

B.Comp. Dissertation

**Placing Objects Onto Shelf: Object Rearrangement Under
Geometric And Commonsense Constraints (Interim Report)**

By

Zhu Tianqi

Department of Computer Science

School of Computing

National University of Singapore

2023/24

Table of Contents

Title	i
1 Introduction	1
1.1 Background	1
1.2 The Problem	2
2 Related Work	4
2.1 Object Rearrangement	4
2.1.1 Inferring Rearrangement Planning Goals From Weak Human Guidance . .	4
2.2 Space Allocation Under Geometric Constraints	7
2.3 Encoding Commonsense and User Preference	8
2.3.1 Sample Commonsense And User Preference From Foundation Model . .	8
2.3.2 Learning Commonsense And User Preference	9
2.4 Object Centric Relational Representation	9
3 Problem Analysis And Proposed Solution	11
3.1 Analysis of the Problem	11
3.2 Plans For Next Semester	13
3.2.1 November and December	13
3.2.2 January and Feburary	13
3.2.3 March	13
3.2.4 April	13
4 Other topics that I explored	14
4.1 Large Language Model for Planning	14
References	16

Chapter 1

Introduction

1.1 Background

To function effectively in the dynamic and unstructured settings of household environments, robots require a nuanced understanding of several key factors. First and foremost, they need to grasp human habits and preferences, as these dictate the practical usage of space and objects within the home. Additionally, an awareness of the physical constraints inherent in these environments is crucial. This includes understanding spatial limitations, the varying nature of surfaces, and the need for safe navigation and manipulation. Lastly, comprehending the spatial relationships among different objects and how they are used in conjunction with each other is essential. This comprehensive understanding enables robots to navigate and perform tasks in a manner that is both efficient and aligned with the habits and expectations of human users.

In this work, we attempt to address the challenge of robots placing unknown objects onto multi-layer shelves already containing items. The robot must first assess whether there's adequate space for the new items. If space is limited, it may need to reorganize existing items to accommodate the additions. This task is particularly pertinent in household environments, where ease of locating objects is crucial for both humans and robots. To achieve this, the robot should arrange items based on user habits and common sense, grouping similar categories together and placing items in their typical positions. For instance, on a kitchen shelf, a new can

of juice might be placed alongside existing Coke cans, while on a bedroom shelf, slippers would likely be placed at the bottom.

While previous studies have focused on individual aspects of this problem, they haven't holistically addressed the dual challenge of space optimization and intuitive arrangement. Our goal is to develop a comprehensive approach that simultaneously satisfies these constraints, enabling more effective and user-friendly object rearrangement by robots.

1.2 The Problem

There are several variants of the object placing onto shelf problem concerning how many more objects to place and how many objects are already on the shelf:

1. Given a set of objects, place it nicely on an empty shelf.
2. Given a shelf with x existing objects, find a place to put one or more additional objects.
3. Given a shelf with x existing objects, rearrange the objects on the shelf to make it look nicer or more conformal with commonsense.

We focus on generating object rearrangement goals for the second task variant. We define geometric constraint as objects not overlapping with each other and the shape and volume of the shelf can hold the objects, and semantic constraint as objects are placed at a user-friendly location on the shelf, i.e. the relative positions among objects locally and global object position to the whole shelf complies with user habit and commonsense.

For a set of objects (o_1, o_2, \dots, o_n) on a shelf, define initial object set $O_{init} = \{o_1, o_2, \dots, o_n\}$, and initial position set $P_{init} = \{p_1, p_2, \dots, p_n\}$, the state of the objects could be represented as the position of each object on the shelf

$$s_{init} = ((o_1, p_1), (o_2, p_2), \dots, (o_n, p_n))$$

We use abstract symbols to represent the concepts first. The exact attribute set to represent the objects, and the exact coordinates systems to define the positions are left for further discussion.

Given a new set of objects $O_{new} = \{o_{n+1}, o_{n+2}, \dots, o_{n+m}\}$, denote the concatenation of tuple as \oplus , the initial configuration becomes:

$$\begin{aligned} s'_{init} &= s_{init} \oplus s_{new} \\ &= ((o_1, p_1), (o_2, p_2), \dots, (o_n, p_n), (o_{n+1}, \text{null}), (o_{n+2}, \text{null}), \dots, (o_{n+m}, \text{null})) \end{aligned} \tag{1.1}$$

we aim to find a goal configuration

$$\begin{aligned} s_{goal} &= ((o_1, p'_1), \dots, (o_n, p'_n), (o_{n+1}, p'_{n+1}), \dots, (o_{n+m}, p'_{n+m})) \\ &= ((o_1, p'_1), \dots, (o_n, p'_n)) \oplus ((o_{n+1}, p'_{n+1}), \dots, (o_{n+m}, p'_{n+m})) \\ &= F_{move}(s_{init}) \oplus s_{add} \end{aligned} \tag{1.2}$$

that satisfies the geometric constraints and the semantic constraints.

Denoting the configuration space S , we define functions to capture the semantics and how humans perceive the shelf as follows. We aim to find a semantic projection function $f_{\theta_1} : S \mapsto A$ that projects the configuration space to the semantic space A , a semantic distance function that denotes the distance between two configurations in the semantic space,

$$D_{\theta_2}(a_1, a_2) = D_{\theta_2}(f_{\theta_1}(s_1), f_{\theta_1}(s_2)), a_1, a_2 \in A, s_1, s_2 \in S$$

, and another compliance score function that scores how well a configuration aligns with commonsense or more specific user habits $g_{\theta_3} : A \mapsto R$.

Denote the subspace of all geometrically valid configurations in the configuration space as S_{valid} . Given the initial configuration, to find an optimal goal configuration lies in S_{valid} , we aim to explicitly or implicitly minimize the semantic distance between the two configurations and maximize the compliance score that defines how well the configurations align with user habit.

Chapter 2

Related Work

2.1 Object Rearrangement

The task that we are trying to solve belongs to the task category of object rearrangement (Batra, Chang, Chernova, Davison, Deng, Koltun, Levine, Malik, Mordatch, Mottaghi, & others, 2020). Object rearrangement is traditionally solved by task and motion planning (TAMP) (Garrett, Chitnis, Holladay, Kim, Silver, Kaelbling, & Lozano-Pérez, 2021). Given an initial configuration and a predefined goal configuration, a rearrangement plan is generated via searching to solve the hybrid constraint satisfaction problem. It relies on predefined symbolic rules and fine-grained dynamics models and suffers from high computation complexity when scaling up. It usually works in 2D space and geometric constraints beyond cluttered situations are not fully tapped. Recent deep learning based methods that learn rearrangement policies from data enable generalization to unknown objects (Qureshi, Mousavian, Paxton, Yip, & Fox, 2021), and scaling to more complex multi-level structures (Kulshrestha & Qureshi, 2023).

2.1.1 Inferring Rearrangement Planning Goals From Weak Human Guidance

While traditional methods require programmers to specify the rearrangement goal configurations explicitly and accurately as 2D or 3D positions or poses, recent advances enable robots to deduce those configurations from weaker human supervision like natural language instruction, symbolic graphs, predicate satisfaction, or even commonsense to rearrange less known objects,

which equips robots with more flexible behaviors to operate in less structured environments like households. Following similar setups, we aim to deduce the rearrangement planning goals purely from observation, pretrained foundation models, and more specified smaller models that generate goal configurations.

Specify Rearrangement Goal With Scene Graph

SG-Bot (Zhai, Cai, Huang, Di, Manhardt, Tombari, Navab, & Busam, 2023) represents the scene as a scene graph, allowing a commonsense mode that automatically generates the graph from foundation models or a user-defined mode that allows users to specify the graph directly. The scene graph is later fed into a Graph-to-3D model (Dhamo, Manhardt, Navab, & Tombari, 2021) that maps the graph to 3D point cloud as a fine-grained rearrangement goal.

Ground Rearrangement Goal With Natural Language

StructFormer (Liu, Paxton, Hermans, & Fox, 2022) trains an object selection network to decide which object to rearrange and another pose generator network that samples the goal configuration to perform rearrangement plans that satisfy a natural language goal like putting objects into a shape or setting this table. Following a similar setup to StructFormer, StructDiffusion (Liu, Du, Hermans, Chernova, & Paxton, 2023b) incorporates a transformer-based diffusion model to generate 3D point cloud goal configurations that satisfy natural language instructions, and uses an additional classifier to reject physically invalid poses.

SREM (Gkanatsios, Jain, Xian, Zhang, Atkeson, & Fragkiadaki, 2023) generates 2D goal configurations that satisfy language instructions by using energy-based models (EBM) to ground spatial concepts composablely. Specifically, it utilizes a BinaryEBM to ground binary language concepts (e.g., left of) and a MultiAryEBM to ground multi-ary language concepts (e.g., circle). The models are learned over the coordinates of centroids of detection bounding boxes generated by open vocabulary detectors, enabling generalization to novel objects and invariance to environment perturbations. The composableity of energy-based models also allows jointly optimizing over an arbitrary number of constraints, allowing for the satisfaction of a language

instruction composed of multiple implicit constraints at the same time.

Learn Semantically Meaningful Rearrangement Goals Without Grounding

TarGF represents the objects as graphs and uses a score-based generative model to model a Target Gradient Field that encodes the likelihood of a configuration satisfying the tidiness goal, which provides moving direction for motion planning or reward specification for reinforcement learning to execute rearrangement plans (Wu, Zhong, Xia, & Dong, 2022). Similarly, LEGO-Net learns a score-based model that generates neat configurations of a 2D scene that satisfy tidiness and alignment constraints but encodes the objects with positions and types and uses a transformer as an architecture (Wei, Ding, Park, Sajnani, Poulenard, Sridhar, & Guibas, 2023). Besides, NeatNet (Kapelyukh & Johns, 2022) utilizes a variational autoencoder (VAE) based on GNN scene encoders to learn more personalized tidiness preferences in the latent space and generate tidy configurations for the positions of furniture for room rearrangement.

Deduce Rearrangement Goal From Semantic Relationships Among Objects

ConSOR (Zhu, Tremblay, Birchfield, & Zhu, 2021) shows that contextual cues from partially arranged scenes provide sufficient context to enable robots to perform object rearrangement without any explicit user goal specification. Given a fixed set of objects already in some uniform receptacles and some new objects to place into the receptacles, ConSOR uses a learned Transformer encoder to generate an object-centric latent embedding space from the partially arranged initial state that mimics the object grouping in the desired goal state. The object-centric embeddings are then clustered to determine object placements in the predicted goal state. The representations are augmented with commonsense knowledge from ConceptNet to generalize to novel object categories.

In order to tidy rooms by detecting out-of-place objects and put the objects in the correct position, TIDEE (Sarch, Fang, Harley, Schydlo, Tarr, Gupta, & Fragkiadaki, 2022) augments an image detector to predict the objects' semantic categories and detects if an object is visually out of place, and trains a language classifier to detect if an object is semantically out of place.

They infer plausible receptacles for the object to be placed onto through graph inference over the union of a neural memory graph of objects and spatial relations and the scene graph of the room at hand. Similarly, Housekeep (Kant, Ramachandran, Yenamandra, Gilitschenski, Batra, Szot, & Agrawal, 2022) proposes a benchmark and defines the household tidying task as robot automatically decides out-of-place objects, and puts them back to the correct locations, and provides baseline solutions of stacking MLP with LLM embedding or Masked Language Modeling (MLM) to infer object receptacle relationships. For more specialized user habits, TidyBot deduced user preference for object receptacle relationships by using LLM to summarise a general rule from a few examples about user preference history to put misplaced objects back to a proper position (Wu, Antonova, Kan, Lepert, Zeng, Song, Bohg, Rusinkiewicz, & Funkhouser, 2023).

Sample Rearrangement Goal Directly From Foundation Model

DALL-E-Bot samples images of how tables should be set from DALL-E, a pre-trained image generative model (Kapelyukh, Vosylius, & Johns, 2023), that was post-processed to be used as goal specification for table setting, while Ding et al. obtain similar information by querying LLM for the relative position and distance among the forks and plates (Ding, Zhang, Paxton, & Zhang, 2023).

2.2 Space Allocation Under Geometric Constraints

Generating proposals of object placement goals under geometric constraints is usually modeled as 2D or 3D bin packing problem: a set of items of various sizes has to be packed into a minimum number of identical bins, or various-sized bin packing problems when bins are not identical.

Early approaches to the bin packing problem were primarily heuristic-based, like the First Fit and Best Fit algorithms, due to their simplicity and reasonable performance for 2D problems. However, as highlighted by Martello and Toth, these heuristics often fell short in efficiently solving more complex 3D problems, prompting a shift towards more sophisticated methods (Martello & Toth, 1990). While the incorporation of genetic algorithms (Bortfeldt & Gehring,

2001) solves 3D bin packing with known objects, recent advances in deep reinforcement learning also enabled efficient online 3D bin packing when objects are not fully known (Zhao, She, Zhu, Yang, & Xu, 2021).

Besides methods specifically designed to solve the bin packing problems, generic methods for resolving constraint satisfaction problems also apply. Yang et al. leverage a graph to capture the dependencies between different decision variables and condition variables to model continuous constraint satisfaction problems and capture the conditional distribution of variable assignment via diffusion models, which provides an effective solution for 2D triangle packing with collision constraint, 2D shape arrangement with qualitative constraints, and 3D block packing with stability constraints (Yang, Mao, Du, Wu, Tenenbaum, Lozano-Pérez, & Kaelbling, 2023).

Other problems are also formulated to optimize over spatial constraints. For example, Optimal Shelf Arrangement (OSA) aims to optimize the arrangement of objects on a shelf for access time given an access frequency and movement cost for each object, and can be solved by mixed integer linear programming (Chen, Huang, Danielczuk, Ichnowski, & Goldberg, 2022a).

2.3 Encoding Commonsense and User Preference

Besides providing direct cues for generating goals for object rearrangement tasks, commonsense and user preferences have been leveraged to contribute to other tasks as well.

2.3.1 Sample Commonsense And User Preference From Foundation Model

Foundation models, trained on internet scale data, are believed to capture commonsense and human behavior that could be utilized by various robotics tasks (Bommasani, Hudson, Adeli, Altman, Arora, von Arx, Bernstein, Bohg, Bosselut, Brunskill, & others, 2021). Xie et al. use a large language model (LLM) to translate natural language instruction into planning goals (Xie, Yu, Zhu, Bai, Gong, & Soh, 2023). Subsequently, LLM-MCTS (Zhao, Lee, & Hsu, 2023) samples LLM for commonsense of where objects usually are, and uses LLM as a heuristic to guide Monte Carlo tree search to scale up task planning. LLM is also used for inferring object relationships. For example, Chen et al. query LLM for commonsense of object room affinities

for 3D scene understanding (Chen, Hu, Talak, & Carbone, 2022b).

2.3.2 Learning Commonsense And User Preference

Before the rise of deep learning, Abdo et al. learned user preference for object-object relationships and object-container relationships through collaborative filtering (Abdo, Stachniss, Spinello, & Burgard, 2016). Schuster et al. model human organization habits as a classification task (Schuster, Jain, Tenorth, & Beetz, 2012). They represent the objects using their class, physical constraints, purpose and packing, and use traditional machine learning classifiers such as support vector machine and naive Bayes to classify the items.

To tidy up a table with various objects, Xu et al. (Xu & Hsu, 2023) leverage LLM to capture the semantic object information in a scene and propose possible rearrangement actions, and train an additional visual tidiness score to select among the action proposals. Fusing both visual tidiness and semantic tidiness, the solution outperforms action proposals solely by LLM.

Giuliari et al. introduce Spatial Commonsense Graphs (SGG) for object localization in partial scenes (Giuliari, Skenderi, Cristani, Wang, & Del Bue, 2022). To embed commonsense knowledge into a geometric scene graph extracted from a partial scan of an area, they incorporate proximity edge to relate pairwise distance among objects, atLocation and UsedFor edge retrieved from ConceptNet to describe where the object is usually located in and used for. To localize unseen objects, they proposed a Proximity Prediction Network(PPN) on top of the SCG that predicts the pairwise distances between the unseen target object and the objects in the partially known scene.

2.4 Object Centric Relational Representation

As the local and global relationships among the objects are essential components of the semantics, incorporating an object-centric relational representation may offer a systematic way to encode the semantics, and possibly provide an effective interface for planners.

3D Scene graph, a unified representation to capture the relative spatial relationships among objects and hierarchical relationships among floors, rooms, and objects, has later been used as

an intermediary for various tasks (Armeni, He, Gwak, Zamir, Fischer, Malik, & Savarese, 2019). Ravichandran et al. trains navigation policy via reinforcement learning over 3D scene graph (Ravichandran, Peng, Hughes, Griffith, & Carlone, 2022). Capturing occupancy and semantic content and explicitly retaining memory of the robot trajectory, they show that their method outperforms normal visual-motor policy for a multi-object search task.

Besides, dynamic and uncertain scene graphs are also invented. Scene Graph Memory (SGM) has been proposed as a state representation to aggregate an agent’s observation of a dynamic environment over time, based upon which a Node Edge Predictor model could address dynamic link prediction problems in partial observable settings (Kurenkov, Lingelbach, Agarwal, Jin, Li, Zhang, Fei-Fei, Wu, Savarese, & Martín-Martin, 2023). Besides, scene graph has also been grounded with POMDP for effective object search (Amiri, Chandan, & Zhang, 2022).

Furthermore, scene graphs are also used as an intermediary for more effective task and motion planning. Grounding 3D scene graph with LLM, SayPlan enables task planning of an impressive scale with only simple human instructions as input (Rana, Haviland, Garg, Abou-Chakra, Reid, & Suenderhauf, 2023). Moreover, Zhu et al. utilize a symbolic scene graph encoding abstract semantic relationships among objects and the robot for task planning and a geometric scene graph depicting the 6-DoF poses of objects and the relative spatial relationships for motion planning (Zhu et al., 2021). Task planning is carried out by a modified version of the Regression Planning Network (RPN) (Xu, Martín-Martín, Huang, Zhu, Savarese, & Fei-Fei, 2019) that outputs the next planning subgoal, which together with the geometric scene graph is fed into another motion planning model to output a continuous parameter in the action space for a selected motion primitive. Leveraging the geometric information from perception otherwise abstracted away by TAMP, they show that the method achieves better computational complexity and generalization to unseen tasks. Jiao et al. also invent a contact graph that encodes the geometric dependencies, proximal relations as well as task-dependent supporting and statute attributes among objects for task planning for sequential manipulation, upon which planning is carried out by topological sorting (Jiao, Niu, Zhang, Zhu, Zhu, & Liu, 2022).

Chapter 3

Problem Analysis And Proposed Solution

3.1 Analysis of the Problem

There are several constraints that need to be captured to solve the problems above:

1. Shelf shape vs object shape
2. Shelf volume vs object volume
3. Object poses. For example, a booklet 30 degrees tilted over an open can of salt might not be nice.
4. Local semantics: Objects of similar categories need to be placed together.
5. Global semantics: Objects put in the correct layers of the shelf. For example, slippers at the very bottom.

Here are some additional parameters that could matter for the complexity of the problem:

1. Shape or categorical uniformity of objects
2. Shape uniformity of shelf structure
3. Total volume of objects vs total available space of the shelf

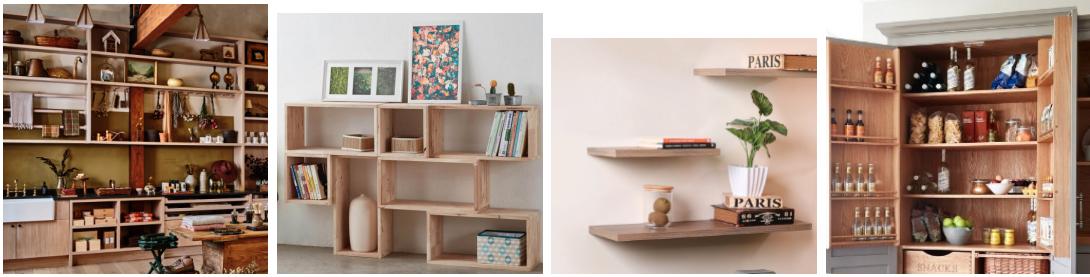


Figure 3.1: From left to right, (1) Large shelf with diverse usage for each cell, and implicit convention of how each cell should be used. (2) Complicated shelf structure that matters for both physical constraints of if the shelf can hold something, and semantic constraints of whether something is put at a place, for example, putting the books at the place where the big vase is standing is usually not appropriate. (3) Half open shelf space that requires interpolation of shelf structure from perception.(4) Perceiving natural clustering of objects in a continuous space.

For the user preference, there is a difference between the general preference among the population and the specific preference for each individual. We focus on the general preference first to find the semantic space projection and scoring function.

To enable the robot to ‘understand’ the constraints and relationships, we may need to find a unified representation to encode the spatial constraints and semantic commonsense constraints, building upon which rearrangement proposals and could be generated effectively.

We could utilize a hierarchical object-centric relational representation and either condition the discrete graph on some continuous model or store some numerical values in the nodes and edges to encode the spatial information that is abstracted out by the graph. For example, connect two objects with edges that have a certain probability of existence dependent on how relevant their categories are and how far they are placed from each other.

To process the graph, we could introduce additional models to output commonsense compliance scores and utilize an energy-based model (EBM) or diffusion model to generate proposals for rearrangement goals.

The construction process of this graph could be interpreted as projecting to the semantic space and the energy that is minimized by the EBM or diffusion model could be the semantic

scoring function.

3.2 Plans For Next Semester

3.2.1 November and December

1. Scope the problem better.
2. Investigate existing space allocation solutions' applicability to the shelf case.
3. Investigate existing user preference and commonsense scoring solutions' applicability to the shelf case.
4. Modify the existing solutions so the two constraints can be more easily combined.
5. Build simulating environments to test the algorithms.
6. Propose preliminary constraint combining solutions.

3.2.2 January and February

1. Implement and test the preliminary solution.
2. Refine the solution from testing results.

3.2.3 March

1. Continue testing and refining the solution.
2. Try plugging the goal proposal into a planner if have more time.

3.2.4 April

1. Finalise the solution and prepare report for submission.

Chapter 4

Other topics that I explored

Before coming to this topic, I also explored other possible topics that I decided not to take up.

4.1 Large Language Model for Planning

Believed to incorporate commonsense and is "sparks of AGI" (Bubeck, Chandrasekaran, Eldan, Gehrke, Horvitz, Kamar, Lee, Lee, Li, Lundberg, & others, 2023), large language models are recently utilized to help with task planning in robotics. While early works directly prompt LLM to output planning policies (Huang, Abbeel, Pathak, & Mordatch, 2022), LLM is also used as heuristics (Zhao et al., 2023), or outputs planning problem specification like PDDL domain and problem files (Liu, Jiang, Zhang, Liu, Zhang, Biswas, & Stone, 2023a). When combined with open vocabulary perception modules and foundation model controllers, it enables execution of tasks in the open world (Wang, Cai, Liu, Ma, & Liang, 2023). To alleviate the correctness and excitability issues of policies directly output by LLM, methods have been proposed to train external models to rank the correctness and executability of actions output by LLM (Hazra, Martires, & De Raedt, 2023). Structural Causal Model has also been utilized to reduce the spurious output from LLM planners (Lu, Feng, Zhu, Xu, Wang, Eckstein, & Wang, 2023). Besides, conformal prediction has been leveraged to calibrate the uncertainty of the actions proposed by LLM-based policies to trigger human help automatically when uncertainty exceeds a certain level (Ren, Dixit, Bodrova, Singh, Tu, Brown, Xu, Takayama, Xia, Varley, Xu, Sadigh,

Zeng, & Majumdar, 2023).

Despite its impressive performance, LLM’s limitation for planning is not fully known, and whether the mistakes that it makes are accidental has not been thoroughly investigated. While Dziri et al. show by experiment that LLM struggle with increasing breadth and depth for reasoning when solving problems (Dziri, Lu, Sclar, Li, Jian, Lin, West, Bhagavatula, Bras, Hwang, & others, 2023), the reason why this happen is still not clear.

References

- Abdo, N., Stachniss, C., Spinello, L., & Burgard, W. (2016). Organizing objects by predicting user preferences through collaborative filtering. *The International Journal of Robotics Research*, 35(13), 2016, 1587–1608.
- Amiri, S., Chandan, K., & Zhang, S. (2022). Reasoning with scene graphs for robot planning under partial observability. *IEEE Robotics and Automation Letters*, 7(2), 2022, 5560–5567.
- Armeni, I., He, Z.-Y., Gwak, J., Zamir, A. R., Fischer, M., Malik, J., & Savarese, S. (2019). 3d scene graph: A structure for unified semantics, 3d space, and camera. *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 5664–5673), 2019.
- Batra, D., Chang, A. X., Chernova, S., Davison, A. J., Deng, J., Koltun, V., Levine, S., Malik, J., Mordatch, I., Mottaghi, R., et al. (2020). Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, , 2020.
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, , 2021.
- Bortfeldt, A., & Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1), 2001, 143–161.
- Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, , 2023.
- Chen, L. Y., Huang, H., Danielczuk, M., Ichnowski, J., & Goldberg, K. (2022a). Optimal shelf arrangement to minimize robot retrieval time. *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)* (pp. 993–1000), IEEE, 2022.
- Chen, W., Hu, S., Talak, R., & Carlone, L. (2022b). Leveraging large language models for robot 3d scene understanding.
- Dhamo, H., Manhardt, F., Navab, N., & Tombari, F. (2021). Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 16352–16361), 2021.
- Ding, Y., Zhang, X., Paxton, C., & Zhang, S. (2023). Task and motion planning with large language models for object rearrangement. *arXiv preprint arXiv:2303.06247*, , 2023.

- Dziri, N., Lu, X., Sclar, M., Li, X. L., Jian, L., Lin, B. Y., West, P., Bhagavatula, C., Bras, R. L., Hwang, J. D., et al. (2023). Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*, , 2023.
- Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., & Lozano-Pérez, T. (2021). Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1), 2021, 265–293.
- Giuliari, F., Skenderi, G., Cristani, M., Wang, Y., & Del Bue, A. (2022). Spatial commonsense graph for object localisation in partial scenes. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 19518–19527), June, 2022.
- Gkanatsios, N., Jain, A., Xian, Z., Zhang, Y., Atkeson, C., & Fragkiadaki, K. (2023). Energy-based Models are Zero-Shot Planners for Compositional Scene Rearrangement. *Robotics: Science and Systems*, 2023.
- Hazra, R., Martires, P. Z. D., & De Raedt, L. (2023). Saycanpay: Heuristic planning with large language models using learnable domain knowledge. *arXiv preprint arXiv:2308.12682*, , 2023.
- Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *International Conference on Machine Learning* (pp. 9118–9147), PMLR, 2022.
- Jiao, Z., Niu, Y., Zhang, Z., Zhu, S.-C., Zhu, Y., & Liu, H. (2022). Sequential manipulation planning on scene graph. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 8203–8210), IEEE, 2022.
- Kant, Y., Ramachandran, A., Yenamandra, S., Gilitschenski, I., Batra, D., Szot, A., & Agrawal, H. (2022). Housekeep: Tidying virtual households using commonsense reasoning. *European Conference on Computer Vision* (pp. 355–373), Springer, 2022.
- Kapelyukh, I., & Johns, E. (2022). My house, my rules: Learning tidying preferences with graph neural networks. In A. Faust, D. Hsu, & G. Neumann (Eds.), *Proceedings of the 5th Conference on Robot Learning*, Vol. 164 of *Proceedings of Machine Learning Research* (pp. 740–749), 08–11 Nov, 2022: PMLR.
- Kapelyukh, I., Vosylius, V., & Johns, E. (2023). Dall-e-bot: Introducing web-scale diffusion models to robotics. *IEEE Robotics and Automation Letters*, , 2023.
- Kulshrestha, M., & Qureshi, A. H. (2023). Structural concept learning via graph attention for multi-level rearrangement planning. *7th Annual Conference on Robot Learning*, 2023.
- Kurenkov, A., Lingelbach, M., Agarwal, T., Jin, E., Li, C., Zhang, R., Fei-Fei, L., Wu, J., Savarese, S., & Martin-Martin, R. (2023). Modeling dynamic environments with scene graph memory. *International Conference on Machine Learning* (pp. 17976–17993), PMLR, 2023.
- Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., & Stone, P. (2023a). Llm+p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, , 2023.
- Liu, W., Du, Y., Hermans, T., Chernova, S., & Paxton, C. (2023b). Structdiffusion: Language-guided creation of physically-valid structures using unseen objects. *RSS 2023*, 2023.

- Liu, W., Paxton, C., Hermans, T., & Fox, D. (2022). Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects. *2022 International Conference on Robotics and Automation (ICRA)* (pp. 6322–6329), IEEE, 2022.
- Lu, Y., Feng, W., Zhu, W., Xu, W., Wang, X. E., Eckstein, M., & Wang, W. Y. (2023). Neuro-symbolic procedural planning with commonsense prompting. *The Eleventh International Conference on Learning Representations*, 2023.
- Martello, S., & Toth, P. (1990). Lower bounds and reduction procedures for the bin packing problem. *Discrete applied mathematics*, 28(1), 1990, 59–70.
- Qureshi, A. H., Mousavian, A., Paxton, C., Yip, M. C., & Fox, D. (2021). Nerp: Neural rearrangement planning for unknown objects. *arXiv preprint arXiv:2106.01352*, , 2021.
- Rana, K., Haviland, J., Garg, S., Abou-Chakra, J., Reid, I., & Suenderhauf, N. (2023). Say-plan: Grounding large language models using 3d scene graphs for scalable task planning. *7th Annual Conference on Robot Learning*, 2023.
- Ravichandran, Z., Peng, L., Hughes, N., Griffith, J. D., & Carlone, L. (2022). Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. *2022 International Conference on Robotics and Automation (ICRA)* (pp. 9272–9279), IEEE, 2022.
- Ren, A. Z., Dixit, A., Bodrova, A., Singh, S., Tu, S., Brown, N., Xu, P., Takayama, L., Xia, F., Varley, J., Xu, Z., Sadigh, D., Zeng, A., & Majumdar, A. (2023). Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*, 2023.
- Sarch, G., Fang, Z., Harley, A. W., Schydlo, P., Tarr, M. J., Gupta, S., & Fragkiadaki, K. (2022). Tidee: Tidying up novel rooms using visuo-semantic commonsense priors. *European Conference on Computer Vision* (pp. 480–496), Springer, 2022.
- Schuster, M. J., Jain, D., Tenorth, M., & Beetz, M. (2012). Learning organizational principles in human environments. *2012 IEEE International Conference on Robotics and Automation* (pp. 3867–3874), 2012.
- Wang, Z., Cai, S., Liu, A., Ma, X., & Liang, Y. (2023). Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, , 2023.
- Wei, Q. A., Ding, S., Park, J. J., Sajnani, R., Poulenard, A., Sridhar, S., & Guibas, L. (2023). Lego-net: Learning regular rearrangements of objects in rooms. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 19037–19047), 2023.
- Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., & Funkhouser, T. (2023). Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*, , 2023.
- Wu, M., Zhong, F., Xia, Y., & Dong, H. (2022). Targf: Learning target gradient field to rearrange objects without explicit goal specification. *Advances in Neural Information Processing Systems*, 35, 2022, 31986–31999.

- Xie, Y., Yu, C., Zhu, T., Bai, J., Gong, Z., & Soh, H. (2023). Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, , 2023.
- Xu, D., Martín-Martín, R., Huang, D.-A., Zhu, Y., Savarese, S., & Fei-Fei, L. F. (2019). Regression planning networks. *Advances in Neural Information Processing Systems*, 32, 2019.
- Xu, Y., & Hsu, D. (2023). How to tidy up a table: Fusing visual and semantic commonsense reasoning for robotic tasks with vague objectives. *arXiv preprint arXiv:2307.11319*, , 2023.
- Yang, Z., Mao, J., Du, Y., Wu, J., Tenenbaum, J. B., Lozano-Pérez, T., & Kaelbling, L. P. (2023). Compositional Diffusion-Based Continuous Constraint Solvers. *Conference on Robot Learning*, 2023.
- Zhai, G., Cai, X., Huang, D., Di, Y., Manhardt, F., Tombari, F., Navab, N., & Busam, B. (2023). Sg-bot: Object rearrangement via coarse-to-fine robotic imagination on scene graphs. *arXiv preprint arXiv:2309.12188*, , 2023.
- Zhao, H., She, Q., Zhu, C., Yang, Y., & Xu, K. (2021). Online 3d bin packing with constrained deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35 (pp. 741–749), 2021.
- Zhao, Z., Lee, W. S., & Hsu, D. (2023). Large language models as commonsense knowledge for large-scale task planning. *arXiv preprint arXiv:2305.14078*, , 2023.
- Zhu, Y., Tremblay, J., Birchfield, S., & Zhu, Y. (2021). Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6541–6548), IEEE, 2021.

B.Comp. Dissertation

**Placing Objects Onto Shelf: Object Rearrangement Under
Geometric And Commonsense Constraints**

By

Zhu Tianqi

Department of Computer Science

School of Computing

National University of Singapore

2023/24

Abstract

In this work, we tested VLM and LLM’s ability for various properties related goal proposals for large language model constraints. Specifically, we tested VLM’s ability on pattern deduction, goal proposal under geometric constraint, and goal proposal under stability constraints, and generated shelf arrangement goals on a 2d shelf.

Table of Contents

Title	i
Abstract	ii
Acknowledgement	iii
1 Introduction	1
1.1 Background	1
1.2 The Problem	2
2 Related Work	3
2.1 Object Rearrangement	3
2.1.1 Inferring Rearrangement Planning Goals From Weak Human Guidance	3
2.2 Space Allocation Under Geometric Constraints	6
2.3 Encoding Commonsense and User Preference	7
2.3.1 Sample Commonsense And User Preference From Foundation Model	7
2.3.2 Learning Commonsense And User Preference	8
2.4 Object Centric Relational Representation	8
3 Problem Analysis And Proposed Solution	10
3.1 Testing of GPT4-v	10
3.1.1 Pattern Deduction	10
3.1.2 Proposing Arrangement Goals under Stability Constraints	11
3.1.3 Proposing Arrangement Goals under Geometrical Constraints	13
3.2 Proposing object arrangement goals	13
4 Other topics that I explored	15
4.1 Large Language Model for Planning	15
References	17
A Example for Pattern Deduction	A-1
B Example for Geometrical Constraint Goal Proposal	B-1
C Example for Stability Testing	C-1
D Sample Images Generated by LLM Arrangement	D-1
E Sample Images Generated by Relationship Proposal	E-1

Chapter 1

Introduction

1.1 Background

To function effectively in the dynamic and unstructured settings of household environments, robots require a nuanced understanding of several key factors. First and foremost, they need to grasp human habits and preferences, as these dictate the practical usage of space and objects within the home. Additionally, an awareness of the physical constraints inherent in these environments is crucial. This includes understanding spatial limitations, the varying nature of surfaces, and the need for safe navigation and manipulation. Lastly, comprehending the spatial relationships among different objects and how they are used in conjunction with each other is essential. This comprehensive understanding enables robots to navigate and perform tasks in a manner that is both efficient and aligned with the habits and expectations of human users.

In this work, we attempt to address the challenge of robots placing unknown objects onto multi-layer shelves already containing items. The robot must first assess whether there's adequate space for the new items. If space is limited, it may need to reorganize existing items to accommodate the additions. This task is particularly pertinent in household environments, where ease of locating objects is crucial for both humans and robots. To achieve this, the robot should arrange items based on user habits and common sense, grouping similar categories together and placing items in their typical positions. For instance, on a kitchen shelf, a new can

of juice might be placed alongside existing Coke cans, while on a bedroom shelf, slippers would likely be placed at the bottom.

While previous studies have focused on individual aspects of this problem, they haven't holistically addressed the dual challenge of space optimization and intuitive arrangement. Our goal is to develop a comprehensive approach that simultaneously satisfies these constraints, enabling more effective and user-friendly object rearrangement by robots.

1.2 The Problem

There are several variants of the object placing onto shelf problem concerning how many more objects to place and how many objects are already on the shelf:

1. Given a set of objects, place it nicely on an empty shelf.
2. Given a shelf with x existing objects, find a place to put one or more additional objects.
3. Given a shelf with x existing objects, rearrange the objects on the shelf to make it look nicer or more conformal with commonsense.

We focus on generating object rearrangement goals for the first task variant. We define geometric constraint as objects not overlapping with each other and the shape and volume of the shelf can hold the objects, semantic constraint as objects are placed at a user-friendly location on the shelf, i.e. the relative positions among objects locally and global object position to the whole shelf complies with user habit and commonsense, and stability / physical constraint as object wouldn't slip from the base objects that were placed upon.

For a set of objects (o_1, o_2, \dots, o_n) and a shelf with cells c_1, c_2, \dots, c_n . We aim to look for a cell and a position inside the cell for each object.

Chapter 2

Related Work

2.1 Object Rearrangement

The task that we are trying to solve belongs to the task category of object rearrangement (Batra, Chang, Chernova, Davison, Deng, Koltun, Levine, Malik, Mordatch, Mottaghi, & others, 2020). Object rearrangement is traditionally solved by task and motion planning (TAMP) (Garrett, Chitnis, Holladay, Kim, Silver, Kaelbling, & Lozano-Pérez, 2021). Given an initial configuration and a predefined goal configuration, a rearrangement plan is generated via searching to solve the hybrid constraint satisfaction problem. It relies on predefined symbolic rules and fine-grained dynamics models and suffers from high computation complexity when scaling up. It usually works in 2D space and geometric constraints beyond cluttered situations are not fully tapped. Recent deep learning based methods that learn rearrangement policies from data enable generalization to unknown objects (Qureshi, Mousavian, Paxton, Yip, & Fox, 2021), and scaling to more complex multi-level structures (Kulshrestha & Qureshi, 2023).

2.1.1 Inferring Rearrangement Planning Goals From Weak Human Guidance

While traditional methods require programmers to specify the rearrangement goal configurations explicitly and accurately as 2D or 3D positions or poses, recent advances enable robots to deduce those configurations from weaker human supervision like natural language instruction, symbolic graphs, predicate satisfaction, or even commonsense to rearrange less known objects,

which equips robots with more flexible behaviors to operate in less structured environments like households. Following similar setups, we aim to deduce the rearrangement planning goals purely from observation, pretrained foundation models, and more specified smaller models that generate goal configurations.

Specify Rearrangement Goal With Scene Graph

SG-Bot (Zhai, Cai, Huang, Di, Manhardt, Tombari, Navab, & Busam, 2023) represents the scene as a scene graph, allowing a commonsense mode that automatically generates the graph from foundation models or a user-defined mode that allows users to specify the graph directly. The scene graph is later fed into a Graph-to-3D model (Dhamo, Manhardt, Navab, & Tombari, 2021) that maps the graph to 3D point cloud as a fine-grained rearrangement goal.

Ground Rearrangement Goal With Natural Language

StructFormer (Liu, Paxton, Hermans, & Fox, 2022) trains an object selection network to decide which object to rearrange and another pose generator network that samples the goal configuration to perform rearrangement plans that satisfy a natural language goal like putting objects into a shape or setting this table. Following a similar setup to StructFormer, StructDiffusion (Liu, Du, Hermans, Chernova, & Paxton, 2023b) incorporates a transformer-based diffusion model to generate 3D point cloud goal configurations that satisfy natural language instructions, and uses an additional classifier to reject physically invalid poses.

SREM (Gkanatsios, Jain, Xian, Zhang, Atkeson, & Fragkiadaki, 2023) generates 2D goal configurations that satisfy language instructions by using energy-based models (EBM) to ground spatial concepts composablely. Specifically, it utilizes a BinaryEBM to ground binary language concepts (e.g., left of) and a MultiAryEBM to ground multi-ary language concepts (e.g., circle). The models are learned over the coordinates of centroids of detection bounding boxes generated by open vocabulary detectors, enabling generalization to novel objects and invariance to environment perturbations. The composableity of energy-based models also allows jointly optimizing over an arbitrary number of constraints, allowing for the satisfaction of a language

instruction composed of multiple implicit constraints at the same time.

Learn Semantically Meaningful Rearrangement Goals Without Grounding

TarGF represents the objects as graphs and uses a score-based generative model to model a Target Gradient Field that encodes the likelihood of a configuration satisfying the tidiness goal, which provides moving direction for motion planning or reward specification for reinforcement learning to execute rearrangement plans (Wu, Zhong, Xia, & Dong, 2022). Similarly, LEGO-Net learns a score-based model that generates neat configurations of a 2D scene that satisfy tidiness and alignment constraints but encodes the objects with positions and types and uses a transformer as an architecture (Wei, Ding, Park, Sajnani, Poulenard, Sridhar, & Guibas, 2023). Besides, NeatNet (Kapelyukh & Johns, 2022) utilizes a variational autoencoder (VAE) based on GNN scene encoders to learn more personalized tidiness preferences in the latent space and generate tidy configurations for the positions of furniture for room rearrangement.

Deduce Rearrangement Goal From Semantic Relationships Among Objects

ConSOR (Zhu, Tremblay, Birchfield, & Zhu, 2021) shows that contextual cues from partially arranged scenes provide sufficient context to enable robots to perform object rearrangement without any explicit user goal specification. Given a fixed set of objects already in some uniform receptacles and some new objects to place into the receptacles, ConSOR uses a learned Transformer encoder to generate an object-centric latent embedding space from the partially arranged initial state that mimics the object grouping in the desired goal state. The object-centric embeddings are then clustered to determine object placements in the predicted goal state. The representations are augmented with commonsense knowledge from ConceptNet to generalize to novel object categories.

In order to tidy rooms by detecting out-of-place objects and put the objects in the correct position, TIDEE (Sarch, Fang, Harley, Schydlo, Tarr, Gupta, & Fragkiadaki, 2022) augments an image detector to predict the objects' semantic categories and detects if an object is visually out of place, and trains a language classifier to detect if an object is semantically out of place.

They infer plausible receptacles for the object to be placed onto through graph inference over the union of a neural memory graph of objects and spatial relations and the scene graph of the room at hand. Similarly, Housekeep (Kant, Ramachandran, Yenamandra, Gilitschenski, Batra, Szot, & Agrawal, 2022) proposes a benchmark and defines the household tidying task as robot automatically decides out-of-place objects, and puts them back to the correct locations, and provides baseline solutions of stacking MLP with LLM embedding or Masked Language Modeling (MLM) to infer object receptacle relationships. For more specialized user habits, TidyBot deduced user preference for object receptacle relationships by using LLM to summarise a general rule from a few examples about user preference history to put misplaced objects back to a proper position (Wu, Antonova, Kan, Lepert, Zeng, Song, Bohg, Rusinkiewicz, & Funkhouser, 2023).

Sample Rearrangement Goal Directly From Foundation Model

DALL-E-Bot samples images of how tables should be set from DALL-E, a pre-trained image generative model (Kapelyukh, Vosylius, & Johns, 2023), that was post-processed to be used as goal specification for table setting, while Ding et al. obtain similar information by querying LLM for the relative position and distance among the forks and plates (Ding, Zhang, Paxton, & Zhang, 2023).

2.2 Space Allocation Under Geometric Constraints

Generating proposals of object placement goals under geometric constraints is usually modeled as 2D or 3D bin packing problem: a set of items of various sizes has to be packed into a minimum number of identical bins, or various-sized bin packing problems when bins are not identical.

Early approaches to the bin packing problem were primarily heuristic-based, like the First Fit and Best Fit algorithms, due to their simplicity and reasonable performance for 2D problems. However, as highlighted by Martello and Toth, these heuristics often fell short in efficiently solving more complex 3D problems, prompting a shift towards more sophisticated methods (Martello & Toth, 1990). While the incorporation of genetic algorithms (Bortfeldt & Gehring,

2001) solves 3D bin packing with known objects, recent advances in deep reinforcement learning also enabled efficient online 3D bin packing when objects are not fully known (Zhao, She, Zhu, Yang, & Xu, 2021).

Besides methods specifically designed to solve the bin packing problems, generic methods for resolving constraint satisfaction problems also apply. Yang et al. leverage a graph to capture the dependencies between different decision variables and condition variables to model continuous constraint satisfaction problems and capture the conditional distribution of variable assignment via diffusion models, which provides an effective solution for 2D triangle packing with collision constraint, 2D shape arrangement with qualitative constraints, and 3D block packing with stability constraints (Yang, Mao, Du, Wu, Tenenbaum, Lozano-Pérez, & Kaelbling, 2023).

Other problems are also formulated to optimize over spatial constraints. For example, Optimal Shelf Arrangement (OSA) aims to optimize the arrangement of objects on a shelf for access time given an access frequency and movement cost for each object, and can be solved by mixed integer linear programming (Chen, Huang, Danielczuk, Ichnowski, & Goldberg, 2022a).

2.3 Encoding Commonsense and User Preference

Besides providing direct cues for generating goals for object rearrangement tasks, commonsense and user preferences have been leveraged to contribute to other tasks as well.

2.3.1 Sample Commonsense And User Preference From Foundation Model

Foundation models, trained on internet scale data, are believed to capture commonsense and human behavior that could be utilized by various robotics tasks (Bommasani, Hudson, Adeli, Altman, Arora, von Arx, Bernstein, Bohg, Bosselut, Brunskill, & others, 2021). Xie et al. use a large language model (LLM) to translate natural language instruction into planning goals (Xie, Yu, Zhu, Bai, Gong, & Soh, 2023). Subsequently, LLM-MCTS (Zhao, Lee, & Hsu, 2023) samples LLM for commonsense of where objects usually are, and uses LLM as a heuristic to guide Monte Carlo tree search to scale up task planning. LLM is also used for inferring object relationships. For example, Chen et al. query LLM for commonsense of object room affinities

for 3D scene understanding (Chen, Hu, Talak, & Carbone, 2022b).

2.3.2 Learning Commonsense And User Preference

Before the rise of deep learning, Abdo et al. learned user preference for object-object relationships and object-container relationships through collaborative filtering (Abdo, Stachniss, Spinello, & Burgard, 2016). Schuster et al. model human organization habits as a classification task (Schuster, Jain, Tenorth, & Beetz, 2012). They represent the objects using their class, physical constraints, purpose and packing, and use traditional machine learning classifiers such as support vector machine and naive Bayes to classify the items.

To tidy up a table with various objects, Xu et al. (Xu & Hsu, 2023) leverage LLM to capture the semantic object information in a scene and propose possible rearrangement actions, and train an additional visual tidiness score to select among the action proposals. Fusing both visual tidiness and semantic tidiness, the solution outperforms action proposals solely by LLM.

Giuliari et al. introduce Spatial Commonsense Graphs (SGG) for object localization in partial scenes (Giuliari, Skenderi, Cristani, Wang, & Del Bue, 2022). To embed commonsense knowledge into a geometric scene graph extracted from a partial scan of an area, they incorporate proximity edge to relate pairwise distance among objects, atLocation and UsedFor edge retrieved from ConceptNet to describe where the object is usually located in and used for. To localize unseen objects, they proposed a Proximity Prediction Network(PPN) on top of the SCG that predicts the pairwise distances between the unseen target object and the objects in the partially known scene.

2.4 Object Centric Relational Representation

As the local and global relationships among the objects are essential components of the semantics, incorporating an object-centric relational representation may offer a systematic way to encode the semantics, and possibly provide an effective interface for planners.

3D Scene graph, a unified representation to capture the relative spatial relationships among objects and hierarchical relationships among floors, rooms, and objects, has later been used as

an intermediary for various tasks (Armeni, He, Gwak, Zamir, Fischer, Malik, & Savarese, 2019). Ravichandran et al. trains navigation policy via reinforcement learning over 3D scene graph (Ravichandran, Peng, Hughes, Griffith, & Carlone, 2022). Capturing occupancy and semantic content and explicitly retaining memory of the robot trajectory, they show that their method outperforms normal visual-motor policy for a multi-object search task.

Besides, dynamic and uncertain scene graphs are also invented. Scene Graph Memory (SGM) has been proposed as a state representation to aggregate an agent’s observation of a dynamic environment over time, based upon which a Node Edge Predictor model could address dynamic link prediction problems in partial observable settings (Kurenkov, Lingelbach, Agarwal, Jin, Li, Zhang, Fei-Fei, Wu, Savarese, & Martín-Martin, 2023). Besides, scene graph has also been grounded with POMDP for effective object search (Amiri, Chandan, & Zhang, 2022).

Furthermore, scene graphs are also used as an intermediary for more effective task and motion planning. Grounding 3D scene graph with LLM, SayPlan enables task planning of an impressive scale with only simple human instructions as input (Rana, Haviland, Garg, Abou-Chakra, Reid, & Suenderhauf, 2023). Moreover, Zhu et al. utilize a symbolic scene graph encoding abstract semantic relationships among objects and the robot for task planning and a geometric scene graph depicting the 6-DoF poses of objects and the relative spatial relationships for motion planning (Zhu et al., 2021). Task planning is carried out by a modified version of the Regression Planning Network (RPN) (Xu, Martín-Martín, Huang, Zhu, Savarese, & Fei-Fei, 2019) that outputs the next planning subgoal, which together with the geometric scene graph is fed into another motion planning model to output a continuous parameter in the action space for a selected motion primitive. Leveraging the geometric information from perception otherwise abstracted away by TAMP, they show that the method achieves better computational complexity and generalization to unseen tasks. Jiao et al. also invent a contact graph that encodes the geometric dependencies, proximal relations as well as task-dependent supporting and statute attributes among objects for task planning for sequential manipulation, upon which planning is carried out by topological sorting (Jiao, Niu, Zhang, Zhu, Zhu, & Liu, 2022).

Chapter 3

Problem Analysis And Proposed Solution

3.1 Testing of GPT4-v

We tested the ability of vision language model and large language model for solving the problem.

3.1.1 Pattern Deduction

To propose arrangement goals complying with a specific user's preference over spatial arrangements, such as a left-handed person's preference over placing stationaries on the left-hand side, existing work learns a distribution of human preference by a generative model, specifically a GNN-based VAE. It was trained on arrangement data collected from particular users by asking the user to arrange some sample scenes. However, in real-life, the arrangement of data is tedious to generate for a particular human. Besides, the patterns are predefined by humans and it is unlikely to specify all patterns in the entire world by hand. For household shelves where designs vary and both low-level shape and high-level design style matter for the particular arrangement pattern to be chosen, we may need a more general-purpose pattern deducer to capture the preference over different patterns for a particular user.

Specifically, if we have a household robot that is about to help the user arrange his or her shelf, the robot needs to first capture an image of the existing shelf and deduce the qualities

of the shelf like design patterns, proper item types etc. Then, it searches for similar types of the shelf online for references for arrangement goals to be generated and prompts a user with a sidebar ranking how adorable it is on a scale between 1-5. A large vision language model could then deduce the patterns from the goals images with higher rankings. Besides, we could also deduce preferred patterns from the existing arrangement of objects in the user’s environment.

We tested VLM’s ability to extract organizational patterns from images. Given an input image, we first asked VLM to describe the image by ”Tell me something about the organizational pattern on this shelf”, the model will then output a language description of the existing patterns that it observes from the shelf image. The model could then break the more complicated patterns down to some basic relations or properties. And a compositional diffusion model could be utilised to solve the continuous constraint satisfaction problem. We leave the diffusion model training as future work.

3.1.2 Proposing Arrangement Goals under Stability Constraints

We also tested vlm’s ability to propose object arrangement goals for object stacking. We assume that the object weights are evenly distributed inside the object, and check if the model outputs the most stable and appropriate configuration possible. For objects that could stand stably on multiple surfaces, we consider the more commonly seen pose as a better configuration, for example, for an instant noodle cup, it could stand stably as both on its normal base or up-side-down, we consider the normal upright position as better. For objects that could stand stably on only one surface, we consider the more stable placing as better.

By inputting images of objects placed on a desk, we asked vlm to propose a goal configuration that is stable. We define the existing orientation of the objects as upright and ask the model to output the relative position between each objects and their orientation. We tested with only small amount of objects (2-4) and the model managed to identify the object types correctly. Despite not explicating requested, the model also output the tips for object placement, such as ’aligning at the center’, ’ensure the bottom of the box are flat’ etc.

It successfully predicted the arrangement in the following situations:

1. For two cans and one additional flat rectangular container, it proposed to place the container on top of the based formed by two cans next to each other.
2. For one rectangular object with larger horizontal area, a bottle with uneven lid, and a can, it proposed to put the rectangular object below and the other two objects on top of the larger surface.
3. For three objects with flat top surfaces, it proposed to place the object with larger horizontal surface at the very bottom, and the smallest at the very top.
4. For two rectangular objects and one bottle with smaller upper surface, it proposed to place the two rectangular objects according to their weight estimation (heavier at bottom), and the bottle on top.
5. One tall bottle with flat surface, two rectangular objects of different types, it proposed to place it according to surface areas and weight estimation.
6. Three objects of different sizes, it placed the largest at the bottom.
7. An open paper cup and a bottle, it proposed to place the paper cup upside down as the base and the bottle on top.

It has made the following errors:

1. For three objects of different shapes, it was unable to recognize the shape of an object (milk container) is not flat, and cannot be stably placed upon.
2. For three objects of different heights, it proposed to place an object on two objects that are obviously not the same heights.
3. For one bottle with uneven top, one rectangular object with even top, and one rectangular object with uneven top, it proposed to lay a cylindrical object horizontally as the base because the possible weight concerns.
4. For one bottle with uneven top, two rectangular objects with even top, it proposed to lay the cylindrical bottle horizontally.

5. For two bottles and one instant noodle cup that has enough surface area for two bottles to stand upon, it proposed to stack two bottles with very small tops together.

The most frequent not-so-stable arrangement of the model proposes to lay a bottle flat.

3.1.3 Proposing Arrangement Goals under Geometrical Constraints

Given an image of objects placed on a shelf, there is limited space, and we prompt the vision language model to move an object to a cell with existing objects. Specifically, we asked the vision language model to propose a goal configuration for both the new object to be placed and the existing objects. We aim to test if the vision language model knows if the objects need to be moved and propose the correct goal.

3.2 Proposing object arrangement goals

For the scope of this work, we assume that the shelf is deep enough and but we need to make sure all items need to be fully visible from the front view for robot’s easy retrieval. Thus, we aim to solve the problem below. Given a list of daily items as 2d rectangles, and a shelf with multiple cells, we aim to assign each item to an appropriate cell in the shelf and set exact coordinates for each item in the cell. We aim to generate a physically valid configuration that is both aesthetically pleasing and convenient.

For grouping the objects into different cells of the shelf, we need to consider both the type of the object and the shape (if there is enough space in the cell). This is very similar to the bin packing problem, but the number of bins is fixed and our objective is minimizing a semantic compliance objective.

For the first step, there are several possible ways, we decided to prompt a large language model as it has the potential to utilize all the aspects of the scenarios. We tried both prompting with the size of the shelf cell and only including the objects and cell number in the first step.

We generated a list of commonly used object type with their width and height using GPT4, and randomly sampled each object configuration to be assigned onto the shelf from the list. We first asked the model to group the object into cells, then asked the model to propose the

arrangement goal by its relative position or coordinates, and finally, given the number of rows, columns, assign each cluster to a cell that takes the positions of the cell into consideration.

We defined the following language to describe the supporting relationships for a vertical shelf:

1. $\text{LeftMost}(A, B)$ means A, B are on the same object and A is to the left of B, and there is no object between them.
2. $\text{Below}(A, B)$ means A is below B and object A,B are in contact, which means when moving A, object B would be affected.
3. If multiple objects A_1, A_2, \dots, A_n are placed on top of the same object B, we represent the relationship between object A_x and object B separately as: $\text{Below}(B, A_x)$, and $\text{LeftMost}(A_i, A_{i+1})$.
4. If a bigger object A is placed on top of multiple smaller objects $B_1, B_2 \dots B_n$ sorted from left to right, and we include object B_x as long as A is in contact with B_x . We represent the relationship as $\text{Below}(\text{Left}(B_1, B_2 \dots B_n), A)$.

By sorting the language represented from down to up, left to right, it outputs a valid motion plan for placing the objects onto the shelf. For the relationship output format, we requested the model in order, to simulate the object placing sequences. We parse all the relationships among the objects and generate the pose of the objects greedily in order, by assuming no space between objects down to up, and even spacing between objects left and right.

Chapter 4

Other topics that I explored

Before coming to this topic, I also explored other possible topics that I decided not to take up.

4.1 Large Language Model for Planning

Believed to incorporate commonsense and is "sparks of AGI" (Bubeck, Chandrasekaran, Eldan, Gehrke, Horvitz, Kamar, Lee, Lee, Li, Lundberg, & others, 2023), large language models are recently utilized to help with task planning in robotics. While early works directly prompt LLM to output planning policies (Huang, Abbeel, Pathak, & Mordatch, 2022), LLM is also used as heuristics (Zhao et al., 2023), or outputs planning problem specification like PDDL domain and problem files (Liu, Jiang, Zhang, Liu, Zhang, Biswas, & Stone, 2023a). When combined with open vocabulary perception modules and foundation model controllers, it enables execution of tasks in the open world (Wang, Cai, Liu, Ma, & Liang, 2023). To alleviate the correctness and excitability issues of policies directly output by LLM, methods have been proposed to train external models to rank the correctness and executability of actions output by LLM (Hazra, Martires, & De Raedt, 2023). Structural Causal Model has also been utilized to reduce the spurious output from LLM planners (Lu, Feng, Zhu, Xu, Wang, Eckstein, & Wang, 2023). Besides, conformal prediction has been leveraged to calibrate the uncertainty of the actions proposed by LLM-based policies to trigger human help automatically when uncertainty exceeds a certain level (Ren, Dixit, Bodrova, Singh, Tu, Brown, Xu, Takayama, Xia, Varley, Xu, Sadigh,

Zeng, & Majumdar, 2023).

Despite its impressive performance, LLM’s limitation for planning is not fully known, and whether the mistakes that it makes are accidental has not been thoroughly investigated. While Dziri et al. show by experiment that LLM struggle with increasing breadth and depth for reasoning when solving problems (Dziri, Lu, Sclar, Li, Jian, Lin, West, Bhagavatula, Bras, Hwang, & others, 2023), the reason why this happen is still not clear.

References

- Abdo, N., Stachniss, C., Spinello, L., & Burgard, W. (2016). Organizing objects by predicting user preferences through collaborative filtering. *The International Journal of Robotics Research*, 35(13), 2016, 1587–1608.
- Amiri, S., Chandan, K., & Zhang, S. (2022). Reasoning with scene graphs for robot planning under partial observability. *IEEE Robotics and Automation Letters*, 7(2), 2022, 5560–5567.
- Armeni, I., He, Z.-Y., Gwak, J., Zamir, A. R., Fischer, M., Malik, J., & Savarese, S. (2019). 3d scene graph: A structure for unified semantics, 3d space, and camera. *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 5664–5673), 2019.
- Batra, D., Chang, A. X., Chernova, S., Davison, A. J., Deng, J., Koltun, V., Levine, S., Malik, J., Mordatch, I., Mottaghi, R., et al. (2020). Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, , 2020.
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, , 2021.
- Bortfeldt, A., & Gehring, H. (2001). A hybrid genetic algorithm for the container loading problem. *European Journal of Operational Research*, 131(1), 2001, 143–161.
- Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, , 2023.
- Chen, L. Y., Huang, H., Danielczuk, M., Ichnowski, J., & Goldberg, K. (2022a). Optimal shelf arrangement to minimize robot retrieval time. *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)* (pp. 993–1000), IEEE, 2022.
- Chen, W., Hu, S., Talak, R., & Carlone, L. (2022b). Leveraging large language models for robot 3d scene understanding.
- Dhamo, H., Manhardt, F., Navab, N., & Tombari, F. (2021). Graph-to-3d: End-to-end generation and manipulation of 3d scenes using scene graphs. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 16352–16361), 2021.
- Ding, Y., Zhang, X., Paxton, C., & Zhang, S. (2023). Task and motion planning with large language models for object rearrangement. *arXiv preprint arXiv:2303.06247*, , 2023.

- Dziri, N., Lu, X., Sclar, M., Li, X. L., Jian, L., Lin, B. Y., West, P., Bhagavatula, C., Bras, R. L., Hwang, J. D., et al. (2023). Faith and fate: Limits of transformers on compositionality. *arXiv preprint arXiv:2305.18654*, , 2023.
- Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., & Lozano-Pérez, T. (2021). Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1), 2021, 265–293.
- Giuliari, F., Skenderi, G., Cristani, M., Wang, Y., & Del Bue, A. (2022). Spatial commonsense graph for object localisation in partial scenes. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 19518–19527), June, 2022.
- Gkanatsios, N., Jain, A., Xian, Z., Zhang, Y., Atkeson, C., & Fragkiadaki, K. (2023). Energy-based Models are Zero-Shot Planners for Compositional Scene Rearrangement. *Robotics: Science and Systems*, 2023.
- Hazra, R., Martires, P. Z. D., & De Raedt, L. (2023). Saycanpay: Heuristic planning with large language models using learnable domain knowledge. *arXiv preprint arXiv:2308.12682*, , 2023.
- Huang, W., Abbeel, P., Pathak, D., & Mordatch, I. (2022). Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *International Conference on Machine Learning* (pp. 9118–9147), PMLR, 2022.
- Jiao, Z., Niu, Y., Zhang, Z., Zhu, S.-C., Zhu, Y., & Liu, H. (2022). Sequential manipulation planning on scene graph. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 8203–8210), IEEE, 2022.
- Kant, Y., Ramachandran, A., Yenamandra, S., Gilitschenski, I., Batra, D., Szot, A., & Agrawal, H. (2022). Housekeep: Tidying virtual households using commonsense reasoning. *European Conference on Computer Vision* (pp. 355–373), Springer, 2022.
- Kapelyukh, I., & Johns, E. (2022). My house, my rules: Learning tidying preferences with graph neural networks. In A. Faust, D. Hsu, & G. Neumann (Eds.), *Proceedings of the 5th Conference on Robot Learning*, Vol. 164 of *Proceedings of Machine Learning Research* (pp. 740–749), 08–11 Nov, 2022: PMLR.
- Kapelyukh, I., Vosylius, V., & Johns, E. (2023). Dall-e-bot: Introducing web-scale diffusion models to robotics. *IEEE Robotics and Automation Letters*, , 2023.
- Kulshrestha, M., & Qureshi, A. H. (2023). Structural concept learning via graph attention for multi-level rearrangement planning. *7th Annual Conference on Robot Learning*, 2023.
- Kurenkov, A., Lingelbach, M., Agarwal, T., Jin, E., Li, C., Zhang, R., Fei-Fei, L., Wu, J., Savarese, S., & Martin-Martin, R. (2023). Modeling dynamic environments with scene graph memory. *International Conference on Machine Learning* (pp. 17976–17993), PMLR, 2023.
- Liu, B., Jiang, Y., Zhang, X., Liu, Q., Zhang, S., Biswas, J., & Stone, P. (2023a). Llm+p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, , 2023.
- Liu, W., Du, Y., Hermans, T., Chernova, S., & Paxton, C. (2023b). Structdiffusion: Language-guided creation of physically-valid structures using unseen objects. *RSS 2023*, 2023.

- Liu, W., Paxton, C., Hermans, T., & Fox, D. (2022). Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects. *2022 International Conference on Robotics and Automation (ICRA)* (pp. 6322–6329), IEEE, 2022.
- Lu, Y., Feng, W., Zhu, W., Xu, W., Wang, X. E., Eckstein, M., & Wang, W. Y. (2023). Neuro-symbolic procedural planning with commonsense prompting. *The Eleventh International Conference on Learning Representations*, 2023.
- Martello, S., & Toth, P. (1990). Lower bounds and reduction procedures for the bin packing problem. *Discrete applied mathematics*, 28(1), 1990, 59–70.
- Qureshi, A. H., Mousavian, A., Paxton, C., Yip, M. C., & Fox, D. (2021). Nerp: Neural rearrangement planning for unknown objects. *arXiv preprint arXiv:2106.01352*, , 2021.
- Rana, K., Haviland, J., Garg, S., Abou-Chakra, J., Reid, I., & Suenderhauf, N. (2023). Say-plan: Grounding large language models using 3d scene graphs for scalable task planning. *7th Annual Conference on Robot Learning*, 2023.
- Ravichandran, Z., Peng, L., Hughes, N., Griffith, J. D., & Carlone, L. (2022). Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. *2022 International Conference on Robotics and Automation (ICRA)* (pp. 9272–9279), IEEE, 2022.
- Ren, A. Z., Dixit, A., Bodrova, A., Singh, S., Tu, S., Brown, N., Xu, P., Takayama, L., Xia, F., Varley, J., Xu, Z., Sadigh, D., Zeng, A., & Majumdar, A. (2023). Robots that ask for help: Uncertainty alignment for large language model planners. *arXiv preprint arXiv:2307.01928*, 2023.
- Sarch, G., Fang, Z., Harley, A. W., Schydlo, P., Tarr, M. J., Gupta, S., & Fragkiadaki, K. (2022). Tidee: Tidying up novel rooms using visuo-semantic commonsense priors. *European Conference on Computer Vision* (pp. 480–496), Springer, 2022.
- Schuster, M. J., Jain, D., Tenorth, M., & Beetz, M. (2012). Learning organizational principles in human environments. *2012 IEEE International Conference on Robotics and Automation* (pp. 3867–3874), 2012.
- Wang, Z., Cai, S., Liu, A., Ma, X., & Liang, Y. (2023). Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*, , 2023.
- Wei, Q. A., Ding, S., Park, J. J., Sajnani, R., Poulenard, A., Sridhar, S., & Guibas, L. (2023). Lego-net: Learning regular rearrangements of objects in rooms. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 19037–19047), 2023.
- Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., & Funkhouser, T. (2023). Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*, , 2023.
- Wu, M., Zhong, F., Xia, Y., & Dong, H. (2022). Targf: Learning target gradient field to rearrange objects without explicit goal specification. *Advances in Neural Information Processing Systems*, 35, 2022, 31986–31999.

- Xie, Y., Yu, C., Zhu, T., Bai, J., Gong, Z., & Soh, H. (2023). Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, , 2023.
- Xu, D., Martín-Martín, R., Huang, D.-A., Zhu, Y., Savarese, S., & Fei-Fei, L. F. (2019). Regression planning networks. *Advances in Neural Information Processing Systems*, 32, 2019.
- Xu, Y., & Hsu, D. (2023). How to tidy up a table: Fusing visual and semantic commonsense reasoning for robotic tasks with vague objectives. *arXiv preprint arXiv:2307.11319*, , 2023.
- Yang, Z., Mao, J., Du, Y., Wu, J., Tenenbaum, J. B., Lozano-Pérez, T., & Kaelbling, L. P. (2023). Compositional Diffusion-Based Continuous Constraint Solvers. *Conference on Robot Learning*, 2023.
- Zhai, G., Cai, X., Huang, D., Di, Y., Manhardt, F., Tombari, F., Navab, N., & Busam, B. (2023). Sg-bot: Object rearrangement via coarse-to-fine robotic imagination on scene graphs. *arXiv preprint arXiv:2309.12188*, , 2023.
- Zhao, H., She, Q., Zhu, C., Yang, Y., & Xu, K. (2021). Online 3d bin packing with constrained deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35 (pp. 741–749), 2021.
- Zhao, Z., Lee, W. S., & Hsu, D. (2023). Large language models as commonsense knowledge for large-scale task planning. *arXiv preprint arXiv:2305.14078*, , 2023.
- Zhu, Y., Tremblay, J., Birchfield, S., & Zhu, Y. (2021). Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6541–6548), IEEE, 2021.

Appendix A

Example for Pattern Deduction

T You



Tell me something about the pattern of object organization that is shown in this image

ChatGPT

The objects on this bookshelf are arranged with a clear sense of order and aesthetic. There's a mix of decorative and functional items:

1. Decorative items include the elephant figurines, globe, and the ornamental plant to the right, which add a touch of personality and warmth.
2. Functional items are the books, camera, headphones, and binders, which suggest that this is an active, working space as well as a display area.
3. The shelves seem to be organized by item type, with books clustered together, decorative items paired or alone, and work-related items like binders and a headset grouped for easy access.

The top shelf is dedicated to display, with the elephants and globe being the focal points. The second shelf down features predominantly books, with a camera indicating an interest in photography. The third shelf contains work or study-related items like binders, suggesting organization and professionalism. Finally, on the fourth shelf, there's a balance between the books and decorative vases, implying a blend of utility and design.

Overall, the arrangement balances form and function, creating a pleasing yet practical space.



You

summarize the patterns by a list of self-explanatory pattern names



ChatGPT

Certainly, here's a list of pattern names that could summarize the organizational method displayed on the bookshelf:

1. **Zonal Segregation:** Each shelf has a specific category or type of items, creating distinct zones for different purposes.
2. **Decorative Balance:** Symmetrical placement of decorative items for aesthetic appeal.
3. **Functional Clustering:** Grouping of functionally similar items, like books or office supplies, for easy access.
4. **Vertical Rhythm:** The vertical arrangement of items creates a visual rhythm that leads the eye from top to bottom.
5. **Personal Showcase:** Display of personal items, like the camera and headphones, suggesting individual interests.
6. **Visual Anchoring:** Larger, more visually striking items (like the globe and plant) are placed at the ends or in the center to anchor the arrangement.
7. **Mixed Media Display:** Combination of various types of items (books, decorative objects, electronics) on the same shelf.
8. **Open Space Utilization:** Strategic use of open space around objects to avoid clutter and maintain a clean look.



You

represent each pattern mathematically as a function of object_type, object_shape, and object_pose



ChatGPT

Representing these patterns mathematically can be quite abstract, as the concepts of object type, shape, and pose are qualitative rather than quantitative. However, we can define some functions that might represent the patterns by assigning hypothetical numerical values to the attributes of each object. Here's a conceptual representation:



1. **Zonal Segregation:**

```
1 & \text{if object is decorative} \\
2 & \text{if object is a book} \\
3 & \text{if object is electronic} \\
4 & \text{if object is plant} \\
\end{cases} ]
```

Each object type is assigned to a particular zone (shelf).

2. Decorative Balance:

$$f_{\text{balance}}(\text{object_type}, \text{object_shape}) = \text{object_shape} \times \delta(\text{object_type}, \text{'decorative'})$$

Where δ is a Kronecker delta function that is 1 when the object is decorative and 0 otherwise.

3. Functional Clustering:

$$f_{\text{cluster}}(\text{object_type}) = \sum_{i=1}^n \delta(\text{object_type}, \text{'functional'})_i$$

Summing the occurrences of functional items to form clusters.

4. Vertical Rhythm:

$$f_{\text{rhythm}}(\text{object_pose}) = a \cdot \text{object_pose}_{\text{vertical}}$$

Where a is a constant that scales the pose value, representing the vertical placement.

5. Personal Showcase:

```
1 & \text{if object is personal} \\
0 & \text{otherwise} \\
\end{cases} ]
```

A binary function indicating the presence of personal items.

6. Visual Anchoring:

$$f_{\text{anchor}}(\text{object_shape}) = \max(\text{object_shape})$$

The larger the shape value, the stronger the anchoring effect.



7. Mixed Media Display:

$$f_{\text{mixed}}(\text{object_type}) = \text{entropy}(\text{object_types on shelf})$$

Entropy could represent the variety of object types on a shelf; the higher the entropy, the more mixed the shelf.

8. Open Space Utilization:

$$f_{\text{open}}(\text{object_shape}, \text{object_pose}) = \frac{\text{Total Space} - \sum \text{object_shape}}{\text{Total Space}} \cdot$$

$$\delta(\text{object_pose}, \text{'optimally positioned'})$$

The fraction of space not occupied by objects, multiplied by a factor that is 1 when objects are optimally positioned to use open space effectively.

Each function f attempts to capture the essence of the patterns described using object attributes. In reality, translating these patterns into mathematical functions would require more precise definitions and possibly more attributes to fully capture the complexity of spatial arrangements and design principles.



You

represent each pattern in pythonic functions



ChatGPT

! Error analyzing ▾

Always expand output?

python

 Copy code

```
# To represent these patterns in Pythonic functions, we would first need a way to
# For simplicity, let's assume that we can quantify object_type, object_shape, and
# We'll define a class to hold these properties for an object.

class ShelfObject:
    def __init__(self, object_type, object_shape, object_pose):
        self.object_type = object_type # Decorative, Book, Electronic, Plant, Personal
        self.object_shape = object_shape # A numerical representation of the object
        self.object_pose = object_pose # A numerical representation of the object's orientation

    # Example function for Zonal Segregation pattern
    def zonal_segregation(objects):
        zones = {'Decorative': 1, 'Book': 2, 'Electronic': 3, 'Plant': 4}
        return [zones[obj.object_type] for obj in objects]

    # Example function for Decorative Balance pattern
    def decorative_balance(objects):
        return [obj.object_shape for obj in objects if obj.object_type == 'Decorative']

    # Example function for Functional Clustering pattern
    def functional_clustering(objects):
        return len([1 for obj in objects if obj.object_type == 'Functional'])

    # Example function for Vertical Rhythm pattern
    def vertical_rhythm(objects):
        return [obj.object_pose for obj in objects]

    # Example function for Personal Showcase pattern
    def personal_showcase(objects):
        return [1 if obj.object_type == 'Personal' else 0 for obj in objects]

    # Example function for Visual Anchoring pattern
    def visual_anchoring(objects):
        return max([obj.object_shape for obj in objects])

    # Example function for Mixed Media Display pattern
    def mixed_media_display(objects):
        from collections import Counter
        object_types = [obj.object_type for obj in objects]
        type_count = Counter(object_types)
        # Entropy can be a measure of variety
        entropy = -sum((count / len(objects)) * math.log(count / len(objects))) for count in type_count.values()
        return entropy

    # Example function for Open Space Utilization pattern
    def open_space_utilization(objects, total_space):
```

```
used_space = sum(obj.object_shape for obj in objects)
optimal_positions = [obj.object_pose for obj in objects if obj.object_pose == 3]
return (total_space - used_space) / total_space * len(optimal_positions)

# We'll create a few example objects to demonstrate these functions.
example_objects = [
    ShelfObject('Decorative', 10, 'mid-level'),
    ShelfObject('Book', 5, 'low-level'),
    ShelfObject('Electronic', 7, 'high-level'),
    ShelfObject('Plant', 8, 'top-level'),
    ShelfObject('Personal', 6, 'mid-level')
]
```

Appendix B

Example for Geometrical Constraint Goal Proposal

T1

You

You are a helpful robot assistant. If you want to add an object to a shelf with existing objects, what are the procedures that you need to take?

ChatGPT

Adding an object to a shelf with existing objects involves a few careful steps to ensure that the shelf remains organized, and the objects are stored safely. Here are some procedures to follow:

- 1. Assess the Space:** Before adding an object, check the available space on the shelf to ensure there is enough room for the new item without overcrowding.
- 2. Consider the Weight:** Evaluate the weight of the new object and the capacity of the shelf. Placing too heavy items on a weaker shelf could risk damage or collapse.
- 3. Organize by Relevance or Size:** Depending on the nature of the items, you might want to organize them by size, type, or frequency of use. For example, heavier items usually go on lower shelves, and items used more frequently should be more accessible.
- 4. Rearrange if Necessary:** If the existing objects need to be rearranged to make space or to maintain balance and aesthetics, do so carefully. This might involve temporarily removing some items to better organize them.
- 5. Clean the Shelf and Objects:** Before placing the new object, it might be a good opportunity to dust the shelf and clean the items that are already there.
- 6. Place the New Object:** Once you've made the space and considered the arrangement, carefully place the new object on the shelf. Make sure it is stable and not at risk of falling.
- 7. Final Check for Stability and Accessibility:** After placing the object, ensure that the shelf isn't overloaded and that all items, including the newly added one, are stable and easily accessible.
- 8. Maintain Regular Checks:** Periodically check the shelf for any signs of stress or overloading and reorganize if necessary to ensure ongoing safety and convenience.

These steps help in maintaining an organized, safe, and aesthetically pleasing shelf.



T1

You





How to move the object on the top layer of the shelf to the bottom layer?



ChatGPT

To move the object from the top layer to the bottom layer of the shelf, follow these steps:

- 1. Prepare the Area:** Ensure the bottom shelf area is clear of any obstacles that could hinder the placement of the object or cause it to be unstable once placed.
- 2. Ensure Stability:** Make sure that when you take the object from the top shelf, other items on the same shelf are stable and won't fall or shift dangerously.
- 3. Handle with Care:** Carefully reach for the object on the top shelf. Use both hands if necessary to secure a firm grip, especially if the item is heavy or fragile.
- 4. Move the Object:** Gently bring the object down, being mindful of the distance to the bottom shelf to avoid any collisions with other objects.
- 5. Place on the Bottom Shelf:** Upon reaching the bottom shelf, carefully place the object in the desired spot, ensuring it is stable and won't easily tip over. Adjust other items if necessary to make room.
- 6. Check for Balance:** Make a final check for the balance and stability of both shelves, ensuring that the weight is evenly distributed and there's no risk of anything falling.
- 7. Final Adjustments:** If any of the other items have been disturbed or if there's an opportunity to better organize the bottom shelf, make those adjustments now.

Remember to lift with your legs, not with your back, to avoid strain or injury, especially if the object is heavy.



You

refer to the image, tell me exactly how to move the existing objects and where the new object should be placed by indicating its relative position with the existing objects



ChatGPT

In the image, there are two shelves with various items. To move the object on the top shelf (a bag of what appears to be nuts) to the bottom shelf, you would do the following:

- 1. Clear a Space:** Begin by making space on the bottom shelf. You might want to temporarily move the two bottles on the left closer to the box on the right to create an open area on the left side of the bottom shelf.
- 2. Secure the Object:** Gently grab the bag from the top shelf. Be careful as shelves might be high, and the item might be heavier than it appears or could have contents that shift.
- 3. Move the Object Down:** Carefully bring the bag down without disrupting the other items on the top shelf.
- 4. Place the Object:** On the bottom shelf, place the bag to the left of the two bottles you moved earlier, or between the two bottles and the box, whichever fits best. Ensure the bag is upright if possible, to prevent its contents from spilling.
- 5. Readjust Surrounding Items:** Adjust the bottles and the box to ensure everything sits securely on the shelf and there's no risk of items tipping over. The bag should be positioned in such a way that it does not impede the view or access to the other items.
- 6. Final Inspection:** Check the stability of the items on both the top and bottom shelves to ensure everything is safe and secure after the rearrangement.

T1

You

represent the moving by the (object_name, moving direction, moving goal), and the final configuration in the following format: We define a language representing relationship indicating object positions in the 2d space in the following format:

LeftMost(A, B) means A, B are on the same object and A is to the left of B, and there is no object between them.

Below(A, B) means A is below B and objec'  are in contact, which means when moving A, object B would be affected.

If multiple objects A1, A2, ..., An are placed on top of the same object B, we represent the relationship between object Ax and object B separately as: Below(B, Ax), and LeftMost(A_i, A_{i+1}).

If a bigger object A is placed on top of multiple smaller objects B1, B2 ... Bn sorted from left to right, and we include object Bx as long as A is in contact with Bx. We represent the relationship as Below(Left(B1, B2 ... Bn), A).



ChatGPT

From the image provided and the task to move the bag of nuts from the top shelf to the bottom shelf, the operation and final configuration can be described as follows:

Moving Operation:

- (Bag_of_Nuts, Downward, Bottom_Shelf)

Final Configuration:

- LeftMost(Water_Bottle_1, Water_Bottle_2)
- LeftMost(Water_Bottle_2, Bag_of_Nuts)
- LeftMost(Bag_of_Nuts, Box)
- LeftMost(Box, Pineapple_Roll)
- Below(Top_Shelf, Water_Bottle_1)
- Below(Top_Shelf, Water_Bottle_2)
- Below(Top_Shelf, Box)
- Below(Top_Shelf, Pineapple_Roll)



This final configuration assumes that the bag of nuts is placed on the bottom shelf to the left of the box but to the right of the two water bottles, with no other items in between these. It's also assumed that the bag of nuts does not need to support any weight from above and is not in direct contact with the top shelf.



You

are you sure you only need to move the nuts?



ChatGPT

Apologies for the oversight. Based on the image provided and the constraints of the language for representing object positions, the moving operation for the bag of nuts indeed necessitates an adjustment of the other items on the bottom shelf to accommodate it. Here's the correct sequence of moves and the final configuration:

Moving Operations:

- (Water_Bottle_1, Rightward, Next_to_Water_Bottle_2)
- (Bag_of_Nuts, Downward, Bottom_Shelf_Between_Bottles_and_Box)

Final Configuration:

- LeftMost(Water_Bottle_1, Water_Bottle_2)
- LeftMost(Water_Bottle_2, Bag_of_Nuts)
- LeftMost(Bag_of_Nuts, Box)
- LeftMost(Box, Pineapple_Roll)
- Below(Top_Shelf, Pineapple_Roll)
- Below(Bottom_Shelf, Water_Bottle_1)
- Below(Bottom_Shelf, Water_Bottle_2)
- Below(Bottom_Shelf, Bag_of_Nuts)
- Below(Bottom_Shelf, Box)

- Below(Bottom_Shelf, Pineapple_Roll)

After moving Water Bottle 1 to the right so that it is next to Water Bottle 2, the bag of nuts from the top shelf can be placed on the bottom shelf between Water Bottle 2 and the Box. All the objects are assumed to be in contact with the bottom shelf, and their left-to-right order is preserved as described.



Appendix C

Example for Stability Testing



How to stack these objects together? We define the current orientation of the objects as the upright position. Please identify the goal pose by their relative position and orientation



ChatGPT

To stack these two items — the bottle of "YOU-C10 Vitamin Apple" and the can of "Yeo's Soy Bean Drink" — with their current orientation defined as upright, you can proceed as follows:

1. Goal Pose:

- The bottle should be on top of the can.
- The bottom of the bottle will be in contact with the top of the can.
- The central axis of the bottle should align with the central axis of the can, so the bottle is balanced and the stack is stable.

2. Steps:

- Start by securely placing the can on a stable, flat surface to ensure it doesn't tip over easily.
- Carefully align the bottle above the can, entering it as best as possible.
- Slowly lower the bottle onto the can so that the bottom of the bottle makes contact with the top end of the can.
- Adjust the bottle so that it's as centered as possible to avoid it tipping over.

Please note that stacking rounded items like these can be unstable and may easily fall over, so do this with caution. If you intend for this to be a display or for practical use, it may be advisable to secure them with temporary adhesive or putty if appropriate.



Is this conversation helpful so far?

Appendix D

Sample Images Generated by LLM Arrangement

T

You

You are a helpful robot assistant. There are 4 even cells with width 80cm, height 60cm, in a shelf inside a kitchen, here are a list of objects, help me assign each object to a cell of the shelf: [{'type': 'Sugar bowl', 'width': 10, 'height': 8, 'number': 1}, {'type': 'Glass pitcher', 'width': 15, 'height': 25, 'number': 1}, {'type': 'Spice jar', 'width': 5, 'height': 8, 'number': 3}, {'type': 'Oven mitt', 'width': 15, 'height': 30, 'number': 3}, {'type': 'Bowl', 'width': 20, 'height': 8, 'number': 3}, {'type': 'Whisk', 'width': 5, 'height': 25, 'number': 1}, {'type': 'Wine rack', 'width': 40, 'height': 30, 'number': 3}]



C

ChatGPT

To efficiently assign each object to a cell on the shelf, we'll need to consider both the dimensions of each cell and the dimensions and quantities of the items. Let's start by listing out the total space each type of item will occupy based on their quantity and then we can plan how to fit them into the cells.

Here's the total space each item type will require:

1. **Sugar bowl:** 1 x (10cm width, 8cm height) = 10cm width total, 8cm height total
2. **Glass pitcher:** 1 x (15cm width, 25cm height) = 15cm width total, 25cm height total
3. **Spice jar:** 3 x (5cm width, 8cm height) = 15cm width total, 8cm height total
4. **Oven mitt:** 3 x (15cm width, 30cm height) = 45cm width total, 30cm height total
5. **Bowl:** 3 x (20cm width, 8cm height) = 60cm width total, 8cm height total
6. **Whisk:** 1 x (5cm width, 25cm height) = 5cm width total, 25cm height total
7. **Wine rack:** 3 x (40cm width, 30cm height) = 120cm width total, 30cm height total

Each shelf cell is 80cm wide and 60cm high. Let's plan the distribution:



- **Cell 1:** Can hold items with smaller footprints. Place the Sugar bowl, Glass pitcher, and Whisk here. Total width = $10 + 15 + 5 = 30$ cm. All fit well within the width limit.
- **Cell 2:** Place the Spice jars here. Total width = 15cm. Fits easily and leaves plenty of space for access or other smaller items if needed.
- **Cell 3:** Place one Wine rack. This will fill the width of the shelf cell, using up 40cm of the 80cm width, and will fill 30cm of the 60cm height.
- **Cell 4:** As the Wine rack and Oven mitts are both large, you might need to split these among more cells. Place two Oven mitts here which takes up 30cm width x 30cm height, filling the cell effectively with 60cm width used.

This arrangement places some of the larger and more used items in separate cells for easier access, while optimizing the use of space in each cell.



T

You

For each cell, mark object name as object_type_index, assign an object to a position by outputting the coordinate of the down left corner of the rectangle, define the coordinate as (x, y) where x is the horizontal distance from the down left corner of the cell, and y is the vertical distance from the

same corner



ChatGPT

To assign each object to a position within the cells with coordinates marking the bottom-left corner of the rectangle for each item, we need to allocate the positions based on the total dimensions and previously determined organization. Here are the specific placements for each item:



Cell 1:

- **Sugar_bowl_1**: Place at the bottom-left corner of the cell. Coordinate: (0, 0)
- **Glass_pitcher_1**: Place next to the Sugar bowl. Coordinate: (10, 0) (since the sugar bowl takes up 10cm)
- **Whisk_1**: Place next to the Glass pitcher. Coordinate: (25, 0) (since the Glass pitcher ends at 25cm from the left)

Cell 2:

- **Spice_jar_1**: Place at the bottom-left corner. Coordinate: (0, 0)
- **Spice_jar_2**: Place next to the first Spice jar. Coordinate: (5, 0)
- **Spice_jar_3**: Place next to the second Spice jar. Coordinate: (10, 0)

Cell 3:

- **Wine_rack_1**: Place at the bottom-left corner. Coordinate: (0, 0)

Cell 4:

- **Oven_mitt_1**: Place at the bottom-left corner. Coordinate: (0, 0)
- **Oven_mitt_2**: Place above the first Oven mitt (vertically aligned). Coordinate: (0, 30) (since the height of one Oven mitt is 30cm)
- **Oven_mitt_3**: Place next to the first Oven mitt. Coordinate: (15, 0) (since the width of one Oven mitt is 15cm)

This arrangement effectively utilizes the space within each cell while ensuring that items are not overlapping and are organized for easy access.



You

This shelf has 2 rows and 2 columns convert the cell_name index with the column and row number



ChatGPT

Given that the shelf is organized into 2 rows and 2 columns, we can reassign the items based on this configuration. The cells can be labeled as follows:

- **Cell 1**: Top left (Row 1, Column 1)
- **Cell 2**: Top right (Row 1, Column 2)

- **Cell 3:** Bottom left (Row 2, Column 1)
- **Cell 4:** Bottom right (Row 2, Column 2)

Here's how each item can be positioned within these cells, keeping the same assignments but updating the labels:

Row 1, Column 1 (Top Left):

- **Sugar_bowl_1:** Coordinate: (0, 0)
- **Glass_pitcher_1:** Coordinate: (10, 0)
- **Whisk_1:** Coordinate: (25, 0)

Row 1, Column 2 (Top Right):

- **Spice_jar_1:** Coordinate: (0, 0)
- **Spice_jar_2:** Coordinate: (5, 0)
- **Spice_jar_3:** Coordinate: (10, 0)

Row 2, Column 1 (Bottom Left):



- **Wine_rack_1:** Coordinate: (0, 0)

Row 2, Column 2 (Bottom Right):

- **Oven_mitt_1:** Coordinate: (0, 0)
- **Oven_mitt_2:** Coordinate: (0, 30)
- **Oven_mitt_3:** Coordinate: (15, 0)

This configuration maps each item to a specific position within the cells labeled according to their row



You

continue



ChatGPT

and column in the shelf structure. Here is the continuation of the configuration:

Row 2, Column 2 (Bottom Right) continued:

- **Bowl_1:** Position next to Oven_mitt_3. Coordinate: (30, 0) (since Oven_mitt_3 uses 15cm width and starts at 15cm)
- **Bowl_2:** Above Bowl_1. Coordinate: (30, 8) (as each bowl is 8cm tall)
- **Bowl_3:** Next to Bowl_2. Coordinate: (50, 8) (using 20cm width for each bowl)

This completes the arrangement of all items on the shelf. Each object is placed in such a way that it utilizes the space efficiently within the constraints of each cell's width and height, respecting the needs for access and organization.



**You**

create a json file with the following {cell_row, cell_column, [{item_name, position}]}

**ChatGPT**

I've created the JSON file with the shelf layout as specified. You can download it using the link below:

[Download Shelf Layout JSON \[→\]](#)

**You**

convert to json formatwith the following {cell_row, cell_column, [{item_name, position}]}

**ChatGPT**

Here's the JSON data in the requested format for the shelf layout:

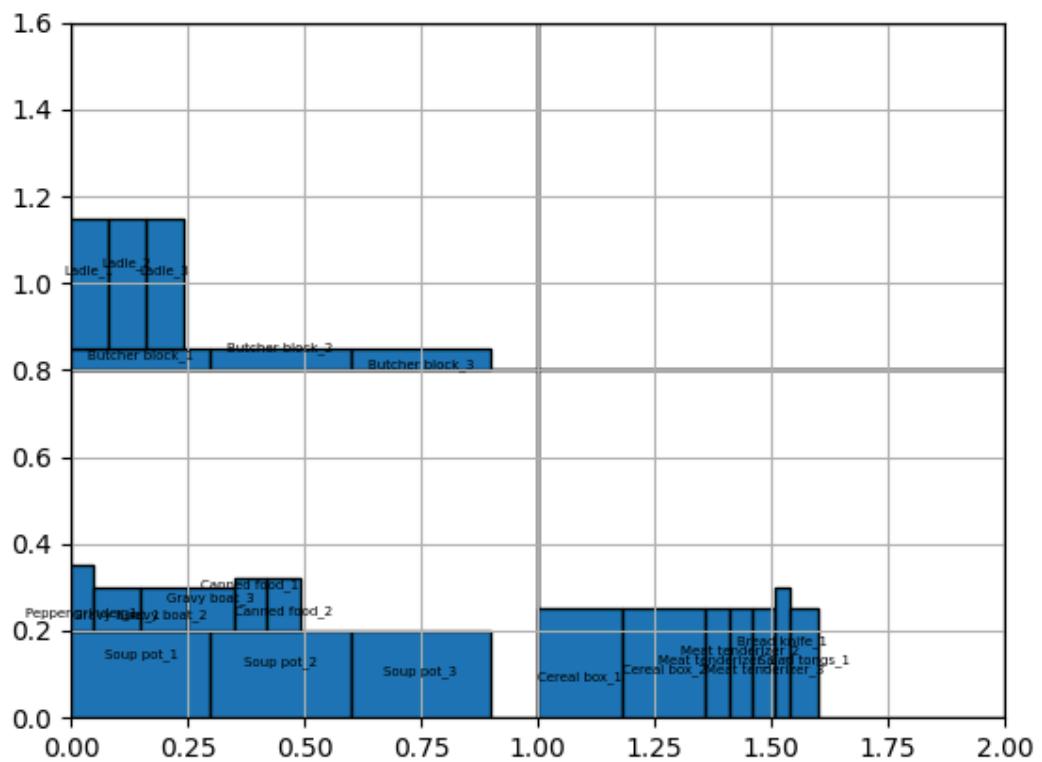
```
json
Copy code

{
  "shelf": [
    {
      "cell_row": 1,
      "cell_column": 1,
      "items": [
        {"item_name": "Sugar_bowl_1", "position": "(0, 0)"}, ↓
        {"item_name": "Glass_pitcher_1", "position": "(10, 0)"}, 
        {"item_name": "Whisk_1", "position": "(25, 0)"}
      ]
    },
    {
      "cell_row": 1,
      "cell_column": 2,
      "items": [
        {"item_name": "Spice_jar_1", "position": "(0, 0)"}, 
        {"item_name": "Spice_jar_2", "position": "(5, 0)"}, 
        {"item_name": "Spice_jar_3", "position": "(10, 0)"}
      ]
    },
    {
      "cell_row": 2,
      "cell_column": 1,
      "items": [
        {"item_name": "Wine_rack_1", "position": "(0, 0)"}
      ]
    },
    {
      "cell_row": 2,
```

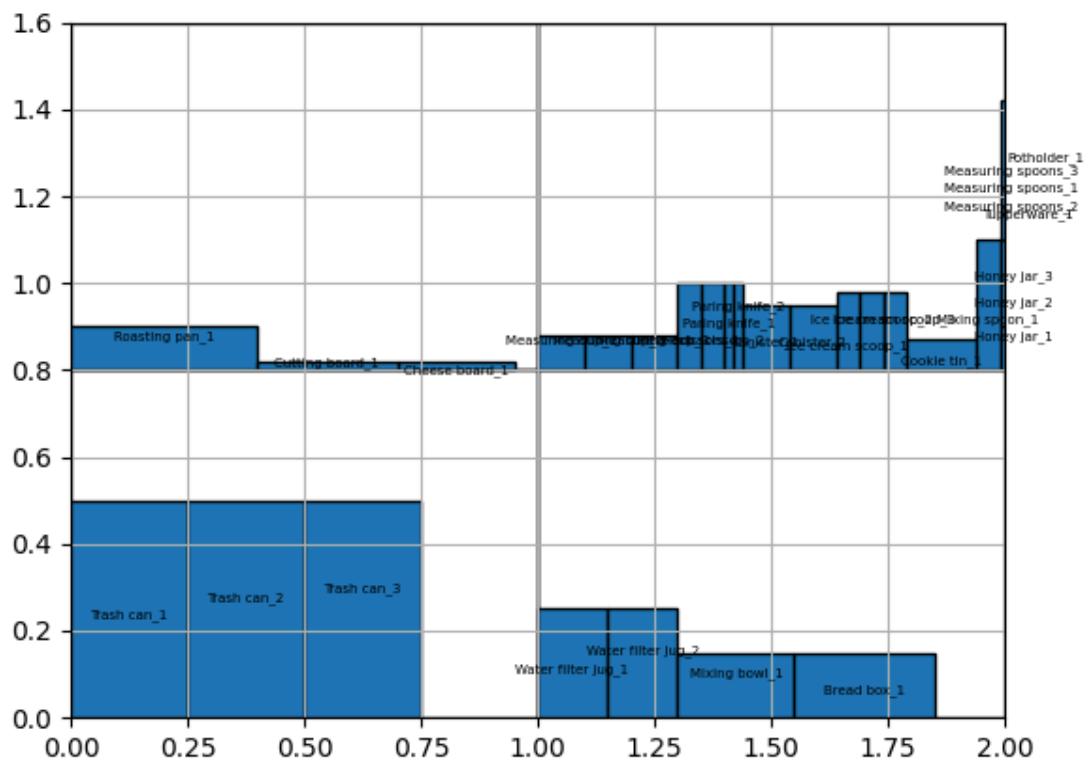
```
        "cell_column": 2,  
        "items": [  
            {"item_name": "Oven_mitt_1", "position": "(0, 0)"},  
            {"item_name": "Oven_mitt_2", "position": "(0, 30)"},  
            {"item_name": "Oven_mitt_3", "position": "(15, 0)"},  
            {"item_name": "Bowl_1", "position": "(30, 0)"},  
            {"item_name": "Bowl_2", "position": "(30, 8)"},  
            {"item_name": "Bowl_3", "position": "(50, 8)"}  
        ]  
    }  
]  
}
```

This JSON structure represents the placement of various items in a 2×2 shelf with each item's name and position specified in a tuple format for clarity.

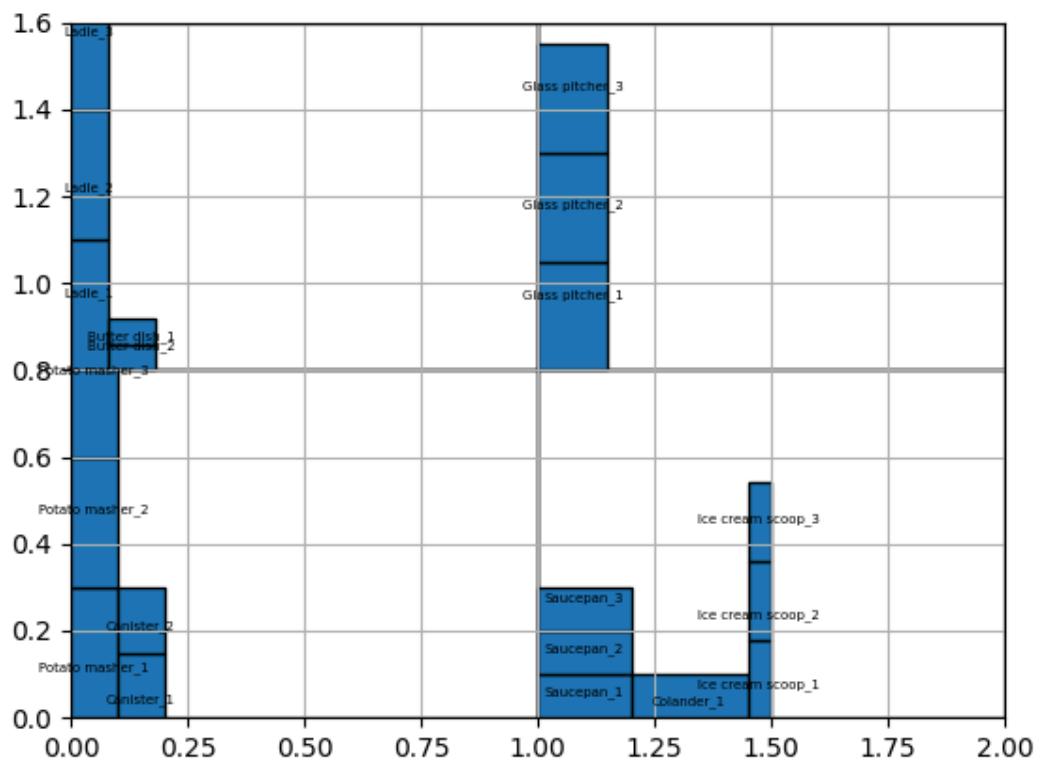




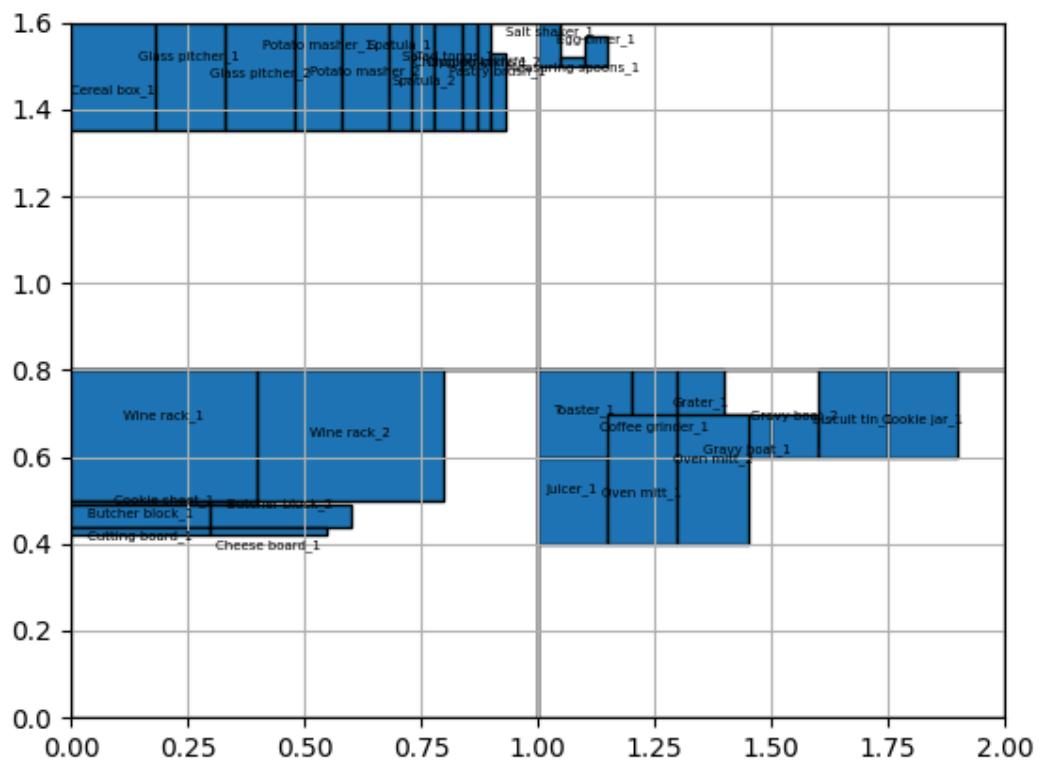
Failure:



Violate Geometrical Constraints(Not enough space)



Uneven arrangement / doesn't comply with commonsense



Not Physically Valid

Appendix E

Sample Images Generated by Relationship Proposal

You

You are a helpful robot assistant. There are 4 even cells in a shelf inside a kitchen, here are a list of objects, we assume that objects can't occlude each other and should be fully visible when looking from the front view, help me assign each object to a cell of the shelf: [{type: 'Cookie sheet', 'width': 40, 'height': 1, 'number': 1}, {type: 'Coffee grinder', 'width': 10, 'height': 20, 'number': 2}, {type: 'Ice cream scoop', 'width': 5, 'height': 18, 'number': 2}, {type: 'Bread knife', 'width': 3, 'height': 30, 'number': 1}, {type: 'Bread box', 'width': 30, 'height': 15, 'number': 2}, {type: 'Butcher block', 'width': 30, 'height': 5, 'number': 1}, {type: 'Honey jar', 'width': 6, 'height': 10, 'number': 3}, {type: 'Cake server', 'width': 5, 'height': 25, 'number': 3}].



ChatGPT

To effectively assign each object to a shelf cell in a way that optimizes the available space and ensures that the objects are fully visible, we should consider the width and height of each object relative to the dimensions of the shelf cells.

We will assume the following for each shelf cell:

- Sufficient height to accommodate the tallest object (at least 30 cm based on the bread knife's height).
- Sufficient width to accommodate the objects together or individually without exceeding the total shelf width.

Given this, let's proceed by categorizing the objects by their number, size, and type to find an optimal placement strategy.

Object List:

1. **Cookie sheet** - 1 unit, 40 cm width, 1 cm height
2. **Coffee grinder** - 2 units, 10 cm width, 20 cm height
3. **Ice cream scoop** - 2 units, 5 cm width, 18 cm height
4. **Bread knife** - 1 unit, 3 cm width, 30 cm height
5. **Bread box** - 2 units, 30 cm width, 15 cm height
6. **Butcher block** - 1 unit, 30 cm width, 5 cm height
7. **Honey jar** - 3 units, 6 cm width, 10 cm height
8. **Cake server** - 3 units, 5 cm width, 25 cm height

Shelf Organization Strategy:

- The first step is to consider placing the widest and largest objects first. The cookie sheet and bread box take significant width.
- Balance out the number of objects per shelf.

- Place objects with similar numbers together if possible to create a visually balanced and easily accessible arrangement.

Assignment Proposal:

Shelf 1:

- Bread knife (3 cm, 30 cm) - tall and narrow, fits well without taking much width.
- Bread box (30 cm, 15 cm) - the larger object, using much of the width.

Shelf 2:

- Cookie sheet (40 cm, 1 cm) - quite wide and low height; it can lie flat.

Shelf 3:



- Coffee grinder x2 (10 cm each, 20 cm) - these together take 20 cm width.
- Ice cream scoop x2 (5 cm each, 18 cm) - these together add another 10 cm width.
- Butcher block (30 cm, 5 cm) - not very high, can fit alongside others.

Shelf 4:

- Honey jar x3 (6 cm each, 10 cm) - total of 18 cm width.
- Cake server x3 (5 cm each, 25 cm) - total of 15 cm width, also quite tall.

This layout ensures that all objects are visible from the front, utilizes the space efficiently, and objects of similar types or needs are grouped together. If there are specific size constraints per shelf or other considerations, adjustments may be needed.

T1

You

We need to describe the object placement by assuming each object in a rectangular box and indicating the object relative relationships in the 2d space. We define a language representing relationship indicating object positions in the 2d space in the following format:

LeftMost(A, B) means A, B are on the same object and A is to the left of B, and there is no object between them.

Below(A, B) means A is below B and object A,B are in contact, which means when moving A, object B would be affected.

If multiple objects A1, A2, ..., An are placed on top of the same object B, we represent the relationship between object Ax and object B separately as: Below(B, Ax), and LeftMost(A_i, A_{i+1}).

If a bigger object A is placed on top of multiple smaller objects B1, B2 ... Bn sorted from left to right, and we include object Bx as long as A is in contact with Bx. We represent the relationship as Below(Left(B1, B2 ... Bn), A).

For this task, mark each different item for the same type with an index, represent each object as

type_index.



ChatGPT

Based on your description, we will define the 2D spatial relationships for the objects on each shelf using the provided formal language. Each object is assigned an index to differentiate between multiple instances of the same type.

Object Naming:

- **Cookie sheet** - CookieSheet_1
- **Coffee grinder** - CoffeeGrinder_1, CoffeeGrinder_2
- **Ice cream scoop** - IceCreamScoop_1, IceCreamScoop_2
- **Bread knife** - BreadKnife_1
- **Bread box** - BreadBox_1, BreadBox_2
- **Butcher block** - ButcherBlock_1
- **Honey jar** - HoneyJar_1, HoneyJar_2, HoneyJar_3
- **Cake server** - CakeServer_1, CakeServer_2, CakeServer_3

Spatial Relationship Descriptions:

Shelf 1:

- **LeftMost(BreadKnife_1, BreadBox_1)** - The bread knife is to the left of the bread box, and there are no objects between them.

Shelf 2:



- **LeftMost(CookieSheet_1)** - The cookie sheet occupies the entire shelf space.

Shelf 3:

- **LeftMost(CoffeeGrinder_1, CoffeeGrinder_2)** - The first coffee grinder is to the left of the second one, no objects between them.
- **LeftMost(CoffeeGrinder_2, IceCreamScoop_1)** - The second coffee grinder is to the left of the first ice cream scoop.
- **LeftMost(IceCreamScoop_1, IceCreamScoop_2)** - The first ice cream scoop is to the left of the second one.
- **LeftMost(IceCreamScoop_2, ButcherBlock_1)** - The second ice cream scoop is to the left of the butcher block.

Shelf 4:

- **LeftMost(HoneyJar_1, HoneyJar_2)** - The first honey jar is to the left of the second one.

- **LeftMost(HoneyJar_2, HoneyJar_3)** - The second honey jar is to the left of the third one.
- **LeftMost(HoneyJar_3, CakeServer_1)** - The third honey jar is to the left of the first cake server.
- **LeftMost(CakeServer_1, CakeServer_2)** - The first cake server is to the left of the second one.
- **LeftMost(CakeServer_2, CakeServer_3)** - The second cake server is to the left of the third one.

This layout uses a formal language to describe the relative positioning of each object on the shelves, ensuring that each placement and sequence is clearly defined in terms of their left-to-right arrangement on the shelves.



You

The shelf has 2 rows and 2 columns, assign each cell to an appropriate position on this shelf



ChatGPT

Given that the shelf has two rows and two columns, creating a total of four cells (each cell can be considered as a section of the shelf), let's assign each object or group of objects to a specific cell. The cells can be designated as follows:

- Top-Left (TL)
- Top-Right (TR)
- Bottom-Left (BL)
- Bottom-Right (BR)

We will rearrange the objects to fit into these cells, considering the width and height of each object to maintain visibility and access.

Revised Assignment with Rows and Columns:

Top-Left (TL) Cell:

- **Objects:** Bread Knife (BreadKnife_1), Coffee Grinders (CoffeeGrinder_1, CoffeeGrinder_2)
- **Spatial Description:**
 - LeftMost(BreadKnife_1, CoffeeGrinder_1)
 - LeftMost(CoffeeGrinder_1, CoffeeGrinder_2)

Top-Right (TR) Cell:

- **Objects:** Butcher Block (ButcherBlock_1), ↓ Cream Scoops (IceCreamScoop_1, IceCreamScoop_2)
- **Spatial Description:**
 - LeftMost(ButcherBlock_1, IceCreamScoop_1)
 - LeftMost(IceCreamScoop_1, IceCreamScoop_2)

Bottom-Left (BL) Cell:

- **Object:** Cookie Sheet (CookieSheet_1)
- **Spatial Description:**
 - LeftMost(CookieSheet_1) - Occupies the entire cell width

Bottom-Right (BR) Cell:

- **Objects:** Honey Jars (HoneyJar_1, HoneyJar_2, HoneyJar_3), Cake Servers (CakeServer_1, CakeServer_2, CakeServer_3)
- **Spatial Description:**
 - LeftMost(HoneyJar_1, HoneyJar_2)
 - LeftMost(HoneyJar_2, HoneyJar_3)
 - Below(HoneyJar_3, CakeServer_1)
 - LeftMost(CakeServer_1, CakeServer_2)
 - LeftMost(CakeServer_2, CakeServer_3)

This layout optimizes space while ensuring the items are organized neatly and are easily accessible. The placement allows each type of object to fit comfortably in the assigned cells, considering their dimensions and the number of units for each type.



TI

You

Represent each cell as {Cell_index, row, column, [a list of the relations defined above sorted from left to right, down to up]}



ChatGPT

Let's precisely define each shelf cell using the format you specified. The cell index will be a sequential number starting from 1, and we'll specify the row and column for each, along with the relationships of the objects within that cell. Each list of relationships will be sorted from left to right and from bottom to top.

Detailed Cell Representation:

Cell 1: Top-Left (TL)

- **Index: 1, Row: 1, Column: 1**
- **Content:**
 - `'[LeftMost(BreadKnife_1, CoffeeGrinder_1), LeftMost(CoffeeGrinder_1, CoffeeGrinder_2)]`

Cell 2: Top-Right (TR)

- **Index: 2, Row: 1, Column: 2**
- **Content:**
 - `'[LeftMost(ButcherBlock_1, IceCreamScoop_1), LeftMost(IceCreamScoop_1, IceCreamScoop_2)]`

Cell 3: Bottom-Left (BL)



- **Index: 3, Row: 2, Column: 1**
- **Content:**
 - `'[LeftMost(CookieSheet_1)]`

Cell 4: Bottom-Right (BR)

- **Index: 4, Row: 2, Column: 2**
- **Content:**
 - `'[LeftMost(HoneyJar_1, HoneyJar_2), LeftMost(HoneyJar_2, HoneyJar_3), Below(HoneyJar_3, CakeServer_1), LeftMost(CakeServer_1, CakeServer_2), LeftMost(CakeServer_2, CakeServer_3)]`

Each cell description provides clarity on the spatial arrangement of the objects within that section of the shelf, following the specified language for representing object positions in 2D space. This format supports understanding how items are organized visually and spatially within the shelf structure.

< 1 / 2 >



You

Refine the object arrangement as follows: We need to describe the object placement by assuming each object in a rectangular box and indicating the object relative relationships in the 2d space. We define a language representing relationship indicating object positions in the 2d space in the following format:

LeftMost(A, B) means A, B are on the same object and A is to the left of B, and there is no object between them.

Below(A, B) means A is below B and object A,B are in contact, which means when moving A, object B would be affected.



If multiple objects A1, A2, ..., An are placed on top of the same object B, we represent the relationship between object Ax and object B separately as: Below(B, Ax), and LeftMost(A_i, A_{i+1}).

If a bigger object A is placed on top of multiple smaller objects B1, B2 ... Bn sorted from left to right, and we include object Bx as long as A is in contact with Bx. We represent the relationship as Below(Left(B1, B2 ... Bn), A).

For this task, mark each different item for the same type with an index, represent each object as type_index.

Represent each cell as {Cell_index, row, column, [a list of the relations defined above sorted from left to right, down to up]} in json format



ChatGPT

Based on your request for a more refined JSON representation that follows the specified spatial language rules, we will redefine the object placement within the shelf's cells. The format will explicitly detail the relative relationships of objects in a rectangular box, using indices for each type.

Updated Object Placement:

Cell 1: Top-Left (TL)

- **Index: 1, Row: 1, Column: 1**
- **Content:**
 - Objects: BreadKnife_1, CoffeeGrinder_1, CoffeeGrinder_2
 - Relationships: `[[{"LeftMost": ["BreadKnife_1", "CoffeeGrinder_1"]}, {"LeftMost": ["CoffeeGrinder_1", "CoffeeGrinder_2"]}]]`



Cell 2: Top-Right (TR)

- **Index: 2,



You

Refine the object arrangement as follows: We need to describe the object placement by assuming each object in a rectangular box and indicating the object relative relationships in the 2d space. We define a language representing relationship indicating object positions in the 2d space in the following format:

LeftMost(A, B) means A, B are on the same object and A is to the left of B, and there is no object between them.

Below(A, B) means A is below B and object A,B are in contact, which means when moving A, object B would be affected.

If multiple objects A1, A2, ..., An are placed on top of the same object B, we represent the relationship between object Ax and object B separately as: Below(B, Ax), and LeftMost(A_i, A_{i+1}).

If a bigger object A is placed on top of multiple smaller objects B1, B2 ... Bn sorted from left to right, and we include object Bx as long as A is in contact with Bx. We represent the relationship as Below(Left(B1, B2 ... Bn), A).

For this task, mark each different item for the same type with an index, represent each object as type_index.

Represent each cell as a list of python string with the following information: [Cell_index, row_number, column_number, [a list of the relations defined above sorted from left to right, down

to up]]



ChatGPT

To create a refined representation of object arrangement on the shelf with the specified relationships and using indices for each type, let's prepare the format as requested. Each cell will be described with its index, row, and column position, and the relationships among the items in the cell will be detailed. We'll adjust the indices and relationships based on the current understanding of the shelf configuration.

Cell Description in Python List Format:

python

Copy code

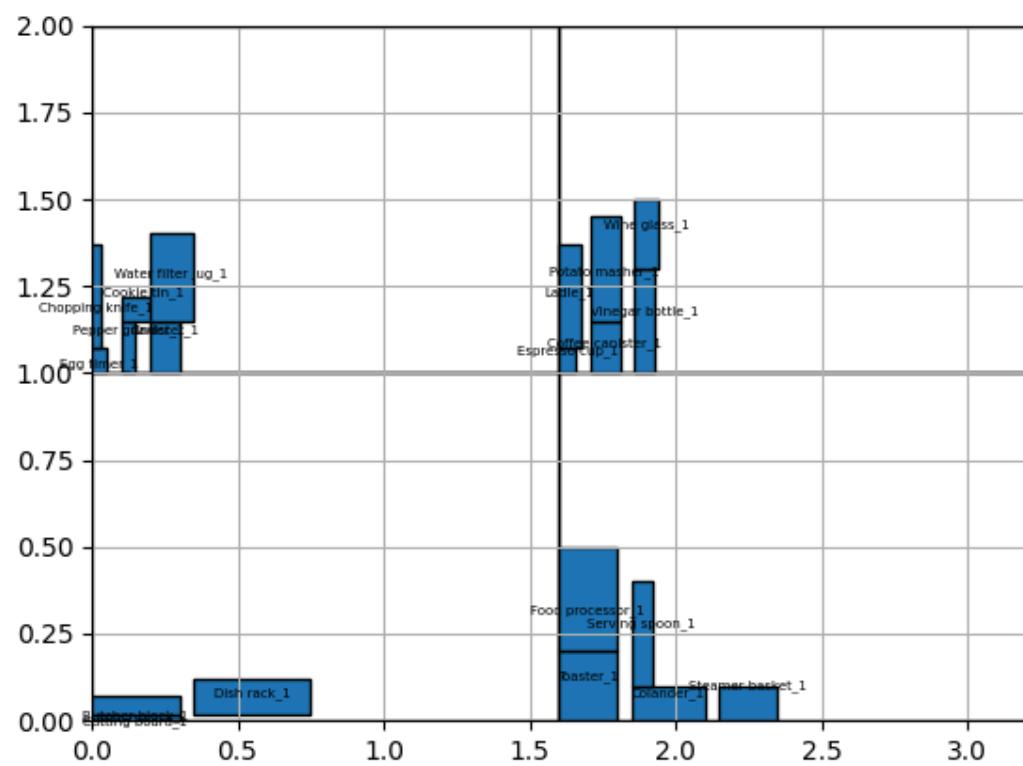
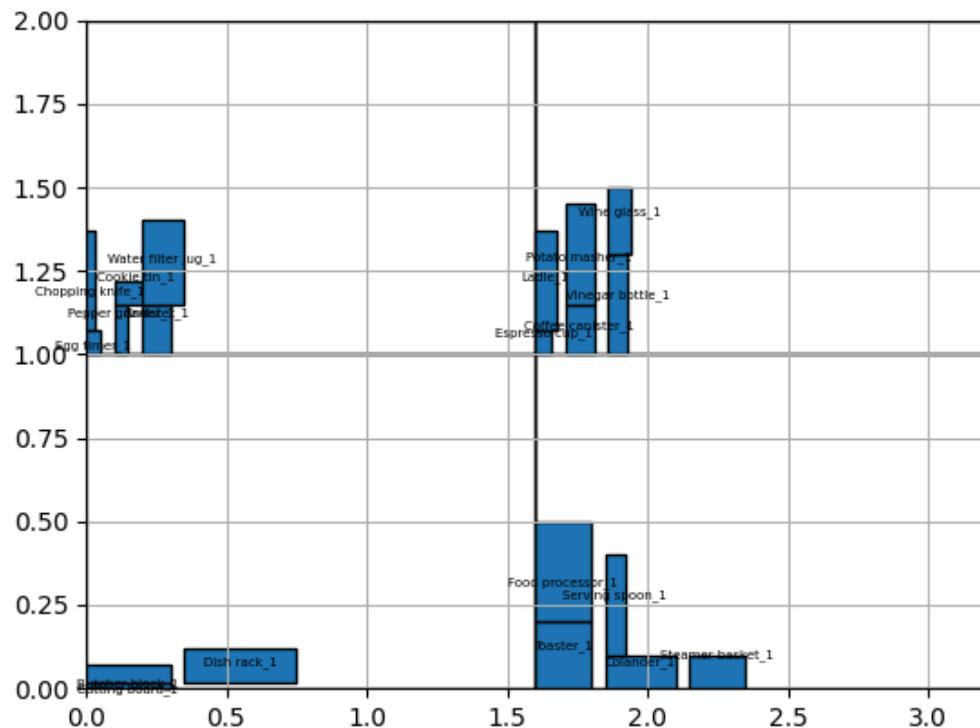
```
[  
  ["1", "1", "1", [  
    "LeftMost(BreadKnife_1, CoffeeGrinder_1)",  
    "LeftMost(CoffeeGrinder_1, CoffeeGrinder_2)"  
  ]],  
  ["2", "1", "2", [  
    "LeftMost(ButcherBlock_1, IceCreamScoop_1)",  
    "LeftMost(IceCreamScoop_1, IceCreamScoop_2)"  
  ]],  
  ["3", "2", "1", [  
    "LeftMost(CookieSheet_1)"  
  ]],  
  ["4", "2", "2", [  
    "LeftMost(HoneyJar_1, HoneyJar_2)",  
    "LeftMost(HoneyJar_2, HoneyJar_3)",  
    "Below(HoneyJar_3, CakeServer_1)",  
    "LeftMost(CakeServer_1, CakeServer_2)",  
    "LeftMost(CakeServer_2, CakeServer_3)"  
  ]]  
]
```

Each list item describes a cell on the shelf:

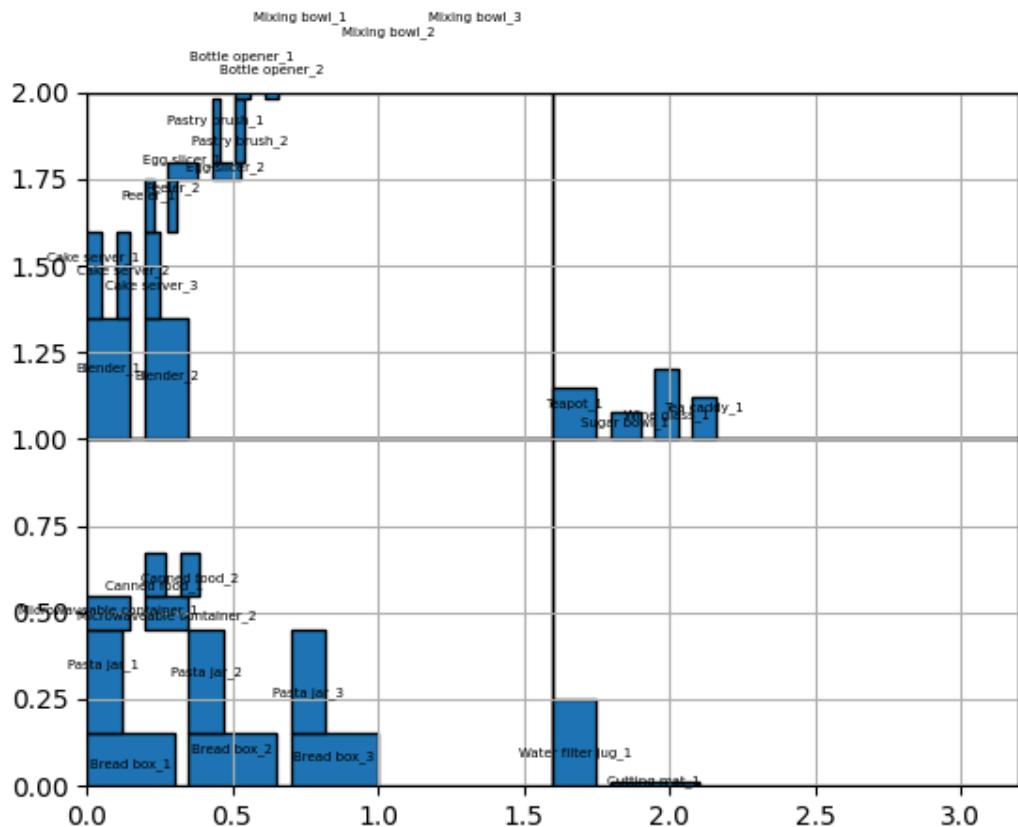
- The first element is the cell index.
- The second and third elements denote the row and column positions respectively.
- The fourth element is a list containing the relationships among the objects placed within that cell, expressed in the language specified.

This format reflects the spatial distribution and orientation of objects within each cell on a two-row, two-column shelf, ensuring clarity in the relationships and positions of all items.

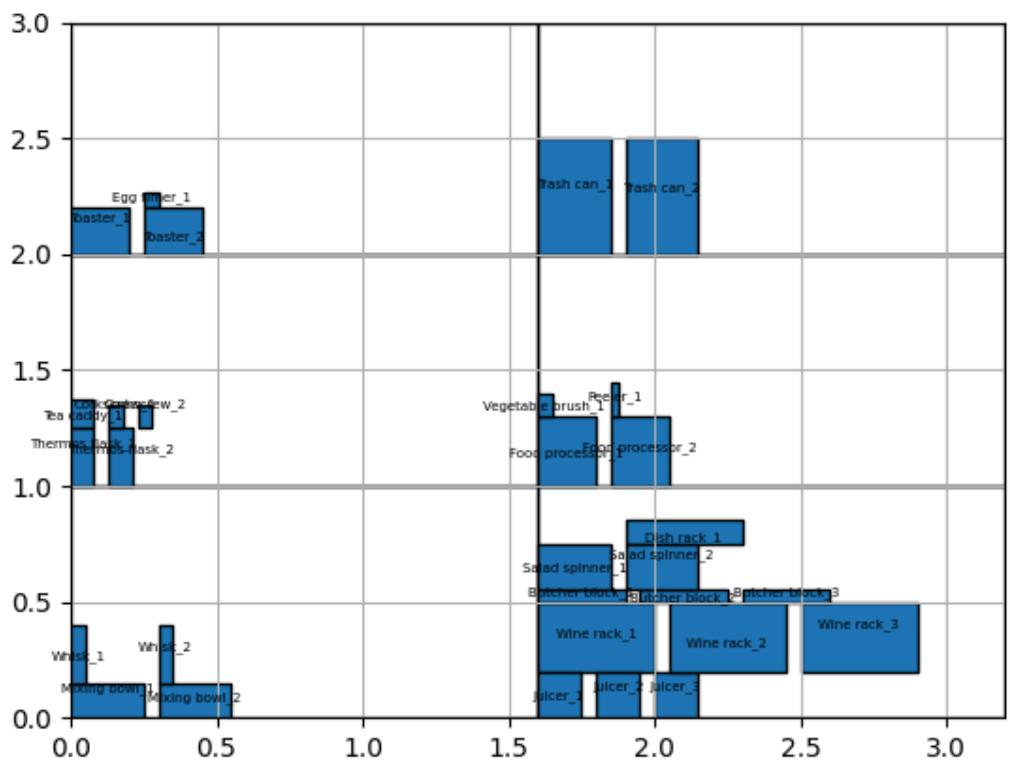
Success:



Failure:



Above: Missing relationship description tuple



Above: physically invalid relationships