Project 4 Writeup

Max Conway, Nicholas Rummel

December 18, 2024

Abstract

Effective autonomous systems must perceive and reason about the world around them. Historically, image classification paired with custom algorithms have been coupled to guide autonomous systems. Recent research efforts have focused on building Scene Graphs to give an effective framework to accomplish more complicated tasks. Building Scene Graphs is challenging and usually requires training custom machine learning algorithms to accomplish this task. This approach lacks generalizablity. There is a lack of clean labeled training data to build Scene Graphs directly. Our approach instead chooses to make use of modern open world models in conjunction to create a robust Scene Graph pipeline. The algorithm's abilities are demonstrated on proof of concept lab experiments. Our code is publicly available at `https://github.com/mconway2579/OW_PSG`.

1 Background

Building a representation of your environment is a critical capability for autonomous systems. Early robots made a grid world assumption which discretized the world into typed cells. Modern robots create a scene graph where nodes are objects in the scene and edges are relationships between objects. Existing Scene Graph (SG) generation methods classify objects in a scene from a closed set of labels [4]. This approach is limited because of a lack of generalizability. Objects that are not in the set of labels are often misclassified and poor decision making results.

Recent advances in machine learning have allowed for open world vision models, their usage in environment understanding is currently a research frontier. For instance, Concept Graph [2] leverages foundation models to create an open world semantic map. Unfortunately, this work suffers from two drawbacks that we hope to address. First, nodes can be mislabeled causing false positives. Second, their algorithm relies fully built map. Advances in machine learning have allowed for open world vision models, their usage in environment understanding is currently a research frontier. For instance, Concept Graph [2] leverages foundation models to create an open world semantic map. Unfortunately, this work suffers from two drawbacks that we hope to address. First, nodes can be mislabeled caus incrementally, and the down stream algorithm necessitates fully built 3D map and produces a single concept graph.

We address these complications in with the Concept Graph algorithm, by developing an iterative scene graph method that updates the SG as more data becomes available. We

assume that both imagery data and depth information can be obtained by the autonomous system. For each sample, a pipeline of minimal wrapping code calls general open world to generate a SG for the new sample. Then, the SG for the most recent sample is joined with the current estimate of the full world SG. This addresses the Concept Graph's shortcomings giving a current best estimate of the full SG, and reducing false positives by taking many samples from more general tools.

2 System Overview

Generalizablity is at the forefront of our pipeline's design. Our system assumes an RGBD camera as well as 6-dof pose estimation. For our testing, we use an intel realsense camera mounted on a universal robotics arm (UR5). For every (RGBD, Pose) pair we first prompt ChatGPT4o [7] with the RGB components of the image to output a SG represented by a state JSON. The output contains *objects* and *object relationships*. This is a programatic representation of the SG where objects are nodes and object relationships are edges. The state JSON is then passed as an input to OWLv2 [6] to create bounding boxes in the RGB image of each *object*. We then use the *object* bounding boxes with SAM-2 [8] to obtain a pixel mask of the *object*. This mask is applied to RGBD image to get a point cloud for each *object*. Finally we form a graph where *objects* and their corresponding point clouds are nodes and *object relationships* with their string descriptor are edges.

The first SG is used as the full world SG, and subsequent graphs are combined with the current full world SG. To combine nodes we first match nodes between graphs using a semantic distance metric, then all matched nodes have their point clouds combined and are added to a new graph, nodes that are not matched between graphs are then also added to the graphs. After all the nodes have been combined between two graphs we match edges and transfer matched edges from both graphs into the new combined graph, finally edges that were only in one graph are also transferred to the new graph.

3 Use of gpt-4o

The first step of the pipeline is to obtain an estimate of a SG for a particular sample from gpt-4o. This is accomplished by providing the current RGB image, system prompt and a user prompt to gpt-4o. The system prompt constrains the output to of this algorithm to a usable format and encourages predictable behavior. The user prompt provides call specific information to tailored to the current scenario. This step is prone to error because gpt-4o inherently has probabilistic outputs. This LLM was developed as general tool, but hallucination and other erratic behavior has been well documented when applied to subject specific applications [1, 5]. Erratic behavior is mitigated to some degree by making multiple calls to gpt-4o and taking the output that is identified as the best. This is discussed in more detail in Section 3.4.

3.1 Input Prompts

Below are the prompts currently used in our pipeline. The system prompt has be significantly tailored to improve and constrain the output of gpt-4o. As the algorithm is developed further, we expect to possibly leverage more specific user prompts. Currently, we use the minimal user prompt shown.

```
system_prompt="""You are a robots eyes. Your task is to analyze the scene, determine the objects present, and infer their relationships through detailed chain of thought reasoning.

\# Instructions

You should output a JSON object containing the following fields:

- **objects**: A list of strings where the string describes the object visible in the scene descriptions should be percise nouns.

- **object\_relationships**: A list of tuples describing relationships between the objects. Each tuple should be in the format (OBJECT1, RELATIONSHIP, OBJECT2), where OBJECT1 is related to OBJECT2.

\# Chain of Thought Reasoning

1. **Identify Objects**: Begin by analyzing the scene to identify all visible objects.
    - List each object and the number of instances of that object.

2. **Determine Object Positions**: For each object, determine its placement in relation to other objects:
    - Is the object spatially related to another object?
    - Capture as many relationships as you can

3. **Establish Relationships**: Once object positions are determined, establish relationships following these rules:
    - Each relationship is a triple (OBJECT1, RELATIONSHIP, OBJECT2), where OBJECT1 is related to OBJECT2 by RELATIONSHIP.

4. **Verify Completeness**: Ensure that all objects are covered in the relationships and that none remain unrelated.

\# Output Format

Your output should be formatted as a JSON object, like the example below:

```json
```
```

```
30
31 \{
32   "objects": ["table", "A", "B", "C"],
33
34   "object\_relationships": [["A", "is on", "B"], ["B", "is under", "
     table"], ["C", "is next to", "B"]]
35 \}
36
37 \# Notes
38 - Ensure no object is left unplaced; every object must be included in
     the relationships field either on another object or on the table.
39
40 - Follow the reasoning steps explicitly before outputting to ensure
     correctness and completeness.
41
42 - You cannot have an object in a relationship but not in the object
     list or saftey will be at risk
43
44 - Ensure that the object\_relationships are only made up of objects in
     the objects list
45
46 - Use specific Nouns and advjectives to label objects
47
48 - Include many relationships
49 """
50 user_prompt="How are objects layed out on the table?"
```

3.2 gpt-4o Example Outputs

Below in Figures 1 and 2 we show the the input RGB images and corresponding output for each gpt-v4 call.

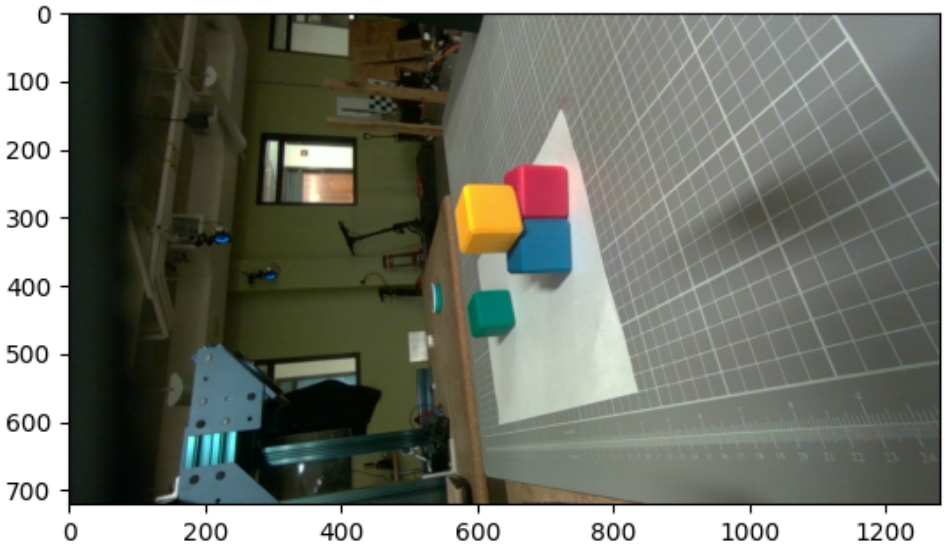

Figure 1: A side view of scene where blocks are stacked on top of paper and a table.

```
{
"objects": [
    "yellow cube",
    "pink cube",
    "blue cube",
    "green cube",
    "grid cutting mat",
    "white paper",
    "table"
],
"object\_relationships": [
    [
        "yellow cube",
        "is next to",
        "pink cube"
    ],
    [
        "pink cube",
```

```
            "is next to",
            "blue cube"
        ],
        [
            "blue cube",
            "is next to",
            "green cube"
        ],
        [
            "yellow cube",
            "is on",
            "white paper"
        ],
        [
            "pink cube",
            "is on",
            "white paper"
        ],
        [
            "blue cube",
            "is on",
            "white paper"
        ],
        [
            "green cube",
            "is on",
            "white paper"
        ],
        [
            "white paper",
            "is on",
            "grid cutting mat"
        ],
        [
            "grid cutting mat",
            "is on",
            "table"
        ]
    ]
}
```

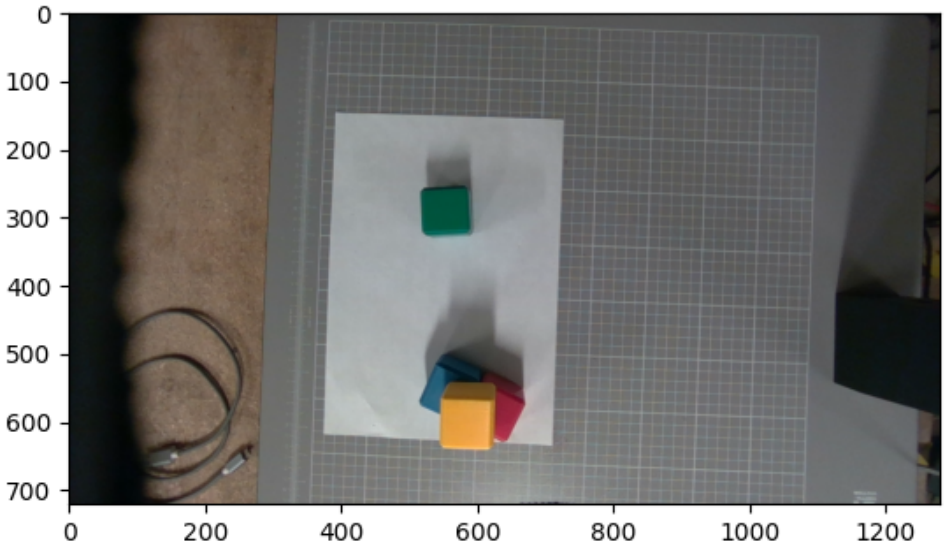

Figure 2: A top view of scene where blocks are stacked on top of paper and a table.

```
{
    "objects": [
        "green cube",
        "blue cube",
        "red cube",
        "yellow cube",
        "paper",
        "table"
    ],
    "object\_relationships": [
        [
            "paper",
            "is on",
            "table"
        ],
        [
            "green cube",
            "is on",
```

```
            "paper"
        ],
        [
            "blue cube",
            "is on",
            "paper"
        ],
        [
            "red cube",
            "is on",
            "blue cube"
        ],
        [
            "yellow cube",
            "is on",
            "blue cube"
        ],
        [
            "red cube",
            "is next to",
            "yellow cube"
        ],
        [
            "green cube",
            "is to the left of",
            "blue cube"
        ]
    ]
}
```

3.3 Discussion of gpt-4o Performance

In this instance, gpt-4o correctly identifies the objects, and correctly identifies some object relationships.

From the side view, it is correctly identified that the blue cube, pink (red) cube and green cube are on the paper, and that the paper is on the cutting mat, and that the cutting mat is on the table. But the yellow cube is incorrectly said to be on the paper as well. Ideally the algorithm would have identified it to be on top of both the red and blue cube.

From the top view, it is correctly identified that the blue cube is on the paper, and that the green cube is near the blue cube. The cutting mat is now neglected, and in its absence the paper is correctly said to be on the table. Now, the yellow cube is correctly said to be on the blue cube, but the red cube is incorrectly said to be next to the yellow cube. Ideally the algorithm would have identified it to be on top of both the red and blue cube.

The gpt-4o model is prone to making such mistakes, this is why several calls to gpt-4o

are made and the best output is then selected through Monte-Carlo estimation described in Section 3.4.

3.4 MonteCarlo State Estimation

Because of unpredictable behavior for individual calls to gpt-4o, multiple calls are made and we identify the best SG of the current scene. We have a user defined hyper parameter N for the number of calls to gpt-4o.This produces a set of candidate SG for this sample, $\{G_n\}_{n=1}^{N}$. The graph edit distance is computed pairwise over this set to create a distance matrix, $\mathbb{D} \in \mathbb{R}^{N \times N}$. Each entry, $D_{n,m}$, is the graph edit distance between state estimate n and state estimate m. The best graph is selected by choosing the graph corresponding to the minimum row sum. This identifies the graph that is the most similar to the most other candidate graphs.

The graph edit distance we use is defined by setting the node/edge replacement cost to the semantic distance between the original node/edge and the replacement node/edge, the node/edge addition, subtraction cost is a constant 1. We use the semantic distance and equality describe in Section 4 to measure distance between nodes and edge identifiers.

We found that this significantly reduces false positives and negatives in each estimate of a sample's SG.

4 Semantic Reasoning

When describing a scene the LLM may use synonyms to describe the same thing, in our example outputs above the LLM used the term pink cube in fig 1 and used the term red cube in fig 2. Neither term is wrong but we need a way to determine whether or not the LLM is referring to the same thing in each instance, to this end we use the cosine distance of text embeddings.

4.1 Semantic Distance between two strings

Given two strings s_1 and s_2 we use a semantic encoder E. Specifically, we make use of the all-MiniLM-L6-v2 hugging face sentence encoder [9]. Each string is encoded into a vector of real values, $E(s_i) \in \mathbb{R}^d$. The semantic distance between the two strings is the cosine distance in the encoding space

$$\mathcal{D}_{\text{semantic}}(s_1, s_2) = 1 - \frac{E(s_1) \cdot E(s_2)}{\|E(s_1)\|\|E(s_2)\|}$$

To determine semantic equality between two strings, we have a threshold the semantic distance, τ, $\mathcal{D}_{\text{semantic}}(s_1, s_2) < \tau$

4.2 Semantic Matchings

Sometimes, like when matching nodes between graphs, we need to match strings between sets, given two sets of strings $S_1 = [s_{1,1}, s_{1,2}...s_{1,N}]$ and $S_2 = [s_{2,1}, s_{2,2}...s_{2,M}]$ we create a complete bipartite graph where the weight of the edge from $s_{1,i}$ to $s_{2,j}$ is $SemanticDistance(s_{1,i}, s_{2,j})$.

We then find the minimum weight matching on this bipartite graph and keep all the matchings with $SemanticDistance < \tau$.

5 3D Scene Information

Given a SG for the current sample, 3D information about the scene is computed. This is done by first obtaining bounding boxes from OWL-v2 described in Section 5.1. Then we compute a pixel-wise mask from SAM-v2 described in Section 5.2. This mask allows us to obtain 3D pose information for *object* in the SG.

5.1 JSON to Bounding Boxes

The SG produced from GPT labels *objeces* in the scene but does not provide localization. To obtain spatial identification of each object, the pipeline makes use of OWLv2 [6]. This algorithm consumes string that describe *objects* and produces bounding boxes for each *object*. More over each object is given a list of potential bounding boxes with a corresponding scores. The pipeline, takes the best scoring bounding box which implicitly assumes there is only one object of each instance. This also is a source of error as sometimes OWLv2 produces inaccurate bounding boxes.

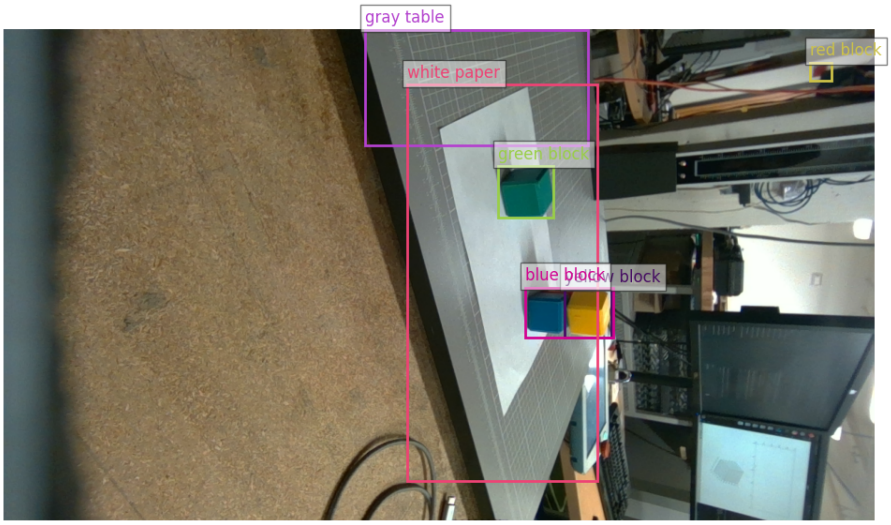

Figure 3: OWLv2 Output

The output of our bounding box step can be seen in Figure 3, note this is the same block positioning as 1 and 2. Unfortunately the output of OWLv2 is not perfect. The main error

is caused by the red block being mostly occluded by the blue block. OWLv2 selects a red blob in the distance instead of the true location of the red block.

5.2 Bounding Boxes to Point Clouds

SAM2 [8] consumes bounding boxes and produces pixel wise masks, we feed SAM2 the bounding boxes from OWLv2 to get pixelwise masks of the detected objects, these makes are then applied to the RGBD image to get a point-cloud of the object in meter scale relative to the cameras position. Given the position of the camera, it is then trivial to transform these points from the camera coordinate frame to the world coordinate frame.

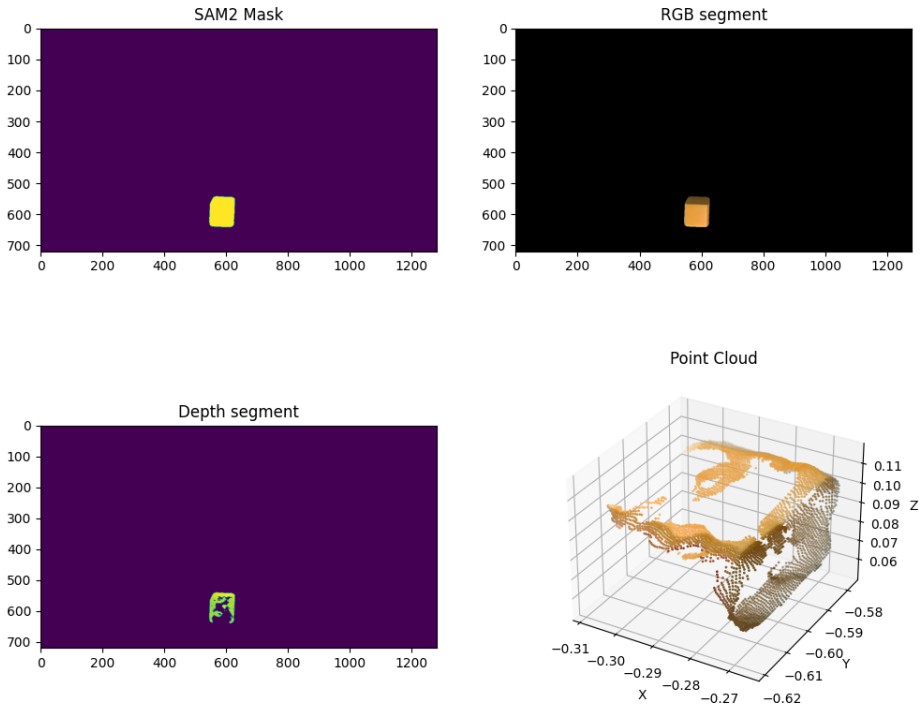

Figure 4: In the top left we can see the mask produced by SAM, in the top right we see that mask applied to the RGB image, in the bottom left we see the mask applied to depth image, and in the bottom right we can see the resulting point cloud

6 Scene Graph Manager

After we obtain a estimate of the SG for the current sample, we want to join this graph into the full world SG. This process is described in Sections 6.1 and 6.2.

6.1 Combining Nodes

Merging graphs the SG for the current sample and the current estimate of the full world SG requires combining two graphs from different poses at different times. To do this we first merge the nodes from two graphs. There are three cases to consider when merging nodes from graphs:

1. An object is represented in both graph one and graph two
2. An object is only represented in graph one
3. An object is only represented in graph two

To find objects represented in both graph one and graph two we use the semantic matchings method detailed in section 4.2. All semantic matchings between graph one and graph two are considered to be describing the same object. We then can add the point clouds for the object in both graphs to get a better point cloud in our combined graph. For objects that were not in both graphs we add the data and point-cloud from the graph it was present in to the new combined graph.

6.2 Combining Edges

After the nodes from the graphs to be combined have been processed we begin processing edges, this time there are five cases to consider

1. A node exists in both graph one and graph two, and the edge exists in both graph one and graph two
2. A node exists in both graph one and graph two, but the edge only exists in graph one
3. A node exists in both graph one and graph two, but the edge only exists in graph two
4. A node only exists in graph one
5. A node only exists in graph two

In case one for each node present in graph one and graph two we find a semantic matching for its edges, then we add each matched edge to the new output graph. For case two, three, four and five, we simply add all the edges.

This edge combining step is a candidate for improving our approach. One source of error here is if the objects have moved between graph one and graph two all the edges will still be transferred to the combine graph even if some of the edges are no longer valid.

7 Future Improvements

There are many future improvements we would like to make before submitting this research to a publication. The four main areas for improvement are:

- Considering time and motion of objects.

- Considering the physics of the environment to validate that objects and relationships are physically valid.
- Allowing for the detection of multiple object instances.
- Comparing against other methods.

7.1 Time Consideration

By considering time in our graph manager we would allow for objects in the scene graph to move and its associated relationships to change, this is a critical capability for robotics systems.

7.2 Physics Validation

By considering the physical relationships between objects and their point clouds we should gain two improvements, the first is that we can reduce redundant nodes such as white paper and paper sheet, or grey table and table in figure 5. These strings may be semantically different but they share many points in the point cloud, allowing us to determine these are the same object. We would also like to validate the relationships between nodes by using physics and the point clouds for example, we should be able to validate the is on relationship by comparing the Z components in two points clouds, similarly we should be able to validate the is next to relationship with the X and Y components of the point clouds. We would like to explore generative approaches to verifying physical relationships so that we do not limit ourselves to a closed set of relationships.

7.3 Multiple object instances

Currently we assume each object only has one instance, when getting the bounding boxes from OWLv2 we simply take the highest score bounding box. This greedy approach occasionally gives false positives like in fig 3 and will not always give a bounding box around the same instance of an object across multiple frames if multiple objects are present. To handle multiple object instances we should again consider the physics of the pointclouds as well as let gpt produce counts for how many objects are present. We can leverage techniques from instance segmentation [3] to keep node references consistent between graphs.

7.4 Comparing Against other methods

While working on this project we became aware of the Seeing and Understanding dataset (SUN)[10] We must compare our method against state of the art approaches like Hydra[4] and ConceptGraph[2] in order to publish this work. The SUN dataset seems like a common, well explored benchmark that will allow us to compare our work against state of the art methods

8 Results

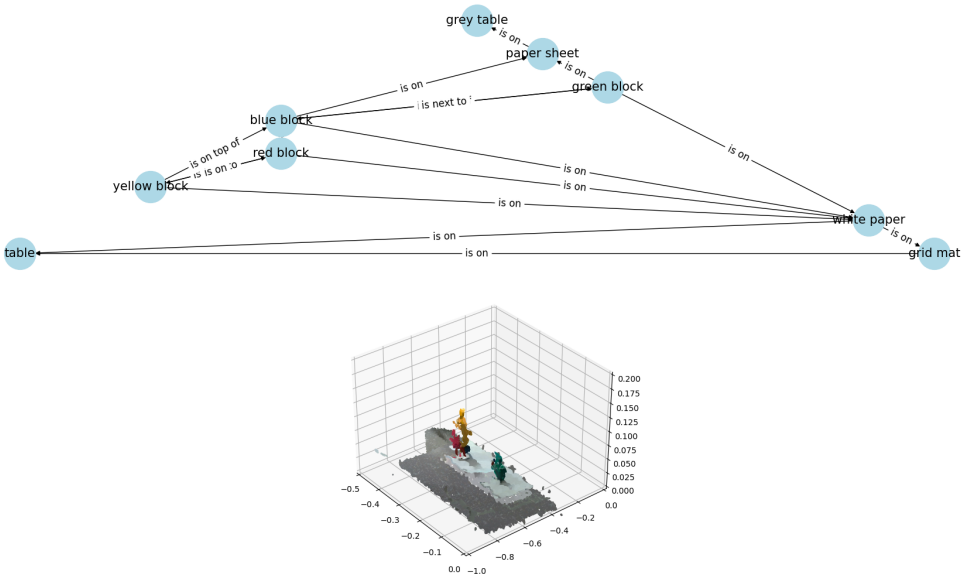

Figure 5: Our approaches output after consuming four images

While our approach is not final we have some promising results. Our approach suffered from miss-classifications as well as repeated nodes, these drawbacks are lessened with our semantic matching and montecarlo state estimation but not entirely eradicated. Some successes in figure 5 are

- Accumulated pointclouds from different views give better object models
- Node redundancy is reduced by our semantic matchings
- Object relationships are all present
- Segmentation errors (like the incorrect red block in fig 3) give obscure background points

Some failures in fig 5 include

- multiple instances the same object (table and grey table) (paper and white paper), these could be reduced by lowering the node semantic match threshold
- the multiple instances could also be reduced by doing a physics based matching with the point clouds instead of just a semantic matching

References

[1] Mikaël Chelli, Jules Descamps, Vincent Lavoué, Christophe Trojani, Michel Azar, Marcel Deckert, Jean-Luc Raynier, Gilles Clowez, Pascal Boileau, and Caroline Ruetsch-

Chelli. Hallucination Rates and Reference Accuracy of ChatGPT and Bard for Systematic Reviews: Comparative Analysis. *Journal of Medical Internet Research*, 26:e53164, May 2024.

[2] Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning, 2023.

[3] Abdul Mueed Hafiz and Ghulam Mohiuddin Bhat. A survey on instance segmentation: State of the art. *CoRR*, abs/2007.00047, 2020.

[4] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception engine for 3d scene graph construction and optimization. *CoRR*, abs/2201.13360, 2022.

[5] Ray Li, Tanishka Bagade, Kevin Martinez, Flora Yasmin, Grant Ayala, Michael Lam, and Kevin Zhu. A Debate-Driven Experiment on LLM Hallucinations and Accuracy, October 2024.

[6] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling Open-Vocabulary Object Detection, May 2024.

[7] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan,

Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. GPT-4 Technical Report, March 2024.

[8] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos, October 2024.

[9] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers, April 2020.

[10] Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *Int. J. Comput. Vision*, 119(1):3–22, August 2016.