A Multimodal Dialogue System for Conversational Image Editing

Tzu-Hsiang Lin[1] **Trung Bui**[2] **Doo Soon Kim**[2] **Jean Oh**[1]
[1]Carnegie Mellon University [2] Adobe Research
{tzuhsial,hyaejino}@andrew.cmu.edu {bui, dkim}@adobe.com

Abstract

In this paper, we present a multimodal dialogue system for Conversational Image Editing. We formulate our multimodal dialogue system as a Partially Observed Markov Decision Process (POMDP) and trained it with Deep Q-Network (DQN) and a user simulator. Our evaluation shows that the DQN policy outperforms a rule-based baseline policy, achieving 90% success rate under high error rates. We also conducted a real user study and analyzed real user behavior.

1 Introduction

Image editing has been a challenging task due to its steep learning curve. Image editing software such as Adobe Photoshop typically have a complex interface in order to support a variety of operations including selection, crop, or slice. They also require users to learn jargons such as *saturation*, *dodge* and *feather*[1]. In order for a user to achieve a desired effect, some combinations of operations are generally needed. Moreover, image edits are usually localized in a particular region, for instance, users may want to add more color in the trees or remove their eyepuffs. In the first case, users need to first select the trees and then adjust the saturation to a certain level. In the second case, users need to select the eyepuffs, apply a reconstruction tool that fills the region with nearby pixels, then apply a patching tool to make the reconstruction more realistic. Such complexity makes the editing task challenging even for the experienced users.

In this paper, we propose a conversational image editing system which allows users to specify the desired effects in a natural language and interactively accomplish the goal via multimodal dialogue. We formulate the multimodal dialogue system using the POMDP framework and train the dialog policy using Deep Q-Network (DQN)[16]. To train the model, we developed a user simulator which can interact with the system through a simulated multimodal dialogue. We evaluate our approach–i.e., DQN trained with the user simulator–by comparing it against a hand-crafted policy under different semantic error rates. The evaluation result shows that the policy learned through DQN and our user simulator significantly outperforms the hand-crafted policy especially under a high semantic error rate. We also conducted a user study to see how real users would interact with our system.

The contributions of the paper are summarized as follows:

- We present a POMDP formulated multimodal dialogue system.
- We developed a multimodal multi-goal user simulator for our dialogue system.
- We present an architecture for Conversational Image Editing, a real-life application of the proposed framework in the domain of image editing
- We present the experiment results of comparing the proposed model against a rule-based baseline.

[1] *saturation*: color intensity; *dodge*: tool to lighten areas; *feather*: tool to create soft edges

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

2 Related Work

The multimodal system PixelTone [11] shows that an interface combining speech and gestures can help users increase more image operations. Building on its success, we propose to build a multimodal image editing dialogue system. Previous research on multimodal dialogue systems mostly focus on the architectures [4] for multimodal fusion and did not adopt the POMDP framework [22]. Since the dialogue managers of these systems are based on handcrafted rules, it cannot be directly optimized. Also, real users are essential in evaluating these systems [21], which can be costly and time inefficient.

Information-seeking dialogue systems such as ticket-booking [7] or restaurant-booking [6, 20] typically focus on achieving one user goal throughout an entire dialogue. Trip-booking [7] is a more complex domain where memory is needed to compare trips and explore different options. The most similar domain is conversational search and browse [9, 8], where the system can utilize gestures and even gazes to help users to locate the desired objects. A recently collected corpus [13] shows that Conversational Image Editing is more challenging, requiring to address not only these aspects but also the composite-task setting [17, 3] where the user may have multiple goals to achieve in a sequential order.

Our task is also related to Visual Dialogue [5] which focuses on the vision and the dialogue jointly. The agent in Visual Dialogue needs to recognize objects, infer relationships, understand the scene and dialogue context to answer questions about the image. Our agent also requires a similar level of understanding, but the focus on vision is more of recognizing a localized region in an image. Another closely related area is vision-and-language navigation [1], since both the navigation instructions (e.g., go upstairs and turn right) and image edit requests [15] (e.g., remove the elephant not looking forward) are mostly in imperative form and relates to what the agent sees.

3 Partially Observed Markov Decision Process Formulation

In this section, we formulate our image editing dialogue system as a Partially Observable Markov Decision Process (POMDP) [22]. POMDP dialogue framework combines *Belief State Tracking* and *Reinforcement Learning*. *Belief State Tracking* represents uncertainty in dialogue that may come from speech recognition errors and possible dialogue paths, whereas *Reinforcement Learning* helps the dialogue manager discover an optimal policy.

POMDP is composed of belief states B, actions A, rewards R, and transitions T. The goal is to optimize a parametrised policy $\pi : B \rightarrow A$ to maximize the expected sum of rewards $R = \sum_t \gamma^t r_t$.

State

Our state space $B = B^u \oplus B^e$ includes the user state B^u and the image edit engine state B^e. B^u, the estimation of the user goal at every step of the dialogue, is modeled as the probability distribution over possible slot values. For gesture related slots (e.g., *gesture_click*, *object_mask_str*), we assume these values hold 100% confidence and assigns probability score 1 if a value is present and 0 otherwise. B^e is the information provided by the engine that is related to the task at hand.

The main difference from convention dialogue systems is that we include information from the engine. Since the image edit engine displays the image, executes edits and stores edit history, we hypothesize that including this information can help our system achieve a better policy. One example is, if the edit history is empty, users will unlikely to request an *Undo*. Examples of our state features is presented in Table 1.

Type	Feature Type	Examples
Speech	Distribution over possible values	*intent, attribute*
Gestures	Binary	*image_path*
Image edit engine	Binary	*has_next_history*

Table 1: Example state features used for dialogue policy learning

Action

We designed 4 actions for our dialogue system: (i) *Request*, (ii) *Confirm*, (iii) *Query*, and (iv) *Execute*. *Request* and *Confirm* actions are each paired with a slot. *Request* asks users for the value of a slot and *Confirm* asks users whether the current value stored by the system is correct. *Query* takes the current value in slot *object* and queries the vision engine to predict segmentation masks of *object*. *Execute* is paired with an intent. *Execute* passes its paired intent and the intent's children slots (Figure 1) to the image edit engine for execution. If any of the arguments are missing, the execution will fail, and the image will remain the same.

Unlike information-seeking dialogue systems [6, 20] which attempt to query a database at every turn, we make *Query* an independent action because of two reasons: (i) modern vision engines are mostly based on Convolutional Neural Networks (CNNs) and frequent queries may introduce performance latency; (ii) segmentation results should be stored, and consecutive queries will override previous results.

Reward

We define two reward functions. The first reward function is defined based on PyDial [19]. The reward is given at the end of a dialogue and defined as $20 * \mathbb{1}(D) - T$, where 20 is the success reward, $\mathbb{1}(D)$ is the success indicator and T is the length of the dialogue. The second reward function gives a positive reward r^p when an user goal is completed, and a negative reward r^n if an incorrect edit is executed. The main idea for the second reward function is that since image editing dialogues have multiple image edit requests, additional supervision reward will better help train the dialogue system. However, we did not observe a huge difference between the two reward functions in our initial experiments. Therefore, we only present the results on the first reward function.

Transition

Our transitions are based on the user simulator and the image edit engine. Every time step t, the system observes belief state b_t and outputs system action a_t. The image edit engine observes system action a_t and update its state to b^e_{t+1}. The user simulator then observes both b^e_t and a_t then updates it state to b^u_{t+1}. Next state $b_t = b^u_t \oplus b^e_t$ will pass to the system for the next turn. Both the user simulator and the image edit engine are updated according to predefined rules.

Dialogue Policy

We present two polices for dialogue management.

Rule-based: We hand-crafted a rule-based policy to serve as our baseline. The rule-based policy first requests the intent from the user. After knowing the intent, it then requests all the slots that correspond to that particular intent and then executes the intent. To obtain the localized region (*object_mask_str*), the rule-based policy first queries the vision engine and then requests *object_mask_str* if the vision engine result is incorrect.

Deep Q-Network: Deep Q-Network [16] combines artificial neural networks and reinforcement learning and takes state b^t as inputs to approximate action values $Q(s_t, a_t)$ for all action a. Deep Q-Networks are shown to succeed in spoken dialogue systems learning [19] due to its capability to model uncertainty in spoken language understanding and large domain space.

4 Conversational Image Editing System

In this section, we first present the ontology used in our system, then describe the role of each system component in further detail.

4.1 Domain Ontology

Conversational Image Editing is a multimodal dialogue domain that consists of multiple user intents and a natural hierarchy. Intents are high level image edit actions that may come with arguments or entities [13, 14]. Also, most edit targets are localized regions in the image. This introduces a

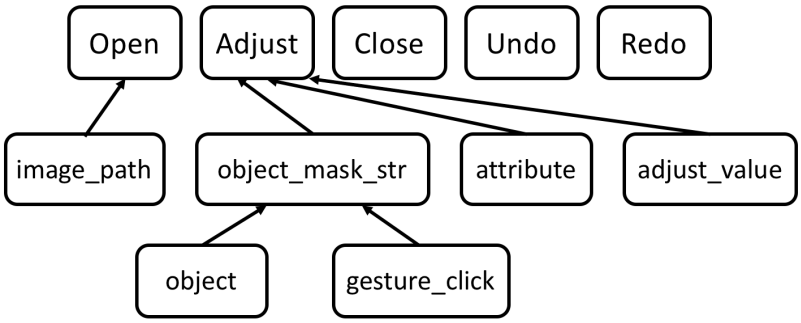

Figure 1: This figure depicts the domain ontology of our conversational image editing system. Top level nodes represent intents; Mid level and low level nodes represent slots. Arrows indicate dependencies. Mid level nodes that directly point to top level nodes are arguments directly associated with that intent. Right three intents do not require additional arguments.

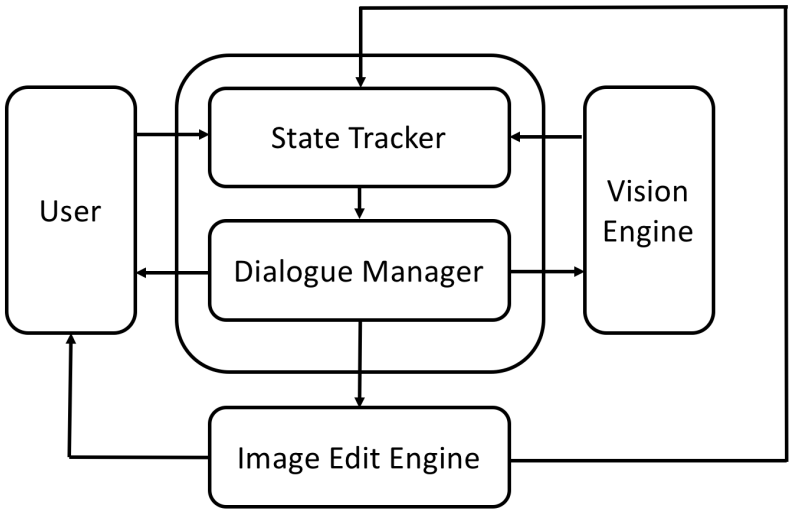

Figure 2: This figure illustrates the components in our system and interactions with the user. State Tracker observes information from User and Image Edit Engine then passes the state to Dialogue Manager. Dialogue Manager then selects an action. If the action is *Query*, Dialogue Manager will query Vision Engine and store the results in the state. The action will then be observed by both the User and Image Edit Engine.

hierarchy where the system has to first request the name of the object and then query vision engine to obtain the object's segmentation mask.

We handpicked a subset of user intents from *DialEdit* [13] and *Editme* [15] corpus. The user intents are *Open, Adjust, Close, Undo, Redo*. *Open* and *Close* are inspired by Manuvinakurike et al. [14]; *Adjust* modifies the attributes of a region; *Undo* reverts incorrect edits and *Redo* can redo edits if *Undo* is accidentally executed. *Open* requires slot *image_path*; *Adjust* requires slots *object_mask_str, adjust_value, and attribute*. Slot *object_mask_str* further depends on slots *object*, and *gesture_click*. *Close, Undo*, and *Redo* do not require slots. Our ontology is depicted in Figure 1.

4.2 Components

Our system architecture consists of four components: (i) Multimodal State Tracker (ii) Dialogue Manager (iii) Vision Engine (iv) Image Edit Engine

Multimodal State Tracker

Input to our state tracker consists of three modalities (i) user utterance, (ii) gestures (iii) image edit engine state. For (i), we trained a two-layer bi-LSTM on the *Editme* [15] corpus for joint intent prediction and entity detection. We select the *Editme* corpus because it is currently the only available dataset which contains image editing utterances. *Editme* has about 9k utterances and 18 different intents. Since *Editme* is collected by crowd-sourcing image edit descriptions, the utterances are often at a high level, and the annotation can span up to a dozen words. For example, in *Editme*, "lower the glare in the back room wall that is extending into the doorway" has the following labels: intent is labeled as "adjust"; attribute is labeled as "the glare"; region is labeled as "in the back room wall that is extending into the doorway". Our tagger achieved 0.80% intent accuracy and 58.4% F1 score on the validation set. Since there exists a discrepancy between our ontology and *Editme*, an additional string matching module is added to detect numerical values and intents not present in *Editme*. For (ii), we directly take the output as state values. That is, if gestures are present, then the gestures slot values will 1. For (iii), we designed several features including ones mentioned in Table Table 1. The final output is the concatenation of the results of (i), (ii) and (iii), which is the belief state B in the previous section.

Dialogue Manager

Dialogue manager is a rule-based policy or a parametrised policy that observes the dialogue state, and performs actions to complete user's image edit requests. Detailed of actions are presented in the previous section.

Vision Engine

Vision engine performs semantic segmentation and is called when system selects action *Query*. Vision engine takes the image and slot *object*'s value as query and outputs a list of candidate masks to be shown to the user. If present, *gesture_click* will be used filter the candidate masks by checking whether it overlaps with any of the candidate masks. We leave extracting state features from vision engine as future work.

Image Edit Engine

Image edit engine is an image edit application that acts as an interface to our user and as an API to our system. At every turn, our system loads the candidate masks stored in slot *object_mask_str* to the engine for display. When system performs an *Execute* action, the executed intent and associated arguments will be passed to the engine for execution. If the intent and slots are valid, then an edit can be performed. Else, the execution will result in failure, and image will remain unchanged.

We developed a basic image edit engine using the open-source OpenCV [2] library. The main features of our engine include image edit actions, region selectors, and history recording.

4.3 Multimodal Multi-goal User Simulator

We developed an agenda based [18] multimodal multi-goal user simulator to train our dialogue system. User agenda is a sequence of goals which needs to be executed in order. A goal is defined as an image edit request that is composed of an intent and it depending slots in the tree Figure 1. A successfully executed goal will be removed from the agenda, and the dialogue is considered as success when all the user goals are executed. If an incorrect intent is executed, the simulator will add an *Undo* goal to the agenda to undo the incorrect edit. On the other hand, if the system incorrectly executes *Undo*, the simulator will add a *Redo* goal to the agenda. Our simulator is programmed not to inform *object_mask_str* unless asked to.

5 Simulated User Evaluation

5.1 Experimental setting

User Goal We sampled 130 images from MSCOCO [12] and randomly split them into 100 and 30 for training and testing, respectively. Goals are randomly generated using our ontology and the dataset.

	Rule-based				DQN			
SER	Turn	Reward	Goal	Success	Turn	Reward	Goal	Success
0.0	7.5	13.50	3.0	1.0	7.43	13.56	3.0	1.0
0.1	7.2	13.80	3.0	1.0	7.27	13.73	3.0	1.0
0.2	9.16	11.13	2.9	0.97	8.33	12.66	3.0	1.0
0.3	15.0	4.6	2.87	0.93	9.23	11.76	3.0	1.0
0.4	17.3	-6.1	2.30	0.53	12.1	8.2	2.93	0.97
0.5	18.2	-16.8	1.50	0.07	13.6	5.3	2.8	0.90

Table 2: Results of our simulated user evaluation. SER denotes semantic error rate. Our DQN policy still retains high success rate even under high semantic errors compared to rule-based baseline.

	Rule-based		DQN	
	Turn	Success	Turn	Success
Real User	5.7	0.8	3.9	0.8

Table 3: Result metrics of our real user study on the 10 sampled goals from the test set. Rule-based and DQN policies have the same 0.8 success rate, and the DQN policy has fewer number of turns.

For simulated users, we set the number of goals to 3 which starts with an *Open* goal, followed by an *Adjust* goal, then ends with a *Close* goal. Maximum dialogue length is set to 20. To simulate novice behavior, a θ parameter is defined as the probability a slot will be dropped in the first pass. We set θ to 0.5.

Vision Engine We directly take ground truth segmentation masks from the dataset as query results.

Training Details For DQN, we set the hidden size to 40. We set batch size to 32 and freeze rate to 100. We used 0.99 for reward decay factor γ. The size of experience replay pool is 2000. We used Adam [10] and set the learning rate to 1e-3.

5.2 Results

We evaluate four metrics under different semantic error rates (SER): (i) Turn (ii) Reward (iii) Goal (iv) Success. (i), (ii), and (iv) follow standard task-oriented dialogue system evaluation. Since image editing may contain multiple requests, having more goals executed should indicate a more successful dialogue, which success rate cannot capture. Therefore, we also include (iii).

Table 2 shows the results of our simulated user evaluation. At low SER (<0.2), we can see that rule-based and DQN can successfully complete the dialogue. As SER increases (>0.2), the success rate of the rule-based policy decreases. On the other hand, our DQN policy manages to learn a robust policy even under high SERs and achieved 90% success rate.

6 Real User Study

While the experiment in the previous section shows effectiveness under a simulated setting, real users may exhibit a completely different behavior in an multimodal dialogue. Therefore, we built a web interface(Appendix A for our system. Our interface allows text input and gestures. Users can input *gesture_click* by putting a marker on the image, and input *object_mask_str* by putting a bounding box on the image as the localized region.

We recruited 10 subjects from the author's affiliation and conducted a study testing them against our Rule-based Policy and the DQN policy. One person among the 10 is familiar with image editing. 10 goals were sampled from the test set, and each person is asked to complete two dialogues of the same policy. In every dialogue, there is only one *Adjust* goal and it is presented to the user in semantic form. Since number of goals is reduced, we set maximum dialogue length to 10.

Table 3 presents the turns and success rate in our real user study. Both policies achieved 0.8 success rate, while the DQN policy has fewer number of turns. Due to the small amount of goals, we focus on the insights gained from manually inspecting the dialogue data.

Turn	Speaker	Utterance	Gesture
1	User	increase the man's saturation by 10	
1	System	What object_mask_str do you want?	
2	User	man	object_mask_str
2	System	Execute: intent=adjust, adjust_value=10 attribute=saturation, object_mask_str=iVBOR	
Success			

Table 4: Success dialogue example with DQN policy. The user first interacts only with text input in the first turn, then inputs both the "man" (*object*) and gesture (*object_mask_str*) in the second turn.

Turn	Speaker	Utterance	Gesture
1	User	make the man 30% less bright	object_mask_str
1	System	Execute: intent=adjust, adjust_value=30 attribute=brightness, object_mask_str=iVBOR	
Failure			

Table 5: Failure dialogue example with DQN policy. In the first turn, our issues a image edit request and also the localized region (*object_mask_str*). However, our state tracker failed to interpret "30% less bright" into the value "-30", and executes with the incorrect value 30 (*adjust_value*).

Turn Number: We discovered the turn number in our study mostly depend on the user behavior. For users who prefer to express more information, the turn number is smaller. For users who prefer to interact mostly with text and leave gestures as a last resort, the turn number is larger.

Success Rate: The semantic form of the user goal makes it easy for the users to express and inform slot values (Appendix A)), so that most dialogues are successful. All the failure cases are incorrect understanding of the slot *adjust_value*, which is a result of the string matching module in our state tracker. Table 4 presents a success dialogue and Table 5 presents a failure dialogue.

7 Conclusions and Future Work

We present a multimodal dialogue system for Conversational Image Editing. We derived the POMDP formulation for Conversational Image Editing, and our simulated evaluation results show that the DQN policy significantly outperforms a sophisticated rule-based baseline under high semantic error rates. Our real user study shows that the language understanding component is crucial to success and real users may exhibit more complex behavior. Future work includes frame-based state tracking, expanding the ontology to incorporate more intents and modeling multimodal user behavior.

Acknowledgement

This work is in part supported through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016. This work should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. We would like to thank the anonymous reviewers for their insightful comments.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2018.

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

[3] Pawel Budzianowski, Stefan Ultes, Pei hao Su, Nikola Mrksic, Tsung-Hsien Wen, Iñigo Casanueva, Lina Maria Rojas-Barahona, and Milica Gasic. Sub-domain modelling for dialogue management with hierarchical reinforcement learning. In *SIGDIAL Conference*, 2017.

[4] Trung Bui. Multimodal dialogue management-state of the art. 2006.

[5] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–495, 2017.

[7] Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 207–219, 2017.

[8] Dilek Hakkani-Tür, Malcolm Slaney, Asli Celikyilmaz, and Larry Heck. Eye gaze for spoken language understanding in multi-modal conversational interactions. In *Proceedings of the 16th International Conference on Multimodal Interaction*, pages 263–266. ACM, 2014.

[9] Larry Heck, Dilek Hakkani-Tür, Madhu Chinthakunta, Gokhan Tur, Rukmini Iyer, Partha Parthasarathy, Lisa Stifelman, Elizabeth Shriberg, and Ashley Fidler. Multi-modal conversational search and browse. In *First Workshop on Speech, Language and Audio in Multimedia*, 2013.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Gierad P Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. Pixeltone: A multimodal interface for image editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2185–2194. ACM, 2013.

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[13] Ramesh Manuvinakurike, Jacqueline Brixey, Trung Bui, Walter Chang, Ron Artstein, and Kallirroi Georgila. DialEdit: Annotations for Spoken Conversational Image Editing. In *Proceedings of the 14th Joint ACL - ISO Workshop on Interoperable Semantic Annotation*, Santa Fe, New Mexico, August 2018. Association for Computational Linguistics. URL https://aclanthology.info/papers/W18-4701/w18-4701.

[14] Ramesh Manuvinakurike, Trung Bui, Walter Chang, and Kallirroi Georgila. Conversational image editing: Incremental intent identification in a new dialogue task. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 284–295, 2018.

[15] Ramesh R. Manuvinakurike, Jacqueline Brixey, Trung Bui, Walter Chang, Doo Soon Kim, Ron Artstein, and Kallirroi Georgila. Edit me: A corpus and a framework for understanding natural language image editing. In *LREC*. European Language Resources Association (ELRA), 2018.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[17] Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Çelikyilmaz, Sungjin Lee, and Kam-Fai Wong. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *EMNLP*, 2017.

[18] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152. Association for Computational Linguistics, 2007.

[19] Stefan Ultes, Lina M. Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gasic, and Steve Young. PyDial: A Multi-domain Statistical Dialogue System Toolkit. In *Proceedings of ACL 2017, System Demonstrations*, pages 73–78, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/P17-4013.

[20] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 438–449, 2017.

[21] Steve Whittaker and Marilyn Walker. Evaluating dialogue strategies in multimodal dialogue systems. In *Spoken Multimodal Human-Computer Dialogue in Mobile Environments*, pages 247–268. Springer, 2005.

[22] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

A System Interface

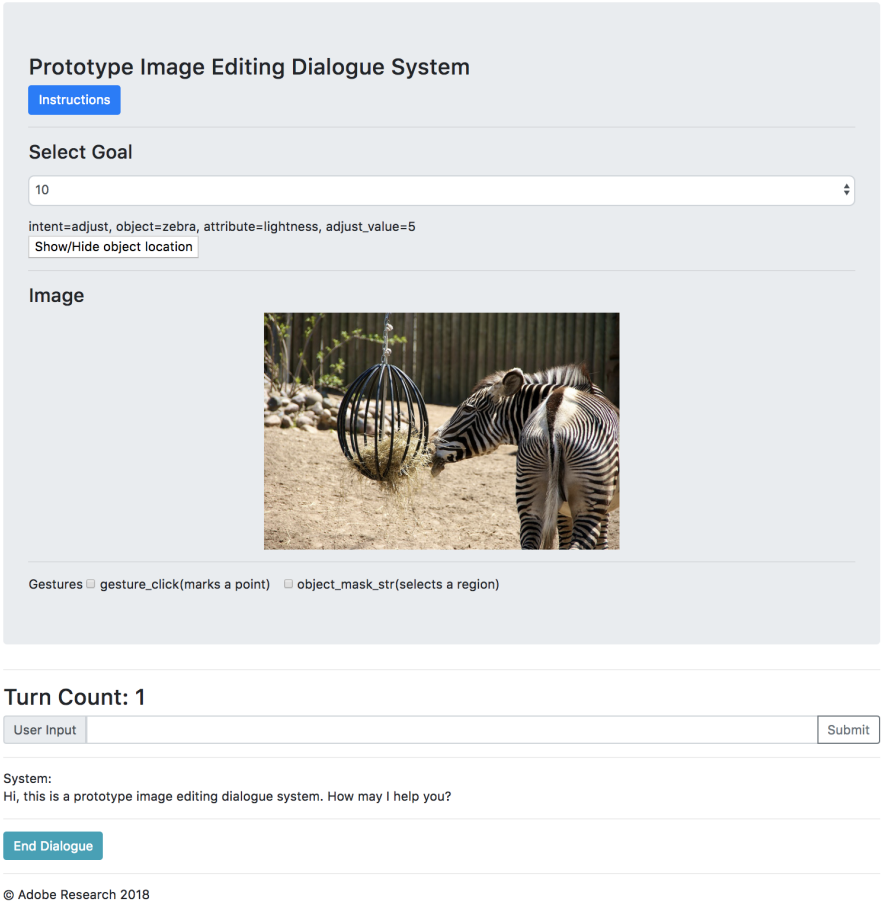

Figure 3: Interface of our system prototype. A goal is presented to the user in semantic form. Users can input a click (gesture_click) or select a bounding box (object_mask_str)