Rearranging the Environment to Maximize Energy with a Robotic Circuit Drawing

Xianglong Tan1,*, Zhikang Liu2,*, Chen Yu3, Sören Schwertfeger3 and Andre Rosendo4

Abstract—The ability of acquiring energy form environments is greatly beneficial for robot to function in uncertain environments. In this work, we present a robot capable of drawing circuits with conductive ink while also rearranging the visual world to receive maximum energy from a power source. A range of circuit drawing tasks is designed to simulate real-world scenarios, including avoiding physical obstacles and regions that would discontinue drawn circuits. We adopt the state-of-the-art Transporter networks for pick-and-place manipulation from visual observation. We conduct experiments in both simulation and real-world settings, and our results show that, with a small number of demonstrations, the robot learns to rearrange the placement of objects (removing obstacles and bridging areas unsuitable for drawing) and to connect a power source with a minimum amount of conductive ink. As autonomous robots become more and more involved in our daily lives, our proposed approach brings a novel way for machines to keep themselves functional by rearranging their surroundings to create their own electric circuits.

I. INTRODUCTION

Recent advances in Robotics have led to progress in various areas and naturally raise the demand for robots to become more autonomous, intelligent and long-lasting [1], [2], [3]. However, as the computing power of robots increases, robots consume more energy to keep functional. Novel approaches are needed to solve the challenges of keeping robots work longer with limited storage capacity of batteries [4].

Previous work on robotic self-powered ability focus on the design of power systems. Solar panels are widely-used for robots to harvest energy from ambient sources to recharge batteries. Solar panels are normally low-cost and light-weighted but suffer from low efficiency and restrictive application environments. In a series of works on energy harvesting [5], [6], [7], researchers demonstrate the powering of nanobots from ambient mechanical energy. In a more recent work [8], a novel robotic power system is powered by scavenging energy from external metals. However, in these works the power harvested from the environment is very limited, and it can be hardly used to power any standard robot.

Manipulation of objects and rearrangement of environments have a long history in robotics. Traditional pick-and-place solutions follow a standard pipeline: object recognition, pose estimation, and then grasp planning. For instance, [9] used handcrafted image features for object perception followed by computation of pose estimation for suction objects. Recent data-driven solutions for manipulation tasks involve imitation learning (IL) and reinforcement learning (RL). [10] used meta-learning as a pre-training procedure that uses demonstrations from different environments. The method is shown to be able to achieve one-shot imitation learning for novel scenarios and scale to raw pixel inputs. [11] also achieve one-shot imitation on a block stacking task. They used soft attention for processing both the state-action sequence that corresponds to the demonstration, and for processing the components of the vector specifying the locations of various blocks. RL algorithms have also shown their ability on learning reliable policy end-to-end with raw image inputs. [12] proposed an RL algorithm that is entirely self-supervised, using only grasp outcome labels that are obtained automatically by the robot for the grasping task. [13] performed RL to learn complementary pushing and grasping policies that operate end-to-end from visual observations to actions. Besides using IL or RL, [14] also proposed a framework called Transporter Networks that rearranges deep features to infer spatial displacements from visual input, which can parameterize robot actions. Experimental results show that it is orders of magnitude more sample efficient than a variety of end-to-end baselines.

This paper presents a novel approach for robots to survive in resource-limited environments. Instead of using wires or cables, the robot learns to rearrange environmental obstacles and useful resources to build an optimal electrical path to a power supply. We develop a path planning algorithm for 3D circuit drawing and adopt a novel network, capable of selecting pick and place pose for the robot to reposition objects. This paper shows that robots can rearrange the physical world to receive energy from the environment to stay alive.

II. ROBOTIC CIRCUIT DRAWING

A. Conductive Ink

Graphene-based conductive ink has shown great potential in printing flexible electronics [15], [15]. Compared to metal-based conductive ink, it is low cost, low-toxic, environment friendly, and easy to make and store [16].

B. Simulation Setup and Tasks

The simulation is set up with PyBullet [17]. We use a Kinova 6 DoF Jaco robot arm to perform circuit drawing on a 0.8×0.6m workspace. The two-finger gripper of the robot

1Hamlyn Centre, Imperial College London, UK.
2Department of Computer Science, University of York, UK.
3School of Information Science and Technology, ShanghaiTech University, China.
4Robotics Engineering, Worcester Polytechnic Institute, USA.
*These authors contributed equally to this work.
enables picking and placing of objects. The electrodes of the robot and the power source are placed on opposite sides of the workspace. The main goal is to draw continuous ink to connect the electrode pair of the robot and the power source.

Circuit drawing tasks are selected based on the possible situation a robot confronts in complex real-world environments. For example, in search and rescue missions, the robot needs to automatically plan the drawing path avoiding obstacles and areas that would discontinue or damage the circuit (e.g., huge bumps, holes, surface with liquid, sand, etc.). We refer to these areas as forbidden zones in this paper. Meanwhile, drawing longer the circuit path leads to higher circuit resistance and more ink and energy used. If any blocking obstacles can be moved by the robot, they should be removed before the drawing. Moreover, if movable bridge-like structures are presented in the workspace, reposition them on top of any forbidden zones could enable drawing above these zones to connect more efficiently. The details of circuit drawing tasks are shown in Fig. 1.

III. METHODS

A. Transporter Networks

We consider the pick-and-place problem with actions \( a_t \) defining the pick and place poses and states \( s_t \) represented by visual observations:

\[
\pi(s_t) \rightarrow a_t = (T_{\text{pick}}, T_{\text{place}}) \in A, \tag{1}
\]

where \( T_{\text{pick}} \) is the pose of the end effector used to pick an object, and \( T_{\text{place}} \) to place the object, where \( T_{\text{pick}}, T_{\text{place}} \in \mathbb{R}^2 \) are 2D coordinates. In our task, the goal is to pick up the bridge-like blocks then place them over the forbidden zones while picking up other blocks then placing them out of the working area.

Other than using end-to-end RL for solving the pick-and-place task [18], [19], [20], [13], we use Transporter Networks [14] to recover the distribution of successful pick poses and the corresponding distribution of successful place poses.

The distributions are trained with pure visual observations, which allows no assumption of prior information about the picking objects. The state \( s_t \) in our experiment is a grid of pixels reconstructed from RGB-D information. We use the pick-and-place actions in the space of 2D translations. We discretize the space of SO(2) rotations into \( k \) bins where the input visual observation \( s_t \) are rotated. \( s_t \) is therefore defined on a grid of \{\( \{x, y, w\} \}\} where \( w \) describes the rotation.

Denote such a pose \( \{x, y, w\} \) as \( \xi \). We use Q value function \( Q_{\text{pick}}(\xi|s_t) \) to model the distribution of the returns among different picking positions and \( Q_{\text{place}}(\xi|s_t, T_{\text{pick}}) \) for placing positions.

Learning Picking and Placing After the Q function \( Q_{\text{pick}}(\xi|s_t) \) is trained by a fully convolutional network (FCN), the picking position can be obtained by

\[
T_{\text{pick}} = \arg\max_{\xi} Q_{\text{pick}}(\xi|s_t). \tag{2}
\]

Denote \( s_t[\xi] \) as a partial crop centered on a pose \( \xi \). We search for the best position for placing that match with the highest feature correlation. The Q function for placing can then be written as

\[
Q_{\text{place}}(\xi|s_t, T_{\text{pick}}) = \psi(s_t[T_{\text{pick}}]) \ast \phi(s_t)[\xi] \tag{3}
\]

where \( \psi(\cdot) \) and \( \phi(\cdot) \) are dense feature embeddings from two deep models. Similarly, the pose of placing is hence obtained by

\[
T_{\text{place}} = \arg\max_{\xi} Q_{\text{place}}(\xi|s_t, T_{\text{pick}}). \tag{4}
\]

Network Architecture We use visual observation as the state \( s_t \), which is a \( 320 \times 240 \) RGB-D image, where each pixel contains both color and distance information. This enables our models to learn both texture features and shape features. The picking model is a FCN that takes the state \( s_t \in \mathbb{R}^{H \times W \times 4} \) as input and outputs dense pixel-wise state value \( V_{\text{pick}} \in \mathbb{R}^{H \times W} = \text{softmax}(Q_{\text{pick}}(\xi|s_t)) \). Specifically, the picking model is an hourglass network with encoder-decoder architecture. It is a 43-layer residual network [21]

![Fig. 1. The four basic circuit drawing tasks in this paper: path planning for circuit drawing (a), pick and remove obstacles (b), reposition ramps to draw through a forbidden zone (brown area) (c), reposition ramps to draw above circuits (d). The workspace is a 0.8×0.6m platform with four electrodes (two from the robot arm and two from a power source) on each side (a1). The basic goal for the robot is to connect electrodes with conductive ink (a2) while avoiding obstacles (a3) and forbidden zones (a4). In task (b), the robot should pick the obstacle (b2) and place it in a zone away from the center (b3) to draw the shortest circuit between electrodes (b4). In task (c), when a forbidden zone is presented (c1), the robot is required to pick a ramp (c2) and place it in the zone (c3) to allow drawing circuit through the zone (c4). In task (d), the electrodes of the robot and the power source are in the opposite position, the robot needs to connect a pair of electrode first (d1), then pick a ramp (d2) and place on the drawn circuit (d3) to connect the rest electrodes without shortcircuiting the power source (d4).](image-url)
with 12 residual blocks, 8-stride, and image-wide softmax. Dilation [22] and ReLU activations [23] are implemented on all layers except the first. It acts as an attention mechanism. The placing model is a two-stream FCN that takes the state \( s_t \in \mathbb{R}^{H \times W \times d} \) as input and outputs two dense feature maps: \( \psi(s_t) \in \mathbb{R}^{H \times W \times d} \) and \( \phi(s_t) \in \mathbb{R}^{H \times W \times d} \), where \( d \) is the number of feature dimensions. The architecture of the placing model is similar to the picking one but without ReLU activations in the last layer. Once the picking pose is obtained by \( T_{\text{pick}} = \arg\max \hat{y}_{\text{pick}} \), the partial crop with size \( c \), \( \psi(s_t[T_{\text{pick}}]) \subset \mathbb{R}^{c \times c \times d} \) centered around \( T_{\text{pick}} \) is transformed by \( \xi \) and cross-correlated with the feature map \( \phi(s_t) \) according to (3). The pixel-wise placing state value can then be obtained by \( y_{\text{place}} \in \mathbb{R}^{H \times W \times k} = \text{softmax}(Q_{\text{place}}(\xi | s_t, T_{\text{pick}})) \). \( T_{\text{place}} = \arg\max y_{\text{place}} \) is the final placing pose that has the highest correlation with the two feature partial crops.

**Learning from Demonstrations** We use the rewards \( \mathcal{R} \) to collect data for a dataset \( \mathcal{D} \) of \( n \) expert-demonstrated trajectories \( \tau_i \), where each trajectory \( \tau_i \) is a sequence of state-action pairs \( (s_t, a_t) \). During the training process, we sample state-action pairs from the dataset \( D \) serving as \( T_{\text{pick}} \) and \( T_{\text{pick}} \) labels. The one-hot pixel maps \( y_{\text{pick}} \in \mathbb{R}^{H \times W} \) and \( y_{\text{place}} \in \mathbb{R}^{H \times W \times k} \) are generated by two training labels. The loss function is the cross-entropy between these one-hot pixel maps and the state values of picking and placing:

\[
L = -E_{y_{\text{pick}}} [\log y_{\text{pick}}] - E_{y_{\text{place}}} [\log y_{\text{place}}].
\]  

(5)

We used 3D path planning for calculating the reward \( r_t \in \mathcal{R} \).

**B. 3D Path Planning**

The path planning is used to not only make the shortest path of the drawing circuit on the ground, but also be treated as a filter to select the high-quality data samples for training. A demonstration of the path planning algorithm is shown in Fig. 2.

**Information Input** The terrain information will just be a visual observation, which is an orthographic top-down view from the agent, and input as a 2D array, representing the situation of a 3-dimensional working area. It is different from 2-dimensional space, there is an additional consideration of displacement in the z index that should be added when we process the algorithm.

**Algorithm** In this case, An double layer A* search algorithm, which is an informed searching algorithm, has been used. Firstly, we start the A* search from the departure point, but we don’t expand the points to the nearest points. It expands to the available grooves instead, whatever it is uphill or downhill, or choose directly to the destination if it is possible and in the same level. Then, the A* will calculate the distance between two nodes generated in the previous layer in the second layer.

This algorithm fits our path planning, which also serves for drawing circuit lines on the ground. The path is also not allowed to go through any steep edges, including cliffs and precipitous slopes. This is solved by defining a minimum and a maximum value to detect slope values. If the depth difference of two adjacent nodes is less than the minimum slope value, then we can treat it as they are at the same level.
We train the networks for 1k iterations with 1, 10, and 100 demonstrations respectively and test the performance with 50 test samples for all the tasks. In each test, the number of steps for pick and place action equals to the movable objects in the workspace. If the position of electrodes are inverted, the robot will connect one of the electrode pairs one step before the maximum action steps. In other cases, the robot only start drawing circuits after all action steps. A demonstration of the heatmap for pick and place positions after the training is shown in Fig. 4.

D. Sim-to-Real Experiment Setup

The experimental setup is shown in Fig. 3. The workspace is a 0.8×0.6m wooden platform. The cardboard is placed on top of the platform and two metal bars are fixed on each side of the cardboard, representing the terminals of the robot and the power source. The conductive ink is prepared and stored in a glass jar placed on a magnetic stirrer to prevent the graphene from solidification. The glass jar is linked to a soft pipe that is connected to a peristaltic pump. The pump pushes the ink towards a nozzle which is held by a 3D-printed dispenser at the end-effector of the Kinova 6DOF Jaco Arm. An Arduino Uno is used to control the speed of ink flow. The robot arm is controlled by Moveit (https://github.com/ros-planning/moveit). We use one of the successful cases from the all-in-one task, which has two obstacles, one forbidden zone, and inverted electrodes, as the demonstration of the sim-to-real control. To achieve sim-to-real control, we use the objects with the same size and shape trained in the simulation to reconstruct the real-world workspace. Each pick and place pose is given by the trained networks. We used the built-in inverse kinematics solver of Moveit to move the robot to the desire poses.

IV. RESULTS

A. Simulation Tasks

In the simulation environment, we have created and contributed a wide variety of tasks, which is shown in Table IV-A. As the difficulty of the task increases, the number of obstacles and forbidden zones has increased as well. We have trained the network in these simulations for 1000 iterations, with respectively 1, 10, and 100 demos.

Trained models There are two types of models trained from the network, including including the attention model and the transport model, are responsible for picking and placing actions respectively. As the observation information inputs, the model will automatically find the maximum Q value and output the pose, including the position and orientation. Then, these models are able to act.

The reward We have evaluated their performance based on the average reward gained during the 100 times test. The reward refers to how far the length of the path from anode to cathode, and the range is 0-1. The closer the final path length is to the direct length from the anode to the cathode, the greater the reward will be.

Performance Evaluation Overall, we can see the performance improves, as the reward is increasing when we
increase the number of demonstrations trained. The more difficult the task is, the lower reward they will perform if there are only a few demonstrations for training. However, they all perform relatively well as the rewards are close to 1 when the network has been trained with 100 demonstrations.

Among these results, remove-obstacle, which is the simplest task, achieves the best reward all the time, which means it performs perfectly.

B. Real-World Experiments

The real-world demonstration is shown in Fig. 5, although the final placement of the objects and circuit are the same as the simulation result, we confronted difficulties when the robot picked objects that we had to repeat the pick action several times. Examples of successful and failed cases are shown in Fig. 6. Overall, the pick action achieves around 60% success rate from 20 trials.

Fig. 6. Examples of pick actions. Successful (a), Failed - gripper position too high, can not pick objects (b), Failed - gripper position too low, objects will bend and damage the nozzle (c).

V. DISCUSSION

To the best of our knowledge, this is the only work where a robot learns to rearrange visual world to receive energy from self-printed circuits. In our previous work [25], a robot learned to draw circuits with Bayesian Optimization to receive maximum energy from a power source, but no vision sensing or manipulation was applied. We also studied the simulation gap in our real-World experiments. The main issue in real-world implementation is the misplacement in z-axis of the gripper. The original z value is generated from the network and worked well in simulation tasks. In simulation a constrain between the object and the gripper is automatically created when the object is within a small distance threshold from the gripper to guarantee a firm grasp. However, in real cases, tolerance for displacement in z-axis is much smaller than in the simulation. In the original paper of Transporter [14], authors use a suction gripper instead of our two-finger gripper, which is much easier to control but has the limitation that the object surface should be regular. Although the success rate for picking in our real-world experiments is around 60%, the use of a finger gripper is necessary considering irregular objects and structures in real-world scenarios.

VI. CONCLUSION

This paper presents a novel approach for robots to access energy to survive in energy-limited and uncertain environments, through self-drawn electrical connections and rearranging the physical world. A few tasks were performed where a robot needs to rearrange obstacles to receive maximum power from the power supply. With properly designed artificial neural network and training methods, the robot can succeed in these tasks within a small number of trials. This study indicates that robots can survive and be self-sufficient by combining self-drawn electric circuits and machine learning. As robots become more present in our society and even reach other planets, maximizing their capacity to keep themselves online is crucial to increase the odds of success.

Apart from energy, other resources are also crucial for robotic survival. Keeping the usage of material (e.g., conductive ink) in the optimization routine is a further direction to explore. For example, circles outperform lines in reducing resistance of the connection but cost more ink. The robot
should consider the trade-off between the usage of resources and the improvement in received energy. Currently, we only implement circuit drawing on flat surfaces. Applying circuit drawing in 3-dimensional space would be another interesting further direction.

REFERENCES

[1] J. Bongard, V. Zykov, and H. Lipson, “Resilient machines through continuous self-modeling,” Science, vol. 314, pp. 1118–1121, Nov 2006.

[2] J. C. Bongard, “Evolutionary robotics,” Commun. ACM, vol. 56, p. 74–83, Aug 2013.

[3] A. Rosendo, M. von Atzigen, and F. Iida, “The trade-off between morphology and control in the co-optimized design of robots,” PLOS ONE, vol. 12, pp. 1–14, Oct 2017.

[4] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobson, V. Kumar, M. McNutt, R. Merrifield, B. J. Nelson, B. Scassellati, M. Taddeo, R. Taylor, M. Veloso, Z. L. Wang, and R. Wood, “The grand challenges of science robotics,” Science Robotics, vol. 3, no. 14, 2018.

[5] Z. W. Pan, Z. R. Dai, and Z. L. Wang, “Nanobelts of semiconducting oxides,” Science, vol. 291, no. 5510, pp. 1947–1949, 2001.

[6] X. Wang, J. Song, J. Liu, and Z. L. Wang, “Direct-current nanogenerator driven by ultrasonic waves,” Science, vol. 316, no. 5821, pp. 102–105, 2007.

[7] S. Xu, Y. Qin, C. Xu, Y. Wei, R. Yang, and Z. L. Wang, “Self-powered nanowire devices,” Nature Nanotechnology, vol. 5, no. 5, pp. 366–373, 2010.

[8] M. Wang, U. Joshi, and J. H. Pikul, “Powering electronics by scavenging energy from external metals,” ACS Energy Letters, vol. 5, no. 3, pp. 758–765, 2020.

[9] R. Jonschkowski, C. Eppner, S. Höfer, R. Martín-Martín, and O. Brock, “Probabilistic multi-class segmentation for the amazon picking challenge,” in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1–7, IEEE, 2016.

[10] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, “One-shot visual imitation learning via meta-learning,” in Conference on Robot Learning, pp. 357–368, PMLR, 2017.

[11] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, “One-shot imitation learning,” in Advances in Neural Information Processing Systems (I. Guyon, U. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[12] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holli, M. Kalakrishnan, V. Vanhoucke, and S. Levine, “Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation,” CoRR, vol. abs/1806.10293, 2018.

[13] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, “Learning synergies between pushing and grasping with self-supervised deep reinforcement learning,” 2018.

[14] A. Zeng, P. Florence, J. Thompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee, “Transporter networks: Rearranging the visual world for robotic manipulation,” Conference on Robot Learning (CoRL), 2020.

[15] N. Karim, S. Afroz, A. Malandraki, S. Butterworth, C. Beach, M. Rigout, K. S. Novoselov, A. J. Casson, and S. G. Yeates, “All inkjet-printed graphene-based conductive patterns for wearable e-textile applications,” J. Mater. Chem. C, vol. 5, pp. 11640–11648, 2017.

[16] K. Pan, Y. Fan, T. Leng, J. Li, Z. Xin, J. Zhang, L. Hao, J. Gallop, K. S. Novoselov, and Z. Hu, “Sustainable production of highly conductive multilayer graphene ink for wireless connectivity and iot applications,” Nature Communications, vol. 9, no. 1, p. 5197, 2018.

[17] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” in GitHub Repository, 2016.

[18] J. Zhang, W. Zhang, R. Song, L. Ma, and Y. Li, “Grasp for stacking via deep reinforcement learning,” in 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 2543–2549, IEEE, 2020.

[19] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, “End-to-end robotic reinforcement learning without reward engineering,” Robotics: Science and Systems, 2019.

[20] V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, “Visual reinforcement learning with imagined goals,” Advances in Neural Information Processing Systems, vol. 31, pp. 9191–9202, 2018.

[21] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016.

[22] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings (Y. Bengio and Y. LeCun, eds.), 2016.

[23] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21–24, 2010, Haifa, Israel (J. Fünkranz and T. Joachims, eds.), pp. 807–814, Omnipress, 2010.

[24] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” 2016.

[25] X. Tan, W. Lyu, and A. Rosendo, “Circuitbot: Learning to survive with robotic circuit drawing,” PLOS ONE, vol. 17(3): e0265340, 2022.