Energy-based Models are Zero-Shot Planners for Compositional Scene Rearrangement

Nikolaos Gkanatsios†, Ayush Jain†, Zhou Xian, Yunchu Zhang, Christopher Atkeson, Katerina Fragkiadaki
Carnegie Mellon University
{ngkanats,ayushj2,xianz1,yunchuz,cga}@andrew.cmu.edu, katef@cs.cmu.edu

***Abstract*—Language is compositional; an instruction can express multiple relation constraints to hold among objects in a scene that a robot is tasked to rearrange. Our focus in this work is an instructable scene-rearranging framework that generalizes to longer instructions and to spatial concept compositions never seen at training time. We propose to represent language-instructed spatial concepts with energy functions over relative object arrangements. A language parser maps instructions to corresponding energy functions and an open-vocabulary visual-language model grounds their arguments to relevant objects in the scene. We generate goal scene configurations by gradient descent on the sum of energy functions, one per language predicate in the instruction. Local vision-based policies then re-locate objects to the inferred goal locations. We test our model on established instruction-guided manipulation benchmarks, as well as benchmarks of compositional instructions we introduce. We show our model can execute highly compositional instructions zero-shot in simulation and in the real world. It outperforms language-to-action reactive policies and Large Language Model planners by a large margin, especially for long instructions that involve compositions of multiple spatial concepts. Simulation and real-world robot execution videos, as well as our code and datasets are publicly available on our website: https://ebmplanner.github.io.**

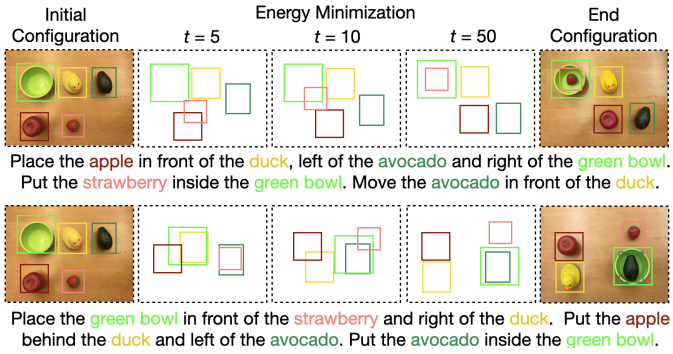

Fig. 1: **Energy-based Models are Zero-Shot Planners for Compositional Scene Rearrangement**. We represent language concepts with energy functions over object locations and sizes. Gradient descent on the sum of energy functions, one per predicate in the instruction, iteratively updates the object spatial coordinates and generates a goal scene configuration that satisfies the instruction, if one exists.

I. Introduction

We consider the scene arrangement task shown in Figure 1. Given a visual scene and an instruction regarding object spatial relations, the robot is tasked to rearrange the objects to their instructed configuration. Our focus is on strong generalization to longer instructions with novel predicate compositions, as well as to scene arrangements that involve novel objects and backgrounds.

We propose generating goal scene configurations corresponding to language instructions by minimizing a composition of energy functions over object spatial locations, where each energy function corresponds to a language concept (predicate) in the instruction. We represent each language concept as an n-ary energy function over relative object poses and other static attributes, such as object size. We train these predicate energy functions to optimize object poses starting from randomly sampled object arrangements through Langevin dynamics minimization [8], using a handful of examples of visual scenes paired with single predicate captions. Energy functions can be binary for two-object concepts such as *left of* and *in front of*, or multi-ary for concepts that describe arrangements for sets of objects, such as *line* or *circle*. We show that gradient descent on the sum of predicate energy functions, each one involving different subsets of objects, generates a configuration that jointly satisfies all predicates, if this configuration exists, as shown in Figure 1.

We propose a robot learning framework that harnesses minimization of compositions of energy functions to generate instruction-compatible object configurations for robot scene rearrangement. A neural semantic parser is trained to map the input instruction to a set of predicates and corresponding energy functions, and an open-vocabulary visual-language grounding model [21, 17] grounds their arguments to objects in the scene, as shown in Figure 2. Gradient descent on the sum of energies with respect to the objects' spatial coordinates computes the final object locations that best satisfy the set of spatial constraints expressed in the instruction. Given the predicted object goal locations, we use vision-based pick-and-place policies that condition on the visual patch around the predicted pick and place locations to rearrange the objects [66]. We call our framework Scene Rearrangement via Energy Minimization (SREM).

We test SREM in scene rearrangement of tabletop environments on simulation benchmarks of previous works [52], as well as on new benchmarks we contribute that involve

†Equal contribution

compositional instructions. We curate multiple train and test splits to test out-of-distribution generalization with respect to (i) longer instructions with more predicates, (ii) novel objects and (iii) novel background colors. We show SREM generalizes zero-shot to complex predicate compositions, such as *"put all red blocks in a circle in the plate"* **while trained from single predicate examples**, such as *"an apple inside the plate"* and *"a circle of blocks"*. We show SREM generalizes to real-world scene rearrangement without any fine-tuning, thanks to the object abstractions it operates on. We compare our model against state-of-the-art language-to-action policies [52] as well as Large Language Model planners [15] and show it dramatically outperforms both, especially for long complicated instructions. We ablate each component of our model and evaluate contributions of perception, semantic parsing, goal generation and low-level policy modules to performance.

Simulation and real-world robot execution videos, as well as our code are publicly available on our website: https://ebmplanner.github.io.

II. Method

The architecture of SREM is shown in Figure 2. The model takes as input an RGB-D image of the scene and a language instruction. A semantic parser maps the instruction to a set of spatial predicate energy functions and corresponding referential expressions for their object arguments. An open-vocabulary visual detector grounds the arguments of each energy function to actual objects in the scene. The goal object locations are predicted via gradient descent on the sum of energy functions. Lastly, short-term vision-based pick-and-place policies move the objects to their inferred goal locations. Below, we describe each component in detail.

A library of energy-based models for spatial concepts In our work, a spatial predicate is represented by an energy-based model (EBM) that takes as input x the set of objects that participate in the spatial predicate and maps them to a scalar energy value $E_\theta(x)$. An EBM defines a distribution over configurations x that satisfy its concept through the Boltzmann distribution $p_\theta(x) \propto e^{-E_\theta(x)}$. Low-energy configurations imply satisfaction of the language concept and have high probability. An example of the spatial concept can be generated by optimizing for a low-energy configuration through gradient descent on (part of) the input x. We represent each object entity by its 2D overhead centroid coordinates and box size. During gradient descent, we only update the center coordinates and leave box sizes fixed. We consider both binary spatial concepts (*in*, *left of*, *right of*, *in front of*, *behind*) as well as multi-ary spatial concepts (*circle*, *line*).

Using an EBM, we can sample configurations from p_θ, by starting from an initial configuration x^0 and refining it using Langevin Dynamics [59]:

$$x^{k+1} = x^k - \lambda \nabla_x E_\theta(x^k) + \epsilon^k z^k, \tag{1}$$

where z^k is random noise, λ is an update rate hyperparameter and ϵ^k is a time-dependent hyperparameter that monotonically decreases as k increases. The role of z^k and decreasing ϵ^k is to induce noise in optimization and promote exploration, similar to Simulated Annealing [24]. After K iterations, we obtain $x^- = x^K$. During training, we iterate over Equation 1 $K = 30$ times, using $\lambda = 1$ and $\epsilon_k = 5{\cdot}10^{-3}$. During inference, we find that iterating for more, e.g. $K = 50$ leads to a better solution. In this case we also linearly decay ϵ_k to 0 for $k > 30$.

We learn the parameters θ of our EBM using a contrastive divergence loss that penalizes energies of examples sampled by the model being lower than energies of ground-truth configuration:

$$\mathcal{L} = \mathbb{E}_{x^+ \sim p_D} E_\theta(x^+) - \mathbb{E}_{x^- \sim p_\theta} E_\theta(x^-), \tag{2}$$

where x^+ a sample from the data distribution p_D and x^- a sample drawn from the learned distribution p_θ. We additionally use the KL-loss and the L2 regularization proposed in [8] for stable training. At test time, compositions of concepts can be created by simply summing energies of individual constituent concept, as shown in Figures 1 and 2.

Semantic parsing of instructions into spatial concepts and their arguments. Our parser maps language instructions to instantiations of energy-based models and their arguments. It is a Sequence-to-Tree model [5] with a copying mechanism [12] which allows it to handle a larger vocabulary than the one seen during training. The input to the model is a natural language instruction and the output is a tree. Each tree node is an operation. The three operations supported are i) `BinaryEBM` which calls a BinaryEBM from our library, ii) `MultiAryEBM` and iii) `VLMGround` which calls the visual-language grounding module. Each node has a pointer to the arguments of the operation, language concepts for EBM calls, e.g., *behind*, and noun phrases for grounding model calls, e.g., *"the green cube"*. Nodes in the parsing tree may also have children nodes, which imply nested execution of the corresponding operations. For example, the instruction *"a circle of cubes inside the plate"* is mapped to a sum of energy functions where each object of the multi-ary concept *circle* participates in the constraining binary concept *in*:

$$\begin{aligned} E^{total} = \; & \texttt{MultiAryEBM}(\textit{circle}, \texttt{VLMGround}(\text{“}\textit{cubes}\text{”})) \\ & + \textstyle\sum_i \texttt{BinaryEBM}(\textit{in}, x_i, \texttt{VLMGround}(\text{“}\textit{plate}\text{”})), \\ & x_i \in \texttt{VLMGround}(\text{“}\textit{cubes}\text{”}). \end{aligned} \tag{3}$$

We ground noun phrases predicted by our parser with an off-the-shelf language grounding model [17], which operates as an open-vocabulary detector. The input is the noun phrase, e.g., *"the blue cube"* and the image, while the output is the boxes of all object instances that match the noun phrase. The open-vocabulary detector has been pre-trained for object detection and referential grounding on MS COCO [29], Flickr30k [44] and Visual Genome [25]. We finetune the publicly available code of [17] on our training data of all tasks jointly.

Short-term vision-based manipulation skills We use short-term manipulation policies built upon Transporter Networks [66] to move the obejcts to their predicted locations. Transporter Networks take as input one or more RGB-D images, reproject them to the overhead birds-eye-view, and

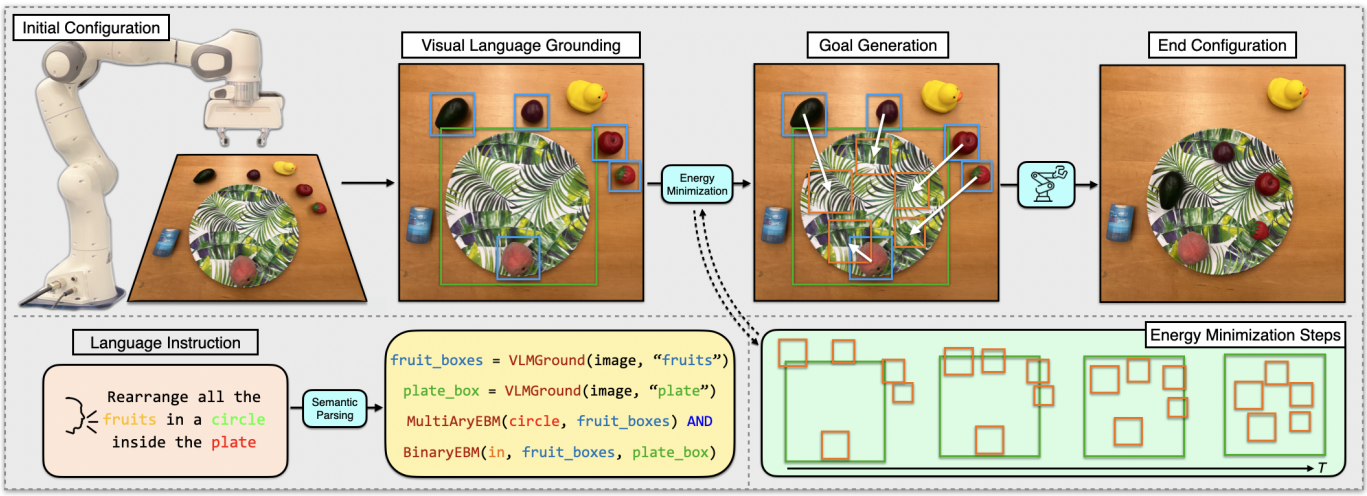

Fig. 2: **Scene rearrangement through energy minimization.** Given an image and a language instruction, a semantic parser maps the language into a set of energy functions (`BinaryEBM`, `MultiAryEBM`), one for each spatial predicate in the instruction, and calls to an open-vocabulary visual language grounder (`VLMGround`) to localize the object arguments of each energy function mentioned in the instruction, here "fruits" and "plate". Gradient descent on the sum of energy functions with respect to object spatial coordinates generates the goal scene configuration. Vision-based neural policies condition on the predicted pick and place visual image crops and predict accurate pick and place locations to manipulate the objects.

predict two robot gripper poses: i) a pick pose and ii) a pick-conditioned placement pose. These networks can model any behaviour that can be effectively represented as two consecutive poses for the robot gripper, such as pushing, sweeping, rearranging ropes, folding, and so on – for more details please refer to [66].

Termination of execution: SREM generates a goal scene by optimizing the relative poses of the objects mentioned in the instruction. We estimate how many objects should be moved by comparing the detected bounding box (by the language grounding model) and the optimized bounding box (by the EBM). For non-compositional tasks that involve binary concepts, we inject the prior that one object is fixed. Then we take as many actions as the number of objects the EBM moved.

Closed-loop execution: SREM first generates a goal scene from the input instruction and then executes it. After execution, we re-detect all relevant objects using our VLM-grounder module to check if they are close to their predicted goal locations. If the re-detected object's bounding box and initially predicted goal bounding box intersect over a certain IoU threshold, we consider the goal to be successfully executed. If we fail to reach the goal, we call again our vision based policies using the current scene configuration. Comparing the post-execution object configuration with the initially imagined goal scene allows to track progress and estimate goal completion as we show in the experimental section and in Section V-G and Table IX of the Appendix.

III. Experiments

We test SREM in its ability to follow language instructions for rearrangement of tabletop scenes in simulation and in the real world. We compare our model against LLM planners [15] and end-to-end language-to-action policies [52].

Method	comp-one-step		comp-group	
	10	100	10	100
Initial (no movement)	0.0	0.0	31.7	31.7
CLIPort (zero-shot)	9.0	12.0	37.4	37.5
CLIPort	13.0	15.0	38.2	38.5
LLMplanner	51.2	53.2	38.6	39.0
SREM (zero-shot)	90.0	91.0	77.2	77.4
SREM (zero-shot + closed-loop)	**91.6**	**92.0**	**80.8**	**81.6**

TABLE I: **Evaluation (TP) of SREM, CLIPort and LLMplanner on compositional tasks**. SREM is trained only on atomic relations and tested zero-shot on tasks with compositions of spatial relations which involve moving one (**comp-one-step**) or multiple (**comp-group**) objects to satisfy all constraints specified by the language. Some language constraints are satisfied already in the initial configuration and the Initial model captures that.

Benchmarks: Existing language-conditioned manipulation benchmarks are usually dominated by a single spatial concept like "inside" [52]. To better illustrate the compositionality of spatial concepts, we introduce the following set of benchmarks, implemented with PyBullet:

- **comp-one-step**, containing compositional instructions with referential expressions in cluttered scenes with distractors that require one object to be re-located to a particular location, e.g. *"put the red bowl to the right of the yellow cube, to the left of the red cylinder, and above blue cylinder"*.

- **comp-group**, containing compositional instructions with referential expressions in cluttered scenes with distractors that require multiple objects to be re-located, e.g., *"put the grey bowl above the brown cylinder, put the yellow cube to the right of the blue ring, and put the blue ring below the grey bowl"*.
- **shapes**, containing instructions for making multi-entity shapes (circles and lines), e.g. *"rearrange all red cubes in a circle"*.

We refer to our Appendix (Section-V-D) for evaluation on the benchmark of [52] as well as ablations.

Evaluation Metrics: (i) **Task Progress (TP)** [66] is the percentage of the referred objects placed in their goal location, e.g. 4/5 = 80.0% for rearranging 4 out of 5 objects specified in the instruction. (ii) **Task Completion (TC)** rewards the model only if the full rearrangement is complete. For the introduced benchmarks we have oracle reward functions that evaluate whether the task constraints are satisfied.

A. Spatial reasoning for scene rearrangement with oracle perception and control

In this section, we compare spatial reasoning for predicting compositional scene subgoals in a language space versus in an abstract visually grounded space. In this section, to isolate this reasoning ability from nuisance factors of visually localizing the objects and picking them up effectively, we consider **oracle object detection, referential grounding and low-level pick-and-place policies.** Specifically, we carry out inferred language subgoals from LLMplanner using oracle controllers that relocate an object in the scene such that it satisfies the predicted subgoals. Note that SREM relies on pick-and-place policies that are not language-conditioned, while LLMplanner relies on language-conditioned policies for object re-location. Thus, the oracle control assumption is less realistic in the latter case. We forego this difference for the sake of comparison.

We show quantitative results of SREM and LLMplanner on the **comp-one-step** and **comp-group** benchmarks in Table II. Our model outperforms LLMplanner and the performance gap is larger in more complex instructions. To elucidate why an abstract visual space may be preferable for planning, we visualize steps of energy minimization for different instructions in Figure 1 and steps of the execution of the LLM prompted by us to the best of our capability in Figure 5. We can see that SREM trained on single-predicate scenes shows remarkable composability in case of multiple predicates. Language planning on the other hand suffers from the ambiguity of translating geometric concepts to language and vice versa: step-by-step execution of language subgoals does not suffice for the composition of the two subgoals to emerge (Figure 5).

B. Spatial scene rearrangement

To evaluate generalization on longer instructions at test time, we show quantitative results in Table I for the benchmarks of **comp-one-step** and **comp-group**. We compare our model with CLIPort trained on atomic spatial relations and zero-shot evaluated on compositional benchmarks. We further fine-tune CLIPort on demos from the compositional benchmarks. SREM is not trained on these benchmarks, because the energy functions are already composable, meaning that we can jointly optimize over an arbitrary number of constraints by simply summing the different energy terms. Under all different settings, we significantly outperform all variants of CLIPort and LLMplanner. We also observe that closed-loop execution boosts our performance further.

Method	**comp-one-step**		**comp-group**	
	TP	TC	TP	TC
LLMplanner w/ oracle	82.0	59.0	75.3	29.0
SREM w/ oracle	**90.8**	**76.0**	**88.7**	**62.0**

TABLE II: **Evaluation of SREM and LLMplanner with oracle perception and oracle low-level execution policies** on compositional spatial arrangement tasks. We report Task Progress (TP) and Task Completion (TC).

Method	**comp-one-step**	**comp-group**	**circles**	**lines**
CLIPort	13.1	22.9	34.0	46.0
LLMplanner	39.5	25.9	-	-
SREM	**85.6**	**75.8**	**94.0**	**90.0**

TABLE III: **Real-world evaluation (TP) of SREM**

Real World: We test our model on a 7-DoF Franka Emika robot, equipped with a parallel jaw gripper and a top-down Azure Kinect RGB-D camera. We do not perform any real-world finetuning. Our test set contains 10 language-guided tabletop manipulation tasks per setting (Comp-one-step, Comp-group, Circles, Lines). We show quantitative results in Table III. SREM generalizes to the real world without any real-world training or adaptation thanks to the open-vocabulary detector trained on real-world images, as well as the object abstractions in the predicate EBMs and low-level policy modules. We encourage readers to refer to our supplementary video and our website for more detailed results.

IV. Conclusion

We introduce SREM, a modular robot learning framework for instruction-guided scene rearrangement that maps instructions to object scene configurations via compositional energy minimization over object spatial coordinates. We test our model in diverse tabletop manipulation tasks in simulation and in the real world. Our model outperforms state-of-the-art end-to-end language-to-action policies, and LLM-based instruction following methods both in in- and out-of-distribution settings. We contribute a new scene rearrangement benchmark that contains more compositional language instructions than previous works, which we make publicly available to the community. Our work shows that a handful of visually-grounded examples suffice to learn energy-based spatial language concepts that can be composed to infer novel instructed scene arrangements, in long and complex compositional instructions.

References

[1] Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Grounding Language to Autonomously-Acquired Skills via Goal Generation. In *ICLR 2021 - Ninth International Conference on Learning Representation*, Vienna / Virtual, Austria, May 2021. URL https://hal.inria.fr/hal-03121146.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[3] Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Ford Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity-driven exploration. *CoRR*, abs/2002.09253, 2020. URL https://arxiv.org/abs/2002.09253.

[4] Cédric Colas, Tristan Karch, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Vygotskian autotelic artificial intelligence: Language and culture internalization for human-like ai, 2022. URL https://arxiv.org/abs/2206.01134.

[5] Li Dong and Mirella Lapata. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*, 2016.

[6] Yilun Du, Toru Lin, and Igor Mordatch. Model based planning with energy based models. *CoRR*, abs/1909.06878, 2019. URL http://arxiv.org/abs/1909.06878.

[7] Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6637–6647. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/49856ed476ad01fcff881d57e161d73f-Paper.pdf.

[8] Yilun Du, Shuang Li, Joshua B. Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *CoRR*, abs/2012.01316, 2020. URL https://arxiv.org/abs/2012.01316.

[9] Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics (TOG)*, 31(6):1–11, 2012.

[10] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Stripstream: Integrating symbolic planners and blackbox samplers. *CoRR*, abs/1802.08705, 2018. URL http://arxiv.org/abs/1802.08705.

[11] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *CoRR*, abs/1912.03263, 2019. URL http://arxiv.org/abs/1912.03263.

[12] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*, 2016.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[14] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022.

[15] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models, 2022. URL https://arxiv.org/abs/2207.05608.

[16] Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*, 2021.

[17] Ayush Jain, Nikolaos Gkanatsios, Ishita Mediratta, and Katerina Fragkiadaki. Bottom up top down detection transformers for language grounding in images and point clouds. In *European Conference on Computer Vision*, pages 417–433. Springer, 2022.

[18] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

[19] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. *CoRR*, abs/1804.01622, 2018. URL http://arxiv.org/abs/1804.01622.

[20] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pages 1470–1477, 2011. doi: 10.1109/ICRA.2011.5980391.

[21] Aishwarya Kamath, Mannat Singh, Yann LeCun, Ishan Misra, Gabriel Synnaeve, and Nicolas Carion. MDETR - modulated detection for end-to-end multi-modal understanding. *CoRR*, abs/2104.12763, 2021. URL https://arxiv.org/abs/2104.12763.

[22] Ivan Kapelyukh, Vitalis Vosylius, and Edward Johns.

Dall-e-bot: Introducing web-scale diffusion models to robotics. *ArXiv*, abs/2210.02438, 2022.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[24] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34: 975–986, 1984.

[25] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *International Journal of Computer Vision*, 123, 2016.

[26] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1205–1211. IEEE, 2019.

[27] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control, 2022. URL https://arxiv.org/abs/2209.07753.

[28] Bill Yuchen Lin, Chengsong Huang, Qian Liu, Wenda Gu, Sam Sommerer, and Xiang Ren. On grounded planning for embodied tasks with language models, 2022. URL https://arxiv.org/abs/2209.00465.

[29] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[30] Nan Liu, Shuang Li, Yilun Du, Joshua B. Tenenbaum, and Antonio Torralba. Learning to compose visual relations. *CoRR*, abs/2111.09297, 2021. URL https://arxiv.org/abs/2111.09297.

[31] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586, 2021. URL https://arxiv.org/abs/2107.13586.

[32] Weiyu Liu, Chris Paxton, Tucker Hermans, and Dieter Fox. Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects. *arXiv preprint arXiv:2110.10189*, 2021.

[33] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[34] Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

[35] Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. *CoRR*, abs/1811.00090, 2018. URL http://arxiv.org/abs/1811.00090.

[36] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention, 2015. URL https://arxiv.org/abs/1511.02793.

[37] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *Robotics Research: The 19th International Symposium ISRR*, pages 132–157. Springer, 2022.

[38] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

[39] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM transactions on graphics (TOG)*, 30(4):1–10, 2011.

[40] Toki Migimatsu and Jeannette Bohg. Object-centric task and motion planning in dynamic environments. *CoRR*, abs/1911.04679, 2019. URL http://arxiv.org/abs/1911.04679.

[41] Igor Mordatch. Concept learning with energy-based models. *CoRR*, abs/1811.02486, 2018. URL http://arxiv.org/abs/1811.02486.

[42] Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *CoRR*, abs/1807.04742, 2018. URL http://arxiv.org/abs/1807.04742.

[43] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

[44] Bryan A. Plummer, Liwei Wang, Christopher M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models. In *Proc. ICCV*, 2015.

[45] Vitchyr H. Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *CoRR*, abs/1903.03698, 2019. URL http://arxiv.org/abs/1903.03698.

[46] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Scalable differentiable physics for learning and control. *arXiv preprint arXiv:2007.02168*, 2020.

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. URL https://arxiv.org/abs/2103.00020.

[48] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott

Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021. URL https://arxiv.org/abs/2102.12092.

[49] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *ArXiv*, abs/2204.06125, 2022.

[50] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. URL https://arxiv.org/abs/2205.11487.

[51] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575. IEEE, 2021.

[52] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[53] Maximilian Sieb, Zhou Xian, Audrey Huang, Oliver Kroemer, and Katerina Fragkiadaki. Graph-structured visual imitation. In *Conference on Robot Learning*, pages 979–989. PMLR, 2020.

[54] Elias Stengel-Eskin, Andrew Hundt, Zhuohong He, Aditya Murali, Nakul Gopalan, Matthew Gombolay, and Gregory Hager. Guiding multi-step rearrangement tasks with natural language instructions. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1486–1501. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/stengel-eskin22a.html.

[55] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 1930–1936. AAAI Press, 2015. ISBN 9781577357384.

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[57] Yushi Wang, Jonathan Berant, and Percy Liang. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1129. URL https://aclanthology.org/P15-1129.

[58] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022. URL https://arxiv.org/abs/2201.11903.

[59] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[60] Hongtao Wu, Jikai Ye, Xin Meng, Chris Paxton, and Gregory Chirikjian. Transporters with visual foresight for solving unseen rearrangement tasks, 2022. URL https://arxiv.org/abs/2202.10765.

[61] Zhou Xian, Shamit Lal, Hsiao-Yu Tung, Emmanouil Antonios Platanios, and Katerina Fragkiadaki. Hyperdynamics: Meta-learning object and agent dynamics with hypernetworks. *arXiv preprint arXiv:2103.09439*, 2021.

[62] Zhou Xian, Bo Zhu, Zhenjia Xu, Hsiao-Yu Tung, Antonio Torralba, Katerina Fragkiadaki, and Chuang Gan. Fluidlab: A differentiable environment for benchmarking complex fluid manipulation. In *International Conference on Learning Representations*, 2023.

[63] Danfei Xu, Roberto Martín-Martín, De-An Huang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Regression planning networks. *CoRR*, abs/1909.13072, 2019. URL http://arxiv.org/abs/1909.13072.

[64] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation, 2022. URL https://arxiv.org/abs/2206.10789.

[65] Lap Fai Yu, Sai Kit Yeung, Chi Keung Tang, Demetri Terzopoulos, Tony F Chan, and Stanley J Osher. Make it home: automatic optimization of furniture arrangement. *ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH 2011, v. 30,(4), July 2011, article no. 86*, 30(4), 2011.

[66] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv:2010.14406*, 2020.

[67] Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. *CoRR*, abs/2012.07277, 2020. URL https://arxiv.org/abs/2012.07277.

V. Appendix

In Section V-A we discuss the related works; in Section V-B we give implementation details for the components of our method; in Section V-C we present in more detail the evaluation metrics for the newly-introduced tasks; in Section V-D we discuss the various baselines we compare our model with; Section V-E analyzes the generalization capabilities of our model compared with the baselines. In Section V-F we discuss ablations of our models to show effectiveness of our proposed model. In Section V-G we show more results in simulation environment and show effectiveness of closed-loop execution for predicting the success or failure of execution and present a detailed error analysis; We discuss the limitations of our model in V-H; we show an example prompt used for our LLMPlanner baseline in Section V-I; we visualize the learned energy landscapes in Section V-J.

A. Related Work

Following instructions for rearranging scenes: Language is a natural means of communicating goals and can easily describe compositions of actions and arrangements [1, 3, 4], providing more versatile goal descriptions compared to supplying one or more goal images. The latter requires the task to be executed beforehand, which defeats the purpose of instruction [42, 45, 51, 60]. We group methods in the literature in the following broad categories:

- *End-to-end language to action policies* [34, 52, 32, 54] map instructions to actions or to object locations directly. We have found that these reactive policies, despite impressively effective within the training distribution, typically do not generalize to longer instructions, new object classes and attributes or novel backgrounds [32, 52].
- *Symbolic planners* such as PDDL (Planning Domain Definition Language) planners [40, 20, 55, 35] use predefined symbolic rules and known dynamics models, and infer discrete task plans given an instruction with lookahead logic search [20, 10, 40, 20, 55, 35]. Symbolic planners assume that each state of the world, scene goal and intermediate subgoal can be sufficiently represented in a logical form, using language predicates that describe object spatial relations. These methods predominantly rely on manually-specified symbolic transition rules, planning domains and grounding, which limits their applicability.
- *Large language models (LLMs)* map instructions to language subgoals [67, 63, 14, 15] or program policies [27] with appropriate plan-like prompts. The predicted subgoals interface with low-level short-term policies or skill controllers. LLMs trained from Internet-scale text have shown impressive zero-shot reasoning capabilities for a variety of downstream language tasks [2] when prompted appropriately, without any weight fine-tuning [58, 31]. The scene description is usually provided in a symbolic form as a list of objects present, predicted by open-vocabulary detectors [21]. Recent works of [27, 28] have also fed as input overhead pixel coordinates of objects to inform the LLM's predictions. The prompts for these methods need to be engineered per family of tasks. It is yet to be shown how the composition of spatial concept functions can emerge in this way.

Language-conditioned scene generation: A large body of work has explored scene generation conditioned on text descriptions [19, 48, 50, 64, 36]. The work of [22] leverages web-scale pre-trained models [13, 47, 49] to generate segmentation masks for each object in the generated goal image. Given an input image, their method generates a text prompt using a captioning model and feeds it to a generative model that outputs a goal image, which is then further parsed into segmentation masks. However, the prompt is limited to contain only names of objects and there is no explicit language-guided spatial reasoning. In this work, we seek to make scene generation useful as goal imagination for robotic spatial reasoning and instruction following. Instead of generating pixel-accurate images, we generate object configurations by abstracting the appearance of object entities. We show this abstraction suffices for a great number of diverse scene rearrangement tasks.

Energy-based models: Our work builds upon existing work on energy-based models (EBMs) [11, 41, 30, 7, 8, 6]. Most similar to our work is that of [41], which generates and detects spatial concepts with EBMs on images with dots, and [7, 30], which demonstrates composability of image-centric EBMs for generating face images and images from CLEVR dataset [18]. In this work, we demonstrate zero-shot composability of EBMs over object poses instead of images, and showcase their applicability on spatial reasoning and instruction following for robotic scene rearrangement.

Constraint-Guided Layout Optimization: Automatic optimization for object rearrangement has been studied outside the field of robotics. [65] and [39] use few user-annotated examples of scenes to adapt the hyperparameters of task-specific cost functions, which are then minimized using standard optimization algorithms (hill climbing and/or simulated annealing). To learn those hyperparameters from data, these approaches fit statistical models, e.g. Mixtures of Gaussian, to the given samples. [9] further employ such optimization constraints into an interactive environment, where the user can provide an initial layout and the algorithm suggests improvements. All these approaches require expert knowledge to manually design rules and cost function, namely [65] identifies seven and [9] eleven expert-suggested criteria for successful rearrangement. Since they are hand-crafted, these methods do not generalize beyond the domain of furniture arrangement. In contrast, energy optimization is purely data-driven and domain-agnostic: a neural network scores layouts, assigning high energy to those that do not satisfy the (implicit) constraints and low energy to those who do, essentially modeling the underlying distribution of valid layouts.

B. Implementation Details

Energy-based Models: We implement two sets of EBMs, a BinaryEBM and a MultiAryEBM for binary (e.g., *left of*) and multi-ary (e.g., *circle*) language concepts, respectively. The

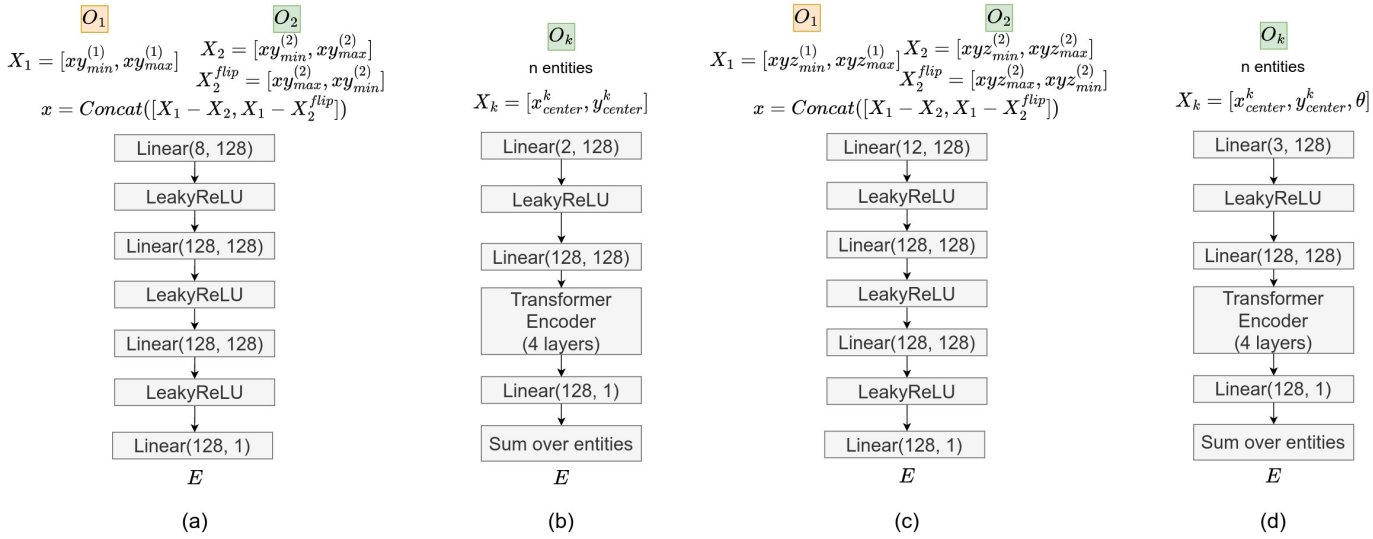

Fig. 3: **(a)**: Architecture of the EBM used for binary concepts such as "right of". The inputs are two boxes O_1 and O_2 and the output is the energy of their relative placement. **(b)**: Architecture of the EBM used for multi-ary concepts such as "circle". The input is a set of n entities $O_k, k = 1, \ldots, n$. The output is the energy of this set of entities wrt the concept. **(c)**: Architecture of the EBM used for 3D binary concepts such as "on". Each object is now represented by a 3D bounding box. **(d)**: Architecture of the EBM used for concepts that involve pose optimization (rotation). Each object is represented with its center and rotation wrt the global coordinate frame.

Operation	**Signature**	**Semantics**
Filter	(ObjectSet, ObjectConcept) $\rightarrow$ ObjectSet	Filter out set of objects based on some *Object Concept* like object name (eg. cube) or property (color, material)
BinaryEBM	(Object A, Object B, Relation) $\rightarrow$ (Pick locations, Place locations)	Executes *BinaryEBMs* for rearranging Object A and Object B to satisfy the given binary *relation* (like left of/right of/inside etc.)
MultiAryEBM	(ObjectSet, Shape Type, Property) $\rightarrow$ (Pick locations, Place locations)	Executes *MultiAryEBMs* for the given *Shape Type* (circle, line, etc.) with specified *Properties* (like size, position etc.) on a set of given objects and generates pick and place locations to complete the shape.

TABLE IV: All operations in the domain-specific language for SREM

BinaryEBM expects two object arguments, each represented by its bounding box. We convert the object bounding box to (top-left corner, bottom-right corner) representation. Then we compute the difference between all corners of the two object arguments and concatenate and feed to a multi-layer perceptron (MLP) that outputs a scalar energy value. Note that the energy function only depends on the relative arrangement of the two objects, not their absolute locations. The MultiAryEBM is used for order-invariant concepts of multiple entities, such as shapes. The input is a set of objects, each represented as a point (box center). We subtract the centroid of the configuration from each point and then featurize each object using an MLP. We feed this set of object features to a sequence of four attention layers [56] for contextualization. The refined features are averaged into an 1D vector which is then mapped to a scalar energy using an MLP. We train a separate EBM for each language concept in our vocabulary using corresponding annotated scenes in given demonstrations. Note that annotated scenes suffice to train the energy functions, kinesthetic demonstrations are not necessary, and in practice each EBM can be trained within a few minutes.

The architectures of the BinaryEBM and MultiAryEBM are shown in Figure 3a and b respectively. We train a separate network for each concept in our library using the same demos that are used to train the other modules. We augment and repeat the samples multiple times to create an artificially larger dataset during training. We use Adam [23] optimizer, learning rate $1e-4$, batch size 128.

Buffer: During training, we fill a buffer of previously gener-

ated examples, following [8]. The buffer is updated after each training iteration to store at most 100000 generated examples. When the buffer is full, we randomly replace older examples with incoming new ones. We initialize x_0 for Equation 1 by sampling from the buffer 70% of the times or loading from the data loader 30% of the times.

Regularization losses: We use the KL-loss from [8], $\mathcal{L}_{\mathcal{KL}} = \mathbb{E}_{x^- \sim p_\theta} \bar{E}_\theta(x^-)$, where the bar on top of $\bar{E}$ indicates the stop-gradient operation (we only backpropagate to E through x^-). We additionally use the L2 energy regularization loss $\mathbb{E}_{x^+ \sim p_D} E_\theta^2(x^+) + \mathbb{E}_{x^- \sim p_\theta} E_\theta^2(x^-)$. We refer the reader to [8] for more explanation on these loss terms.

Extension to tasks with 3D information or pose: An important design choice is what parameters of the input we should be able to edit. We inject the prior knowledge that on our manipulation domain the objects move without deformations, so we fix their sizes and update only their positions. Our EBMs operate on boxes so that they can abstract relative placement without any need for object class or shape information. However, EBMs can be easily extended to optimize other types of representations, such as 3D bounding boxes or pose.

We train EBMs that optimize over 3D locations for relative placement. The architecture is shown in Figure 3c. We adapt the BinaryEBM to represent boxes as (xyz_{min}, xyz_{max}) and then compute the relative representations as in the 2D case. We optimize for one 3D relation, "on". For pose-aware EBMs, we adapt the MultiAryEBM to represent objects as $(x_{center}, y_{center}, \theta)$, where θ is the rotation wrt the world frame. We then simply change the first linear layer of the MultiAryEBM to map the new input tuple to a 128-d feature vector. The architecture is shown in Figure 3d. We show qualitative results that compose these EBMs into new concepts on our website.

Domain-Specific Language: We design a Domain-Specific Language (DSL) which extends the DSL of NS-CL [38] (designed for visual question answering in CLEVR [18]) to further predict scene generations, e.g. *"put all brown shoes in the green box"*. Detailed description of our DSL can be found in Table IV.

Semantic Parser The input utterance is encoded using a pre-trained RoBERTa encoder [33], giving a sequence of contextualized word embeddings and a global representation of the full utterance. Then, a decoder is iteratively employed to i) decode an operation, ii) condition on this operation to decode or copy the arguments for this operation, iii) add one (or more) children node(s). We train our semantic parser on the instructions of all training demonstrations of all tasks jointly, as well as on synthesized instructions paired with programs, each with 1-7 predicates, that we generate by sampling from a grammar, similar to previous works [38, 57]. We then train our parser on all instructions using Adam optimizer with learning rate $1e-3$ and batch size 32. The same parser weights are used across all tasks. For compositional tasks, we train the parser on the descriptions from the demos that are used to finetune our baselines. The parser is the only part of our framework that needs to be updated to handle longer instructions.

Visual-language Grounder: We finetune BEAUTY-DETR [17] on the scenes of the training demos in simulation. BEAUTY-DETR is an encoder-decoder Detection Transformer that takes as input an image and a referential language expression and maps word spans to image regions (bounding boxes). The original BEAUTY-DETR implementation uses an additional box stream of object proposals generated by an off-the-shelf object detector. We use the variant without this box stream for simplicity. BEAUTY-DETR has been trained on real-world images. We finetune it using the weights and hyperparameters from the publicly available code of [17].

Short-term Manipulation Skills Our low-level policy network is based on Transporter Networks. Transporter Networks decompose a given task into a sequence of pick-and-place actions. Given an overhead image, the model predicts a pick location and then conditions on it to predict a place location and gripper pose. We modify Transporter Networks to take as input a small image RGB-D patch, instead of a complete image view. Specifically, we consider as input the image patches around the object pick and object goal locations predicted by our visual grounding and energy-based minimization modules respectively. In this way, the low-level policies know roughly what to pick and where to place it, and only locally optimize over the best pick location, as well as the gripper's relative rotation, within an object of interest, or placement location, at a particular part of the scene, respectively. We show in our ablations (Table VIII) that using learning-based pick-and-place policies helps performance, even if the search space is limited thanks to grounding and goal imagination. We train Transporter Networks from scratch on all our pick-and-place demonstration datasets jointly. The original implementation of Transporter Networks supports training with batch size 1 only. We implement a batch-supporting version and find it more stable. We use batch size 8 and follow the original paper in other hyperparameter values.

C. Benchmark Generation

We extend the Ravens [66] benchmark for spatial reasoning in the PyBullet simulator. For each benchmark, we write a template sentence (e.g. "Arrange OBJ1 into a circle") and then randomly select valid objects and colors from a pre-defined list. To test generalization, we include novel colors or novel objects in the evaluation set. Once the sentence is generated, we programatically define valid regions which satisfy the relation and then sample empty locations from it to specify object goal locations. We start by placing all objects randomly in the scene. Then, an oracle hand-designed policy picks and places the objects to the desired locations and returns a demo trajectory which consists of raw RGB-D images and pick-and-place locations. These can be used then to train a behaviour cloning policy similar to CLIPort.

Evaluation Metrics for Rearrangement Tasks To evaluate make-a-circle task, we fit a best-fit circle for the final configuration predicted by SREM. To do this, we consider the centers of the bounding boxes as points. Then, we compute the centroid of those points and the distance of each point

from the centroid. This is an estimate of "radius". We compute the standard deviation of this radius. If this is lower than 0.03, then we assign a perfect reward. The reward linearly decreases when the std increases from 0.03 to 0.06. Beyond that, we give zero reward. We tuned these thresholds empirically by generating and distorting circle configurations.

We follow a similar evaluation strategy for make-a-line. Here we compute the average slope and fit a line to our data. Then we measure the standard deviation of the distance of each point from the line. We found that the same thresholds we use for circles work well for lines as well.

D. Baselines:

We compare SREM to the following baselines:

- CLIPort [52], a model that takes as input an overhead RGB-D image and an instruction and uses pre-trained CLIP language and image encoders to featurize the instruction and RGB image, respectively; then fuses these with depth features to predict pick-and-place actions using the action parametrization of Transporter Networks [66]. The model capitalizes on language-vision associations learnt by the CLIP encoders. We use the publicly available code of [52]. We train one CLIPort model on all tasks of each benchmark, e.g., one model for **spatial-relations**, a different for **comp-group** etc. Note that the original CLIPort implementation assumes access to oracle success/failure information based on which the model can retry the task for a fixed budget of steps or stop the execution if oracle confirms that the task is completed. We evaluate the CLIPort model without this oracle retry but still with oracle information of how many minimum steps it needs to take to complete the task, so we force CLIPort to take exactly that number of actions.
- LLMplanner, inspired by [15], an instruction-following scene-rearrangement model that uses an LLM to predict a sequence of subgoals in language form, e.g. *"pick the red cube and place it to the right of the blue bowl"*. The generated language subgoals are fed as input to language-to-action policies, such as CLIPort. Scene state description is provided as a list of objects in the scene. LLMplanner does not finetune the LLM but instead uses appropriate prompts so that the LLM adapts its behavior in-context and generates similar statements. The prompts include various previous successful interactions between a human user and the model. We design suitable prompts for our introduced benchmarks and use the LLM to decompose a long instruction into simpler ones (see Figure 4 in the Appendix for an example). Then, we feed each generated instruction to a CLIPort model, trained as described earlier. Lastly, for tabletop manipulation tasks in simulation, the LLMPlanner of [15] assumes access to an oracle success/failure detector. The difference in our implementation is that we do not assume any success detector. The execution terminates when all language subgoals have been fed to and handled by CLIPort.

Note that LLMplanner boils down to CLIPort for non-compositional instructions. As such, we compare with LLMplanner only on **comp-one-step** and **comp-group**, both in simulation and real world.

E. Generalization analysis

We conduct controlled studies of our model's generalization across three axes: a) Novel Colors b) Novel background color of the table c) Novel Objects. In each of these settings, we only change one attribute (i.e. object color, background color or object instance) while keeping everything else constant.

- *Novel colors*: We train the models with ["blue", "red", "green", "yellow", "brown", "gray", "cyan"] colors and evaluate them with unseen ["orange", "purple", "pink", "white"].
- *Novel background colors*: All models are trained with black colored tables and evaluated with randomly sampled RGB color for each instruction.
- *Novel objects*: We train the models on ["ring", "cube", "cylinder", "bowl"] and evaluate them with ["triangle", "square", "plus", "diamond", "pentagon", "rectangle", "flower", "star", "circle", "hexagon", "heart"].
- *Real-World Experiments* In our real-world experiments, we also use novel objects or novel object descriptions like "bananas", "strawberry", "small objects" which the model hasn't seen during training in simulation.

We evaluate our model and CLIPort trained on 10 or 100 demos per task on **spatial-relations** (average performance over all tasks) and **composition** (average performance over all tasks from **comp-one-step** and **comp-group**). The results are summarized in Table-VI. We observe that our model maintains high performance across all axes of generalization, independently of the number of training demos.

Our model's generalization capabilities rely on the open-vocabulary detector and the fact that EBMs and transporter-based low-level execution policy operate on abstracted space in a modular fashion. While CLIPort models can also generalize to novel scenarios by leveraging the CLIP model, the action prediction and perception are completely entangled and hence even if CLIP manages to identify the right objects based on the language, it has trouble predicting the correct pick and place locations.

F. Ablations

We show an error analysis of our model in Table-VIII. First, we remove the goal generation from SREM (SREM w/o goal generation) by conditioning the place network on the language input instead of the EBM-generated goal image, while keeping the pick network and object grounders identical. We observe a drop of 35.1% in accuracy, underscoring the importance of goal generation. We then remove our executor policy (SREM w/o learnable policies) and instead randomly select pick/place locations inside the bounding box of the relevant object. This results in a drop of 16%, showing the importance of robust low-level policies. We do not remove the grounder and parser since they are necessary for goal

Human: Put the cyan bowl in front of the cyan ring and behind the blue cube.
Scene: There is red bowl, green ring, yellow cube, red cylinder, blue cylinder and cyan bowl in the scene.
Robot Thought: The goal state is ['cyan bowl in front of cyan ring", "cyan bowl behind blue cube"]
Robot Action: Put the cyan bowl in front of the cyan ring.
Executor: Done.
Robot Action: Put the cyan bowl behind the blue cube.
Executor: Done.

Fig. 4: **Example prompt used for LLMPlanner**.

generation. We then experiment with oracle visual language grounder (SREM w/ oracle language grounding) that perfectly detects the objects mentioned in the sentence, which results in a performance gain of 5.1%. We finally evaluate with perfect grounding, language parsing and low-level execution (SREM w/ everything oracle except goal) to test the error rate of our goal generator. We obtain an 88.3% accuracy, thus concluding that our goal generator fails in 11.7% cases. For a more detailed error analysis, please refer to V-G in the Appendix.

G. Additional Experiments

Simulation: In this section, we compare our model and the baselines in the task of instruction-guided scene rearrangement. We first show results on **spatial-relations** and **shapes** in Table V. We largely outperform CLIPort, especially when less training demos are considered.

We additionally show results on the CLIPort benchmark in Table VII. We largely outperform CLIPort on almost all tested tasks. Margins are significantly larger when i) less demos are used and ii) the robot has to interact with objects of unseen colors or classes. Most of the failure cases for our model are due to the language grounding mistakes - in particular for assemble-kits-seq we find that the grounder gets confused between letters and letter holes.

Closed-Loop Execution: As we show in the main paper, adding feedback roughly adds 3.5% performance boost on comp-group benchmark and 1% boost on comp-one-step benchmark. Retrying cannot always recover from failure, however, it would be still important to know if the execution failed so that we can request help from a human. Towards this, we evaluate the failure detection capabilities of our feedback mechanism in Table-IX. We observe that all metrics are high for comp-group benchmark. For comp-one-step, however, the recall is very low i.e. only 19.2%. This is because in this benchmark, we only need to move one object to complete the task and hence most failures are due to wrong goal generation. Thus, the failure classifier classifies those demos as success, because indeed the model managed to achieve its predicted goal and hence results in low recall. This motivates the use of external failure classifiers in conjunction with internal goal-checking classifiers. In contrast, comp-group benchmark is harder because it requires the model to move many objects and thus results in higher chances of robot failures. These failures can be better detected and fixed by our goal-checking classifier.

Also, note that while previous works like InnerMonologue [14] show large improvements by adding closed-loop feed-

Human: Put the strawberry to the right of the apple and in front of the green bowl.
Scene: There is an apple, a green bowl and a strawberry in the scene.
Robot Thought: The goal state is ["strawberry right of apple", "strawberry in front of green bowl"]
Robot Action: Put the strawberry to the right of the apple.
Executor: Done.
Robot Action: Put the strawberry in front of the green bowl.
Executor: Done.

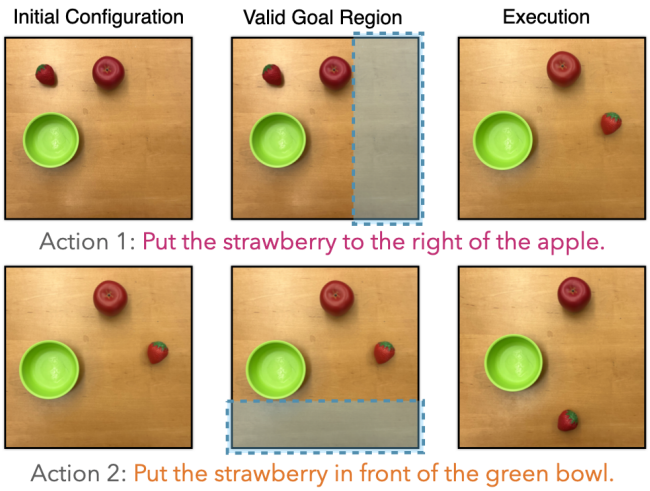

Fig. 5: **Planning in language space with Large Language Models (LLMs).** LLM Planners predict language subgoals that decompose the initial instruction to simpler-to-execute subtasks. Predicted language subgoals are fed to reactive language-to-action policies for execution. In cases where concept intersection is needed, the predicted sequential language subgoal decomposition of instructions can fail. Here, the LLM predicts the first subgoal of putting the strawberry to the right of the apple. The reactive policy can succeed if it places the strawberry anywhere within the shaded region. During execution of the next issued language subgoal of putting the strawberry in front of the bowl, the policy violates the first constraint. Placing the strawberry in the intersection of the two shaded regions may not be achieved by decomposing the two predicates sequentially, as opposed to composing them. Then the burden of handling the compositional instruction is outsourced to the language-to-action policy, which often fails to generalize. Instead, SREM directly addresses compositionality of multiple spatial language predicates.

Method	left-seen-colors		left-unseen-colors		right-seen-colors		right-unseen-colors	
	10 demos	100 demos	10 demos	100 demos	10 demos	100 demos	10 demos	100 demos
CLIPort	13.0	44.0	9.0	33.0	29.0	43.0	28.0	44.0
SREM	**95.0**	**95.0**	**93.0**	**94.0**	**89.0**	**92.0**	**93.0**	**96.0**

Method	behind-seen-colors		behind-unseen-colors		front-seen-colors		front-unseen-colors	
	10	100	10	100	10	100	10	100
CLIPort	24.0	45.0	22.0	51.0	23.0	55.0	13.0	40.0
SREM	**87.0**	**87.0**	**89.0**	**90.0**	**89.0**	**90.0**	**88.0**	**89.0**

Method	circle-seen-colors		circle-unseen-colors		line-seen-colors		line-unseen-colors	
	10 demos	100 demos	10 demos	100 demos	10 demos	100 demos	10 demos	100 demos
CLIPort	34.1	61.5	31.2	55.6	48.6	88.2	48.6	88.5
SREM	**91.3**	**91.5**	**90.2**	**91.2**	**98.1**	**99.0**	**98.4**	**99.4**

TABLE V: **Evaluation (TP) of SREM and CLIPort on spatial-relations and shapes in simulation.**

Novel attribute	Model	spatial-relations		composition	
		10 demos	100 demos	10 demos	100 demos
None	CLIPort	22.0	47.0	25.6	26.8
	SREM	90.0	91.0	83.6	84.2
Color	CLIPort	18.0	39.0	25.1	24.5
	SREM	87.0	85.0	86.5	84.0
Background	CLIPort	10.0	20.0	23.7	23.2
	SREM	79.0	68.0	77.0	72.0
Objects	CLIPort	17.0	19.0	24.5	24.8
	SREM	86.0	86.0	80.9	81.5

TABLE VI: **Generalization experiments of SREM and CLIPort in manipulation tasks in simulation (metric is TP).**

back, the improvements for us are smaller. This is because, by design, our model is more likely to reach its goal - since the EBM generates a visual goal and then the low-level policy predicts a pick-and-place location **within** the predicted goal, it is very likely to satisfy its predicted goal. Indeed, in the comp-one-step, the model satisfies its goal in 93% cases and in 70% cases in the comp-group benchmark. In contrast, InnerMonologue does not have any built-in mechanisms for promoting goal-reaching behaviours and thus the difference in performance with additional goal-satisfying constraints is larger for that method.

There is a lot of potential in designing better goal checkers. As already discussed, we can incorporate external success classifiers like those used by prior literature [14] in conjunction with goal-checking classifiers. Explicit goal generation allows then to check validity of goals directly even before actual robot execution (which makes it safer and less expensive). We leave this for future work. Another promising direction is to add object trackers in the feedback mechanism to keep track of the objects and detect if a failure happened. This is important, if we have multiple objects that are visually similar and hence we would need to keep track of which object corresponds to which goal and retry if it failed to reach it.

Error Analysis: We conduct a detailed error analysis of our model on comp-group benchmark, shown in Table-X. We find that robot failures, i.e. collisions or failure to pick/place objects, result in 6.0% drop in accuracy. Goal generation adds 11.7% to the failure. Visual grounding leads to 5.1% errors while we find language parsing to be nearly perfect.

H. Limitations

Our model presently has the following two limitations: First, it predicts the goal object scene configuration but does not have any knowledge regarding temporal ordering constraints on object manipulation execution implied by physics. For example, our model can predict a stack of multiple objects on top of one another but cannot suggest which object needs to be moved first. One solution to this problem is to heuristically pick the order based on objects that are closer to the floor in the predicted scene configuration. However, more explicit

Method	put-block-in-bowl seen-colors		put-block-in-bowl unseen-colors		packing-google-objects seq-seen-objects		packing-google-objects seq-unseen-objects	
	10 demos	100 demos	10 demos	100 demos	10 demos	100 demos	10 demos	100 demos
CLIPort	31.0	82.1	4.8	17.6	34.8	54.7	27.2	56.4
SREM	**84.3**	**93.8**	**89.0**	**95.3**	**86.8**	**94.8**	**88.0**	**92.9**

Method	packing-google-objects group-seen-objects		packing-google-objects group-unseen-objects		assembling-kits seq-seen-colors		assembling-kits seq-unseen-colors	
	10	100	10	100	10	100	10	100
CLIPort	33.5	61.2	32.2	70.0	38.0	**62.6**	36.8	**51.0**
SREM	**86.1**	**76.8**	**87.2**	**79.6**	**38.4**	42.0	**40.8**	44.0

TABLE VII: **Evaluation (TP) of SREM and CLIPort on CLIPort benchmark in simulation.**

Method	Accuracy
SREM	77.2
SREM w/o goal generation	42.1
SREM w/o learnable policies	61.2
SREM w/ oracle language grounding	82.3
SREM w/ everything oracle except goal	88.3

TABLE VIII: **Ablations of SREM on the benchmark comp-group-seen-colors (metric is TP).**

Benchmark	Precision	Recall	Accuracy
Comp-group	80.5	82.5	85.0
Comp-one-step	71.4	19.2	77.0

TABLE IX: **Performance of failure detection using our generated goal.** We check whether the boxes of the objects in the rearranged scene overlap with the locations the EBM generated. If not, we mark the rearrangement as failed (not satisfying the goal).

encoding of physics priors are important to also identify if the generated configuration is stable or not. A promising direction is to model physics-based constraints as additional energy constraints, and obtain optimization gradients by leveraging either differentiable physics simulators [16, 46, 62] or learned dynamics models [26, 61, 43]. Second, our EBMs are currently parametrized by object locations and sizes, but different tasks need different abstractions. Manipulation of articulated objects, fluids, deformable objects or granular materials, would require finer-grained parametrization in both space and time. Furthermore, even for rigid objects, many tasks would require finer in-space parametrization, e.g., it would be useful to know a set of points in the perimeter of a plate as opposed to solely representing its bounding box for accurately placing things inside it. Considering EBMs over keypoint or object part graphs [53, 37] is a direct avenue for future work.

Error Mode	Error Percentage
Robot failure	6.0
Goal Generation failure	11.7
Grounding failure	5.1
Language Parsing failure	0.0

TABLE X: **Error Analysis of SREM on the benchmark comp-group-seen-colors.**

I. Prompt used for LLMPlanner

We show a prompt used for LLMPlanner for **comp-one-step** in Figure 4.

J. EBM Energy Landscape Visualization

We visualize the energy landscape of our EBMs for different concepts in Figure-6. For binary relations (A, rel, B), we fix B's bounding box in the scene and then move the bounding box of A all over the scene and evaluate the energy of the configuration at each position. For shapes, it is impossible to represent the landscape in 2D or 3D because we need to jointly consider all possible combinations of objects in the scene. Hence, we fix all but one object in a valid circle/line location and move a free box in all possible regions. For compositions of relations, we move only one box and score the sum of energy for all constraints. We expect the energy to be low in regions which satisfy the described concept (relation(s) or shape) and high elsewhere. We observe that the energy landscape is usually smooth with low values in valid regions and high otherwise.

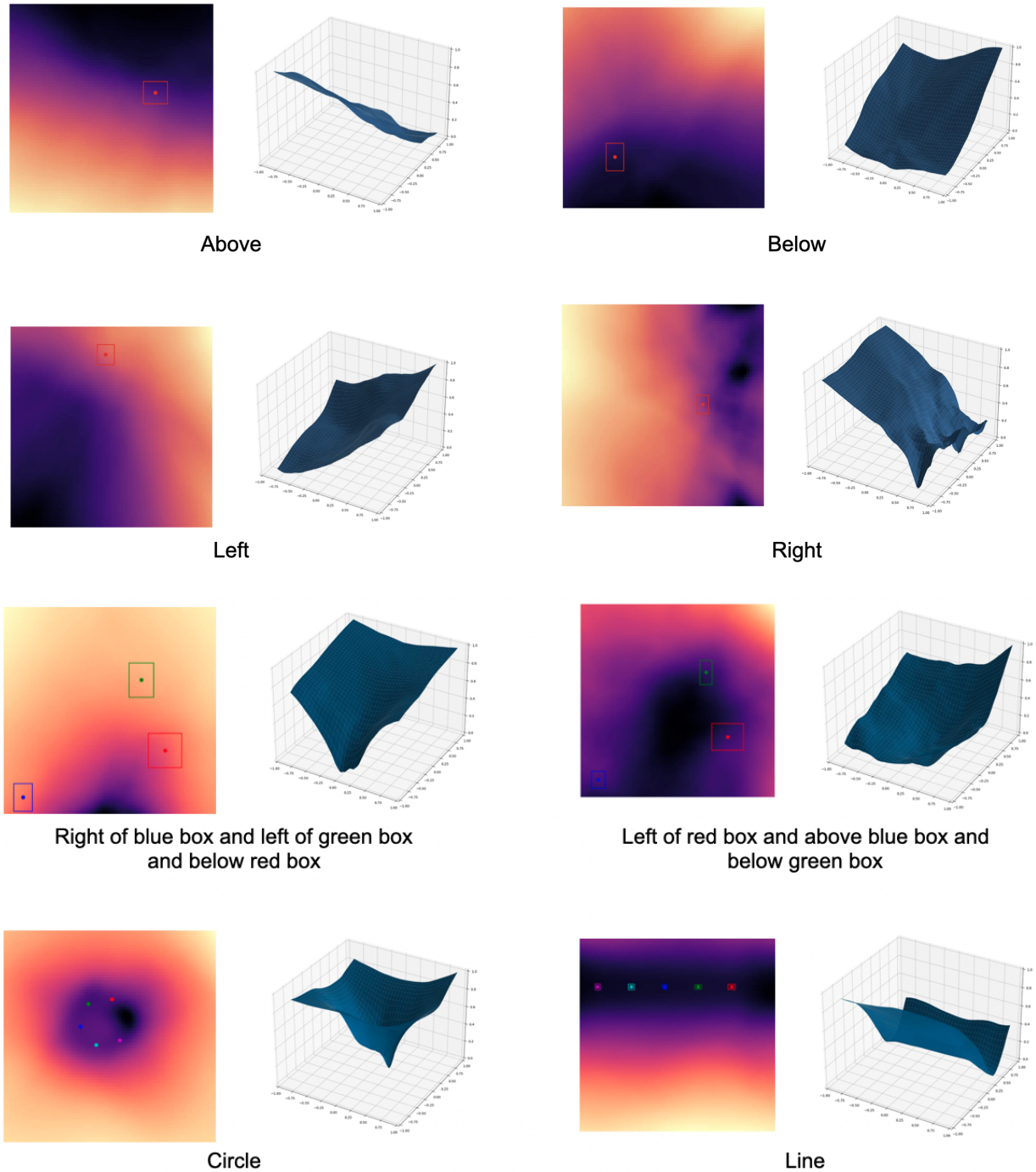

Fig. 6: **EBM Energy Landscape visualization:** The boxes shown here remain fixed and we score the energy by moving another box all along the workspace. Energy decreases from white to orange to purple to black.