Visual Relationship Constructor: Predicting User Preference for Object Arrangement

ABSTRACT
Relationships between objects have played a crucial role in image understanding. Despite the great progress of deep learning techniques in recognizing the relationship between two objects, it is more important how we utilize the relation to help finding a reasonable method for robotic household arrangement. In previous works, people had primarily focused on relation prediction, and some of them start to work on object arrangement on a plane, such as gathering stuff in the same hierarchy together in the same area. However, few people develop a model that performs object arrangement in vertical direction. Therefore, we build Visual Relationship Constructor(VRC), a model that is able to learn its user's arranging rule by parsing visual relationships between objects placed by the user, then check these rules to decide vertical dependency among new objects and complete object arrangement.

Author Keywords
Interaction design;Interaction paradigms;Humanities

CCS Concepts
•Human-centered computing → Human computer interaction (HCI);

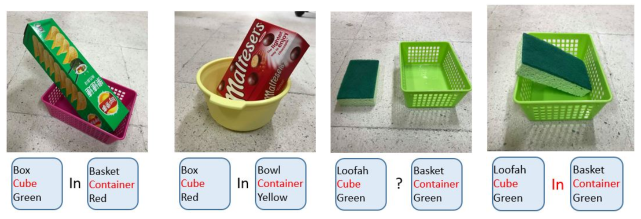
INTRODUCTION
Relationships between objects have played a crucial role in image understanding. Despite the great progress of deep learning techniques in recognizing the relationship between two objects, it is more important how we utilize the relation to help finding a reasonable method for robotic household arrangement. Previous works had primarily focused on relation prediction, others who work on arranging objects by predicting user preferences only deal with tidying up objects by gathering them in the same hierarchy together in the plane, rather than applying them the proper arrangement style to complete further arrangement. While we preferably utilize various visual relation between every object in the image, with several unique attributes of each object, to satisfy as many user preferences as possible when arranging objects, and thus completing the task transfer arrangement. Within our method, we select the common object classes in the daily life to be the sample objects for our task transfer household arrangement. See Figure 1. , With those objects and relations, we use our Visual Relationship Constructor(VRC) to let the machine learn the proper arrange rules from several well-arranged (well-arranged here represents that all objects are arranged by some certain rules) demo cases, and apply this rule to random messy objects for arrangement.

CHI'20, April 25–30, 2020, Honolulu, HI, USA

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-6708-0/20/04. . . $15.00

DOI: https://doi.org/10.1145/3313831.XXXXXXX

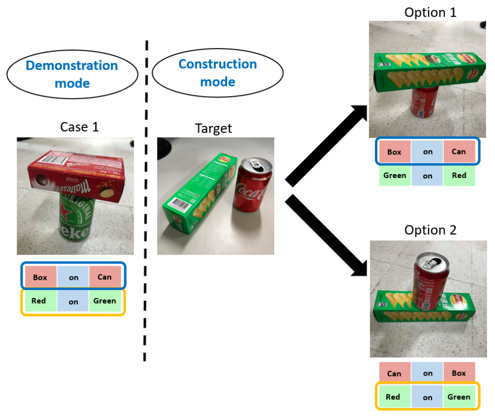
Previous works related to household arrangement can be traced back to the attempt of building an object hierarchy to help classifying objects into different stack[1]. When there is a new object not in the hierarchy list, they can quickly determine which stack it belongs to by adding new objects in one of the root of the list. However, they only deal with tidying up objects by gathering them in the same hierarchy together in the plane, rather than applying them to the up-down/contain-in arrangement style to complete further household arrangement. Instead, we focus on utilizing various of relation results (for example: up-down/contain-in relation) between every object in the image, with several unique attributes of each object, to complete the task transfer household arrangement, not only focusing on gather objects in a stack in a 2D plane.



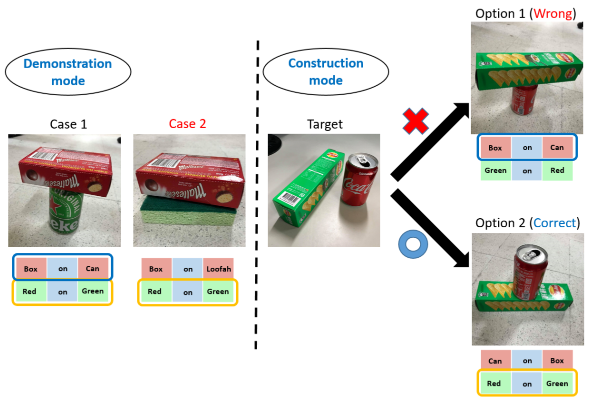
(a) Cases Demonstration (b) Targets (c) Arranged

Figure 1. First, We demonstrate (a) a series of cases that all contain *cube-in-container* rule, (b) when a green loofah and a green basket are asked to be arranged, (c) the model will output green loofah in green basket, since this relation combination match the rule *cube-in-container*, which has been frequently demonstrated before.

We have tried to use deep learning method to solve this arranging problem, but we face two difficulties when implementing architecture. First, inspiring by [2] model, we focus on predicting local relation (i.e. relation between each two objects), so that we can gradually build the whole relation scene. However, when predicting relation between 2 same objects, there are more than one correct prediction (e.g. *obj1 on obj2* and *obj2 on obj1* are both correct), which may be a disaster for error function. Therefore, the error function is hard to define. Second, even if the first difficulty is solved, there still lacks of the overall connection between pairs of object. But sometimes, humans take all objects together into consideration when arranging a place, so it is more complicated than merely focusing on the local relation.



(a)



(b)

Figure 2. (a) Depicts an existing ambiguity concerning that there are more than one arranging options seem to be reasonable and this ambiguity is caused by the variety of human arranging preference. (b) VRC solve this ambiguity by observing more cases from the same user and try to figure out its owner's unique preference.

Traditionally, a machine used to predict the arrangement result may have ambiguity since the user's preference in color or shape might have conflicts when both of these features influence at the same time. See Figure 2(a). There are two users and each of them has their arranging style. Now, they are asked to arrange the red can and green box on the desk. One places the box on the can because he focuses on the type rule - box on can; however, the other puts the box under the can because she prefers arranging them by color rule - red on green. Under this circumstance, though the result could be different, neither of them should be considered wrong. Therefore, building a machine that can fine-tune its arranging rule according to its owner is necessary in this field.

See Figure 2(b). We construct a system called Visual Relationship Constructor (VRC) that can record every demo steps conducted by the users, and simultaneously update the recommendation arrangement rule to fit the user's preference. Which means the final arrangement rule to the messy objects can be determined by every step of the user's demo. By doing so, we can let the machine learn the proper arrange rule from several well-arranged demo cases from the user preference, and apply this rule to random messy objects for arranging. Besides taking user's preference into consideration, our main goal is to learn the arranging rule from tidy demo cases and then help arrange the messy objects. In order to accomplish such an arrangement task, it's important to utilize the relationship between objects in a tidy task to help us building an arranging rule for messy objects. By taking advantage of the relation in a tidy demo tasks, VRC can find a relatively reasonable method for robotic household arrangement for the messy objects. According to our method, we extract some attributes (including name, shape, and color) from every object. These features can form a proper arrangement rule when they combine with the relationships between objects. By learning the features concerning their relations, we can find the arrangement rule from that, and then generate proper arranging relation rules between the messy objects from our VRC.

VRC has the following steps to help arranging random messy objects: (1) Record every tasks of the user's demonstration, and extract their related relations between objects. (2) Simultaneously update the weights of every relation generated by every task and store every relation rules in a rule pool. (3) After finish the demonstration, by calculating every weight of the rules, we'll have a certain relation rule from the rule pool. (4) We generate every possible relation combination for the given objects and evaluate each of these combinations, then finally choose the one with the highest score for arrangement.

RELATED WORK

2.1 . User Preference Arranging Objects

For arranging random messy objects, people often come up with ideas to let the machine learn how we expect them to arrange. There's a quite famous method called collaborative filtering [3] to learn user's preference from some previous choices of the user, and predict the closest answer to the user's next choice. It has widely used in recommendation algorithms, where they use customer's interests as the input to generate a list of recommended items. Frankly speaking, many applications use only the items that customers purchase and explicitly rate to represent their interests, but they can also use other attributes, including items viewed, subject interests, and favorite artists to predict user's preference.

To further utilize this user preference learning method in object arranging, [1] built on the paradigm of collaborative filtering for making personalized recommendations and relies on data from different users which they gathered from multiple users. To deal with novel objects for which we have no data, they propose a method that complements standard collaborative filtering by leveraging information mined from the Web. Later, [4] combined collaborative filtering with a mixture of approaches based on the object hierarchies they mined from the online stores. This can help to compute the semantic similarity of a new object to previously known objects, and help arranging those new messy objects up. Furthermore, [5] presented a method enabling a robot to automatically arrange objects using task and motion planning. By using various relationships extracted from user's positive examples and considering the arranged scene as well as the relationship between objects defined in a dependency graph, the robot is able to arrange objects by using a task and motion planner to shorten the arrangement duration. However, although the above methods can learn to find the belonging locations of new objects quickly and efficiently by a small amount of actions, they mainly focused on 2D plane arrangement. In contrast to those methods, we put our attention in stacking style (i.e. consider to put small objects in the container), which is more related to common household arrangement style. Besides, [6] presented an approach that the use of learned models of arrangement principles can predict the best place for a particular unseen object. They train a classifier to find similarities between objects to put certain objects in their common belonging locations. However, they mainly focus on grouping objects together and where the objects are supposed to be, while we could take the stacking arrangement style (the up-down/contain-in relation) into consideration, and thus flourish the diversity of arrangement ways. In recent years, [7] presented an approach that used the spatial relations between two objects to train a Gaussian mixture model in order to predict the locations of objects which are commonly placed as part of everyday activities. However, they address a problem which needs to local a certain object first.

2.2 . Task Transfer

Recently, machine learning (ML) technologies have already achieved great success in many knowledge engineering areas including classification, regression, and clustering. However, many ML methods work well only under a common assumption: the training and testing data are derived from the same feature source domain and sometimes the same distribution as well. When the distribution changes, almost all the models need to be rebuilt in order to fit the new training data. In many real-world applications, it is expensive or impossible to recollect the training data and rebuild the models. To reduce the need for recollecting data, previous works have taken advantage of task transfer learning to make predictions of new objects. We sometimes have a classification task in one domain of interest, however, we only have enough training data in another domain of interest, where the latter data may follow a different data distribution or be in a different feature space. [8] proposed transfer learning (TL) depends on the amount of labeled data available in the source domain, and the feature-representation TL can be either supervised or unsupervised as well.

[9] comprises a novel recurrent neural network architecture for knowledge transfer which uses factored third-order tensors to encode cross-task and task-specific information. [8] proposed that traditional machine learning techniques try to learn each task from scratch, while transfer learning techniques choose to transfer the knowledge from some previous tasks to a target task when the latter has fewer high-quality training data.

The idea of transfer learning, to some degrees, is similar to the phrase "task transfer" in our article. Transfer learning applies the knowledge of one problem on another different but related one. Similarly, task transfer means that the model applies the arrangement style learning from previous demos on the new task which has never seen before.

[10] propose a CNN model that is able to arrange a group of unknown objects to their specific position and make the whole scene similar to the reference group of object. They implement it by letting the model figure out the similarity between the objects in the reference scene and target scene. Therefore, even objects in two scene may be slightly different, they can still map together and output a reasonable result. This concept inspires us to come up with the idea that we can realize task transfer by using the coherence of attributes of objects.

VISUAL RELATIONSHIP CONSTRUCTOR

See Figure 3. , We build a model named Visual Relationship Constructor (VRC), which can learn certain rules about arranging objects from humans' demonstration. For example, if a homemaker prefer to put the green object on the red object, by demonstrating a few object combinations that contain this rule to VRC, VRC can gradually learn this preference and will automatically put green object on red object when VRC was fed with a object list that contain green one and red one. Besides, simply by classifying attributes to the new objects into existing attribute groups, VRC can apply some rules that learn in previous demo cases on these new objects. Thus, task transfer is complete.

Algorithm 1: VRC

```
1  if initial = True then
2      Clear rule pool $P$ to empty
3  while True do
4      Input: mode M
5      if $M$ = Construction then
6          Input: A list of objects
7          Generate all possible relation vectors of these objects
             and put it in set $V$.
8          foreach $v \in V$ do
9              Evaluate $v$ by evaluator
10         Output: $v$ with the highest score
11     else
12         Input: A list of objects and a relation vector.
13         Build relationship triplets of attribute $r$ between each
             two objects and store them in relation queue $Q$.
14         foreach $r \in Q$ do
15             Search for a rule $r_p$ that match $r$ in $P$ and update
                 its $n_{apply}$
16             Search for rule in the other category in $P$ and
                 update its $n_{apply}$
17         forall rules $\in P$ that dirty bit is True do
18             Apply weight function of its type to update its
                 Weight $W$
```

Input Format and Output Format

At the beginning of every turn, VRC will ask user to select a mode (either **Construction mode** or **Demonstration mode**). It decides what the next step VRC would take.

If **Demonstration mode** is selected, 2 inputs are taken. (1) list of objects: each object will have 3 attributes (name, shape,

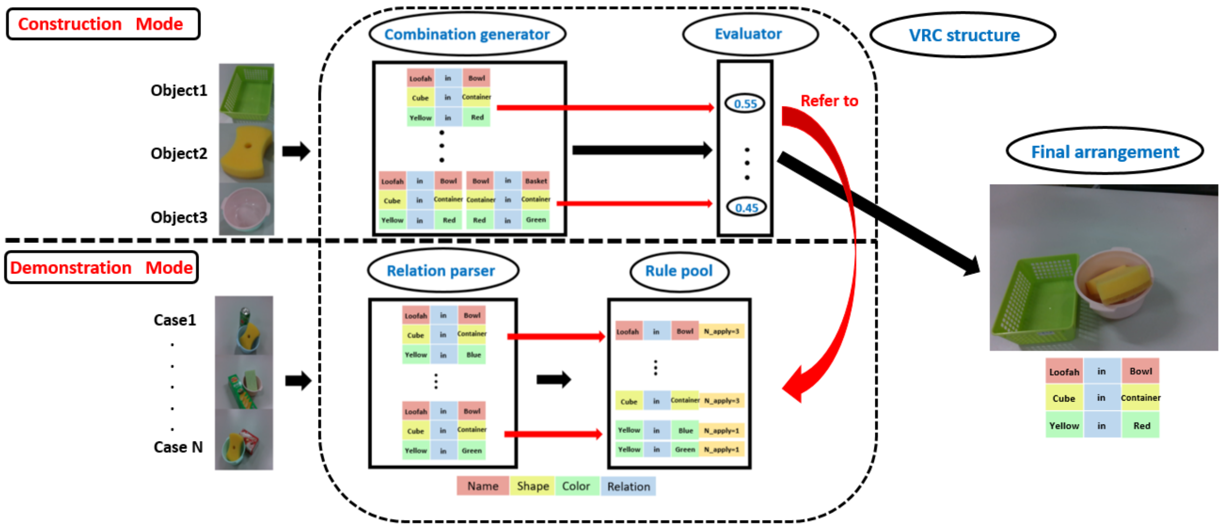


Figure 3. Visual Relationship Constructor pipeline. We divide the whole procedure into two parts. The upper part indicates the construction mode to prepare appropriate arrangement rule for random messy objects. We apply Combination generator to generate relation vectors that represent all kinds of combination and give each of them scores through the Evaluator. The lower part describes the demonstration mode which can learn the arrangement rule from several well-arranged demo cases. We extract every attribute rule through the Relation parser and save the rules into the Rule pool.

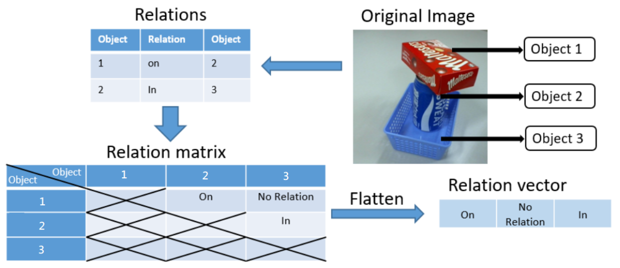


Figure 4. This figure presents how we use a relation vector to represent the visual relationships among objects in a image

color); (2) relation vector: as shown in Figure 4., we transform relations in a image into a relation vector. Besides, there is no output in demonstration mode. If **Construction mode** is selected, different from Demonstration mode, only list of objects are required. The output of this mode will be a relation vector representing relations in the image that all objects are well-arranged.

Processing Inputs

After list of objects and relation vectors are input, we first find the corresponding shape to its name because each name matches a certain shape. Then we transform list of objects and relation vectors into several relation triplets of object in *object1-relation-object2* format, where object1 can be regarded as subject, and object2 can be seen as object. Next, we further divide these object relation triplets attribute by attribute into relation triplets of attribute, denoted by r (i.e. *object1-relation-object2* will be divided into *name1-relation-name2*, *shape1-relation-shape2* and *color1-relation-color2*). Finally, we put every r into a relation queue Q.

Besides, there is one thing to be noticed, only when objects depend on each other directly will we consider there to be some kind of relation between them, otherwise, we consider it to be no relation (e.g. if loofah is on box and that box is in the basket, although the loofah seems to have some certain relation with box, there still be no relation between them).

Rule Pool

Rule Pool, denoted by P, stores information of rules appended at every round of demonstration. Rule pool is divided into 3 independent sub pools: P_{name}, P_{shape} and P_{color} to store rules of its attributes. Every rule r_p is composed of $type$, $attr1$, rel, $attr2$, n_{apply}, $n\_rd\_not\_apply$, $weight$, $dirty$, where $type$ represents attribute type of r_p and also implies which pool this rule is in; $attr1$, rel, $attr2$ have the same definition as relation triplets of attribute mentioned in previous section; n_{apply} shows the number that r_p has been applied and this argument will affect weight directly; $n\_rd\_not\_apply$ is used to record how many rounds r_p has not been applied; $weight$ denotes weight for each rule, higher weight means more important; $dirty$ is a argument used to record whether n_{apply} of r_p has been changed at this round, if $dirty$ is True, then compute new weight of r_p and update it.

Demonstration Mode: Update Of n_{apply}

For every r in Q, we try to find if it is already matched rule r_p existing in the sub pool of its attribute in P (e.g. given r is *red-on-yellow*, if there is a r_p that represents rule *red-on-yellow* in P_{color}, then r_p is the matched rule of r) . If r matched with a rule r_p, then r_p is considered to have been "applied" on this demonstration. Therefore, we add 1 to n_{apply} of r_p and set its $dirty$ to true.

Besides, we divide all rules into 2 categories: rules of general relation (on, in, contain, down) and rules of no relation. When detecting a rule of one category is applied, the n_{apply} of all rules in the other category will be subtracted (e.g. when *red-down-blue* is applied, n_{apply} of *red-no-blue* will decrease), and vice versa. By this mechanism, we solve the problem that

n_{apply} of a rule will rise monotonically resulting from lack of negative feedback.

Demonstration Mode: Weight Update

To prevent one significantly frequently-applied rule from dominating the evaluation result, the weights of rule are confined to interval [-1, 1]. Thus, we feed n_{apply} of rule into a weight function W composed of 2 upside down sigmoid functions and compute its weight.

$$W(n_{apply}) = \begin{cases} \frac{1}{(1+e^{-n_{apply}*\alpha+\beta})} & , n_{apply} > 0 \\ \frac{-1}{(1+e^{n_{apply}*\alpha-\beta})} & , n_{apply} < 0 \\ 0 & , n_{apply} = 0 \end{cases}$$

Here, α is a hyper parameter that can be considered as thriving speed or decline speed of weight. β is a hyper parameter that affect initial value of weight.

Moreover, We find that kind of "no relation" rule may be applied relatively frequently which means its weight increases too fast and will dominate the evaluation result soon. Therefore, we update rule in "no relation" category by a weight function with a smaller α.

In this stage, we only compute new weight for those rules that dirty bit is true and leave weight of others unchanged. Therefore, resource and time can be saved by avoiding doing unnecessary computation.

Construction Mode: Combinations Generator and Evaluator

Given a list of objects, the combination generator will generate all relation vectors that represent each possible relation combination of these objects without violating the restriction that one item cannot have the same relation with two items (e.g. cases like two objects on one object or one object on two objects are illegal). Finally, we put all these relation vector into a set V.

After generating all possible relation vectors, an evaluator will give each relation vector v in set V a score, and here are the grading steps. First of all, we turn v into relation triplets of attributes (as we do in preprocessing part), then find the matched rules in three sub pools of P respectively. Next, we average the weight of these matched rules of same attribute and get 3 average scores: a_{name}, a_{shape} and a_{color}. Finally, we multiply these average scores by their attribute weights and sum up three scores to obtain the final score of each v. The v with highest score will be considered the well-arranged relation vector (the relation vector representing the scene that VRC apply target rule successfully on all pair of object that can be applied and give no relation to the other pairs of object).

$$final\ score = W_{name} * a_{name} + W_{shape} * a_{shape} + W_{color} * a_{color}$$

Here, W_{name}, W_{shape} and W_{color} are three hyper parameters represent weight of each attribute and sum of them must equal to 1.

EXPERIMENTS

We conduct several experiments to evaluate our method.

Dataset

In our experiment, each object has three attributes - name, shape, and color. There are 6 types of name: box, loofah, bowl, basket, can and bottle; 4 types of color: red, yellow, green and blue; 3 types of shape: cube, container and cylinder, and each of the name corresponds to a shape: box (cube), loofah (cube), bowl (container), basket (container), can (cylinder) and bottle (cylinder). The relation between two object includes on, in, no (no relation), contain and down.

Generate Demonstration Cases and Testing Cases

In order to teach VRC some certain rules, we generate cases containing target rules that we want VRC to learn (e.g. if the target rule is *green-on-yellow*, we generate a series of cases with at least one pair of objects that are arranged by this rule and other relations will be randomly generated). Besides, the number of objects in a case is determined by two factors, the number of target rules and the number of extra objects. For example, if we want to generate demonstration cases which contain 2 target rules: *green-on-yellow* and *cube-in-container*, firstly we should generate a pair of objects that one of the colors is assigned green, the other is assigned yellow. Afterward, we generate another pair of objects and repeat the previous steps. For the rest of the attributes of objects that are not assigned in previous steps, we randomly assign one to it.

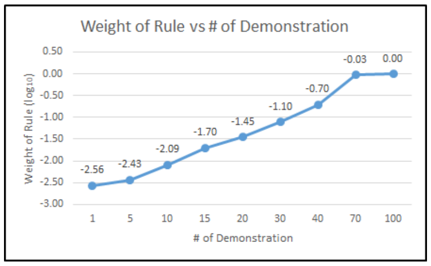
For the testing cases, similar to generating the demonstration cases, we generate a pair of objects with certain attributes that allow VRC to apply target rules on, then randomly generate other objects and attributes.

Relation between Weight of Target Rule And Number Demonstration

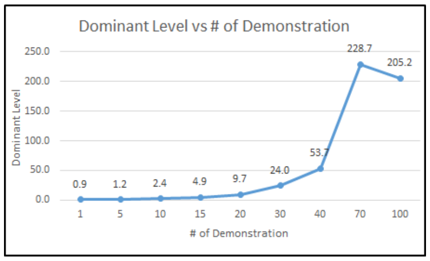
The correctness of arrangement result is directly affected by two factors: (1) weights of target rules and (2) dominant level (the ratio of weight of the target rule with the highest weight in its sub pool and weight of the non-target rule with highest weight in the same sub pool), if the weights of target rules are relatively large (i.e. dominant level is high), they can dominate their attribute pool and therefore, have a great possibility to be applied.

In this experiment, we demo 100 cases containing at least one pair of *yellow-on-red* to a VRC and test it with 10 cases after each demonstration to evaluate the capability of VRC to learn this rule. At the meantime, we record (1) the weight of target rule, (2) the dominant level of the sub pool where target rule is, and (3) the rate of generating well-arranged relation vectors out of 10 testing cases.

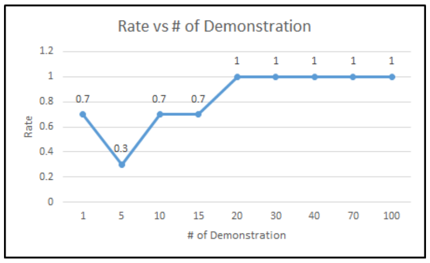
Inferring from curve in Figure 5(a)., we can know that the weight of target rule rises slowly at the beginning, while speeding up at the middle period, and slowing down again and finally approaching to 1 at the end. This is one of the features of adopting sigmoid function as weight function. See Figure5(c)., The rate of well-arranged relation vectors also grow and eventually fixed at 1.



(a)

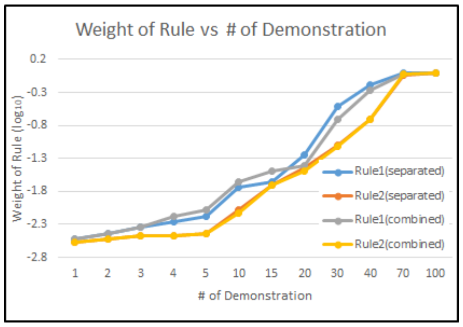


(b)

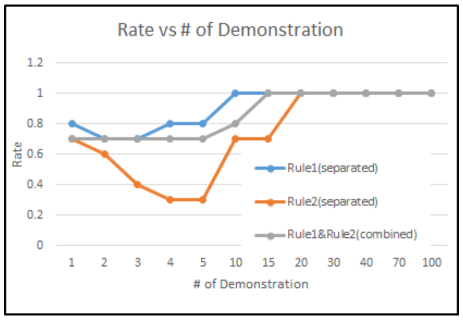


(c)

Figure 5. (a) The growing curve of weight of single target rule (values of y-axis are logarithm of weight to the base 10). (b) The curve of dominant level (c) Rate of well-arranged relation vectors out of 10 testing cases. (In this experiment, α for general relation = 0.2, for no relation = 0.1; β = 6)



(a)



(b)

Figure 6. In the charts, "Separated" right after the curve title suggest that we only demonstrate the cases of same series to VRC; "Combined" right after the curve title suggest that we demonstrate the cases from 2 different series sequentially. (a) Depict the thriving tendency for weight of rules (values of y-axis are logarithm of weight to the base 10). (b) Rate of well-arranged relation vectors out of 10 testing cases (20 testing cases for those curve title followed by "combined"). (In this experiment, α for general relation = 0.2, for no relation = 0.1; β = 6)

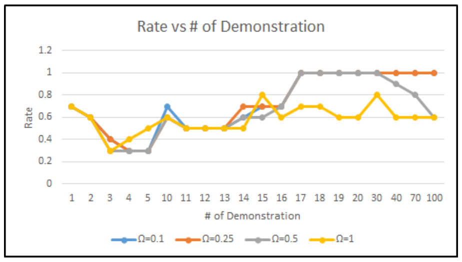


Figure 7. This chart shows how Ω affects rate of well-arranged relation vectors out of 10 testing cases. (In this experiment, α for general relation = 0.2; β = 6)

Multiple Series Of Cases Demonstration

Different series of cases (i.e. cases of different target rule) may lead to some contradictions that lower the learning effectiveness of VRC. Therefore, the weights of target rules cannot rise to 1, which leads to a low-performance model. In this experiment, we use two ways to demonstrate cases of target rule A and target rule B. In the first part, we demonstrate 1 case of target rule B after 1 case of target rule A in every round and repeat this until completing 100 rounds of this kind of demonstration. After each round, we evaluate this model by 10 testing case from series A and 10 from series B. In the second part, We demonstrate cases of two target rule to 2 VRC model respectively and evaluate them with 10 testing cases of their own target rule.

Comparing the evaluation results in Figure 6(a)., we can find that the accuracy and the weight of the target rule of demonstrating rules separately and are approximately the same as those VRC models after 100 demonstration.

Balance Between Thriving Speed Of General Relation and Thriving Speed Of No Relation (Ω)

First, Ω denotes the ratio of thriving speed of weight of no relation and that of general relations(on, in, down, contain), if Ω is too large (i.e. thriving speed of weight of no relationship is too high), it may become dominating rule in a short time and prevent other rule from being chosen. As a result, VRC will output all relation as no relation. On the other hand, if Ω is too small, no relation rule becomes too weak to be chosen. Therefore, VRC may apply another none target rule on a pair of objects that is originally expected to apply no relation on.

Figure 7. shows the result of accuracy versus number of demo with different Ω. The accuracy tend to be unstable irrelevant to the value of Ω at the beginning and become significantly different after several cases demonstration. When Ω equals to 0.1 or 0.25, their accuracy curve are very similar to each other and stably stay at 1 after demonstrating enough cases. When Ω equals to 0.5, the accuracy curve is as stable as the previous two, while drop-down after about 30 demonstration. When Ω equals to 1, the accuracy curve gradually ascents to about 0.7 and start to oscillate between 0.7. Judging from the results above, let Ω equal to 0.1 or 0.25 is suitable for VRC predicting a good relation vectors in this case.

CONCLUSIONS

In this paper, we focus on applying proper arrangement style which is learned from user demonstrations on random messy objects in vertical direction. We propose a new model named Visual Relationship Constructor (VRC), which can learn some rules from objects arranged by user and gradually form a style similar to its owner. We solve task transfer problem by recording not only object name but its color and shape, and then parse the relation between 2 attributes. As a consequence, even though VRC gets objects unseen before, it can still transfer existing arrangement style to this new case by focusing on certain attributes of these objects.

Though VRC has several constraints and difficulties to overcome, it is an essential step for the revolution of multiple objects arrangement. By finding ways to avoid those constraints, there's a great chance that VRC could be applied to more complicated household arrangement problems, such as washing dishes organizing or random surrounding arrangement. We hope that VRC can be implemented on a real robot someday and makes a contribution to the object arrangement field in the near future.

ACKNOWLEDGEMENTS

We thank Charles Tsao, Stanley Chai and Robert Wu for discussing with us and helping us figure out the solutions to those difficulties we met during research. Besides, thanks for Stanley Chai spending time teaching us to use some devices and accompany us to conduct some experiments.

REFERENCES

[1] Nichola Abdo, Cyrill Stachniss, Luciano Spinello, and Wolfram Burgard. Robot, organize my shelves! tidying up objects by predicting user preferences. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 1557–1564. IEEE, 2015.

[2] Hanbo Zhang, Xuguang Lan, Xinwen Zhou, Zhiqiang Tian, Yang Zhang, and Nanning Zheng. Visual Manipulation Relationship Network. *arXiv e-prints*, page arXiv:1802.08857, Feb 2018.

[3] Weijie Cheng, Guisheng Yin, Yuxin Dong, Hongbin Dong, and Wansong Zhang. Collaborative filtering recommendation on usersâĂŹ interest sequences. *PloS one*, 11(5):e0155739, 2016.

[4] Nichola Abdo, Cyrill Stachniss, Luciano Spinello, and Wolfram Burgard. Organizing objects by predicting user preferences through collaborative filtering. *The International Journal of Robotics Research*, 35(13):1587–1608, 2016.

[5] Mincheul Kang, Youngsun Kwon, and Sung-Eui Yoon. Automated task planning using object arrangement optimization. In *2018 15th International Conference on Ubiquitous Robots (UR)*, pages 334–341. IEEE, 2018.

[6] Martin J Schuster, Dominik Jain, Moritz Tenorth, and Michael Beetz. Learning organizational principles in human environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 3867–3874. IEEE, 2012.

[7] Lars Kunze, Chris Burbridge, and Nick Hawes. Bootstrapping probabilistic models of qualitative spatial relations for active visual object search. In *2014 AAAI Spring Symposium Series*, 2014.

[8] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[9] Sigurd Spieckermann. *Multi-Task and Transfer Learning with Recurrent Neural Networks*. PhD thesis, Technische Universität München, 2015.

[10] Philipp Jund, Andreas Eitel, Nichola Abdo, and Wolfram Burgard. Optimization beyond the convolution: Generalizing spatial relations with end-to-end metric learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2018.