# Scene Graph Generation for Object Localization in Smart Homes from User Commands

Md Abdur Rahman Fahad, Md Asif Tanvir
*Department of Computer Science*
*Missouri State University*
Springfield, Missouri, USA
{mf8494s,mt5864s }@missouristate.edu

***Abstract*—With the advancement of smart technologies, many homes now come with integrated smart systems like voice assistants and smart cameras. In this project, we want to use these systems of smart homes and build a solution that would enable users to find objects within their homes very easily by just giving regular human commands. We have integrated natural language processing (NLP) with the scene graph generation technique to build a system that will take the user's command and locate the objects from a camera feed from a complex home environment. This approach allows for accurate object localization and improves smart home assistance, particularly for visually impaired individuals.**

***Index Terms*—Object Localization, Scene Graphs, Iterative refinement, Detectron2, DistilBERT**

## I. Introduction

In today's time, many of the homes come with a smart setting, like having a camera or voice assistant. Day by day, people are also getting accustomed to using voice command-based assistants for their regular tasks. But some tasks are difficult to accomplish even for a smart home. One common task would be the ability to find objects in a cluttered or complex environment, such as asking,"Where is the remote?". Understanding the scene and localizing objects based on user queries is crucial for enhancing the smart home experience.

For solving this issue, one crucial model is understanding the regular objects and their relation with their neighboring objects. In this case, scene graphs can be a really useful method. Scene graphs provide a relationship between objects from a scene. In the paper [1], the authors have provided a method for generating scene graphs using iterative message passing. In their proposed method, it is possible to generate scene graphs that are visually grounded from images. Their model employs a novel approach that iteratively refines its predictions by exchanging contextual information along the structure of a scene graph. The generated scene graphs should look like figure 1.

Another work that is more recent and works on a similar domain is the model stated by Khandewal et al. [3]. Their work is a refinement on the existing work by removing the problem of fixed factorization. They use a transformer-based method to generate the scene graphs. On the back end, they use a message-passing system within the Markov Resource Field. Another work that is also important in the field of generating scene graphs is the work by Wu et al. **??** called

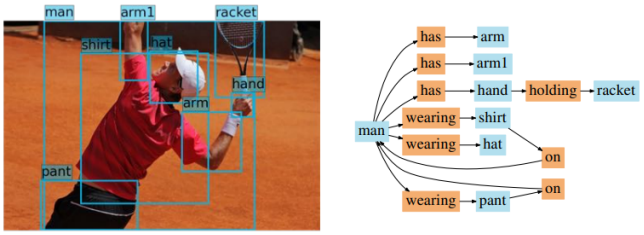


Fig. 1: Example of Scene Graph

Detectron2. This is an object detection model that can detect multiple objects from an image. This object detection method can be used in generating scene graphs.

While scene graph generation is one aspect of addressing this problem, another aspect would be understanding the context from the user's commands. Generally, to understand the context from a sentence, we need some kind of natural language processing model. Sanh et al. [5] proposed a lightweight model based on BERT, DistilBERT. BERT is a popular natural language processing model. In DistilBERT, a smaller language model is pre-trained on a distilled dataset. The architecture is the same as BERT. But the number of layers is reduced by a factor of 2. Similar to BERT, DistilBERT uses bidirectional attention, meaning it considers both preceding and succeeding words to determine context.

In this project, we aim to develop a model capable of generating scene graphs from images to help users locate objects within their home environment based on their query. We have divided the house into multiple rooms, and each of our test images represents different rooms. For the scope of this project, we will not consider voice commands, only text inputs. From this input sequence, we try to identify the context of the sentence. We have used DistilBERT [5] for generating context from the user's queries. Then we will need to detect the objects from each room and generate scene graphs for those detected objects to identify the localizations. For this, we will base our approach on the paper, Scene Graph Generation by Iterative Message Passing [3], leveraging the Visual Genome dataset [2]. The generated scene graph will represent objects, attributes, and relationships, enabling the system to identify the location of queried objects. This has direct implications for

smart home assistance, object tracking, and enhancing overall user interaction. This project can be useful to visually impaired people to find objects in their homes.

The remainder of the paper is organized as follows: Section II describes the approach we have taken for implementing this whole process in detail. We also describe the inherent architecture and their training procedures in this section. In section III we talk about the datasets and their properties. In section IV, we talk about different evaluation metrics. Different real-world examples and scenarios are discussed in the section V. Finally, section VI contains the concluding remarks for our project.

## II. Approach

Our approach has three main modules, which are:

- Context Detection module
- Object Detection module
- Scene Graph Generation module

The three modules are described in different sections below.

### A. Context Detection Module

This module receives user commands or sentences in text format and ggenerates a context that refers to one of the 150 objects in the Visual Genome dataset [2].

*1) Data Preparation:* We have processed object data from the Visual Genome dataset to generate a synthetic dataset for sequence classification. First of all we have collected the synonyms of all the classes of Visual Genome objects and then we have paired them with predefined templates to generate diverse sequences of tokens. The resulting dataset has a size of 47296, which consists of sentences paired with their corresponding object labels (context classes). Then we shuffled the dataset which was used as input for sequence classification models trained to predict object classes from contextual queries.

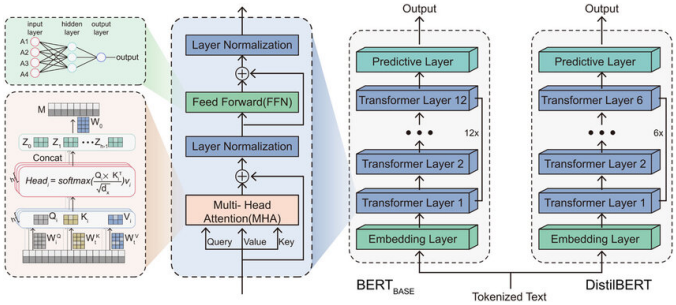


Fig. 2: DistilBERT model architecture

*2) Training:* We have use fine-tuning on DistilBERT model for our sequence classification task using our data prepared from the Visual Genome dataset. First of all the data is tokenized using the DistilBERT tokenizer, padding and truncating sequences to a maximum length of 128 tokens. Then we did a label mapping to encode context labels as integers for classification. Then we processed the tokenized dataset into PyTorch tensors. We fine-tuned a DistilBERT model with a classification head for multi-class classification. We used the Hugging Face *Trainer* API. Our training ran for 15 epochs, with a learning rate of 2e-5, training batch size of 256, and an evaluation batch size of 512, with weight decay set to 0.01. At each epoch, we compute evaluation metrics, such as loss, to monitor performance and optimize the model. After fine-tuning the model, we save both the model and tokenizer for faster inferences in future. Then we created a pipeline to predict object context from input sentences using the saved trained model weights. Figure 3 and 4 shows the training loss and evaluation loss for our training.

We have used Pytorch Library for all our training and inference and We utilized *DistilBertForSequenceClassification, DistilBertTokenizer* from *transformers* library package.


Fig. 3: Training Loss for DistilBert

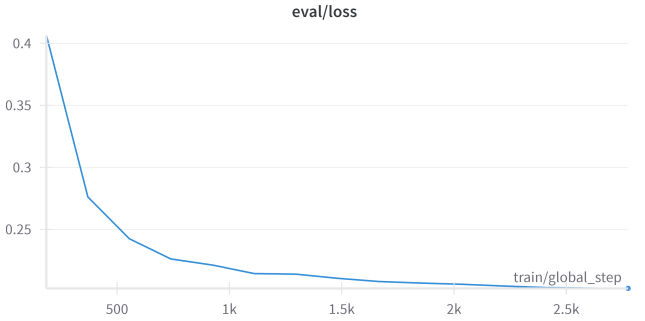


Fig. 4: Evaluation Loss for DistilBert

### B. Object Detection Module

We have used the popular library Detectron2 [4] by Facebook AI Research for detecting objects from images. Detectron2 uses a pre-trained object detection model which is used to detect the objects in the image. Each detected object is represented as a node in the graph. A detailed overview of Detectron2 is provided here.

- **Backbone Network (Resnet-101)**: The backbone extracts hierarchical feature representations from the input image. Detectron2 supports various backbones, including ResNet-101, which is a deep convolutional neural network pre-trained on ImageNet. ResNet-101 consists of

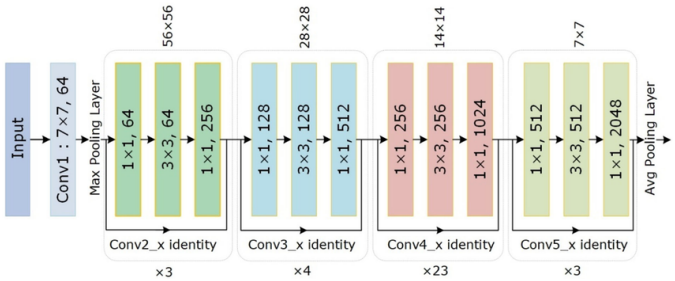


Fig. 5: The architectural view of ResNet-101

101 layers and leverages residual connections to mitigate vanishing gradient issues, making it effective for training very deep networks. It is composed of multiple stages, each containing residual blocks with a combination of 1x1, 3x3, and 1x1 convolutions. As we go deep into the network, the number of features increases and the dimension of the image decreases. The final layer of the Resnet is a fully connected layer which assigns the probability for each output classes.

- **Feature Pyramid Network (FPN)**: Built on top of the backbone, the FPN enhances multi-scale feature extraction by constructing a pyramid of feature maps with varying resolutions. This allows the model to effectively detect objects of different sizes, making it particularly useful for handling diverse datasets.
- **Region Proposal Network (RPN)**: The RPN generates a set of candidate object proposals by predicting bounding boxes that are likely to contain objects. These proposals focus the subsequent stages on regions of interest, significantly reducing the computational burden.
- **Region of Interest (ROI) Heads**: The ROI heads refine the regions proposed by the RPN and perform specific tasks:
  - *Bounding Box Regression*: Adjusting the coordinates of bounding boxes for more precise localization.
  - *Classification*: Assigning object categories to the detected regions.
  - *Mask Prediction*: Producing pixel-level segmentation masks for instance segmentation tasks.

### C. Scene Graph generation Module

We initially proposed the architecture proposed in Scene Graph Generation by Iterative Message Passing [1]. This approach uses a deep neural network to first detect objects in an image and then iteratively pass messages between object nodes in the graph to refine the predictions of relationships and attributes. Then we implemented the architecture of Iterative Scene Graph Generation [3] which is a transformer-based approach. An overview of the module can be seen in figure 7. The different components of Scene Graph Generation module is described here.

**Image Encoder**

For each image, we have used a deep convolutional network ResNet to obtain an image-level spatial map. A 6-layer encoder is used to transform input images into position-aware flattened image feature representation. Each layer has Multiheaded Self-Attention for capturing long-range dependencies, Feedforward Networks (FFN) with two linear layers and dropout and Layer Normalization for stable training.

**Predictor Decoders**

This approach models each of the subject, object, and predicate predictors using a multi-layer transformer decoder. The $t^{th}$ layer of the decoder generates a set of triplet estimates based on fully connected feed-forward layers, which generate the scene graphs at that layer. The decorders are divided into three sets of decoders: subject, object, and relation decoders. Each set has 6 TransformerDecoderLayer modules. Each layer has Self-Attention for intra-query relationships, Cross-Attention to condition on image and prior layer outputs and Feedforward networks with normalization and dropout layers. 300 queries are used in model architecture for both objects and relations. These queries act as placeholders for predicting scene graph components.

**Iterative refinement**

Each layer of the transformer decoder takes the output of the previous layer as input. Predictions are progressively refined across multiple steps. Attention mechanisms are employed to integrate information from both the image features and previous graph estimates

### D. Loss Function

The project uses multi-task loss function. For object detection Detectron2 uses these losses:

- Object Classification Loss: Cross-entropy loss for predicting the correct class labels for the detected objects.
- Bounding Box Regression Loss: Smooth L1 loss for accurate localization of objects.
- Keypoint Loss: L2 loss for estimating keypoint positions.

For Scene Graph Generation module a joint loss is used. In the "Iterative Scene Graph Generation" paper, the authors propose a transformer-based refinement architecture that can be trained in an end-to-end fashion. To ensure the generation of a valid scene graph at each refinement step $t$, they introduce a novel joint loss function applied at every layer of the decoder. The combined loss $L$ can be expressed as:

$$L = \sum_t L^t = \sum_t (L_s^t + L_o^t + L_p^t)$$

where $L_x^t$ (for $x \in \{s, o, p\}$) denotes the loss applied to the $t$-th layer of the decoder for the subject, object, and predicate, respectively. The model generates a fixed-size set of $n$ triplet estimates $\{(ps_i^t, pp_i^t, op_i^t)\}$ at each step $t$, where $n$ exceeds the number of ground truth relations for a given image.

The loss $L_s^t$ is defined as follows:

$$L_s^t = \sum_{i=1}^{n} \left( -\log(p^s_{t,\sigma_p(i),c}) \cdot s_{i,c} + 1_{\text{tri}} \cdot L_{\text{box}}(p^s_{t,\sigma_p(i),b}, s_{i,b}) \right)$$

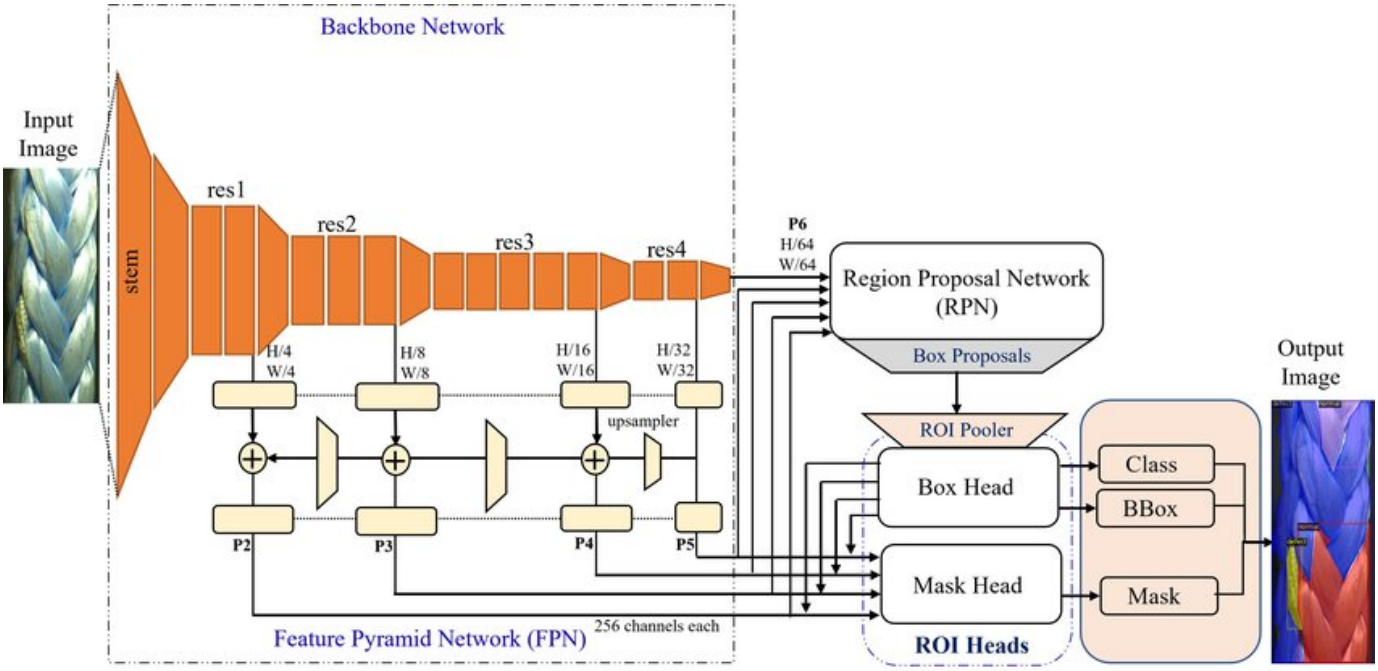


Fig. 6: Architecture of Detectron2 Framework

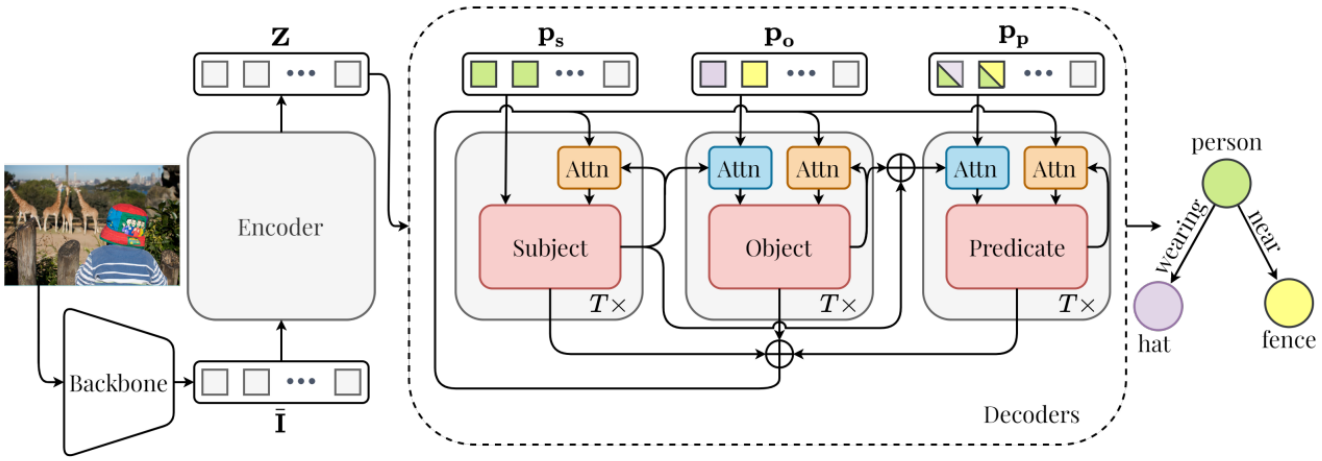


Fig. 7: Architecture of the Transformer based IterativeSG

### *E. Preprocessing Layers*

Preprocessing involves standard image augmentation techniques such as random cropping, resizing, and normalization. These are the steps of the preprocessing:

- Image Resizing: Images are resized to a fixed scale while maintaining their aspect ratio.
- Normalization: Pixel values are normalized by subtracting the mean and dividing by the standard deviation of the dataset (usually based on ImageNet values) to standardize inputs and improve model convergence.
- Data Augmentation: Basic data augmentation techniques, such as random cropping, flipping, rotation, and brightness adjustments, are applied to enhance model robustness and improve generalization.
- Bounding Box and Mask Resizing: Corresponding annotations, such as bounding boxes and segmentation masks, are resized and adjusted to match the transformed image dimensions.
- Metadata Handling: Metadata is also preprocessed for each image, which includes scaling factors, original image sizes, and other relevant information needed for post-processing and evaluation.

## III. Dataset

The Visual Genome dataset [2] is used to train and evaluate the model. It contains 108,000 images, each annotated with objects, attributes, and relationships. This rich dataset provides the necessary information to train models for scene graph generation and object localization tasks.

- Training Set: Images and annotations from Visual Genome are used to train the object detection and relationship prediction models.
- Test Set: A subset of images containing 500 images was used for evaluating the model's ability to generate accurate scene graphs and locate queried objects in unseen images. We will also test the model in our own home/office setup using captured images to see how the model performs.

## IV. Evaluation

To measure the performance of the context detection module, we have run evaluations on our separated test data. The results are shown in Table I.

| Metric | Value |
|---|---|
| Accuracy | 0.8688 |
| Precision | 0.8914 |
| Recall | 0.8688 |
| F1 Score | 0.8550 |

TABLE I: Performance Metrics for the Context Detection

For evaluating the performance of the object detection module, we have used Average Precision for all object classes. AP50 refers to Average Precision for IoU (Intersection over Union) set at 50%. APs, APm, APl refers to the Average Precision for small, medium, and large objects.

| AP | AP50 | AP75 | APs | APm | APl |
|---|---|---|---|---|---|
| 13.309 | 24.861 | 12.074 | 4.137 | 7.732 | 17.963 |

TABLE II: Object Detection Bounding Box Average Precision

To measure the performance of scene graphs, we report results using standard scene graph evaluation metrics, which are Recall (R@K) and Mean Recall (mR@K). While recall is class agnostic, mean recall averages the recalls computed for each predicate category independently. Usually, higher R@K is indicative of better performance in dominant (head) classes, whereas higher mR@K suggests better tail class performance. We have used K = 20, 50, and 100 for evaluation.

We have performed an evaluation on 5000 random images from the Visual Genome dataset. The results can be seen in tables II and III. We have also manually evaluated some real-life test images.

| SGMeanRecall@20 | SGMeanRecall@50 | SGMeanRecall@100 |
|---|---|---|
| 0.1150 | 0.1425 | 0.1601 |

TABLE III: Mean Recall of Scene Graph Evaluation

## V. Results

We first attempted to implement the architecture described in the original paper [1]. However, because this implementation is six years old and the repository has not been updated, the required dependencies could no longer be installed. As a result, we opted to implement a more recent, transformer-based approach using iterative scene graph generation [3]. We trained this updated model on the Visual Genome dataset and generated several sample predictions. Below, we present some of these predictions 8 and 9. The relations for the images are shown in tables IV and V In our predictions, if any objects do not appear in the Visual Genome object list, they are not detected in the predictions.

We have included three test images from our lab environment with orange bounding box for the target object. The results can be seen in figure 10, 11, 12 and the relevant relations in table VII, VI, VIII.


Fig. 8: Test Image 1 (From Visual Genome Dataset)

| Object | Relation (Confidence) | Target |
|---|---|---|
| light | with (0.71), on (0.12) | door |
| clock | with (0.80), on (0.09) | door |
| face | with (0.80), on (0.09) | clock |
| clock | with (0.82), has (0.08) | face |
| plate | with (0.80), on (0.08) | door |
| light | with (0.91), on (0.04) | door |

TABLE IV: Test Image 1 Relations and Confidence scores

| Object | Relation (Confidence) | Target |
|---|---|---|
| pillow | on (0.48), above (0.18) | chair |
| bear | sitting on (0.26), on (0.21) | chair |
| curtain | with (0.31), mounted on (0.23) | window |
| chair | with (0.35), with (0.31) | pillow |
| chair | with (0.40), between (0.23) | table |
| chair | with (0.42), in front of (0.25) | window |
| table | with (0.57), near (0.12) | window |
| table | with (0.51), at (0.13) | table |
| window | with (0.74), behind (0.10) | table |

TABLE V: Test Image 2 Relations and Confidence scores

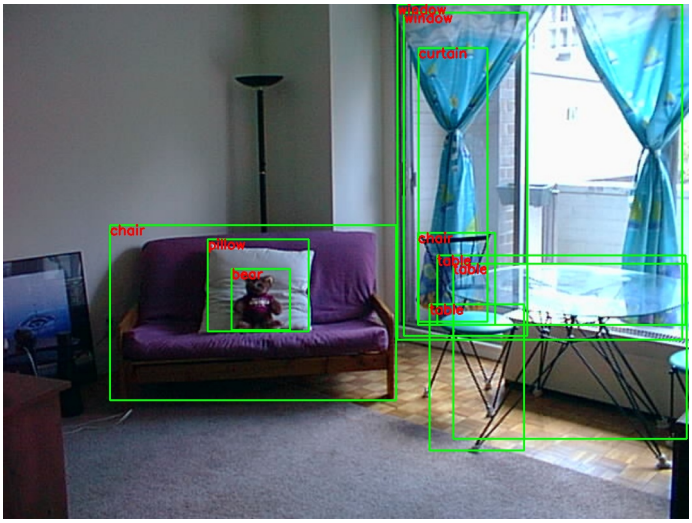


Fig. 9: (Test Image 2 (From Visual Genome Dataset)

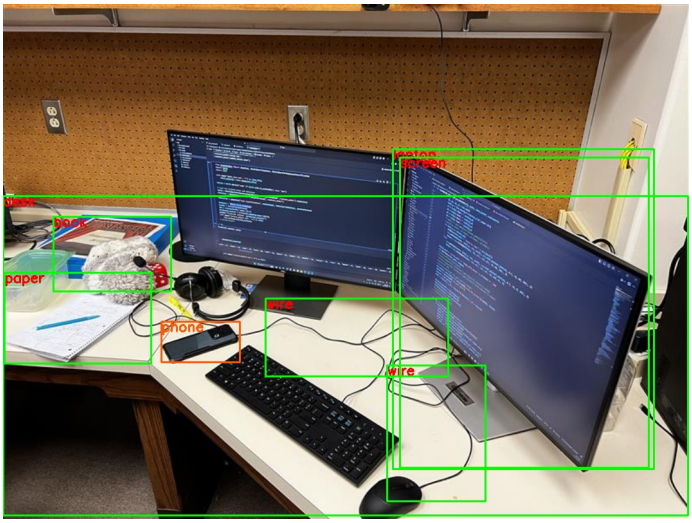


Fig. 10: Demo 1

| Object | Relation (Confidence) | Target |
|---|---|---|
| phone | on (0.39), with (0.36) | desk |
| phone | with (0.93), on (0.02) | laptop |
| phone | with (0.92), on (0.03) | paper |
| phone | with (0.93), and (0.01) | book |

TABLE VI: Relations and confidence scores for Demo 1

| Object | Relation (Confidence) | Target |
|---|---|---|
| bottle | on (0.46), sitting on (0.19) | table |
| bottle | with (0.41), has (0.18) | cap |
| bottle | with (0.76), on (0.11) | table |
| bottle | with (0.91), for (0.03) | phone |
| bottle | with (0.99), on (0.00) | wire |

TABLE VII: Relations and confidence scores for Demo 2

## VI. Conclusion

This project aims to solve a real-world problem in smart home environments by utilizing scene graph generation to assist users in locating objects based on natural queries.

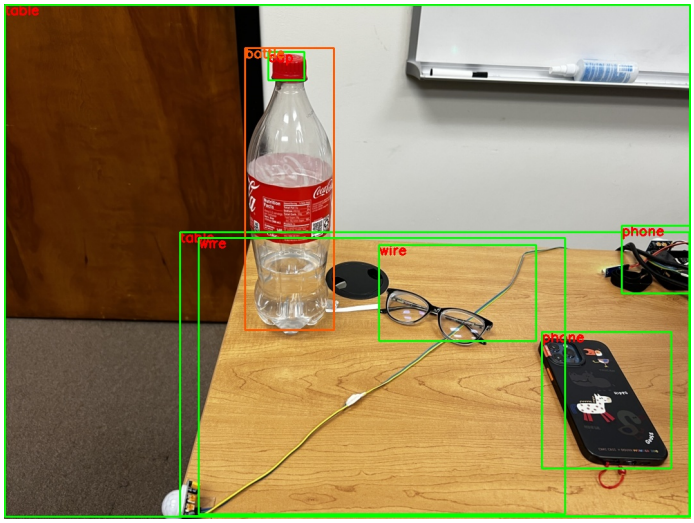


Fig. 11: Demo 2

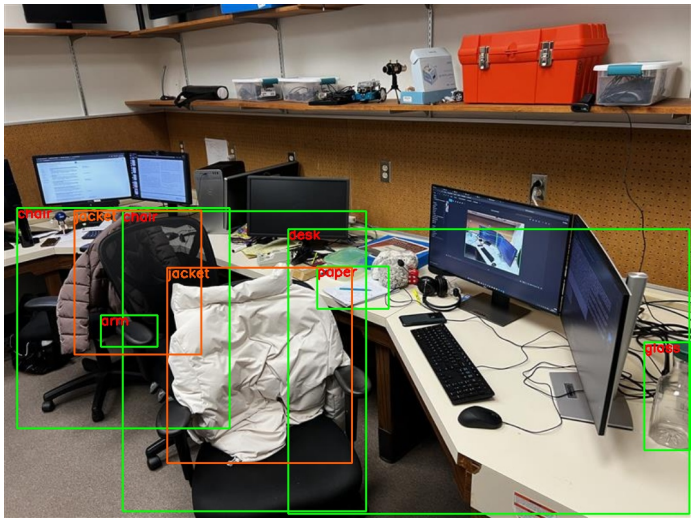


Fig. 12: Demo 3

| Object | Relation (Confidence) | Target |
|---|---|---|
| jacket | hanging from (0.53), with (0.13) | chair |
| jacket | hanging from (0.34), with (0.31) | chair |
| jacket | with (0.25), laying on (0.23) | chair |
| jacket | hanging from (0.41), with (0.24) | chair |
| jacket | with (0.55), at (0.21) | desk |

TABLE VIII: Relations and confidence scores for Demo 3

Currently the scope of this project covers text-based query answering, but this can help visually impaired people or old people locate items at home using voice commands by just adding a voice-to-text module. By using the architecture proposed in Iterative Scene Graph Generation [3] and the Visual Genome dataset [2], we have built a system that can efficiently generate accurate scene graphs and enable object localization.

## REFERENCES


[1] Xu, Danfei, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. "Scene graph generation by iterative message passing." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5410-5419. 2017.

[2] Krishna, Ranjay, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen et al. "Visual genome: Connecting language and vision using crowdsourced dense image annotations." International journal of computer vision 123 (2017): 32-73.

[3] Khandelwal, Siddhesh, and Leonid Sigal. "Iterative scene graph generation." Advances in Neural Information Processing Systems 35 (2022): 24295-24308.

[4] Wu, Yuxin, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. https://github.com/facebookresearch/detectron2.

[5] Sanh, Victor, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter." ArXiv, (2019). Accessed December 9, 2024.