Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

A Discrete Event Approach for Scene Generation conditioned on Natural Language

Yu Cheng, Yan Shi, Zhiyong Sun, Dezhi Feng, and Lixin Dong

***Abstract*— Recognizable scene synthesis conditioned on natural language (NL) descriptions has many applications. The success of existing methods is limited to either clipart scenes or single objects. In this paper, a novel approach to generate realistic complex scenes is proposed. Instead of using a local image asset library or direct text-to-pixel mappings, this method deploys knowledge retrieved from the Internet, which helps to generate diverse and creative scenes. The proposed approach allows to modify the synthesized scene via NL. This helps to generate more complex scenes, and to correct training bias or errors. We validate the proposed approach on challenging MS-COCO dataset and present that our method can improve the semantic alignment to the input language descriptions, and the interpretability and quality of synthesized scenes.**

I. INTRODUCTION

Scene synthesis conditioned on NL can be beneficial and has many applications. One important application is literacy development. For second language learners and children who are learning to read, seeing scenes together with language can enhance learning [1]. Another application is as a reading helper for people with brain damage or learning disabilities. The machines can convert textual menus, signs, and operating instructions into graphical representations. In addition, many industrial applications can benefit from scene generation, such as robotic painting and polishing.

Scene generation conditioned on NL descriptions is challenging. As illustrated in Figure 1, a scene generator is tasked to synthesize a scene according to a NL description "*A lovely dog sits in front of a sofa in a room. A chair is at the left side of the dog*". The generator has to not only infer the context (*room*) and object type (*sofa*, *dog*, *chair*), object property (*lovely*), but also the spatial layout among objects (*in*, *left*, *front*).

There can be a variety of choices for object types and corresponding properties (e.g., shape, color, texture, etc.), and it is nontrivial for a machine to generate recognizable and diverse scenes through a local object library or direct text-to-pixel mappings. The success of existing scene generation approaches is limited to clipart scenes [2][3] or single objects [4]. Generating realistic and complex scenes is still challenging. Instead of generating scenes via a local knowledge base or training mappings from data (which is also based on local labeled datasets), we propose to directly use images searched from the Internet

Yu Cheng, Yan Shi, Zhiyong Sun, and Dezhi Feng are with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, 48823, USA (Email: chengyu9, yanshi3, sunzy, fengdezh@msu.edu). Lixin Dong is with the Department of Biomedical Engineering, City University of Hong Kong, Hong Kong, lixidong@cityu.edu.hk

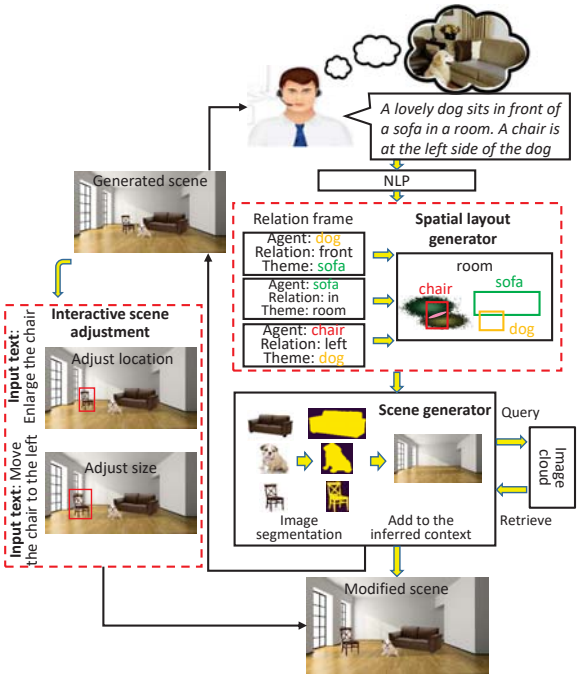


Fig. 1. The overall pipeline of the proposed approach.

for complex scene generation. The Internet can be viewed as a multi-disciplinary knowledge base collected from diverse distributed sources. However, the Internet is "dirty" because it often contains erroneous or corrupted data [5], i.e., a large portion of the retrieved images may be unrelated with the query term. By introducing a mechanism to deal with this situation, the proposed method allows the robot to use results that could be contaminated.

The proposed hierarchical framework for scene generation has two parts: a spatial layout generator and a scene generator. The spatial layout generator infers a spatial configuration of the desired scene. The scene generator first retrieves images through query with object types and object properties from Internet image search engines (in this work, Google image search engine is used. Since major image search engines also use textual cues for image search, they can provide similar performance). The object instances are then segmented from the retrieved images and assembled together with respect to the spatial layout. Additionally, the scene generator deals with unexpected events that may happen during image retrieval and scene generation, and allows users to modify scenes by replacing object instances,

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

changing the location and size of objects via NL.

The main contributions of the paper are as follows:

- Propose a novel approach for scene generation conditioned on NL instructions. Instead of maintaining a knowledge base in limited on-board memory, the proposed method leverages the power of the Internet, which improves the diversity and creativity of synthesized scenes, and to reduce burden on knowledge learning and storage.
- A mechanism is developed to tackle unexpected events to guarantee completion of scene generation.
- The proposed method provides an interactive interface and framework to further improve generated scenes through verbal instructions. This helps to correct possible training bias or training errors using human intuition and experience.
- Quantitative and qualitative evaluations have been conducted to validate improvement in quality of the scene generation over baseline works.

The rest of the paper is organized as follows. Section II briefly reviews related work. Section III provides an overview of the developed framework and Section IV illustrates the proposed approach in detail. Section V discusses the experimental results. Finally, Section VI concludes the paper.

II. RELATED WORK

Scene generation conditioned on text has been researched along two directions: object level and pixel level. Object level scene generation synthesizes scenes by placing object instances with respect to the estimated spatial layout. While pixel level methods map text to most likely object pixel distributions trained from data.

For object level scene synthesis, Zhu et al. [6] use nouns appeared in the text as keywords to search for images of objects and then place the images together to represent the meaning of the input text. Only object type information is employed, visual attributes of objects and spatial configuration are ignored. Zitnick et al. [2] train a conditional random field model to generate clipart scenes using clipart characters. Chang et al. [3] employ Bayesian probability to generate 3D office scenes using objects from a 3D model database. These two works utilize spatial relations between objects. The visual attributes of objects are still not considered in scene generation. The proposed approach takes into consideration both the object properties and the specified spatial configuration in scene generation. In comparison to using a fixed set of objects, our approach utilizes objects retrieved from the Internet. This increases the diversity of generated scenes and removes the need to laboriously compile an object library.

More recent works focus on using generative models for pixel level scene generation. Yan et al. [7] use variational auto-encoder to generate scenes conditioned on visual attributes. Reed et al. [8][9] train generative adversarial networks (GANs) for scene generation. These approaches are limited to single objects, such as birds [4] and flowers [10]. Mansimov et al. [11] synthesize scenes on more complicated scene descriptions from MS-COCO using a variational recurrent autoencoder with attention mechanism. Han et al. [12][13] propose stacked GANs structure to generate and refine scenes for better recognizability than single GANs. Dong et al. [14] use a cascaded convolutional neural network and recurrent neural network with GAN to augment the visual features of objects in the scene. The major focus of pixel level scene generation is on visual features, while spatial mappings are either ignored or modeled implicitly. In comparison, the proposed approach utilizes both types of information explicitly for complex scene generation.

In addition, existing scene generation approaches cannot modify the synthesized scenes. The proposed approach allows users to change objects, modify the size and position of objects in synthesized scenes through NL interactively, which helps to correct possible bias or training errors using human intuition and experience. In addition, the data generated during scene revision can be recorded for further training augmentation.

III. OVERALL PIPELINE

The pipeline of the developed approach is shown in Figure 1. Given a scene description, a scene can be synthesized through the sequence of following modules:

- **Natural language processing (NLP)** module processes language descriptions into relation frames. First, semantic role labeling is implemented for verbs and their arguments identification; second, syntactic parsing is performed to identify object properties and spatial relations between objects (Section IV-A).
- **Spatial layout generator** infers the spatial configuration of objects using relation frames generated from the prior NLP module. A spatial layout is an organized position distribution of objects. An object's position is assumed to be related with other objects around it. Each object position is calculated based on the positional and dimensional information of its dependent objects (Section IV-B).
- **Scene generator** generates a scene according to the inferred layout from previous module. It retrieves images from Internet image search engines using information of object type and properties, segments objects from retrieved images and then merges them together. The generator uses images from diverse cloud resource and an imperfect object detector. Unexpected events may happen and cause the scene synthesis to fail. The scene generator deals with these unexpected events and ensures completion of scene generation. (Section IV-C).

IV. SCENE GENERATION APPROACH

A. Natural Language Processing

Converting the scene descriptions into machine understandable specifications requires linguistic structure identification of the descriptions and then represent them in a formal representation. This section illustrates the conversion process and its implementation. The machine

translates instructions through a sequence of modules which performs syntactic and semantic parsing to create formal representations for spatial layout inference and image query. Different from many NL systems which depend on scenario based grammars that combine semantic and syntactic information [15], this work combines domain-general NLP components for input parsing. The input is firstly parsed by a semantic role labeling (SRL) parser to identify semantic roles (e.g., verbs and arguments) [16]. Each role is recognized as one type of PropBank modifiers [17]. Then Stanford Log-linear POS Tagger is used to identify the subject and corresponding properties for each argument [18]. The identified information is matched to a relation frame (*Agent, Agent property, Verb, Relation, Theme, Theme property*). The expected relation (*Relation*) between the two items (*Agent* and *Theme*), the specification of the items (*Agent property* and *Theme property*), and pairwise position dependency (*Agent* depends on *Theme*) are identified using their tags and syntactic positions. No modifications are required for the core language models across scenarios, which minimizes the cost of adapting the system to new application areas and reduces the impact of hand-engineered grammar.

B. Spatial Layout Generator

The spatial layout is required to synthesize scenes using object instances segmented from images. Relation frames contain partial spatial information and are used for spatial layout inference. Figure 1 presents an example of spatial configuration inference according to the language description *A lovely dog sits in front of a sofa in a room. A chair is at the left side of the dog*. Each positional information is determined by objects around it. For example, the chair's position depends on the dog, and the dog's position depends on the sofa. As a result, only after the positional decisions of the sofa and the dog, the chair's position can be decided based on positional and dimensional information of the dog and the sofa.

In this work, dependency relation between objects is used for spatial configuration inference [19]. An agent is considered to be dependent on its theme. The more objects an object is dependent on, the higher dependency value it has, the latter it will be added into the scene. An ordered object manipulation sequence can be figured out based on the dependency values in an ascending manner. Usually, the first object in the sequence is the context because all the other objects are dependent on it. If the context is not explicitly specified in the language, it will be inferred using the objects appeared in the scene description:

$$P(C|O_{1:m}) = \prod_{i=1}^{m} P(C|O_i) \tag{1}$$

where C represents context and O denotes object. The context type that maximize the probability will be selected. In this work we consider five context types: indoor, road, city plaza, rural field, and sea shore (if a context is explicitly specified in the description, the scene generator can retrieve candidate context images from cloud resources; otherwise, the context will be inferred from the five categories). The priors for object occurrence in different context types are trained using 1708 scenes from MS-COCO training dataset (the object detector used in this work can detect 20 categories of object [20], so the dataset has been filtered for scenes comprised of objects lay in the category set):

$$P(C|O_i) = \frac{count(O_i \text{ in } C)}{count(C)} \tag{2}$$

The spatial knowledge for positional inference is obtained through training with relative location data from [2], which consists of 10020 scenes created using 58 clipart objects. The center position of an object is selected by random from the common set of its relative locations with each dependent object/theme. As circled by the red dashed rectangle shown in Figure 1, the cloud shaped area in dark yellow represents locations recognized as "left" to the dog, and the cloud shaped area in dark green denotes "left" positions to the sofa. Then the center of the chair is chosen randomly from the intersection area (which is marked in pink). The dimensional ratios between object types are set as a priori.

C. Scene Generator

Scene generator does two jobs. First, it retrieves images that have specified object instances from image search results. Second, it synthesizes a scene using segmented object instances from the retrieved images. The object instances are retrieved from a "dirty" image cloud rather than manually labeled datasets [2][3]. Information mismatch and object detector errors may happen and cause scene generation to fail, such as detection failure of objects in the retrieved images. In addition, it would be more practically useful if a human instructor can tune the generated scenes through NL. To this end, a dynamic discrete event controller is developed to tackle possible exceptions in scene generation and modify generated scenes verbally. Supervisory control theory (SCT) is deployed to build the controller and plant models [21]. The modeling process using SCT is illustrated in detail and property analysis of the developed model is conducted for behavior refinement to guarantee successful scene generation.

1) Modeling of Scene Generator: The scene generator retrieves images using relation frames information and assembles object instances together to meet inferred spatial configuration. In addition, it engages in interactive scene modification through NL commands. Figure 2 shows the modular plant models of the generator. Table I gives definitions of the event set and controllability of each event. The centralized plant model is a shuffled product of the modular plant models:

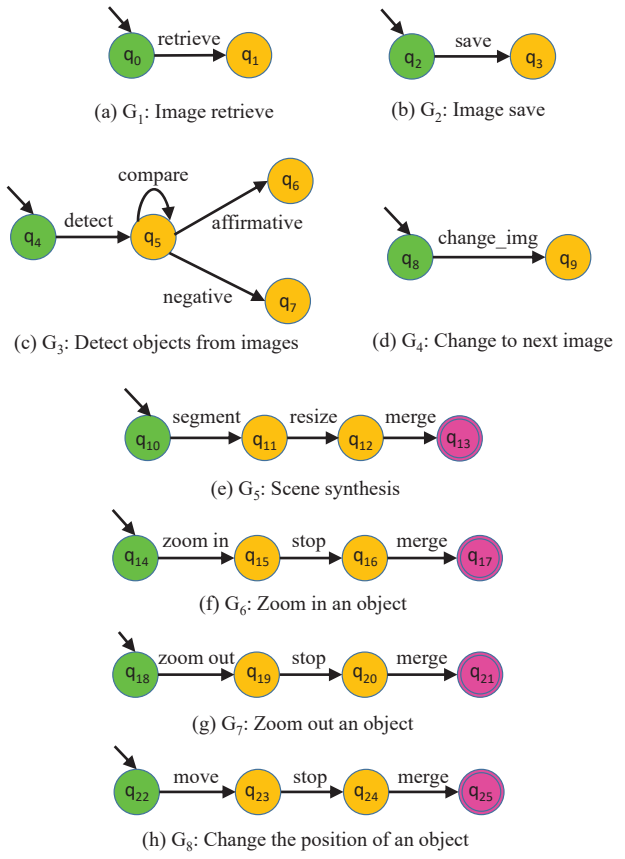
$$G = ||_{i=1}^{8} G_i \tag{3}$$

where $||$ represent parallel composition operation. The shuffled plant model has 896 states and 4096 transitions in total.

TABLE I

LIST OF PRIMITIVE ACTIONS

Primitive Actions	Controllability	Description
retrieve_img (obj)	controllable	Retrieve an image from a local image library or an online image cloud
resize (obj)	controllable	Adjust the size of the object
detect (obj)	controllable	Detect the object category
compare (obj1, obj2)	controllable	Compare category labels between objects
segment (obj)	controllable	Extract objects from images
merge (obj, context)	controllable	Add an object into current scene
affirmative	uncontrollable	Confirm the previous operation
negative	uncontrollable	Deny the previous operation
change_img (obj)	controllable	Switch to the next image
zoom_in (obj)	controllable	Increase the size of an object
zoom_out (obj)	controllable	Reduce the size of an object
move (obj)	controllable	Change the position of an object
save (obj, img)	controllable	Save the image that has the desired object
stop	controllable	Stop current being implemented action



(a) G_1: Image retrieve

(b) G_2: Image save

(c) G_3: Detect objects from images

(d) G_4: Change to next image

(e) G_5: Scene synthesis

(f) G_6: Zoom in an object

(g) G_7: Zoom out an object

(h) G_8: Change the position of an object

Fig. 2. Modular plant models of scene generation. Subfigures (a) to (e) are the models of scene generation using objects extracted from retrieved images. Subfigures (f) to (h) are the models for scene tuning. States in green with an entry arrow represent initial states. Yellow colored states mean intermediate states. States in pink with double circles denote marked states and represent the target state of each behavior.

When the generator receives a scene description, it retrieves images using keyword combinations of $(Agent + Agent\ Property)$ or $(Theme{+}Theme\ Property)$. Then an object detector is deployed to find specified object instances. If an image contains the object instance that meets the description, then the object is segmented from its original image and resized to match the context. Otherwise, the generator picks the next image in the result queue and repeats the process until a scene generation is finished.

The above workflow is ideal for scene generation. This behavior is contained in the synthesized plant model, which captures all the physically possible behaviors of the system, including desired ones (also known as legal behaviors, which mean behaviors leading to the success of tasks) and behaviors that should not be allowed (known as illegal behaviors, which mean behaviors that can cause system failures). As a result, the generator may fail to finish the assigned task due to the illegal behaviors. To guarantee the accomplishment of tasks, a controller is required to make the plant well behaved. To this end, following properties should be held by the controller:

1) **Controllability**: Controllability characterizes the capability of a system to accomplish the assigned task with possible unexpected occurrence of uncontrollable events.

   Definition of Controllability [21]: Let S and $T = \overline{T}$ be two arbitrary languages over event set Σ. S is said to be controllable with respect to T and Σ_{uc} if

   $$\overline{S}\Sigma_{uc} \cap T \subseteq \overline{S}.$$

   Interpreting T as all the physically possible behaviors and S as the legal behaviors. The above definition means S is controllable, if the occurrence of any uncontrollable event does not lead the behavior (that belongs to S) out of the legal range. To ensure successful scene generation, uncontrollable behaviors

should be forbidden by the scene generator.
Controllability can guarantee the task completion under uncontrollable dynamics. However, sometimes the generator can be trapped by a state or a subset of states such that it may not accomplish the task. To prevent this issue, nonblocking property is introduced next for plant behavior analysis and refinement.

2) **Nonblocking**: Nonblocking characterizes an attribute of a system to avoid being stuck at a state or trapped in a subset of states. The following gives the definition of nonblocking property and its checking criteria.
Definition of Nonblocking [21]: Let $T(C/P)$ and $T_m(C/P)$ represent the controlled language and the marked language of the controller, respectively. A set of formal language satisfying

$$\overline{T_m(C/P)} = T(C/P)$$

is said to be nonblocking. An overline of the set denotes all the prefixes of the behavior, including the behavior itself. In this context, marked behaviors represent successfully implemented scene generation. The above definition means any behavior of a nonblocking controller can finally lead to the marker states.
The scene generator may encounter problems that hamper the scene generation. For instance, the mismatch problem between the images and their descriptions, or object detection errors due to the imperfect detector. Then the generator has to keep changing images until the target object has been detected. This can be time-consuming and may trap the process in a livelock. Perform nonblocking check helps to detect these blocking situations. To avoid the blocking, the retrieved image evaluation procedure and scene generation procedure are separated as two independent processes. The retrieved images will be evaluated first to detect desired objects and only the qualified images will be saved to a local image library. Then the generator randomly retrieves an image from the image library and extracts the object instance for scene synthesis. The operations on automata (plant model synthesis, property check, and controller synthesis) are implemented using DESUMA [22].

V. EXPERIMENTAL RESULTS

A. Experimental Setup

Dataset. MS-COCO evaluation dataset [23] is used for evaluation. It is filtered to keep text descriptions that only have objects in the range of processable object categories due to the object detector. As a result, the proposed approach and other baseline works are tested on 128 descriptions.

Baseline works. Performance comparison has been carried out with the following baseline works. **Ground Truth**: original scenes paired with correct scene description from the MS-COCO evaluation dataset. **Random**: scenes chosen by random from the ground truth. **Reed et al.** [8]: scenes generated by a GAN developed by Reed et al. **AttnGAN** [13]: scenes generated by an attentional GAN with multi-stage refinement for text to scene generation. **Ours**: scenes generated by the proposed approach without interactive adjustment. **Ours-Human**: scenes generated by the proposed approach and tuned by instructors through NL.

TABLE II
QUANTITATIVE EVALUATION AND HUMAN STUDY RESULTS

Method	CIDEr	ROUGH_L	METEOR	AAS
Random	0.121	0.256	0.072	1.859
Reed et al.	0.201	0.269	0.092	1.531
AttnGAN	0.220	0.296	0.097	1.951
Ours	**0.416**	**0.335**	**0.134**	**2.737**
Ours-H	**0.654**	**0.362**	**0.151**	**3.201**
GT	1.298	0.476	0.221	4.205

Ours-H denotes Ours-Human.

B. Evaluation

Description generation: To measure the similarity between input text and generated descriptions from the synthesized scenes. The underlying intuition is that one should be able to guess the original text from the synthesized scene if the generated scene is relevant to input description and its contents are recognizable. A description generator [24] trained with MS-COCO data is used to generate text descriptions. One description is generated per scene. Language similarity is reported on three standard check: METEOR [25], CIDEr [26], and Rouge-L [27]. Table II presents the quantitative evaluation results. Higher is better in all columns. The Random approach shows the worst performance. This proves that random scenes rarely convey the same semantic meaning in the ground truth. In addition, the proposed method outperforms the baseline works significantly. Scene revision by verbal interaction also improves the alignment and semantics of generated scenes dramatically. Description generation performance shows that generated descriptions from our synthesized scenes are more correlated with the input text and easier to recognize.

Human evaluation: Using description generator for evaluation is beneficial, especially for large scale data. It can also introduce unintended bias of the generator. Human evaluation has been conducted on Amazon Mechanical Turk to validate the description generation evaluation. For each ground truth text description, six scenes synthesized by different methods are shown to the Turkers. They are asked to score how well the scenes match the text description on a scale of 1 (very poorly) to 5 (very well). The results are presented in the last column of Table II, in which *AAS* represents Average Absolute Score. The results of human studies proves consistency with the description evaluation performance. Figure 3 shows some randomly chosen generated scenes.

Fig. 3. Qualitative examples of synthesized scenes.

VI. CONCLUSION

In this work, we propose a framework for scene generation conditioned on NL descriptions. Instead of learning a direct text-to-pixel mapping or retrieving from a local image asset library, the proposed approach utilizes cloud resources for diverse and creative scene generation. The developed mechanism makes the scene generation reliable under retrieved contaminated data. The interactive modification of scenes helps better alignment with scene descriptions and training error correction.

REFERENCES

[1] R. E. Mayer, "Multimedia learning," in *Psychology of learning and motivation*. Elsevier, 2002, vol. 41, pp. 85–139.

[2] C. L. Zitnick, D. Parikh, and L. Vanderwende, "Learning the visual interpretation of sentences," in *Proc. of IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 1681–1688.

[3] A. X. Chang, M. Savva, and C. D. Manning, "Learning spatial knowledge for text to 3d scene generation." in *EMNLP*, 2014, pp. 2028–2038.

[4] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep. CNS-TR-2011-001, 2011.

[5] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation." *IEEE Trans. Automation Science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[6] X. Zhu, A. B. Goldberg, M. Eldawy, C. R. Dyer, and B. Strock, "A text-to-picture synthesis system for augmenting communication," in *AAAI*, vol. 7, 2007, pp. 1590–1595.

[7] X. Yan, J. Yang, K. Sohn, and H. Lee, "Attribute2image: Conditional image generation from visual attributes," in *European Conference on Computer Vision*. Springer, 2016, pp. 776–791.

[8] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[9] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, "Learning what and where to draw," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 217–225.

[10] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. Indian Conference on Computer Vision, Graphics and Image Processing*, 2008, pp. 722–729.

[11] E. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Generating images from captions with attention," *arXiv preprint arXiv:1511.02793*, 2015.

[12] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *arXiv preprint arXiv:1612.03242*, 2016.

[13] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," *arXiv preprint*, 2017.

[14] H. Dong, J. Zhang, D. McIlwraith, and Y. Guo, "I2t2i: Learning text to image synthesis with textual data augmentation," in *Proc. of IEEE International Conference on Image Processing*. IEEE, 2017, pp. 2015–2019.

[15] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn, "What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution," in *Proc. of IEEE International Conference on Robotics and Automation*, 2009, pp. 4163–4168.

[16] L. He, K. Lee, M. Lewis, and L. Zettlemoyer, "Deep semantic role labeling: What works and what's next," in *Proc. of Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2017, pp. 473–483.

[17] O. Babko-Malaya, "Propbank annotation guidelines," *URL: http://verbs. colorado. edu*, 2005.

[18] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proc. of Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2003, pp. 423–430.

[19] Y. Cheng, J. Bao, Y. Jia, Z. Deng, Z. Sun, S. Bi, C. Li, and N. Xi, "Modeling robotic operations controlled by natural language," *Control Theory and Technology*, vol. 15, no. 4, pp. 258–266, 2017.

[20] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proc. of IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.

[21] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[22] L. Ricker, S. Lafortune, and S. Genc, "Desuma: A tool integrating giddes and umdes," in *International Workshop on Discrete Event Systems*, 2006, pp. 392–393.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[24] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3156–3164.

[25] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *In Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005, pp. 65–72.

[26] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *In Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4566–4575.

[27] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," *Text Summarization Branches Out*, 2004.