SceneScore: Learning a Cost Function for Object Arrangement

Ivan Kapelyukh[1,2], Edward Johns[1]

***Abstract*— Arranging objects correctly is a key capability for robots which unlocks a wide range of useful tasks. A prerequisite for creating successful arrangements is the ability to evaluate the desirability of a given arrangement. Our method “SceneScore” learns a cost function for arrangements, such that desirable, human-like arrangements have a low cost. We learn the distribution of training arrangements offline using an energy-based model, solely from example images without requiring environment interaction or human supervision. Our model is represented by a graph neural network which learns object-object relations, using graphs constructed from images. Experiments demonstrate that the learned cost function can be used to predict poses for missing objects, generalise to novel objects using semantic features, and can be composed with other cost functions to satisfy constraints at inference time. Videos are available at: sites.google.com/view/scenescore.**

I. Introduction

Object rearrangement is a ubiquitous challenge in robotics: given a set of objects, arrange them into a desirable state [1]. Many tasks can be expressed as rearrangement problems, e.g. tidying a room, loading a dishwasher, assembling furniture in a factory, or setting a table in a restaurant. For a robot to be proficient at rearrangement in the real world, it must be able to evaluate the desirability of an arrangement. Our method “SceneScore” learns a cost function for arrangements, such that desirable, human-like arrangements have a low cost, and random arrangements have a higher cost. This cost function can then be minimised to determine low-cost target poses for objects (Fig. 1). This can be used to provide target poses for robotic systems which can physically perform the rearrangement [2], [3], [4].

Prior work [5], [6] has explored the problem of predicting an optimal arrangement for a set of objects. This is a related but different problem. By learning a cost function, our method can also be used to find an optimal arrangement. However, learning a cost function for any given arrangement (an *implicit* approach) gives our method several advantages compared to methods which predict only the optimal arrangement (an *explicit* approach). These benefits include:

(1) Compositionality. The learned cost function can be composed with additional cost functions at inference time. E.g. the time it would take the robot to create an arrangement can be added to the desirability cost of that arrangement. The composed cost function can then be differentiated to efficiently find an arrangement which is both desirable and can be created quickly. Explicit methods cannot be easily composed without re-training, as discussed in [7].

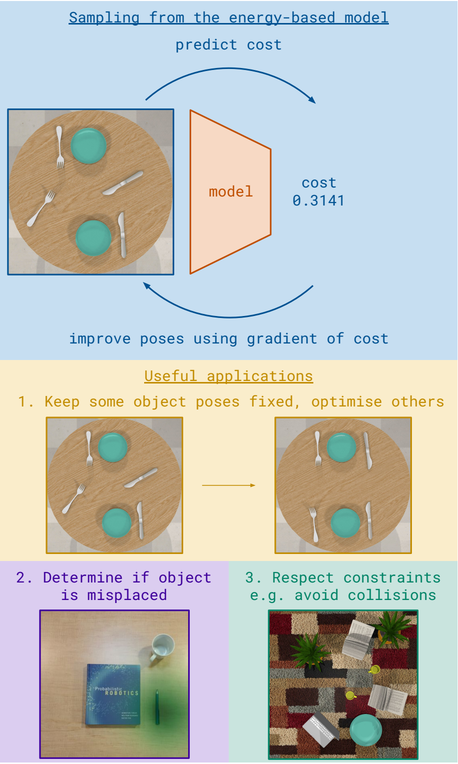
Sampling from the energy-based model
predict cost
model
cost
0.3141
improve poses using gradient of cost
Useful applications
1. Keep some object poses fixed, optimise others
2. Determine if object is misplaced
3. Respect constraints e.g. avoid collisions

Fig. 1: An overview of how to sample from our energy-based model and several useful properties of implicit methods.

(2) Robustness. This is a consequence of compositionality. In real-world scenes, there are many physical constraints which must be satisfied involving robot joints, object collisions, stacking stability, etc. Explicit methods are less robust, since the predicted optimal arrangement may violate constraints. The cost function approach enables the use of efficient gradient-based constrained optimisation to determine a high-quality arrangement which satisfies these constraints.
(3) Theoretical advantages. Prior work [8] has shown that implicit methods have theoretical advantages over explicit methods, such as better handling of discontinuities and stronger generalisation.

When humans evaluate the quality of an arrangement, we consider many factors: aesthetics, convenience, stability, and others. We therefore learn this function from images of human-arranged scenes. This is a scalable research direction as images such as tidy desks and loaded dishwashers are abundant on the Web. Unsupervised learning from this web-scale data can lead to generalist cost functions for object arrangement, analogous to successes in other fields [9], [10].

We learn the distribution of example arrangements using

[1] The Robot Learning Lab at Imperial College London. [2] The Dyson Robotics Lab.

an energy-based model, which can successfully learn complex distributions [7], [8]. However, learning the distribution from images directly is difficult due to their high dimensionality. Instead, we first create an object-centric graph representation of scenes, which separates an object's pose and semantic features. One of our key insights is that this allows us to make the cost function conditional: the semantic identities of the objects must remain fixed, but the robot is free to vary the poses in order to create a higher-quality arrangement.

The main contributions of this paper are as follows:

- **An algorithm for learning the distribution** of example arrangements using techniques from energy-based modelling, which enables a cost function to be learned solely from example object arrangements, without also requiring environment interaction or human supervision.
- **A graph neural network architecture** for predicting the cost of an arrangement, with relative poses as edge features and pose-invariant semantic embeddings as node features. This object-centric, abstracted representation which separates pose and semantics makes generating arrangements offline possible.
- **A vision pipeline for creating graphs** from images of scenes, using a **pre-trained CLIP** [11] model to obtain visual and semantic features for each object, thus allowing generalisation to new objects.

To the best of our knowledge, this is the first neural network method for learning a rearrangement cost function which uses only example images for training. Please visit our website for videos, code, and supplementary materials with real-world demos: sites.google.com/view/scenescore.

II. Related Work

We now summarise several common approaches to object rearrangement, followed by a brief overview of energy-based models, the basis of our approach.

Classification methods select target poses from a fixed set of discrete choices. Generalisation to novel objects can be achieved using object taxonomies [12] or language models [13]. A graph neural network [14] or transformer [15] policy can select the correct goal for each object. This can be trained with reinforcement learning to complete long-horizon rearrangement tasks [16]. While these are effective for task planning, our method addresses the case of predicting a *continuous* target position and orientation for each object.

Explicit methods directly predict target poses for each object. These solve a related but different problem to learning a cost function. NeatNet [5] models user preferences as latent vectors by training as a Variational Autoencoder on scenes. However, this model only uses object names as semantic embeddings, whereas our method also uses visual features. Point clouds can also be used to represent objects as shown in [6], where an autoregressive model conditioned on language commands is used to place objects into the scene. Another line of work takes a "denoising" approach to determine goal arrangements [17], [18], [19]. While this helps avoid collisions in rearrangement, it addresses a different problem: we are interested in developing an implicit method which, given an arrangement, can *evaluate* how desirable it is.

Energy-based models (EBMs) are generative models which approximate the training distribution using an unnormalised energy function. Intuitively, the energy function is usually "pushed down" at training examples, and "pushed up" elsewhere. They have been used for generating high-dimensional images [20]. They have also been used for learning visual and spatial concepts [21], and for learning from demonstrations [8], [22]. We use EBMs to model the distribution of example arrangements. Recent work [23] demonstrates compositionality by training a separate EBM for every spatial relation (such as *"left of"* or *"line"*), and composing them together to comply with detailed user instructions. Our work investigates a different problem: learning to evaluate autonomously whether an arrangement of a scene looks natural, without requiring detailed user instructions. This leads to several differences in our contributions. For example, in Section IV-B, we train a single EBM to learn the joint distribution of object poses (including orientations), capturing what it means for a dining table to be conveniently set for human use, and avoiding the need to train separate EBMs for each low-level spatial relation. Additionally, our EBM is conditioned on visual CLIP features (rather than just object positions and sizes), allowing for semantic generalization. We also show how the performance advantage of implicit methods grows over explicit methods as collision constraints become more restrictive.

Zero-shot approaches use visual-language models (VLMs) to predict desirable target poses without training on example arrangements [24], [25]. Instead, SceneScore allows users to provide their own examples which reflect their needs.

III. Method

We start by formulating the learning of a rearrangement cost function as a density estimation problem (Section III-A). Then, we describe a method for learning this density function from example arrangements (Section III-B), and show how to sample arrangements from this model (Section III-C). Then, we detail the graph neural network architecture used to represent our model (Section III-D). Finally, we illustrate how these graphs can be constructed from images of scenes (Section III-E).

A. Formulating the Problem

To approximate the function that humans use to assess the quality of a scene, we use a neural network model E_θ, with learned parameters θ. The architecture is described in Section III-D. We use the notation E_θ because this network is trained as an EBM. The input to E_θ is a representation of a scene, and the output is a scalar cost.

A scene is represented as an object-centric graph (the input to our model). Each object in a scene is represented by its pose and its semantic embedding. The semantic embedding is a vector which captures visual and semantic features of an object that are useful for arranging it, for example its shape, or its semantic function (e.g. cutlery is often placed together in a drawer). This graph is constructed from an image, as described in Section III-E. Let x denote the set of absolute pose vectors for all the objects in a scene, and s the set of semantic embeddings. Then the cost of the scene according to our model is $E_\theta(x|s)$. This separation between pose and semantics is crucial under the definition of rearrangement: the robot is allowed to vary the pose of objects x, but it cannot alter or discard the objects themselves, so s must remain fixed.

B. Training the Energy-Based Model

We fit the model to the training examples using Maximum Likelihood Estimation (MLE). The probability of an arrangement under the distribution learned by our model is defined as:

$$p_\theta(x|s) = \frac{e^{-E_\theta(x|s)}}{Z_\theta} \qquad Z_\theta = \int_x e^{-E_\theta(x|s)} \mathrm{d}x \tag{1}$$

This probability definition is widely used in EBM literature [26], [7], [8], [27]. Arrangements with a lower cost have a higher probability. The MLE loss is obtained by minimising the negative log-likelihood. We cannot compute an integral over all possible arrangements, but we can approximate the normalisation effect of Z_θ by sampling arrangements from our learned distribution p_θ. The sampling algorithm is described in Section III-C. Replacing Z_θ using the training example i and the sample arrangements indexed by j, we get:

$$L_{\text{MLE}}(\theta) = \sum_i -\log\left(\frac{e^{-E_\theta(x_i|s_i)}}{e^{-E_\theta(x_i|s_i)} + \sum_j e^{-E_\theta(x_j|s_i)}}\right) \tag{2}$$

This takes the form of an InfoNCE-style loss [28], analogous to those used for EBM training in [8], [22]. Note that $s_i = s_j$, i.e. the samples j all have the same semantic embeddings as the training example i. Intuitively, this loss function encourages the model to assign a low cost (i.e. high probability) to the training examples, and a high cost (i.e. low probability) to the generated arrangements which have been sampled from the model – they can be viewed as counter-examples. This creates a high-probability region around the training examples, and "pushes down" the probability elsewhere.

C. Sampling from the Energy-Based Model

We need to sample from the learned distribution $E_\theta(x|s)$ to approximate the normalising constant in the loss function in (2). To sample poses for a given set of objects with semantic embeddings s, we use Langevin Dynamics [29], which is often used to sample from energy-based models [7], [8]. The initial poses x_0 are drawn randomly from a uniform distribution. The poses are then updated in each step t:

$$x_t = x_{t-1} - \lambda_t(\nabla_{x_{t-1}} E_\theta(x_{t-1}|s) + \omega_t) \quad \omega_t \sim \mathcal{N}(0, \sigma_t^2) \tag{3}$$

We take the final poses x as the sampled arrangement, following [29]. Here, $\nabla_{x_{t-1}}$ means taking the gradient of the cost function with respect to the object poses. λ_t is the step size. This update rule is similar to gradient descent, but with an added noise term ω_t. Langevin Dynamics is run for a finite, fixed number of steps. Further details about hyperparameters are in the supplementary material. At inference time, we also use Langevin Dynamics to sample low-cost arrangements.

D. Graph Neural Network Architecture

$E_\theta(x|s)$ is represented by a graph neural network (GNN). Each scene is a fully-connected graph with a node for each object. The edge between two object nodes represents the relative displacement and orientation between them. GNNs are well-suited for this task because they can handle inputs with a variable number of nodes, and they can capture complex, multi-modal training distributions, e.g. if there are multiple acceptable poses for an object. An insight we make is that relative poses between objects often matter more than their absolute pose in the scene, e.g. a pair of slippers being placed together. This motivates our use of relative poses as edge features in the graph, compared to prior work [5] which uses absolute coordinates in the node feature vectors. When sampling, we convert the absolute poses x to relative poses, use the GNN to compute the cost of the scene $E_\theta(x|s)$, and then improve the absolute poses using (3).

We now describe how one layer of our GNN computes the output features for each node, which are used as input features to the next layer. The input features in the first layer are the semantic embeddings. The edge features stay the same at each layer of the network. For node i, the input feature vector is v_i, and the output feature vector is v_i':

$$v_i' = \sum_j f_\phi(v_i, v_j, e_{ji}) \tag{4}$$

For each neighbouring node j in the graph, we compute a message from j to i, and then aggregate all these messages to produce the output feature vector for node i. To compute this message, the feature vectors of nodes i and j are concatenated together, along with the edge feature vector e_{ji}, which represents the pose transformation to get from the pose of j to the pose of i. This concatenated vector is then passed through a linear neural network layer with learned parameters ϕ. The GNN consists of several of these graph layers, each separated by a LeakyReLU non-linearity [30]. We use global add pooling to aggregate the node feature vectors into a single graph encoding vector. This is passed through several linear layers, the output of which is a scalar, which is the cost $E_\theta(x|s)$.

E. Constructing Graphs from Images

The GNN takes as input a graph representing a scene. The system for constructing this graph from an image, and obtaining a cost for this graph, is shown in Fig. 2. First, our method detects objects in the image. We use a pre-trained Mask R-CNN [31] from the detectron2 library [32]. For each object instance, it returns a segmentation mask. We found that our method works even for objects not in the

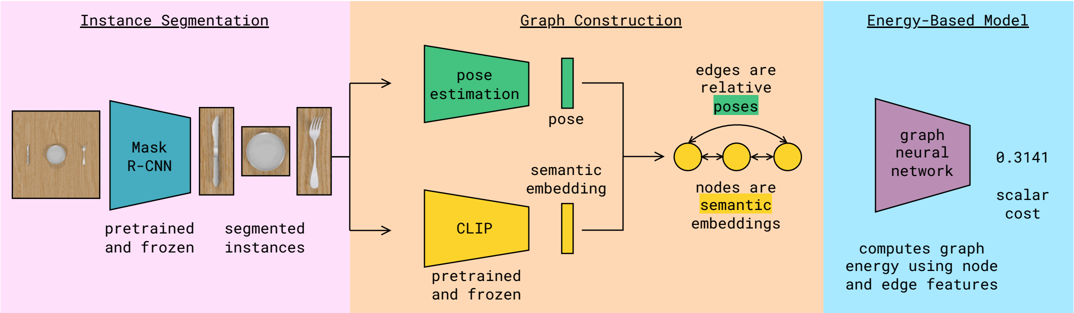

Fig. 2: The pipeline for computing a cost from an image of a scene.

Mask R-CNN training dataset, as long as the mask has an approximately correct shape.

Next, the pose for each object should be estimated. Our method makes it easy to use any existing pose estimation component, including 6-DoF pose estimators. In our experiments, we focus on tabletop scenes where the methods should predict the x and y position of each object, along with a single angle θ along the axis perpendicular to the tabletop. We use a straightforward method for pose estimation based on image moments, which also applies to novel objects. Further details are in the supplementary material. The pose vector for an object is $(x, y, \cos\theta, \sin\theta)$.

To derive an object's semantic embedding we use features from a pre-trained CLIP model [11]. This takes as input an image of the object and returns a 512-dimensional CLIP vector which captures visual features, as well as semantics: e.g. a fork and chopsticks may share some semantic features, which is useful as they are arranged in a related way. The semantic embedding of an object should be pose-invariant: the pose is separate so that we can optimise it. Therefore, we rectify and crop the image of the object before inputting it to CLIP. Details are in the supplementary material. Although the CLIP model weights are frozen, we train a semantic embedding extractor end-to-end, which is a 2-layer MLP that extracts useful features from the CLIP vector. However, when we pre-process an object image to be used as CLIP input, we lose information about the scale of that object, which may be useful for arranging it (e.g. ordering by size). To preserve this information, we compute a scale descriptor (the width and height of the object in its rectified pose), and append this to the output of the semantic embedding extractor to create the object's semantic embedding.

IV. Experiments

We investigate the following research questions: How accurately can our method predict poses for missing objects (Section IV-B)? Can our method generalise to novel objects (Section IV-C)? Can the learned cost function be composed with constraints at inference time (Section IV-D)? Further qualitative results are available on our website: sites.google.com/view/scenescore.

A. Experimental Setup

We create a dataset of images of arranged scenes in simulation, in order to compare predicted poses against ground-truth poses. We use a data-generating process with ground truth distributions, e.g. sampling relative poses between objects from a Gaussian Mixture Model. Details are in the supplementary materials, along with the full datasets. The methods must infer this distribution from example images alone, which are created by rendering the example scenes in a simulator.

B. Predicting Poses for Missing Objects

In this experiment, the method is shown an arranged scene at inference time, with an object missing. The method must predict the correct pose for the missing object, taking into account the poses of the pre-placed objects, and using its learned object-object relations. We compare the following methods:

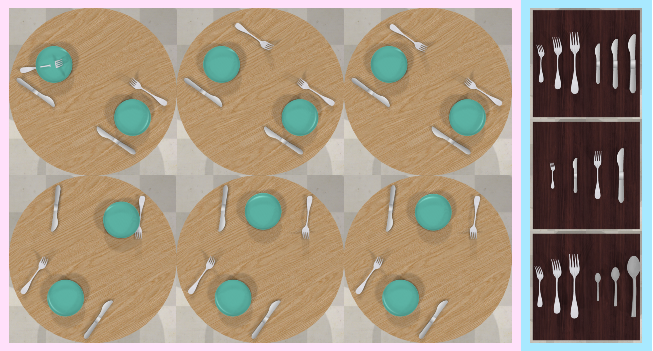

Fig. 3: **Left:** results for placing missing objects. Top: placing fork, bottom: placing bowl. From left to right, methods are: `Nearest-Nbr`, `SceneScore`, ground truth. **Right:** experiment which requires ordering novel objects.

(1) `SceneScore`. Our method learns the distribution of training arrangements. At inference time, the poses of the pre-placed objects are fixed. The missing object's position is randomly initialised, and then optimised using Langevin Dynamics based on the learned cost function. (2) `SceneScore-Abs`. This ablation study is similar to our method, except that absolute poses are included as node features instead of relative poses in edge features. (3)

Nearest-Nbr. This baseline compares the poses of pre-placed objects against each training arrangement to find the closest match, and from that training arrangement returns the pose of the object which is missing from the test arrangement. We expect it to perform well in this experiment because the set of objects is fixed, however it cannot generalise to novel objects. (4) NeatNet [5]. We compare against this prior work for arranging objects, which trains a masked VAE. At inference time, the pose of the missing object is masked out, and is predicted by the decoder. For a fair comparison, we first extend this method to handle orientation. Further implementation details are in the supplementary materials.

For this experiment, we focus on a dining table scenario (Fig. 3 (left)). There are a pair of plates which are placed circularly at any angle around a table. Beside each plate, a fork and knife are placed an approximately equal distance apart, where this distance varies. To test the methods on multi-modal distributions, the fork and knife can be placed on either side for each arrangement. Together, this creates a challenging task because the methods have to learn relative poses in a circular arrangement, handle multi-object relations, place multiple instances of the same class, and learn multi-modal distributions. There are 48 arrangements for training and 16 for testing.

As shown in Table I, our method SceneScore outperforms the baselines, meaning that the cost function is learned correctly. Using relative poses as edge features improves performance, as shown by the comparison with the SceneScore-Abs ablation. Nearest-Nbr is not able to generalise well to novel configurations at inference time. NeatNet struggles to learn relative poses in the complex circular arrangements, and to tell apart instances of the same class, showing that SceneScore's EBM approach is better able to learn these distributions and generalise to new configurations.

C. Generalising to Novel Objects

We now test the method's ability to use semantic features to generalise to novel objects. A common rearrangement task is to categorise and order objects: laying out cutlery in the kitchen, stacking plates in order of size, etc. We use this setting in this experiment (Fig. 3 (right)). The methods are shown training examples where objects of varying size and class are ordered according to some rule, which must be learned by the method from the training images. At test time, a set of novel objects is presented to the method, and it must generalise the learned rule to these novel objects. In the first scenario, the objects are first categorised according to class (fork vs knife), and then ordered according to size within each class. In the second scenario, the objects are all ordered according to size, regardless of class. At test time, the forks and knives are of different sizes to those seen during training. In the third scenario, the model only sees knives and forks during training, arranged as in the first scenario. At inference time, it must generalize to an *unseen class*, i.e. spoons, which appear alongside forks in the test scene. This is challenging because it must group the unseen objects together and generalize the learned ordering pattern to them. There are 16 training and 16 test scenes for each scenario. We compare SceneScore against NeatNet-R [5], which learns to predict poses from semantic embeddings via regression, where the word embedding for each object is derived from the Mask R-CNN class output.

The results are in Table II. Our method SceneScore is better able to generalise to novel objects. NeatNet-R can categorise objects by class using language, but our method allows for more precise placement because it also includes visual features in its semantic embeddings. The Unseen-Class task is more challenging for both methods, causing higher-variance results. Our method uses CLIP features and can generalise the learned object relations to scenes with an unseen class.

D. Composing the Learned Cost Function with Constraints

In this experiment we investigate how constraints can be incorporated into a rearrangement method. This is particularly important for realistic, cluttered scenes. We compare two approaches: an implicit method (ours) which samples a solution and then performs gradient-based constrained optimisation, and an explicit method (NeatNet [5]) which samples solutions and rejects those that violate constraints. We focus on object-object collision avoidance constraints, but this approach can also be used with constraints such as how far the robot arm can reach along a table.

Suppose that the robot learns a model which evaluates the quality of a television and stereo speaker setup on a living room floor, as shown in Fig. 4. For a balanced audio experience, all three objects should be aligned vertically in a straight line. Additionally, the setup should be horizontally symmetrical, such that the left and right speakers are equidistant from the television. During training, the methods are provided with 36 training arrangements, generated using a similar process to Section IV-A. They must learn that distribution of high-quality arrangements. At inference time, the methods must sample quality arrangements which also satisfy constraints that prevent any objects from colliding. Furthermore, there are now several new objects placed in the scene as clutter, which are included in the constraints, but cannot be moved by the method. These clutter objects are excluded from the graph before pose prediction, since they cannot be moved and were not trained on. Each method is allocated an equal budget of 5000 samples they are allowed to draw, and we report the number of correct samples generated by each method. A correct sample is defined as one which has alignment of objects within a fixed threshold, and contains no object collisions. Collisions are detected from the segmentation masks for objects, but in future work a model such as CollisionNet [33] can be used. Further details are in the supplementary material.

For the NeatNet [5] baseline, we sample arrangements from the latent space of the VAE and reject the samples that violate the constraints.

For the SceneScore method, the samples are drawn using Langevin Dynamics, as before: the arrangements are

Method	Bowl	Fork		Knife		Mean	
	t	t	R	t	R	t	R
`NeatNet` [5]	17.6±0.7	24.1±1.8	147.8±21.5	23.6±1.4	145.4±23.5	21.7	146.6
`Nearest-Nbr`	5.4±3.4	9.9±9.0	16.0±11.8	8.4±8.0	17.9±11.5	7.9	16.9
`SceneScore-Abs`	5.2±6.6	6.3±5.3	9.1±4.5	7.8±8.7	20.4±40.0	6.5	14.8
`SceneScore`	**3.4**±2.3	**4.1**±2.4	**6.4**±4.9	**5.6**±4.3	**13.0**±17.9	**4.4**	**9.7**

TABLE I: Placing missing objects. For each object, the distance error t in cm between the predicted and true positions is shown, followed by the orientation error R in degrees (excluding the bowl due to rotational symmetry).

initialised randomly and then optimised using the gradient of the learned cost function, summed with the gradient of the constraint function. To make the constraint function differentiable, we use a standard Hinge Loss, which is zero when two objects are not in collision, and linearly increases as the object overlap increases if there is a collision.

The results are in Fig. 5. **As the scene becomes more cluttered and complex, the performance advantage of our implicit method increases over explicit methods** such as NeatNet. Although it is possible for a solution to be sampled from the VAE which luckily satisfies all the constraints, our gradient-based constrained optimisation approach scales much better as the constraints become more challenging, as is the case in complex real-world scenes.

V. Real-World Demo

We conduct a small-scale demo of our method on a real scene, and visualise qualitative results in Fig. 6. As the pen moves further from the book in the training examples, the mug moves further as well. By fixing the other objects and visualising the cost function for the mug positions, we can see that the model has learned this spatial relation. Additionally, it can generalise this relation when the pen is replaced with an unseen class (a pencil), using semantic features. Further details are in the supplementary material.

VI. Conclusions

Findings. Our method SceneScore learns the distribution of example arrangements from images, using an EBM and a graph representation of scenes. It learns offline, without environment interaction or human supervision. The learned cost function can be used to create low-cost arrangements, generalise to novel objects, and can be composed with constraints at inference time. This is the first method for learning a cost function for arranging objects from images.

Limitations & future work. We currently focus on top-down arrangements. This is sufficient to solve many re-arrangement tasks, but future work can also apply our method in a 3D context, e.g. with shelves. Our experiments are mostly in simulation, since we want to compare our predictions against ground truth poses. There are small-scale real-world demos on our website, but future work can investigate this approach in more complex real-world scenes. A promising direction for future work based on this approach is large-scale learning from in-the-wild images of scenes arranged by humans.

Method	Class-Size	All-Size	Unseen-Class	Mean
`NeatNet-R`[5]	6.55±4.35	12.95±6.84	17.30±11.24	12.27
`SceneScore`	**2.89**±1.53	**2.01**±0.94	**4.76**±3.07	**3.22**

TABLE II: Mean and standard deviation of position error in cm for ordering novel objects.

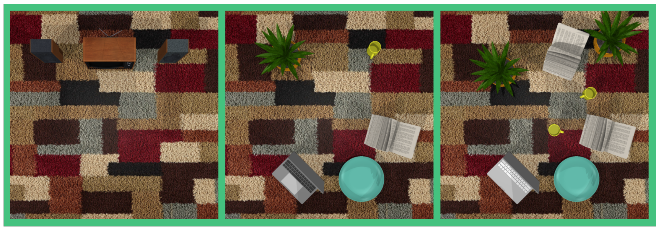

Fig. 4: **Left:** a training example for arranging the television and two stereo speakers. **Centre:** clutter added at inference time. **Right:** even more challenging clutter.

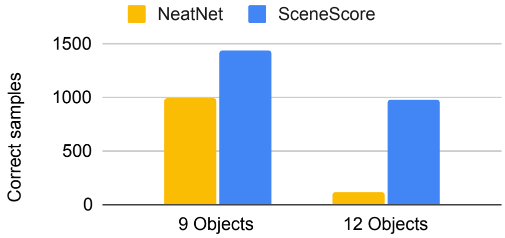

Fig. 5: Number of correct samples from each method as the total number of objects in the scene increases and collisions become harder to avoid.

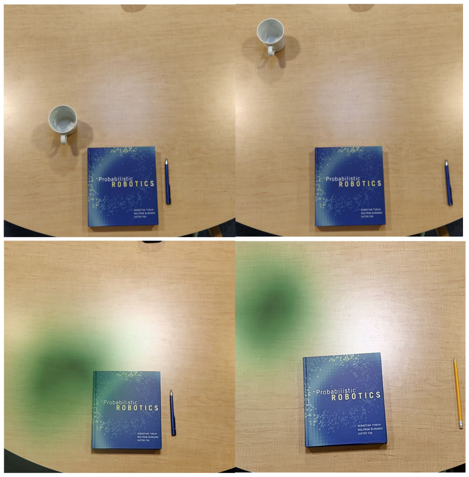

Fig. 6: A small real-world demo. We visualise the learned cost function for placing a mug in a desirable way.

References

[1] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, M. Savva, and H. Su, "Rearrangement: A challenge for embodied AI," 2020.

[2] A. Goyal, A. Mousavian, C. Paxton, Y.-W. Chao, B. Okorn, J. Deng, and D. Fox, "Ifor: Iterative flow minimization for robotic object rearrangement," in *arXiv:2202.00732*, 2022.

[3] V. Vosylius and E. Johns, "Where to start? collision-free transfer of skills to new environments," in *Conference on Robot Learning (CoRL)*, 2022.

[4] A. Murali, A. Mousavian, C. Eppner, A. Fishman, and D. Fox, "CabiNet: Scaling neural collision detection for object rearrangement with procedural scene generation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2023. [Online]. Available: https://arxiv.org/abs/2304.09302

[5] I. Kapelyukh and E. Johns, "My house, my rules: Learning tidying preferences with graph neural networks," in *Conference on Robot Learning (CoRL)*, 2021.

[6] W. Liu, C. Paxton, T. Hermans, and D. Fox, "Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects," *2022 International Conference on Robotics and Automation (ICRA)*, 2022.

[7] Y. Du and I. Mordatch, "Implicit generation and modeling with energy based models," in *Advances in Neural Information Processing Systems*, 2019.

[8] P. Florence, C. Lynch, A. Zeng, O. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," *Conference on Robot Learning (CoRL)*, 2021.

[9] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, "Photorealistic text-to-image diffusion models with deep language understanding," 2022.

[10] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, 2020.

[11] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, 2021.

[12] M. J. Schuster, D. Jain, M. Tenorth, and M. Beetz, "Learning organizational principles in human environments," in *2012 IEEE International Conference on Robotics and Automation*, 2012.

[13] Y. Kant, A. Ramachandran, S. Yenamandra, I. Gilitschenski, D. Batra, A. Szot, and H. Agrawal, "Housekeep: Tidying virtual households using commonsense reasoning," 2022.

[14] Y. Lin, A. S. Wang, E. Undersander, and A. Rai, "Efficient and interpretable robot manipulation with graph neural networks," 2021.

[15] V. Jain, Y. Lin, E. Undersander, Y. Bisk, and A. Rai, "Transformers are adaptable task planners," in *6th Annual Conference on Robot Learning*, 2022. [Online]. Available: https://openreview.net/forum?id=Eal_lL08v_l

[16] N. Funk, G. Chalvatzaki, B. Belousov, and J. Peters, "Learn2Assemble with structured representations and search for robotic architectural construction," in *5th Annual Conference on Robot Learning*, 2021.

[17] W. Liu, T. Hermans, S. Chernova, and C. Paxton, "StructDiffusion: Object-centric diffusion for semantic rearrangement of novel objects," *arXiv*, 2022.

[18] M. Wu, fangwei zhong, Y. Xia, and H. Dong, "TarGF: Learning target gradient field for object rearrangement," in *Conference on Neural Information Processing Systems*, 2022.

[19] Q. A. Wei, S. Ding, J. J. Park, R. Sajnani, A. Poulenard, S. Sridhar, and L. Guibas, "Lego-net: Learning regular rearrangements of objects in rooms," *arXiv*, 2023.

[20] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced score matching: A scalable approach to density and score estimation," 2019.

[21] I. Mordatch, "Concept learning with energy-based models," 2018.

[22] A. Ganapathi, P. Florence, J. Varley, K. Burns, K. Goldberg, and A. Zeng, "Implicit kinematic policies: Unifying joint and cartesian action spaces in end-to-end robot learning," 2022.

[23] N. Gkanatsios, A. Jain, Z. Xian, Y. Zhang, C. G. Atkeson, and K. Fragkiadaki, "Energy-based models are zero-shot planners for compositional scene rearrangement," *Robotics: Science and Systems XIX*, 2023.

[24] I. Kapelyukh, V. Vosylius, and E. Johns, "Dall-e-bot: Introducing web-scale diffusion models to robotics," in *IEEE Robotics and Automation Letters (RA-L)*, 2023.

[25] I. Kapelyukh, Y. Ren, I. Alzugaray, and E. Johns, "Dream2Real: Zero-shot 3D object rearrangement with vision-language models," in *NeurIPS Robot Learning Workshop*, 2023.

[26] Y. Song and D. P. Kingma, "How to train your energy-based models," 2021.

[27] C. Finn, P. F. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *ArXiv*, 2016.

[28] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018.

[29] M. Welling and Y. W. Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011.

[30] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013.

[31] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[32] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," 2019.

[33] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-DOF grasping for target-driven object manipulation in clutter," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.