential frames

Our test agent uses Monte-Carlo Tree Search (MCTS) [1] planning to choose actions at each time step based on the future model which was trained on the agent-independent environment only.

1. At each decision time step, the agent queries the two-stage model for one-step ahead predictions, given spatial conditioning in the form of the Z representation of the previous 4 observations.
2. The agent performs MCTS rollouts on the predicted future environment frames to choose an appropriate action.

Performance

We compare agents using our forward model to an agent which has access to an oracle of the environment. The oracle agent is used as an upper-bound on performance, as although this perfect representation of the future environment is not available in realistic tasks it is the theoretical best we can expect generative model to do.

Code and example playout gifs can be found online at:
`http://github.com/johannah/trajectories`

Steps	1				3				5				10			
Model	G	T	D	S	G	T	D	S	G	T	D	S	G	T	D	S
2×O	100	0	0	34x±17	100	0	0	36±18	100	0	0	45±28	100	4	0	68±49
2×M	78	0	22	33±17	88	0	12	40±18	91	2	7	65±40	52	25	23	111±67
2×5S	84	0	16	34±17	**94**	1	5	46±27	89	5	6	75±51	55	23	22	112±70
2×10S	85	0	15	35±18	88	0	12	46±26	89	9	2	76±56	55	31	14	124±68
1×O	72	25	3	187±151	67	32	1	209±154	60	40	0	224±64	66	34	0	216±156
1×5S	31	3	55	97 ± 107	46	21	33	196±153	41	3	27	259±155	39	46	15	294±143

Table: This table compares agents using MCTS for forward planning on varying models (O is oracle and our VQ-VQE+PCNN approach with varying levels of sampling from the generative model represented by S or midpoint M), rollout lengths (1, 3, 5 and 10), and agent speed (2X agents are twice as fast as the goal and 1X agents are the same speed as the goal). All agents were tested over the same set of 100 random episodes, with MCTS performing 100 rollouts at each decision time. The values in columns *G*, *T*, and *D* stand for the number of games in which the described agent reached the goal (*G*), ran out of time before reaching the goal (*T*), or died (*D*) by running into an obstacle. The *S* column describes the number of steps completed on average by an agent, calculated only from episodes in which the agent avoided dying (smaller is better), along with the standard deviation. When tested on the same episodes, a random agent reached the goal once at 2X speed and never at 1X speed.

Example

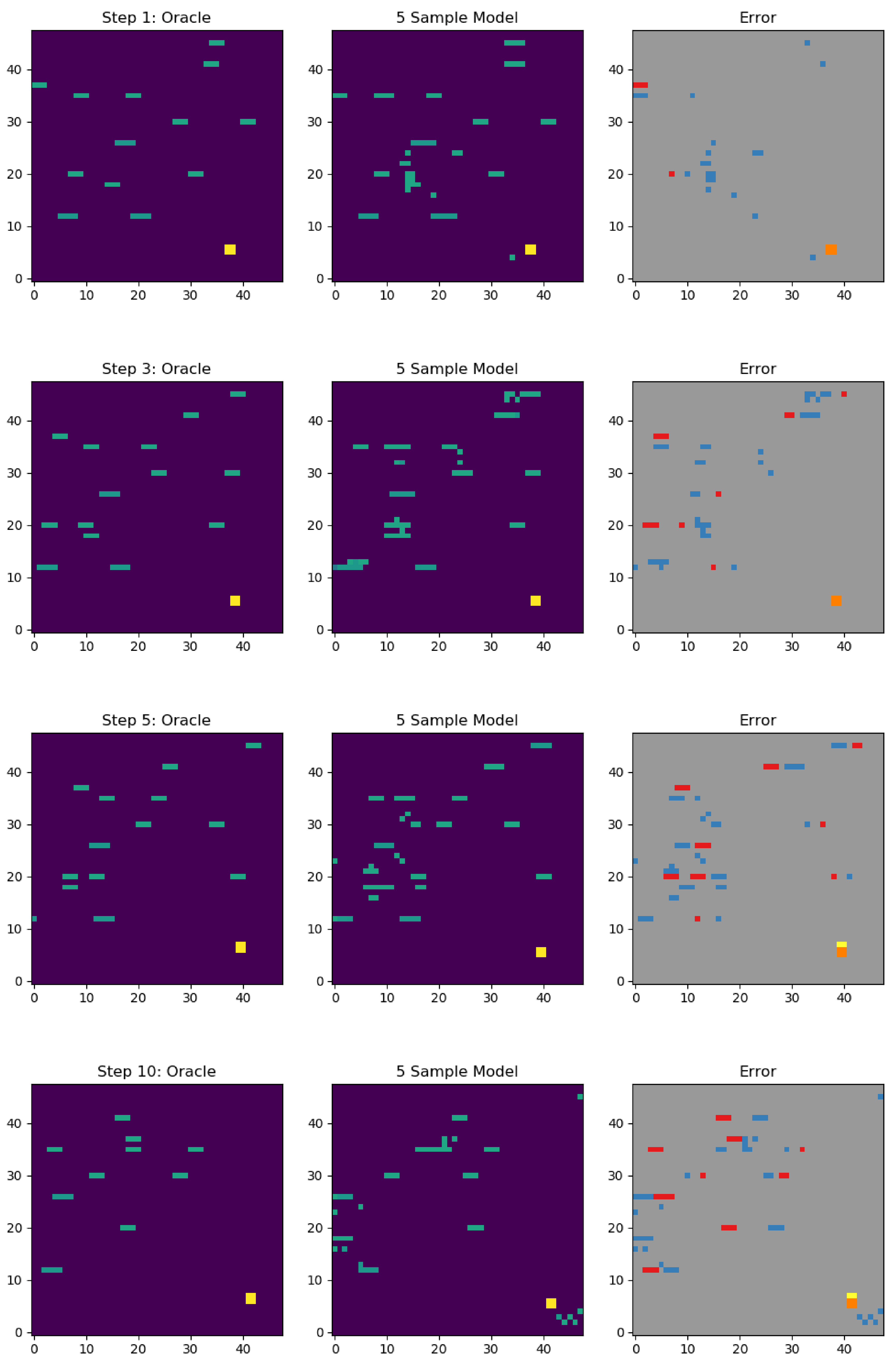

Figure: This figure illustrates forward rollout steps by the oracle (left column), our 5 sample model (middle column), and the error in the model (right column). The number of steps from the given state *t* is indicated in the oracle plot's title. In the first two columns, free space is violet, moving obstacles are cyan, and the goal is yellow. In the third column, we illustrate obstacle error in the model as follows: false negatives (predicted free space where there should be an obstacle) are red and false positives (predicted obstacle where there was free space) are blue. The true goal is plotted in yellow and the predicted goal is plotted in orange (perfect goal prediction is orange).

References

L. Kocsis and C. Szepesvári.
Bandit based monte-carlo planning.
In *In: ECML-06. Number 4212 in LNCS*, pages 282–293. Springer, 2006.

A. van den Oord, N. Kalchbrenner, L. Espeholt, k. kavukcuoglu, O. Vinyals, and A. Graves.
Conditional image generation with pixelcnn decoders.
In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4790–4798. Curran Associates, Inc., 2016.

A. van den Oord, O. Vinyals, and k. kavukcuoglu.
Neural discrete representation learning.
In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6306–6315. Curran Associates, Inc., 2017.

Johanna Hansen [1] Kyle Kastner [2] Aaron Courville [2 3] Gregory Dudek [1]

Mobile Robotics Lab, McGill University [1] Mila, Université de Montréal [2] CIFAR Fellow [3]