2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Autonomous Tool Construction with Gated Graph Neural Network

Chenjie Yang, Xuguang Lan, Hanbo Zhang and Nanning Zheng

***Abstract*— Autonomous tool construction is a significant but challenging task in robotics. This task can be interpreted as when given a reference tool, selecting some available candidate parts to reconstruct it. Most of the existing works perform tool construction in the form of action part and grasp part, which is only a specific construction pattern and limits its application to some extent. In general scenarios, a tool can be constructed in various patterns with different part pairs. Therefore, whether a part pair is most suitable for constructing the tool depends not only on itself, but on other parts in the same scene. To solve this problem, we construct a Gated Graph Neural Network (GGNN) to model the relations between all part pairs, so that we can select the candidate parts in consideration of the global information. Afterwards, we embed the constructed GGNN into a RCNN-like structure to finally accomplish tool construction. The whole model will be named Tool Construction Graph RCNN (TC-GRCNN). In addition, we develop a mechanism that can generate large-scale training and testing data in simulation environments, by which we can save the time of data collection and annotation. Finally, the proposed model is deployed on the physical robot. The experiment results show that TC-GRCNN can perform well in the general scenarios of tool construction.**

I. INTRODUCTION

Tool construction plays an important role in the development of human civilization. For robotics, autonomous tool construction is also a significant task and can be interpreted as when given a reference tool, selecting some available candidate parts to reconstruct it [1]. However, since this task is full of challenges, there are few works that concentrate on it for a long time. Until recent years, [2], [3] propose two theoretical frameworks for tool construction. Most significantly, Nair et al. firstly perform autonomous tool construction on physical robots, which achieve great performances in some specific tool construction scenarios [1], [4].

These existing works are undoubtedly full of value. However, almost all of these works can only be performed in some specific tool construction scenarios. As in [1], the authors firstly segment the reference tool into a grasp part and an action part. Then they substitute each of the segmented parts with a most similar candidate part placed on the table. In [4], the authors use a similar strategy, but they select the candidate parts through Neural Networks (NN) which are trained by the 3D models segmented as in [1]. Within the framework of these two works, the tool must be constructed in the form of action part and grasp part, which limits their application to some extent. For example, as in Fig. 1, the

Chenjie Yang and Xuguang Lan are with the Institute of Artificial Intelligence and Robotics, the National Engineering Laboratory for Visual Information Processing and Applications, School of Artificial Intelligence, Xi'an Jiaotong University, No.28 Xianning Road, Xi'an, Shaanxi, China. cjyang2017@outlook.com, xglan@mail.xjtu.edu.cn

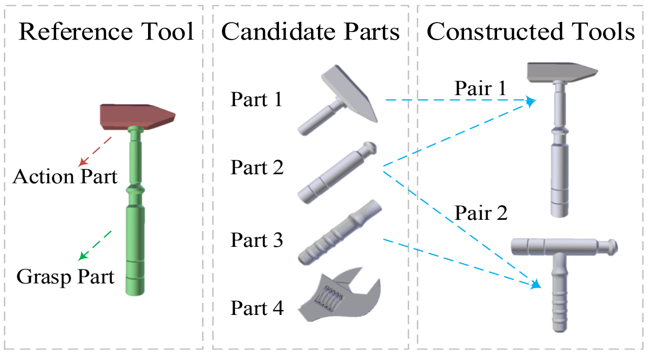


Fig. 1. The illustration of autonomous tool construction. In this figure, we are supposed to select the most suitable pair of candidate parts and use them to construct the reference tool.

most suitable parts that should be selected to construct the hammer are part 1 and part 2, which are not in line with the action part and the grasp part. Therefore, in this paper, we propose a vision-based model named Tool Construction Graph RCNN (TC-GRCNN), which can perform well in general tool construction scenarios. Besides, to train TC-GRCNN, we develop a mechanism that can generate large-scale training data for our task by simulation.

TC-GRCNN takes the depth map of the reference tool and the depth map of the candidate parts as inputs. For each pair of candidate parts, we use a RCNN-like structure to extract its feature. In general, the RCNN header can be directly applied on these features to perform classification and regression. However, in our task, whether a part pair is most suitable for tool construction depends not only on itself, but on other candidate parts in the scene. For example, in Fig. 1, if part 1 not exists, pair 2 which contains part 2 and part 3 will be most suitable for constructing the hammer. However, if part 1 exists, pair 1 will become the most suitable part pair for constructing the hammer and pair 2 is supposed to be suppressed. This means that the property of pair 2 is greatly affected by the existence of part 1, which can not be simply tackled by the general Neural Networks. To solve this problem, we construct a fully-connected Gated Graph Neural Network (GGNN) [5] to model the relations between all pairs of candidate parts in the scene. Through this model we can detect the properties of each pair of candidate parts in consideration of the global information. Finally, the experimental results suggest that the proposed approach can greatly deal with the problem of general tool construction.

The contributions of this paper are summarized as follows:

- We develop a mechanism that can generate large-scale training and testing data for general tool construction.

- A GGNN is constructed to model the relations between all pairs of candidate parts, so that the global information can be taken into consideration when selecting the most suitable part pair. Then we embed it into a RCNN-like structure to finally accomplish tool construction. The whole model will be named TC-GRCNN.
- The proposed model TC-GRCNN is deployed on the physical robot, Baxter. The experimental results suggest that our model can be generalized to real-world scenes.

The rest of this paper is organized as follows: The related works and problem formulation are introduced in Section II and III. The proposed approach is presented in detail in Section IV. Finally, our experiments are illustrated in Section V and the conclusions are discussed in Section VI.

II. RELATED WORK

Robot Tool Construction: Tool construction is a significant topic and has been studied in many fields, such as biological science [6], [7] and manufacturing [8]. However, in robotics, there are few works concentrating on it for a long time on account of its challenges. Until recent years, the development of artificial intelligence provided necessary conditions for the completion of this task. Firstly, the researchers focused on some preliminary works, such as tool substitution [9], [10]. In addition, [2], [3] provided their solutions to this problem from a theoretical perspective. More significantly, Nair et al. firstly performed autonomous tool construction on physical robots [1], [4]. However, these two works only perform tool construction in the form of action part and grasp part, which limits their application to some extent. In this paper, we propose a approach to complete this task in general scenarios.

Simulation Data Generation: To save the time of data collection and annotation, a lot of works generated training and testing data in simulation environments. For robotics, in demand of the interaction with external environment, this approach was widely adopted in many fields, such as vision-based grasping [11]–[13] and reinforcement learning [14], [15]. Notably, since it is difficult to render photo-realistic RGB images in simulation environments, most of these works took depth maps or point clouds as their inputs.

Graph Neural Network (GNN): General neural networks are usually used to process Euclidean data, such as images and text [16]–[19]. However, recently, the focus of artificial intelligence was gradually converted from recognizing into reasoning. In this case, GNN was proposed to process data with graph structure. As a connectionist model, GNN can capture the dependency between nodes while maintaining the expressive ability of standard neural networks. For this reason, the model was used in many fields, such as knowledge graphs [20], natural science [21] and social science [22]. Especially, GNN was usually deployed in conjunction with object detection networks [23]–[25] and achieved great performances in many high-level visual tasks [26], [27].

III. PROBLEM FORMULATION

The objective of this paper is to perform general tool construction on physical robots, which will be interpreted as when given a reference tool, the robot can select the most suitable parts to reconstruct it. Within the framework of the proposed approach, the problem can be refined as follows.

When given the depth maps of the reference tool and the candidate parts, the proposed approach is supposed to detect the bounding boxes of the reference tool and the candidate parts. On this basis, for each pair of detected parts, we are supposed to determine its fitness score, regress its attach points and rotation quaternions which will be used in tool construction. All these items will be interpreted as follows.

Inputs:

Depth Map of Reference Tool: The depth map which contains only one reference tool. In general, this tool will be placed on the plane with a random pose.

Depth Map of Candidate Parts: The depth map which contains three to six candidate parts. In general, these parts will be randomly scattered on the plane.

Outputs:

Bounding Boxes: The bounding boxes of the reference tool and all candidate parts.

Fitness Scores: The score that indicates whether a pair of parts is most suitable for constructing the reference tool.

Attach Points: The locations at which the parts can be attached together. For each pair of detected parts, we are supposed to generate two attach points, which are corresponding to these two parts respectively.

Rotation Quaternions: The quaternions as which the parts are supposed to be rotated when performing tool construction. It should be noticed that the rotation takes the pose of the reference tool as the target. Similarly, for each pair of parts we are supposed to generate two rotation quaternions.

Finally, the physical robot is supposed to select the most suitable pair of candidate parts and perform tool construction according to the attach points and rotation quaternions.

IV. PROPOSED APPROACH

In this section, we firstly introduce the mechanism that we adopt to generate training and testing data for general tool construction. Afterwards, a vision-based model TC-GRCNN is proposed to perform tool construction. In this model, the embedded GGNN can model the relations between all pairs of candidate parts from a global perspective, which greatly improves its performance in general tool construction.

A. Data Generation

For general tool construction, since the reference tool can be constructed in various patterns, it is difficult to collect all possible corresponding parts of the tool in the real world. Therefore, in this paper, we generate the training and testing data by randomly segmenting or combing 3D models in simulation environments. Each sample of the generated data consists of a depth map of the reference tool, a depth map of the candidate parts and the corresponding annotations. As a foundation, we firstly annotate the bounding boxes of the reference tool and the candidate parts. Then for each candidate part, we annotate whether it is of the most suitable parts to construct the reference tool. Finally, for each selected

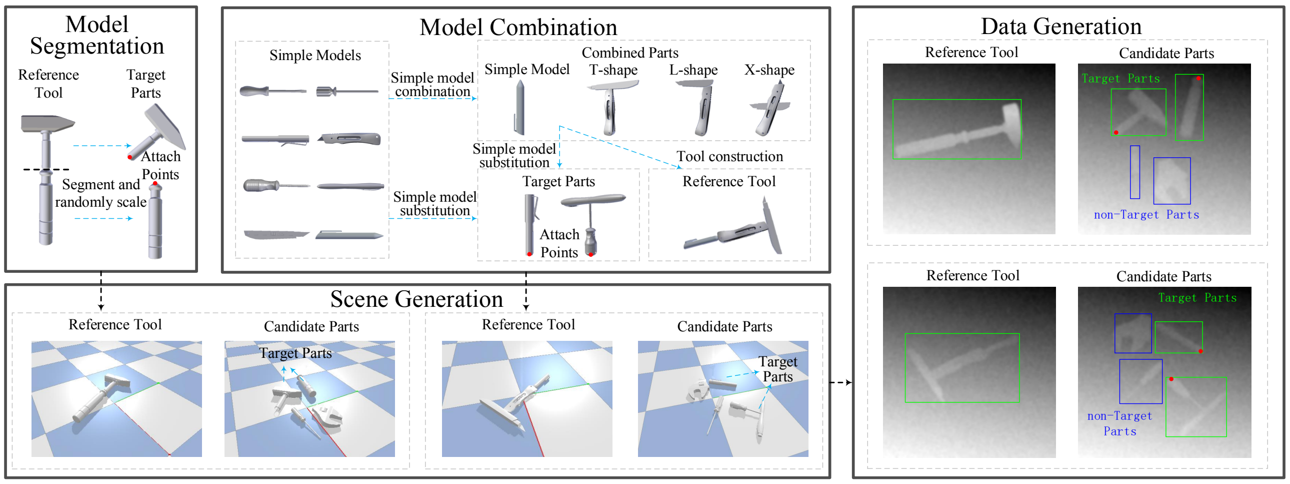


Fig. 2. Process of data generation. To generate training and testing data in simulation environments, we firstly collect 150 3D models of tools. Afterwards, we obtain the reference tool and its corresponding parts (target parts) by Model Segmentation and Model Combination. Then in pybullet, we drop the reference tool on the ground to get the tool scene. And its corresponding parts are also dropped on the ground with other random parts to get the part scene. Finally, the scenes are loaded in blender to generate the depth maps and annotations (bounding boxes, targets, attach points, rotation quaternions).

candidate part we annotate its attach point and quaternion of rotation for performing construction. Through this approach we can get enough training and testing data while saving the time of data collection and annotation. The process of data generation will be introduced as follows and in Fig. 2.

3D Models' Collection: To generate enough training and testing data, we firstly collect 150 3D models of tools and scale them to a appropriate size. These models are selected from YCB dataset [28], ShapeNet [29], ModelNet [30], ToolWeb dataset [9] and Trimble 3D Warehouse [31].

Model Segmentation: In order to obtain the part models which can be used to construct the reference tool, we firstly segment the tool by a random plane in blender [32]. And then the generated parts will be regarded as the most suitable parts which can be used to construct the reference tool. Specifically, the angle between the plane normal and the main axis of the tool will be limited into 45°, by which we can avoid some extremely imbalanced segmentation. Finally, the attach points of these parts will be set to the centers of the cut surfaces. In addition, we scale the generated parts by a random factor of 0.8 to 1.2, which can help alleviate overfitting when training models on the generated data.

Model Combination: The parts directly generated by random model segmentation can match the reference tool almost perfectly. However, in general, we are only allowed to construct the tool similarly but not perfectly. In addition, since the morphology of the generated parts are various, it is difficult for the robot to select similar parts to replace the generated parts. To solve this problem, we firstly select some simple models with cylindricality. Afterwards, like [15], we combine these simple models into parts with T-shapes, L-shapes and X-shapes. Finally, we use these parts to construct tool models. Within this framework, if we want to replace the original part with another similar part, we can just replace the simple models that make up this part.

Data Generation: Before rendering the depth maps, we generate the corresponding scenes in a real-time simulator, pybullet [33]. Firstly, we drop the reference tool on the ground to generate the tool scene. Then the parts corresponding to the reference tool will also be dropped on the ground with other random parts. Similarly, the final steady state will be stored as the part scene. Afterwards, the generated tool scene and part scene will be loaded in blender. In this environment we can get depth maps by reading the z-channel of the camera. On this basis, we add gaussian noise with zero mean on the generated depth maps to make them realistic. Finally, we will apply camera transformation on the reference tool and all of these candidate parts, so that we can obtain the bounding boxes, attach points and rotation quaternions in the image coordinate system, which are the final annotations of the generated training and testing data.

B. *Tool Construction Graph RCNN*

When given the tool depth map and the parts depth map, our approach is supposed to regress the bounding boxes of the reference tool and the candidate parts. Afterwards, for each pair of detected parts we will further determine whether it is the most suitable part pair to construct the reference tool and regress its attach points and rotation quaternions which will be used in tool construction. In this section, we propose a model named TC-GRCNN to perform this task.

TC-GRCNN takes the tool depth map and the parts depth map as inputs. And the first three layers of ResNet-101 [19] will be applied on these two depth maps as feature extractors. Then based on the extracted features, we construct a Region Proposal Network (RPN) [23] to regress the bounding boxes of the reference tool and the candidate parts. Afterwards, we use the generated Regions of Interest (RoI) to crop the extracted features, followed by a 7×7 adaptive pooling layer. In general, the bounding boxes generated by RPN can be directly treated as RoIs. However, if the cropped features have random scales and aspect ratios, adaptive pooling will

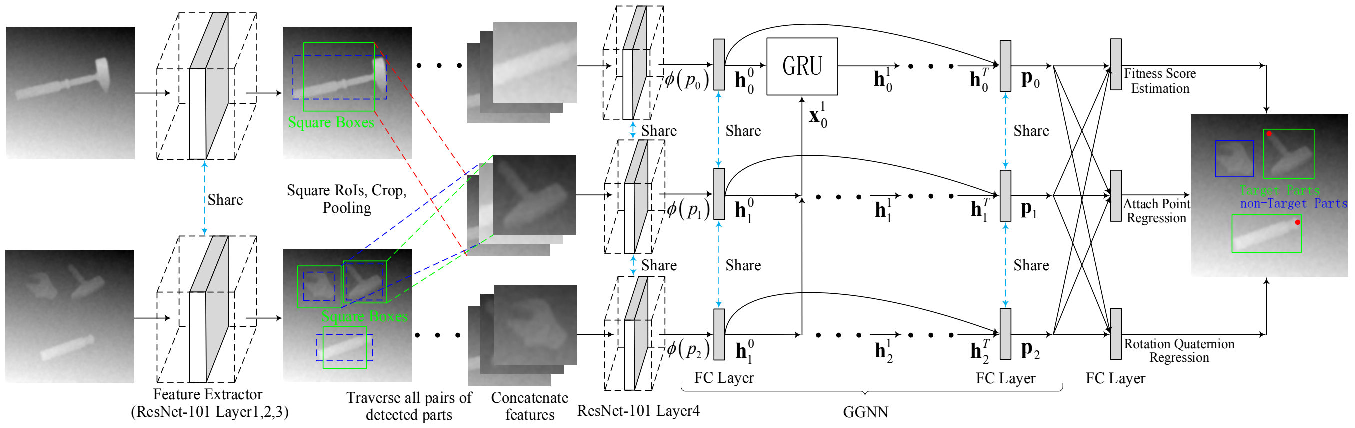


Fig. 3. Overview of TC-GRCNN. In this framework, firstly a RPN will be applied on the extracted features to obtain the bounding boxes of the reference tool and the candidate parts. Then we use the processed square boxes to crop the feature maps. After a 7×7 adaptive pooling layer, we traverse all pairs of detected parts and concatenate their pooled features with the reference tool's pooled features, followed by ResNet-101 Layer4 to compress the feature dimension. Afterwards, a GGNN will be constructed to model the relations between part pairs. Finally, on this basis, we perform fitness sore estimation, attach point regression and rotation quaternion regression. In this figure, the modules annotated ”share” will share a set of parameters with each other.

destroy their size information, which plays an important role in our task. Therefore, instead of original bounding boxes, we take square boxes with fixed edge lengths (200 pixels for candidate part and 300 pixels for reference tool) as RoIs, which greatly solve the problem of size information loss.

To select the most suitable parts combination, we firstly traverse all pairs of the detected parts. And for each pair of parts, we concatenate the pooled features of part 1, part 2 and the reference tool into an integrated feature map, followed by the fourth layer of ResNet-101 to compress the feature map into one dimension. Afterwards, in general, the RCNN Header can be directly applied on the generated feature vector to determine its category. However, in our task, whether a pair of parts is most suitable for tool construction depends not only on itself, but on the features of other candidate parts, as illustrated in Section I. To solve this problem, we construct a GGNN [5] to model the relations between all pairs of candidate parts in the scene.

The constructed GGNN takes each pair of candidate parts as a node. In addition, since all part pairs in the scene can interact with each other, we set the corresponding nodes to be fully connected. For clarity, we let p_i represent the i-th pair of candidate parts and $\phi(p_i)$ represent the corresponding original feature vector. Firstly, we initialize the hidden vector of each node as follows:

$$\mathbf{h}_i^0 = f(\mathbf{W}_\phi \phi(p_i) + \mathbf{b}_\phi) \tag{1}$$

where $\mathbf{h}_i^0$ represents the initial hidden vector of node i, f represents the ReLU activation function and $\mathbf{W}_\phi$, $\mathbf{b}_\phi$ are the parameters of the linear layer.

Within the framework of GGNN, for each node we are supposed to aggregate information from its neighbors. However, since the constructed graph is fully-connected, we are supposed to aggregate information from all other nodes, which can be interpreted as follows:

$$\mathbf{x}_i^t = \sum_{i \neq j} \mathbf{W}_a \mathbf{h}_j^{t-1} + \mathbf{b}_a \tag{2}$$

where $\mathbf{x}_i^t$ represents the fusion feature of all other nodes, $\mathbf{h}_j^{t-1}$ represents the hidden vector of node j at step $t-1$ and $\mathbf{W}_a$, $\mathbf{b}_a$ are the parameters of the linear aggregator.

In the step of propagation, we use Gate Recurrent Unit (GRU) [34] to further fuse the features. This model is full of expressive capability and can greatly deal with the problem of long-term propagation in graph structures. We take the aggregated vector $\mathbf{x}_i^t$ as input and pass the previous hidden vector $\mathbf{h}_i^{t-1}$ into the state line. Then we can obtain the hidden vector at this step by the update formulas as follows:

$$\begin{aligned}
\mathbf{z}_i^t &= \sigma(\mathbf{W}_z \mathbf{x}_i^t + \mathbf{U}_z \mathbf{h}_j^{t-1} + \mathbf{b}_z) \\
\mathbf{r}_i^t &= \sigma(\mathbf{W}_r \mathbf{x}_i^t + \mathbf{U}_r \mathbf{h}_j^{t-1} + \mathbf{b}_r) \\
\tilde{\mathbf{h}}_i^t &= \tanh(\mathbf{W}_h \mathbf{x}_i^t + \mathbf{U}_h (\mathbf{r}_i^t \odot \mathbf{h}_j^{t-1}) + \mathbf{b}_h) \\
\mathbf{h}_i^t &= (1 - \mathbf{z}_i^t) \odot \mathbf{h}_j^{t-1} + \mathbf{z}_i^t \odot \tilde{\mathbf{h}}_i^t
\end{aligned} \tag{3}$$

where $\mathbf{h}_i^t$ represents the hidden vector of node i at step t, σ represents the sigmod activation function and $\mathbf{W}_*$, $\mathbf{U}_*$, $\mathbf{b}_*$ are the parameters of GRU. The processes of aggregation and propagation will be performed T steps. According to our observation, after about five steps, increasing T will not further improve the recognition results. Therefore in this paper, the step number T will be set to five.

Afterwards, for each node i, the initial hidden vector $\mathbf{h}_i^0$ and the final hidden vector $\mathbf{h}_i^T$ will be concatenated together. Then the union feature will be passed into a linear layer and a ReLU activation function to get the final fusion feature $\mathbf{p}_i$:

$$\mathbf{p}_i = f(\mathbf{W}_p[\mathbf{h}_i^0; \mathbf{h}_i^T] + \mathbf{b}_p) \tag{4}$$

where f represents the ReLU activation function and $\mathbf{W}_p$, $\mathbf{b}_p$ are the parameters of the linear layer. Afterwards, we apply three linear layers on the feature $\mathbf{p}_i$ to determine the category of part pair i, regress the attach points and quaternions of these two candidate parts respectively.

TABLE I

Accuracy on Testing data of Our Constructed Model

Author	Algorithm	ATS	ATS-T(0.5)	ATS-T(0.7)	ATS-T(0.9)	APE	RQE	Speed(fps)
Ren et al.	**TC-RCNN(OR)** [23]	32.2	31.0	16.2	2.2	26.32	0.19	7.6
Ours	**TC-GRCNN(OR)**	27.6	27.4	19.4	3.6	24.26	0.26	7.1
	TC-RCNN(SR)	56.0	54.4	46.4	20.2	7.05	**0.05**	**7.9**
	TC-GRCNN(SR)	**63.8**	**63.8**	**63.4**	**53.6**	**5.15**	0.06	7.2

V. EXPERIMENT

In this paper, we train the models on simulated data which are generated in real time. With the learning rate of 0.001 and the momentum of 0.9, for each model we will train 120000 iterations in total. We will evaluate the performances of these trained models on simulated data and the physical robot.

A. Experiments on Simulated Data

Since the data generated in simulation environments are richly annotated, which can help us to evaluate the proposed approach quantitatively and comprehensively. In this section, we firstly perform the experiments on the simulated data.

The innovations of the proposed approach are mainly two points. Firstly, we transform original RoIs into square RoIs with fixed side lengths to preserve size information. What's more, we construct a GGNN to model the relations between all pairs of candidate parts. To evaluate the performance improvements of these two points, we take the model which uses the original RoIs and directly regresses the information of each part pair as baseline. Afterwards, we will make a comparison between the proposed approaches and the baseline approach. Notably, for each approach, we evaluate its performance on 500 simulated scenes generated in real time. The evaluation results will be recorded in Table I.

The algorithms in Table I will be introduced as follows:

TC-RCNN(OR): Tool Construction RCNN (Original RoIs). This model uses the original RoIs to crop the feature map and directly regresses the information of each part pair, which is a simple transfer of Faster-RCNN [23]. In our experiments we take it as the baseline.

TC-RCNN(SR): Tool Construction RCNN (Square RoIs). Based on TC-RCNN(OR), this model uses square boxes with fixed side lengths to crop the feature map, which can preserve the size information in the process of RoI pooling.

TC-GRCNN(OR): Tool Construction Graph RCNN (Original RoIs). This model takes the original bounding boxes as RoIs. However, it uses the constructed GGNN to model the relations between the part pairs, through which we can perform detection in consideration of the global information.

TC-GRCNN(SR): Tool Construction Graph RCNN (Square RoIs). This model takes the square boxes with fixed side lengths as RoIs. Meanwhile, it also uses the constructed GGNN to model the relations between the part pairs.

The metrics in Table I will be introduced as follows:

ATS: Accuracy of Top Selection. This metric is used to evaluate the performance of candidate parts selection. Firstly, the algorithm is supposed to select the pair of candidate parts with top fitness score. If this pair of candidate parts is most suitable to be used to construct the reference tool, this detection will be regarded as a positive example. Otherwise this detection will be regarded as a negative example. In our experiments, ATS is defined as the number of positive examples divided by the number of all examples.

ATS-T: Accuracy of Top Selection above Threshold. This metric is also used to evaluate the performance of candidate parts selection. However, in this metric, in addition to the conditions proposed in ATS, the fitness score of the positive example is supposed to be greater than a threshold. Otherwise it will be treated as a negative example.

APE: Attach Points Error. This metric is used to evaluate the performance of attach points regression. We take the Least Absolute Error (LAE) between the predicted attach points and the ground truth attach points as APE.

RQE: Rotation Quaternions Error. This metric is used to evaluate the performance of rotation quaternions regression. In our experiments, we also take the Least Absolute Error (LAE) between the predicted rotation quaternions and the ground truth rotation quaternions as RQE.

From the results shown in Table I we can observe the following three points: 1) The transformation of the RoIs from the original bounding boxes to the square boxes is the basis for our approach to complete the task. Without the RoI transformation, whether the GGNN is implemented or not, the performance of our approach cannot reach a satisfactory level. 2) The deployment of the GGNN can greatly improve the performance on candidate parts selection (ATS, ATS-T). But for the regressions of attach points and rotation quaternions, the GGNN does not bring significant performance improvements. This may be caused by the performance bottleneck on these two metrics. 3) For the metric ATS-T, the higher the threshold we take, the more obvious the performance improvements brought by the GGNN. This suggests that the GGNN can approximate the fitness score more accurately and make our approach more robust.

B. Experiments on Physical Robot

To evaluate the performance of the proposed approach in real-world scenarios, we implement our models on a Baxter with two 7-DoF arms and two parallel jaw grippers, which is designed by Rethink Robotics. Instead of the original RGB camera, we use an external Kinect2 to obtain the depth maps. Besides, the reference tool and the candidate parts

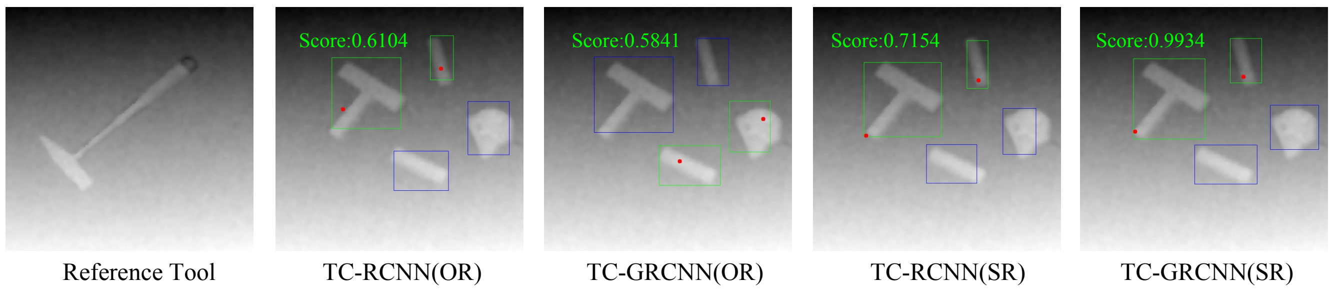
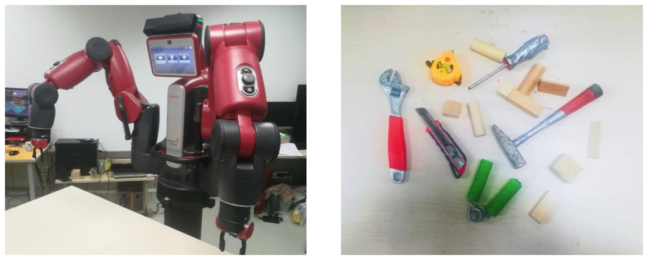


Fig. 4. An example of the detection results in real-world scenarios. In this figure, the detected parts with green bounding boxes are the selected candidate parts and the red points are the attach points. The "Score" with green color is the generated fitness score of the selected part pair.



(a) Environment Configuration (b) Tools and Parts

Fig. 5. Experiment configurations. (a) An overview of our environment configurations. (b) Some reference tools and candidate parts.

will be placed on a workbench with the size of 1m×1m. The complete environment will be shown in Fig. 5(a).

In our experiments, firstly we will randomly place the reference tool on the workbench and get its depth map. After that, we will randomly place 3-6 candidate parts which contain the corresponding parts of the reference tool on the workbench. Similarly, we will store the depth map of the candidate parts. Afterwards, the proposed approach will be deployed on these two depth maps to select the most suitable parts and regress their attach points and rotation quaternions. On this basis, to perform tool construction in real world, for each selected part we firstly calculate its minimum enclosing rectangle. Then the grasp will be applied on the center of the rectangle with a direction perpendicular to its long edge. Above all, in the process of our experiments, we find the accuracy of rotation quaternion regression is insufficient for the completion of tool construction. Therefore, we firstly limit the minimum enclosing rectangles to be perpendicular or parallel to each other. Then we select the most approximate directions according to the generated rotation quaternions, by which we can solve this problem in the majority of situations. Finally, we match the attach points of the selected candidate parts to accomplish tool construction. In this section, for each algorithm we implement 40 experiments and 10 for each part number. The success rate of tool construction is taken as the evaluation metric. And the results are recorded in Table II.

From the results we can observe that, firstly, the model trained on the simulated data can be generalized to real-world scenarios well. What's more, the application of square RoIs and the constructed GGNN can greatly improve the success

TABLE II
SUCCESS RATE OF TOOL CONSTRUCTION

Algorithm	Part Number				Total
	3	4	5	6	
TC-RCNN(OR)	2/10	2/10	0/10	1/10	12.5%
TC-GRCNN(OR)	2/10	1/10	0/10	1/10	10.0%
TC-RCNN(SR)	6/10	4/10	4/10	4/10	45.0%
TC-GRCNN(SR)	8/10	6/10	7/10	6/10	67.5%

rate of tool construction. In addition, we draw the detection results of a real-world scenario in Fig. 4. The results suggests that, firstly, the attach points generated by the models with original RoIs (OR) are biased toward the centers of the parts, which cannot be used to perform tool construction. Secondly, while the results of TC-RCNN(SR) and TC-GRCNN(SR) are both correct, TC-GRCNN(SR) generates higher fitness score, which means that this model is of better robustness.

VI. CONCLUSIONS

In this paper, we firstly develop a mechanism that can generate large-scale training and testing data for tool construction by simulation. Afterwards, we construct a model named TC-GRCNN. TC-GRCNN models the relations between the part pairs and can perform well in the scenes of general tool construction. Finally, the proposed model is deployed on the physical robot and achieve a success rate of 67.5% in general tool construction. The results show that firstly, the model trained on the simulated data can be directly generalized to real-world scenarios, which demonstrates the effectiveness of our data generation mechanism. Secondly, the model TC-GRCNN can accomplish the task of general tool construction well. The shortcoming of our approach is that the quaternion prediction is not accurate enough in real-world scenarios, which we will concentrate on in future works.

ACKNOWLEDGMENT

This work was supported in part by Trico-Robot plan of NSFC under grant No.91748208, National Major Project under grant No. 2018ZX01028-101, Shaanxi Project under grant No.2018ZDCXLGY0607, NSFC under grant No.61973246, and the program of the Ministry of Education for the university.

References

[1] L. Nair, J. Balloch, and S. Chernova, "Tool macgyvering: Tool construction using geometric reasoning," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[2] D. Choi, P. Langley, and S. T. To, "Creating and using tools in a hybrid cognitive architecture," in *2018 AAAI Spring Symposium Series*, 2018.

[3] V. Sarathy and M. Scheutz, "The macgyver test-a framework for evaluating machine resourcefulness and creative problem solving," *arXiv preprint arXiv:1704.08350*, 2017.

[4] N. S. Lakshmi Nair, Z. Erickson, and S. Chernova, "Autonomous tool construction using part shape and attachment prediction."

[5] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[6] C. Boesch and H. Boesch, "Tool use and tool making in wild chimpanzees," *Folia primatologica*, vol. 54, no. 1-2, pp. 86–99, 1990.

[7] T. B. Jones and A. C. Kamil, "Tool-making and tool-using in the northern blue jay," *Science*, vol. 180, no. 4090, pp. 1076–1078, 1973.

[8] M. Khisamutdinov, R. Khisamutdinov, and S. Kugultinov, "Tool creation and operation system development for large engineering enterprises," 2014.

[9] P. Abelha, F. Guerin, and M. Schoeler, "A model-based approach to finding substitute tools in 3d vision data," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2471–2478.

[10] M. Schoeler and F. Wörgötter, "Bootstrapping the semantics of tools: Affordance analysis of real world objects on a per-part basis," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, no. 2, pp. 84–98, 2015.

[11] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[12] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust robot vacuum suction grasp targets in point clouds using a new analytic model and deep learning," *arXiv preprint arXiv:1709.06670*, 2017.

[13] R. Detry, J. Papon, and L. Matthies, "Task-oriented grasping with semantic and geometric scene understanding," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3266–3273.

[14] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6284–6291.

[15] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese, "Learning task-oriented grasping for tool manipulation from simulated self-supervision," *arXiv preprint arXiv:1806.09266*, 2018.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[19] 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*, 2016, pp. 770–778.

[20] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, "Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 1802–1808.

[21] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 6530–6539.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[26] J. Sawatzky, Y. Souri, C. Grund, and J. Gall, "What object should i use?-task driven object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7605–7614.

[27] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph r-cnn for scene graph generation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 670–685.

[28] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 international conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 510–517.

[29] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[30] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[31] *Trimble 3D Warehouse*. [Online]. Available: https://3dwarehouse.sketchup.com/

[32] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, 2016. [Online]. Available: http://www.blender.org

[33] E. Coumans and Y. Bai, "pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org/, 2016–2017.

[34] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.