$\mathcal{T}$wist Rollout

Volodymyr Kyrylov
volodymyr.kyrylov@usi.ch

June 2, 2023

Abstract

This work explores agent environment modeling by learning a language model over discrete codes of 3 million actions and observations using neural networks. The code is available at https://github.com/proger/twist-rollout. Video demo is available at https://youtu.be/X_1c_l8gb8Q.

1 Introduction

Reinforcement learning agents can learn to perform tasks by maximizing state-value function from scratch [Mnih et al., 2013]. By maximizing cumulative reward alone the agent implicitly learns the model of the world. In this work I'm exploring explicitly pretraining the world model by modeling camera observations conditioned on control inputs.

The project has been performed in five stages: task & environment design, data collection, image tokenizer training, world model training, and simulation.

2 Task and Environment

An agent implemented as a ROS [Macenski et al., 2022] node performs random exploration by executing random trajectories. Simulated Thymio [Guzzi et al., 2018] goes forward until is hits a wall, performs a rotation maneuver for random amount of time and continues motion. Thymio can randomly choose to perform a rotation maneuver in the middle of any trajectory. All robot observations are recorded alongside control inputs, and the task of the model is to reproduce an observation given past observations and control inputs, as shown on Figure 1.

I modified a room-like Coppelia [Rohmer et al., 2013] environment provided to me by the Robotics course team for a Thymio assignment by mapping random textures on walls and laying carpet, as shown on Figure 2. Texturing is used to provide additional information to the model.

3 Data Collection

The agent node is logging all camera observations to a gzipped numpy array after downsampling to $64 \times 64 \times 3$ using nearest neighbors, storing each color value as unsigned 8-bit integer. On each camera message an agent makes a decision on a forward, left or right twist and logs that control message to another gzipped binary

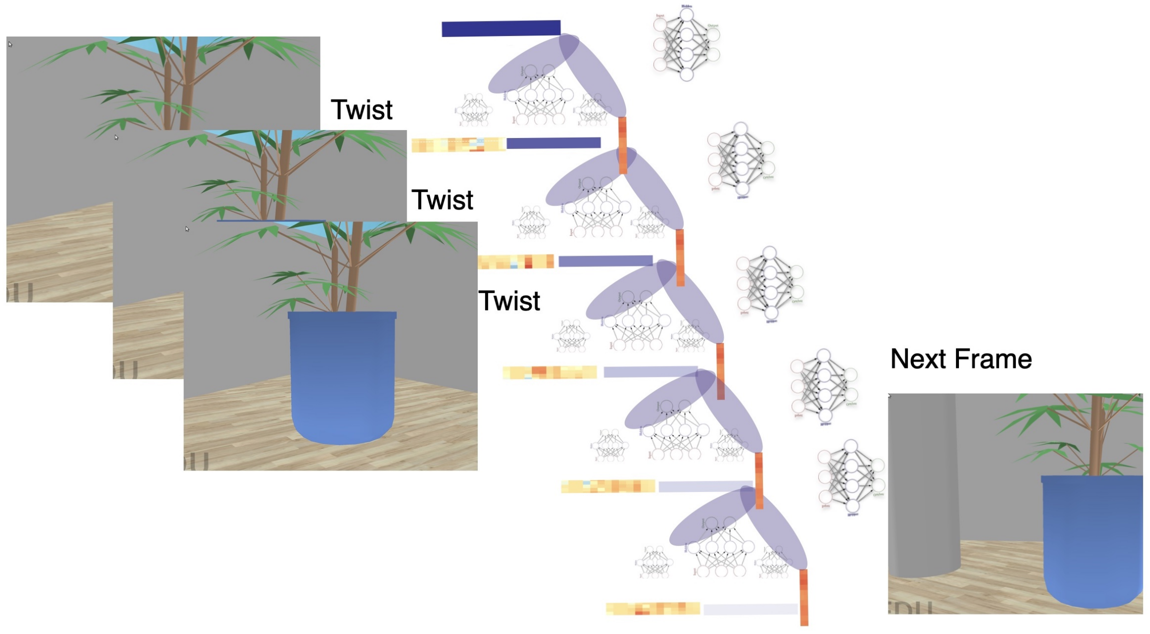

Figure 1: Task Illustration. Given a history and a current twist, we output the next frame.

numpy array. Odometry and sonar proximity messages are also logged to separate gzipped binary files.

I made the decision to log messages using gzip and numpy after experimenting with ROS bags: `ros2 bag` would fill up 10 gigabytes disk space in mere 10 minutes of simulation and compression did not seem to apply to images, making it unfit for data collection in a constrained environment.

I ran data collection for 3 million frames that took about 10 hours to collect overnight.

4 Image Tokenizer

Image representation required careful consideration, because architecture choices impact the design of the world model. Continuous image inputs are natural to work with and are easily handled by convolutional neural networks.

However, *outputs* are a bit more tricky: the task of image prediction can be performed either by regression or by classification. The former is straightforward to implement at the cost of inability to represent a probability distribution over outputs, as the network outputs a single image. Cross entropy-based classification needs a finite set of possible outputs. Step-by-step distributions over outputs allow sampling possible futures and exploring uncertainty.

It's possible to represent continuous distributions using mixture density networks, as shown in [Graves, 2013] and applied to world modeling in [Ha and Schmidhuber, 2018]. Mainstream iamge sequence modeling methods are based on VQ autoencoders, such as [Esser et al., 2020] and applied to world modeling in [Micheli et al., 2022].

[Micheli et al., 2022] implements a system to train reinforcement learning agents on Atari games by interacting with a Gym [Brockman et al., 2016] environment and training tokenizer, world model and actor-critic systems in stages. A *tokenizer* is a VQ-VAE is trained to represent images as sequences of 16 integers.

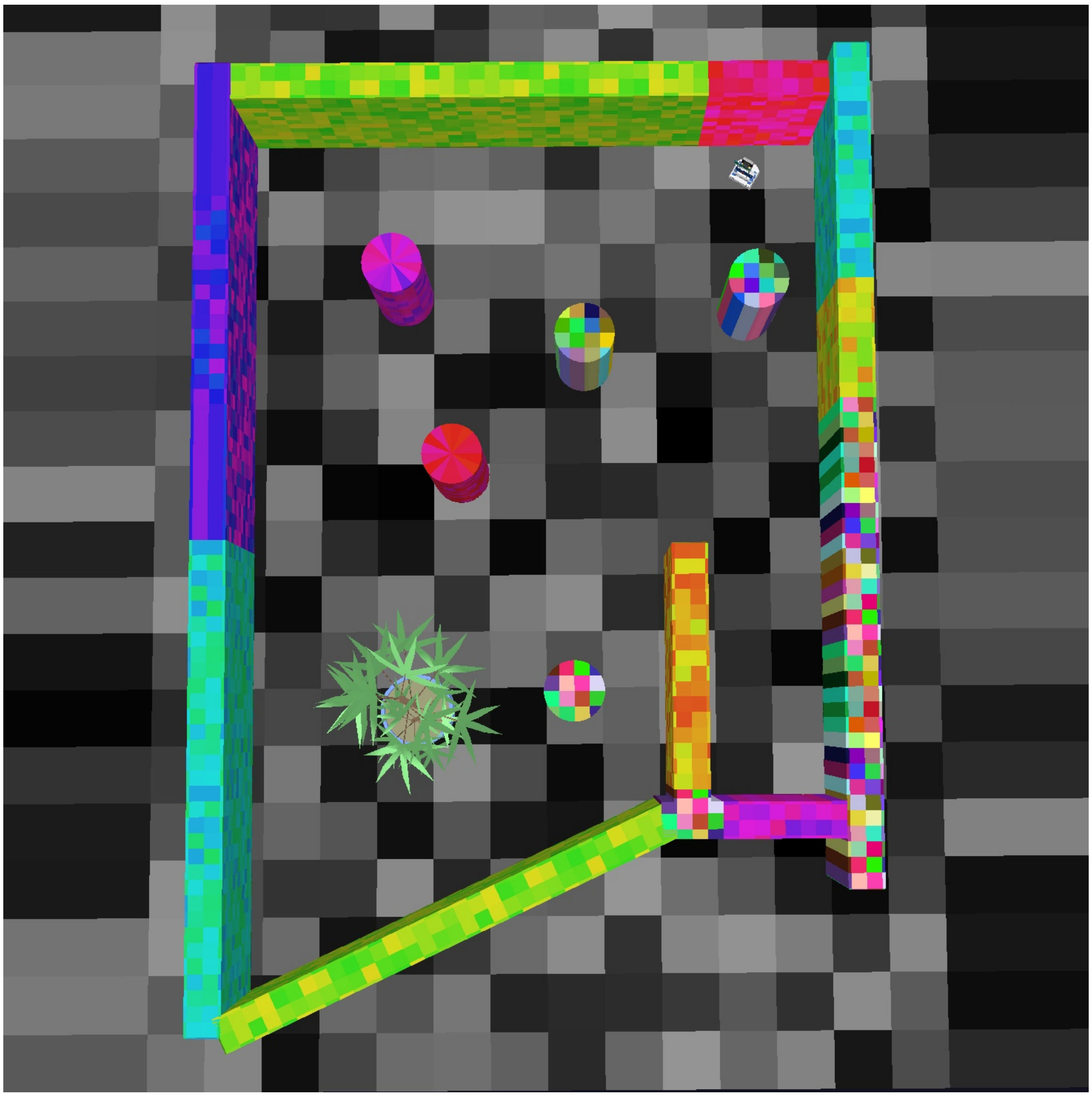

Figure 2: Environment with textured surfaces. Two opposite walls have the same texture.

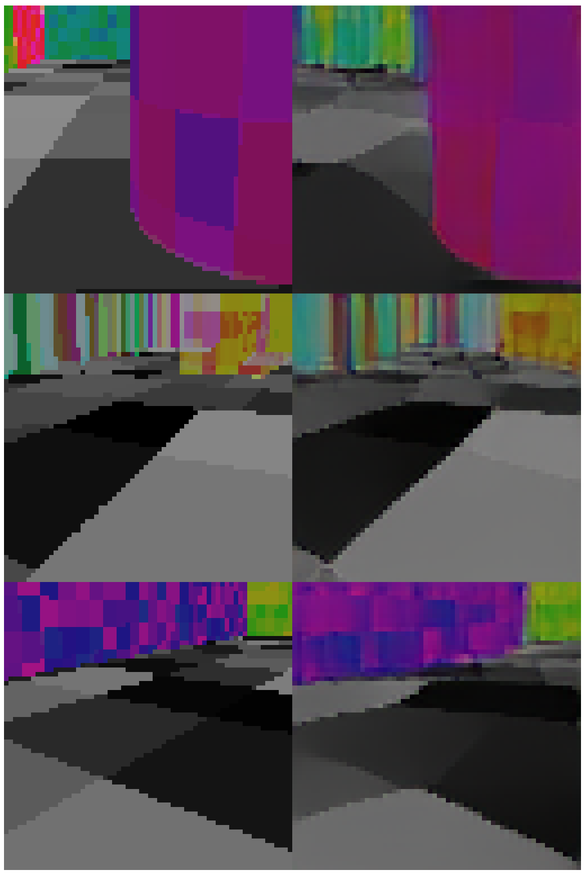
Figure 3: Observation images on the left, VQ-VAE reconstructions on the right.

I've implemented a stub Gym enviroment that reads observations from a file in a fork of `https://github.com/eloialonso/iris` and trained the tokenizer along with the rest of the modules for a bit under 44 hours on RTX 3090. The training run is logged on wandb: `https://wandb.ai/stud76/iris/runs/ck5jz3ql`. Figure 3 shows reconstruction examples.

Once the tokenizer is trained, all collected environment observations are converted using the VQ-VAE encoder to sequences of integers, mixed with manually quantized twist observations in a integer training file for the world model.

5 World Model

I chose an 3-layer LSTM with 1024 units per layer trained on autoregressive sequence generation by supervising each integer on the next one in the sequence, inspired by [Graves, 2013]. The model was trained for 50 epochs with batch size 32 and truncated BPTT of 512 steps, 50 epochs with truncated BPTT of 2048 steps, 50 more epochs with batch size 64, and 100 more epochs with truncated BPTT of 4096 steps. I based an implementation of the world model on my own language modeling code: `https://github.com/proger/haloop`. All training runs for the world model are logged here: `https://wandb.ai/stud76/twist`.

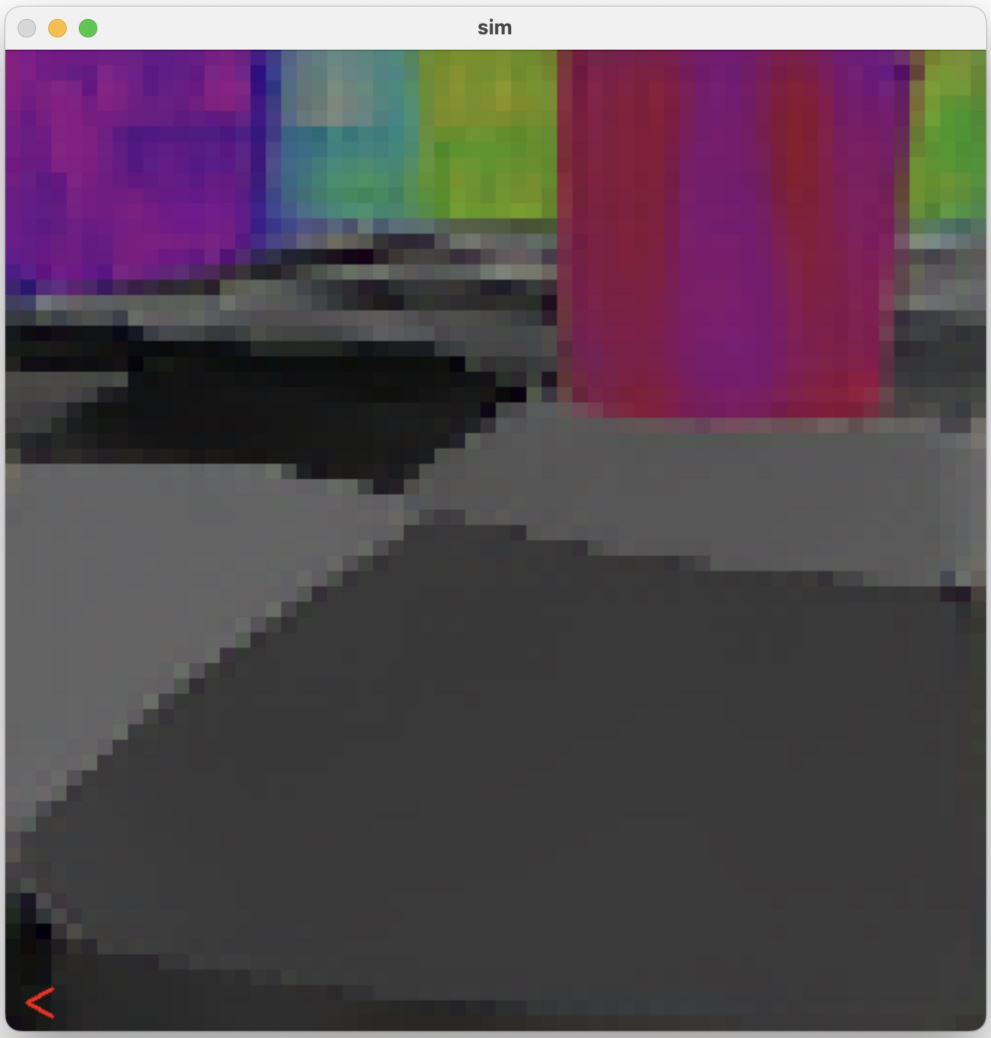

Figure 4: Simulation window. The arrow on the bottom left shows last taken action.

6 Simulation

Simulation debugger is implemented as a python script that loops over user input using OpenCV [Bradski, 2000]. It loads image tokenizer and world models, prepares the hidden state of the world model and converts arrow key input into world model inputs. For each arrow key input the simulator reads out 16 image tokens and renders them using the tokenizer decoder, as shown on Figure 4.

7 Discussion

Video demo is available here: https://youtu.be/X_1c_l8gb8Q. The RNN successfully models short-term transitions in the world.

Due to aliasing in the environment (e.g. two opposite green walls, red walls in different locations or two sides of the yellow wall seen on Figure 2) the agent can be "teleported" to different parts of the world. The floor carpeting ended up being too uniform to disambiguate aliased locations and RNN hidden state in the current iteration does not contain enough information to avoid teleporting.

Streaming compression allowed for significant disk space savings, however a bug in pose saving code caused loss of pose information. I chose not to use global pose information in training due to time constraints. I would like to explore world models that contain latent pose variables in the future — knowing true global pose seems like luxury.

References

[Bradski, 2000] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools.*

[Brockman et al., 2016] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *ArXiv*, abs/1606.01540.

[Esser et al., 2020] Esser, P., Rombach, R., and Ommer, B. (2020). Taming transformers for high-resolution image synthesis.

[Graves, 2013] Graves, A. (2013). Generating sequences with recurrent neural networks. *ArXiv*, abs/1308.0850.

[Guzzi et al., 2018] Guzzi, J., Giusti, A., Di Caro, G. A., and Gambardella, L. M. (2018). Mighty thymio for higher-level robotics education. In *AAAI*.

[Ha and Schmidhuber, 2018] Ha, D. R. and Schmidhuber, J. (2018). World models. *ArXiv*, abs/1803.10122.

[Macenski et al., 2022] Macenski, S., Foote, T., Gerkey, B., Lalancette, C., and Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074.

[Micheli et al., 2022] Micheli, V., Alonso, E., and Fleuret, F. (2022). Transformers are sample efficient world models. *ArXiv*, abs/2209.00588.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602.

[Rohmer et al., 2013] Rohmer, E., Singh, S. P. N., and Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326.