

ColorGridWorld: A Miniaturized Test Environment For Natural Language-Instructed Agents

Michael Ogezi
Department of Computer Science,
University of Alberta, Edmonton, AB.
Email: ogezi@ualberta.ca

Abstract—This work seeks to train simulated agents to dynamically align their goals with natural language instructions. We create a modified GridWorld environment which we call ColorGridWorld and give our agent instructions via the environment state. Our results show that our agent can learn useful and partially transferable knowledge representations. This allows the agent to modify ad hoc its behaviour and goals based on the natural language instructions given to it at the start of each episode.

Keywords—Robotics, Reinforcement Learning, Natural Language Processing, Natural Language-Instructed Robotics.

I. INTRODUCTION

Over the years, robots have taken on increasingly important roles in our daily lives. Tools such as modern-day cars and automated vacuum cleaners fall under the purview of robotics. However, how we communicate with these robots and tell them what to do remains largely rule-based. Let us consider an automated vacuum cleaner such as the *Roomba 800* series. It has two active cleaning modes, namely, clean mode and spot mode. In clean mode, the *Roomba* analyses a room and tries to clean every part of it. In spot mode, the *Roomba* focuses on a small area. To activate either of these modes in the *800*, you have to press its mapped button. A more intuitive way to communicate with the *Roomba* would be to provide it with natural language commands. So, instead of pressing a button, you could say: “*Roomba*, clean this room” or “*Roomba*, clean this spot”. We would expect these verbal commands to activate clean mode and spot mode respectively.

In this work, we primarily seek to synthesize an actionable framework that allows robots to correctly carry out open-ended natural language commands that we present to them. We note that although current state-of-the-art (SOTA) automatic speech recognition (ASR) systems are imperfect, we will assume they are since they are not the core focus of this work. We seek to convert the natural language instructions into a series of physical actions carried out by the robot in the context of its surrounding environment.

II. MOTIVATION

To get everyday people to buy into robotics, they must be easy to use. Think about mobile phones. There was

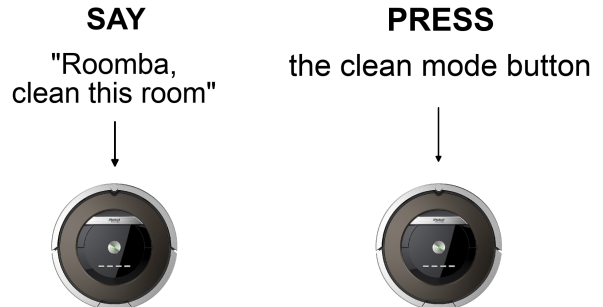


Fig. 1. **Planned** approach

Fig. 2. **Traditional** approach

no true mass adoption until proper touchscreen interfaces became the norm. Before then, mobile phones were a niche product used by the affluent and hobbyists. To go mainstream, consumer robotics must follow a similar path.


That said, the problem of getting robots to understand their environment and ground natural language instructions in that environment seems like a daunting one. Perhaps, only strong AI solutions will be able to do this. However, since we have no proof that this is the case, even with the short nature of this work and our limited resources, we must thoroughly experiment to see how far we can go.

III. PROBLEM FORMULATION: COLORGRIDWORLD

We formulate our task through ColorGridWorld. Much like GridWorld, the agent can be in only one tile before and after each step. However, in this case, the tiles are coloured. We go into greater detail in explaining our formulation below.

A. Basic Formulation Elements

At the most basic level, our formulation consists of the following:

- $C = \{\text{red, green, yellow, blue, magenta, cyan, white}\}$, which represents the 7 possible tile colors.
- An $N \times N$ grid, G , consisting of N^2 tiles, each independently colored with a random $c \in C$.
- A learning agent, , and its current position, p , in G .

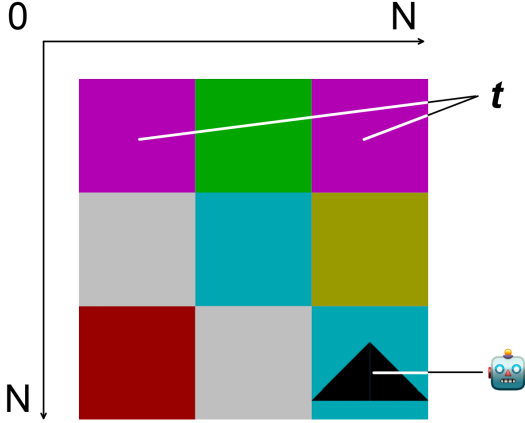


Fig. 3. An example of a 3×3 board for ColorGridWorld with the agent as a black triangle at $p = p_0$, $t = \text{magenta}$, and $N = 3$.

- A target colour, t , indicating the colour of the tile that the agent needs to go to. Note that t could consist of multiple valid positions if there are multiple instances of the target colour in G .
- An natural language instruction, i , telling the agent which colour of the tile to stop at. Eg. $i = \text{"Go to red"}$.
- The environment state, S , consisting of a combination of $f_e(i)$, G , and p . Here, f_e encodes the text into an embedding.
- $A = \{\text{up, right, down, left, stop}\}$, which represents the action space.

B. Task Initialization

Our task is both episodic and stochastic. At the start of each episode, we do the following:

- 1) Reset G and randomly select new colors for the N^2 tiles.
- 2) Select a target color, t , from the tile colors.
- 3) Build the natural language instruction, $i = \text{"Go to } t\text{"}$.
- 4) Randomly select a starting position for our agent, a .

C. Goal

At each step, the agent can select an action $a \in A$. The ultimate goal is for the agent to condition itself on both i and G in order to chart a good path to t , then *stop*. This must be done within the time constraint, τ . τ is typically set as:

$$\tau = 2(N - 1) + 1 \quad (1)$$

The expression in Equation 1 is the largest number of steps required by a fully efficient agent to reach the goal.

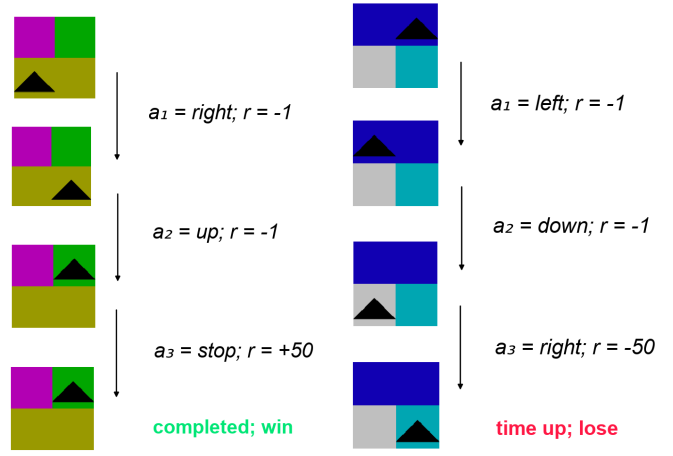


Fig. 4. The agent wins ($t = \text{green}$)

Fig. 5. The agent loses ($t = \text{cyan}$)

D. Rewards

- A **win** happens when the agent successfully stops at t . The reward is $+50$.
- An **invalid stop** happens when the agent tries to *stop* at a non- t tile. The reward is -5 .
- A **move** happens when the agent takes a non-*stop* action ie $a \in A \setminus \{\text{stop}\}$. The reward is -1 . This encourages the agent to make rapid progress.
- An **invalid move** happens when the agent takes an action that would take it out of G . For example, moving *up* on the top row, or *down* on the bottom row. The reward is -1 , the same as a normal move's.
- A **loss** happens when the agent has not completed its task after τ steps. The reward is -50 .

E. Instruction Embeddings

Neural networks cannot process text as is, therefore, we must convert our natural language instruction, i into an embedding vector $f_e(i)$. For this, we use contextual BERT [1] embeddings, which have a strong track record of generating deeply contextual representations [3]. We use a mean pool of the final layer's hidden states to convert the embeddings for each token to a sentence embedding for all of i . The BERT configuration we employ nominally referred to as BERT *tiny*, is trained on uncased English, has two layers, two heads and a 128-length hidden state.

IV. EXPERIMENTS

With our experiments, we aim to rigorously check our hypothesis that an agent can learn to align changing natural language instructions with its goals in a stochastic setting. We trained a Deep Q-Network [5] to predict the best state-action q based on $S = \text{concat}(G, f_e(i), p)$. We linearly annealed ϵ in ϵ -greedy from 0.95 to 0.05, and had our discount factor, $\gamma = 0.99$.

A. Results

The chance baseline for both the 2×2 and 3×3 configurations is approximately a **15%** win rate. The

chance baseline always takes a random action with uniform probability. Note that while one-hot representations demonstrably perform better, the results for the 3×3 section of TABLE I use index-based color representation which performs **13.74%** worse in the 2×2 case. We did not have time to complete the 3×3 experiment with one-hot representations, but extrapolate that it would perform better than what obtains in TABLE I. We set τ based on Equation 1.

TABLE I
WIN RATES FOR DIFFERENT N -CONFIGURATIONS OVER 10^6 STEPS.

$N \times N$	τ	Progress Range	Steps	Win
2×2	3	0-25%	107,856	30.28%
		25-50%	107,856	54.86%
		50-75%	107,856	76.19%
		75-99%	103,542	92.46%
		99-100%	4,314	99.33%
		0-100%	431,426	63.51%
3×3	5	0-25%	51,792	13.41%
		25-50%	51,792	10.11%
		50-75%	51,792	6.68%
		75-99%	49,721	2.40%
		99-100%	2,072	0.10%
		0-100%	207,169	8.13%

B. Interpretations

We see that the 2×2 configuration performs very well and can learn to align the natural language instruction with the stochastic ColorGridWorld environment. This shows the promise of our hypothesis.

Unfortunately, we have considerably worse results with the 3×3 configuration. It even performs worse than chance and trends worse as our ϵ cools off. Of course, we had time and compute constraints, but those may not fully explain the poor performance. We suspect that our same-size model finds it harder to learn with larger values of N , as we used the same neural network architecture for both configurations.

C. Error Analyses and Possible Ways Forward

- 1) **CPU-only Training:** In general, there seems to be a positive correlation between the agent’s win rate and the number of steps. Since we used a version of Tensorflow (2.8) that was incompatible with our system-wide CUDA installation (11.4), we couldn’t train on our GPUs. This, unfortunately, limited our throughput and the extent to which we could carry out exploratory experiments. We also could not run an important experiment for the 3×3 configuration as we explained above. In future work, we look to rewrite our experiments to make use of an earlier Tensorflow version or make use of Tensor Processing Units (TPUs) [4].
- 2) **Color Encoding Formats:** We tried out different representations for tile colours in our grid. Some of them are:

- a) 8-bit: Here, the first 3 bits represent red, the second three represent green, and the last two represent blue.
- b) Index-based: Here, we give each colour an id based on its index in the C . The first colour is 0 while the last colour is 6.
- c) One-hot-based: Here, each colour is represented by an $|C|$ -length array where all elements are zeros except at the index corresponding to the colour. We went with this because it gave the best results, but likely at some cost to zero-shot generalization.

In our experiments here, one-hot-based seemed to perform best, however, we seek to test out other methods for representing the colours since different schemes affect the quality of learning.

- 3) **Zero-shot Color Generalization:** This indicates the ability of our system to navigate to colours that it had not seen during training. Although we performed significantly better than chance, our system showed limited skill in this area. In future work, we seek to examine this more closely and unearth ways to improve generalization. This may be related to the earlier point about colour representation formats. Another potential avenue would be to represent the grid state in a visual format and pass that to a convolution neural network (CNN) [6] or visual transformer (ViT) [2].
- 4) **Embedding Similarity:** Instruction embeddings for different instructions are very similar. The instructions for $t = \text{red}$ and $t = \text{blue}$ have embeddings with a cosine similarity greater than 0.85. This similarity slows down learning considerably, and it is a known problem without a good solution.

V. CONCLUSION AND FURTHER WORK

In general, our results are positive. We initially set out to teach an agent to follow instructions in a stochastic environment. We found success in this, albeit only for the 2×2 configuration. However, the ability of our agent to zero-shot generalize to unseen colours leaves more to be desired. We look to tackle this in future work. We make all our source code available here: <https://github.com/mikeogezi/ColorGridWorld>

ACKNOWLEDGMENTS

We thank Professor Greg Kondrak, Bradley Hauer, and Karim Ali for administering the compute resources that made our extensive experiments possible.

We also thank Professor Matt Taylor and Professor Cynthia Matuszek for the ideas that they shared during their lectures helping to inspire this work.

REFERENCES

- [1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirec-

tional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

- [2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. URL <https://arxiv.org/abs/2010.11929>.
- [3] Kawin Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1006. URL <https://aclanthology.org/D19-1006>.
- [4] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmamghami, Rajendra Gotipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit, 2017. URL <https://arxiv.org/abs/1704.04760>.
- [5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- [6] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks, 2015. URL <https://arxiv.org/abs/1511.08458>.