Spatial and Semantic Scene Graph Generation in Retail

Amaan Sheikh [1]

Abstract

Retail shelves are a challenging problem in computer vision because products appear in dense layouts with occlusion and small visual differences between brands. In this project, we build a practical pipeline that turns a shelf image into a structured graph representation that supports planogram style reasoning. We will be focusing on classifying popular beverages only. Our system localizes products using a fine tuned detector, assigns brand labels using a frozen CLIP model with a few shot visual anchor bank, and then computes spatial and semantic relationships using geometry. We generate interactive semantic and spatial graphs and expose the results through a Streamlit interface for inspection and debugging.

1. Introduction

Retail auditing and planogram compliance require knowing what products are present and whether they follow expected shelf layouts. This is hard for standard detection and classification approaches because shelves contain many objects, strong occlusions, reflective packaging, and visually similar variants. In our case, the goal is to recognize a targeted set of popular beverage brands (Coca-Cola, Pepsi, Sprite, Fanta, Mountain Dew, 7 Up) and to represent their arrangement in a way that supports downstream reasoning like adjacency checks.

Instead of training an end to end model to directly output a scene graph, we build a modular pipeline. First, we detect product bounding boxes as a category agnostic object class. Second, we label each detected crop using a frozen CLIP model and a memory bank of visual anchors, which functions as a few shot nearest neighbor classifier in embedding space. Third, we compute spatial relationships using simple geometry on bounding boxes. Finally, we build and visualize graphs that link physical instances to brand concepts and show relationships like `next_to` and `left_of`. We also include a Streamlit interface that shows each stage output so the pipeline is easy to demo and debug.

2. Related Work

Dense retail object detection. Dense retail shelves pose a challenging detection problem due to heavy object crowding, small object sizes, and significant visual similarity between products. The SKU-110K dataset formalized this setting and demonstrated the limitations of standard object detectors in densely packed retail scenes (Goldman et al., 2019). More recent retail-focused detection studies build on this idea by applying modern YOLO-style architectures to improve localization accuracy and throughput in real store environments (Pannoy et al., 2024). Our work follows this line by treating detection as a category-agnostic localization problem tailored to dense shelves.

Scene graph generation. Scene graph generation (SGG) aims to jointly model objects and their relationships, typically requiring explicit relationship annotations and complex multi-head architectures. Comprehensive surveys highlight transformer-based and message-passing approaches that learn object-relation pairs end-to-end (Zhu et al., 2022). In contrast to learned relationship prediction methods such as SGTR (Li et al., 2021), we compute spatial relationships deterministically from geometry after detection. This design significantly reduces annotation requirements and is well-suited to an applied retail setting.

Detection to knowledge graph construction. Several works have explored converting object detections into structured knowledge representations. Prior research has shown that detected objects can be projected into graph structures to support higher-level reasoning and monitoring tasks, such as construction progress tracking and digital twins (Pfitzner et al., 2024; Fang et al., 2017). These approaches motivate our use of graph-based representations, though our focus is on static shelf layouts and planogram-style spatial reasoning rather than temporal analytics.

Embedding-based recognition and few-shot learning. Large vision-language models such as CLIP learn a shared embedding space that enables zero-shot and few-shot recognition via similarity matching (Radford et al., 2021). Instead of fine-tuning a classifier, we treat recognition as a nearest-neighbor retrieval problem in embedding space using a small bank of visual anchors per brand. This non-parametric ap-

proach aligns with prior work on retrieval-based recognition and allows easy extension to new brands without retraining.

3. Problem Setting and Target Brands

Given a retail shelf image containing the target brands, we want to output:

- A set of detected product instances with bounding boxes.
- A brand label for each instance from a targeted beverage brand list or `UNKNOWN`.
- A graph representation that connects each physical item node to a brand concept node and includes spatial relationships.

Our target brands are: {`Coca-Cola, Pepsi, Sprite, Fanta, Mountain Dew, 7 Up`}.

4. Datasets

We use multiple datasets because detection and brand recognition require different supervision.

4.1. Detection Dataset

We use **SKU-110K** to train category-agnostic product detection with a single class (`product`). This dataset closely matches the dense retail shelf setting we target, where shelves contain a large number of tightly packed products with heavy overlap (Goldman et al., 2019). SKU-110K consists of 11,689 images with a total of 1,723,135 labeled objects.There are 3 splits in the dataset: train with 8185 images, test with 2920 images, and val with 584 images (Ninja, 2025).

4.2. Classification Datasets for Visual Anchors

For brand recognition, we construct a dedicated classification dataset to support the CLIP visual anchor method. We initially use the Refrescos dataset from Roboflow, which provides augmented images for the following beverage brands: Coca-Cola, Sprite, Pepsi, and Fanta (Project1, 2025).

During exploratory analysis, we observed a noticeable class imbalance for Pepsi, with fewer high-quality examples compared to other brands. To address this, we supplement the dataset with additional Pepsi images from a separate Roboflow dataset (sj moon, 2022).

Our end-to-end pipeline testing images also contain additional beverage brands, specifically 7 Up and Mountain Dew, which were not sufficiently covered in the initial datasets. To include these brands and improve overall coverage, we incorporate the Cold Drinks dataset from Roboflow (cold drinks, 2023). This dataset required substantial manual filtering, as many images contained large black padding regions or low-quality crops that were unsuitable for anchor construction.

Because the available data for 7 Up and Mountain Dew remained limited relative to the other brands, we apply data augmentation to these classes to increase diversity in lighting conditions, viewpoints, and background context. This helps reduce bias in the anchor bank and improves robustness during recognition.

After preprocessing and augmentation, we split the dataset into training and test sets. The training split is used to construct the visual anchor bank, while the held-out split is used for evaluation. In our final setup, we use approximately **450 training images per class** and **130 test images per class**.

4.3. Pipeline Testing Datasets

For end-to-end qualitative testing of the full pipeline, we use a small set of beverage shelf images from Kaggle datasets, specifically the Cold Drinks Inventory dataset and the Beverage Detection dataset (Faseeh001, 2022; Goheer, 2021). These images are used to validate that the complete detection-to-graph pipeline functions correctly on realistic shelf scenes.

5. Method

5.1. Overview

Our pipeline has four stages:

1. **Detection and Localization:** detect product bounding boxes.
2. **Recognition:** classify each detected crop into a target brand using CLIP visual anchors.
3. **Spatial Relationship Computation:** compute relationships from bounding box geometry.
4. **Graph Construction and Interface:** build semantic and spatial graphs and render them in a UI.

5.2. Stage 1: Detection and Localization

We train and compare three detection architectures using Ultralytics: YOLOv11n (Nano), YOLOv11L (Large), and RT-DETR (Large)(Jocher & Qiu, 2024) (Lv et al., 2023). Each model is fine tuned on SKU-110K and evaluated with standard detection metrics (mAP@50, mAP@50–95, precision, recall). We first run a small hyperparameter search on each model and then do a longer final training run on the best-performing model.

Table 1. Hyperparameter search spaces used for detector tuning. We use different ranges for YOLOv11 (CNN-based) and RT-DETR (Transformer-based) to account for architectural differences in optimization stability and augmentation sensitivity.

Hyperparameter	**YOLOv11n/YOLOv11L Space**	**RT-DETR Space**	**Reasoning**
Learning Rate (`lr0`)	10^{-4} to 10^{-2}	10^{-5} to 10^{-3}	Transformers typically need lower and steadier learning rates to stabilize attention optimization.
Final LR (`lrf`)	0.01 to 1.0	0.01 to 1.0	Controls how aggressively the learning rate decays over training.
Momentum	0.7 to 0.98	0.9 to 0.95	RT-DETR benefits from tighter momentum ranges to reduce training instability in transformer blocks.
Weight Decay	0.0 to 0.001	10^{-4} to $5 \cdot 10^{-4}$	Regularization to reduce overfitting to repetitive retail textures and packaging patterns.
Box Loss Gain	0.02 to 0.2	0.02 to 0.2	Relative weight of the bounding-box regression loss.
Cls Loss Gain	0.2 to 4.0	0.5 to 4.0	RT-DETR often needs stronger classification weighting to encourage cleaner separation under the same objective.
HSV-H Augment	0.0 to 0.05	N/A (default)	Hue shifts help CNN detectors generalize across store lighting; we keep RT-DETR defaults.
Mosaic Augment	0.0 to 1.0 (high)	0.0 to 0.5 (low)	Heavy mosaic distortion can be harder for transformer attention to learn robustly than CNN features.
MixUp Augment	0.0 to 1.0 (high)	0.0 to 0.2 (low)	Large mixup ratios can blur instance boundaries and confuse attention-based matching more than CNNs.

5.3. Hyperparameter Search

We use the native Ultralytics tuning API (PyTorch backend) with distinct search spaces for YOLO style and RT-DETR style models. The main idea is to search learning rate schedules, regularization, and augmentation settings, and then pick the best run based on validation fitness. We summarize the tuning search spaces in Table 1.

5.4. Stage 2: Retrieval-based Recognition via CLIP Visual Anchors

The detection dataset is class agnostic, so we add brand recognition separately. We use CLIP (Radford et al., 2021) as a frozen embedding model and build a **visual anchor bank** for each target brand. Each anchor is a real product image from the classification dataset. At inference time, each detected crop is embedded using CLIP and compared against all anchor embeddings using cosine similarity. We take the nearest neighbor label if the similarity exceeds a threshold, otherwise we output `UNKNOWN`. This is a non parametric few shot recognition method.

Why anchors help. Anchors let us capture multiple packaging appearances for the same brand, which is important for real retail images where lighting and viewpoints change. It also makes the system easy to extend: to add a new brand, we can add anchor images without retraining a classifier head.

Recognition pipeline. For each detected product crop, we apply the following steps:

- Embed the crop using a frozen CLIP image encoder.
- Compare the embedding against a bank of labeled visual anchor embeddings using cosine similarity.
- Assign the label of the nearest anchor if the similarity exceeds a threshold.
- Otherwise, label the crop as `UNKNOWN`.

5.5. Stage 3: Spatial Relationship Computation

After detection and recognition, we compute spatial relationships using bounding box centers and normalized image thresholds. Specifically, we compute:

- `next_to`: items aligned in the same row and within a horizontal distance threshold.
- `left_of`: a directed relation to the nearest neighbor on the right within the same row.
- `above` (optional): a vertical adjacency relation based on relative box center heights, which is computed but not explicitly visualized.

These simple geometric relationships provide a structured representation that supports planogram-style reasoning while remaining easy to interpret and debug.

5.6. Stage 4: Graph Construction and Streamlit Interface

We build a directed graph where each physical item is a node (`Item_i`) with attributes (bounding box, confidence, predicted label). Each target brand also exists as a **brand concept** node. We connect `Item_i` to its brand node with an `is_brand` edge and add spatial edges between items. Figure 1 illustrates the Streamlit interface used to visualize each stage of the pipeline.

We provide two visualizations:

- An interactive semantic plus spatial graph for inspection.
- A planogram flow view based on `left_of` edges that renders a left-to-right layout.

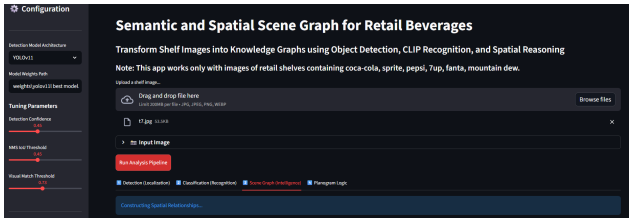

Figure 1. Streamlit interface showing the four stages: detection, recognition, graph generation, and planogram flow.

6. Experiments

6.1. Detection Training Setup

We fine tune YOLOv11n, YOLOv11L, and RT-DETR on SKU-110K. We run 3 to 5 tuning iterations per model with 10 to 15 epochs during tuning, and then perform a longer training run for the best performing configuration.

6.2. Recognition Evaluation Setup

We evaluate brand recognition using a held out classification split. The training split is used to build the anchor bank, and the test split is used to report accuracy. We report standard accuracy and show a confusion matrix.

6.3. End to End Pipeline Testing

We test the full pipeline on a small set of shelf images from Kaggle datasets. We report the fraction of crops confidently classified vs labeled `UNKNOWN` and show counts per detected brand. This is intended to validate the system behavior rather than provide a large scale benchmark.

7. Results

7.1. Detection Results: Model Comparison

YOLOv11 is a convolutional one-stage detector optimized for fast dense object localization, while RT-DETR is a transformer-based detector that models global context using attention mechanisms. We include both architectures to compare CNN-based and transformer-based detection behavior in dense retail scenes.

We report results for the best tuned iteration of each architecture:

Table 2. Detection performance on the test set of SKU-110K (best tuned run per model).

Model	mAP@50	mAP@50–95	Precision	Recall
YOLOv11n	0.8794	0.5460	0.8853	0.7940
YOLOv11L	0.9158	0.5878	0.9030	0.8436
RT-DETR-L	0.8918	0.5720	0.8590	0.8478

Initial experiments were run on NVIDIA L4 and A100 GPUs via Google Colab for rapid iteration, followed by final training and evaluation on an NVIDIA GeForce RTX 5090 using RunPod. This setup allowed stable large-batch training and faster convergence during extended runs.

Figures 2–5 show the training and validation curves for bounding box loss, classification loss, and distribution focal loss, along with detection metrics including precision, recall, mAP@50, and mAP@50–95 over training epochs. The loss curves illustrate how localization and classification objectives evolve during optimization, while the metric plots track detection performance on validation data.

YOLOv11n

We trained YOLOv11n for 10 epochs for 5 iterations with a batch size of 16. The tuning results are present in Table 2.

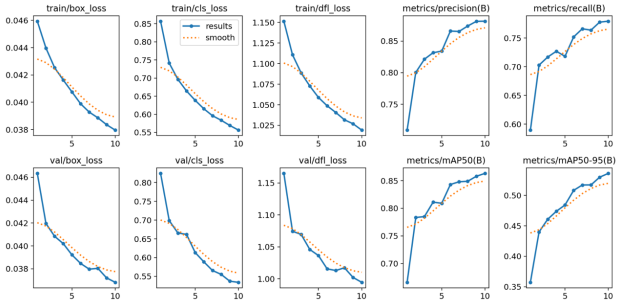

Figure 2. Detection training curves and metrics plots for YOLOV11n.

YOLOv11L

We trained YOLOv11L for 10 epochs for 3 iterations with a batch size of 16. The tuning results are present in Table 2.

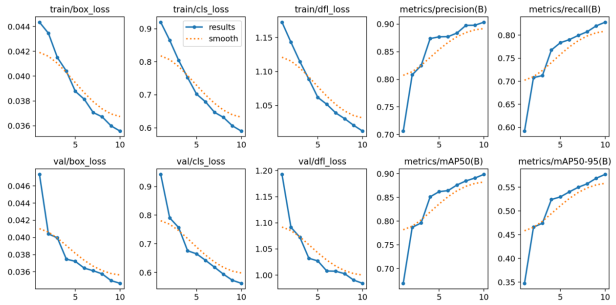

Figure 3. Detection training curves and metrics plots for YOLOV11L.

RT-DETR (Large)

We trained RT-DETR (Large) for 10 epochs for 3 iterations with a batch size of 16. The tuning results are present in Table 2.

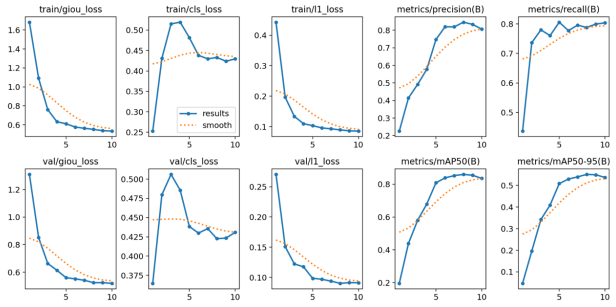

Figure 4. Detection training curves and metrics plots for RT-DETRL.

7.2. Final Detection Training Results

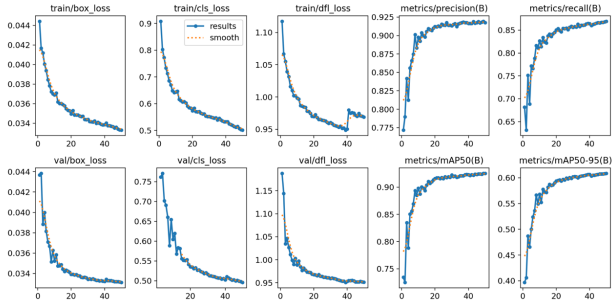

Figure 5. Final YOLOv11L training curves and metrics plots (50 epochs).

After selecting YOLOv11L, we train for 50 epochs using the final tuned parameters:

Final tuned parameters: lr0 = 0.01, lrf = 0.01, momentum = 0.937, weight_decay = 0.0005, box = 0.2, cls = 0.5, hsv_h = 0.015, mosaic = 1.0, mixup = 0.0.

Final test results (50 epochs): mAP@50 = 0.9395, mAP@50–95 = 0.6191, Precision = 0.9171, Recall = 0.8802.

Table 3. Final YOLOv11L results after extended training (50 epochs).

Model	mAP@50	mAP@50–95	Precision	Recall
YOLOv11L (50 ep)	0.9395	0.6191	0.9171	0.8802

7.3. CLIP Recognition Results

Using the visual anchor bank and nearest neighbor matching, we obtain **98.97%** classification accuracy on the held out classification split.

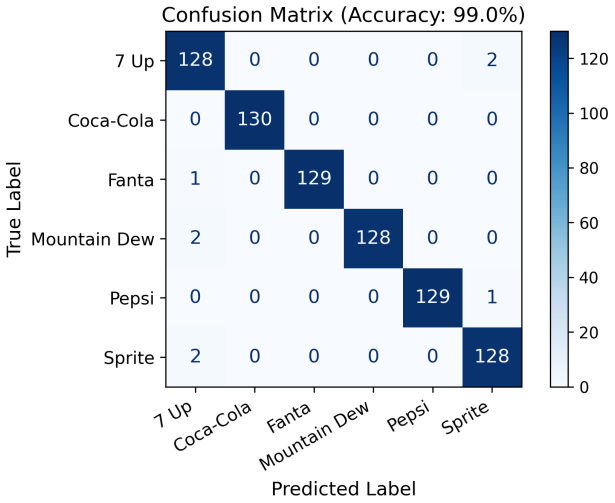

Figure 6. Confusion matrix for CLIP visual anchor recognition on the held out classification test split.

7.4. End to End Pipeline Testing

We evaluate the full pipeline on a small set of shelf images from Kaggle datasets. We report the fraction of detected crops that are confidently labeled vs `UNKNOWN` under a conservative similarity threshold, and we report counts per brand. The average confidence of CLIP was 80.04%

Pipeline thresholds. Detection and graph construction use fixed thresholds chosen empirically for stable behavior across shelf images. We use a YOLO detection confidence threshold of 0.25 and an NMS IoU threshold of 0.45 to

balance recall and duplicate suppression. For recognition, we apply a CLIP cosine similarity threshold of 0.75, below which crops are labeled `UNKNOWN`. Spatial relationships are computed using normalized image-relative thresholds: 0.30× image width for horizontal adjacency, 0.20× image height for same-row alignment, and 0.10× image height for vertical adjacency. These values were selected to tolerate perspective distortion and slanted shelves while preserving interpretable spatial structure.

As shown in Figure 7, we detected close to 16000 products and we were able to classify 67.9% of them.

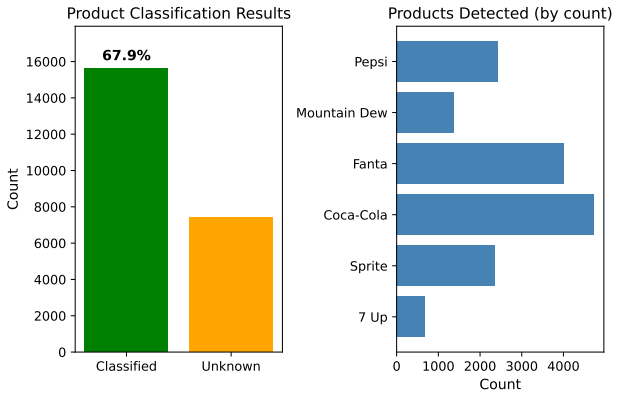

Figure 7. Pipeline testing summary: fraction classified vs unknown and per brand counts.

7.5. Graph Outputs

To illustrate the graph structure, consider a shelf image where multiple instances of the same brand appear on the top row. For example, three Pepsi bottles placed adjacently on the upper shelf are detected as separate item nodes, each linked to the Pepsi brand concept node. Because their bounding boxes are aligned within the same row and fall within the horizontal distance threshold, next_to and left_of edges connect them sequentially. Their vertical position relative to other products places them above lower-shelf items, which is reflected implicitly in the resulting planogram-style flow.

Figure 8 and Figure 9 show the detection and classification stages' output for an input image.

Figure 8. Detection output from stage 1.

Figure 9. Classification output from stage 2.

Based on the results from the Stages 1 and 2, we generate the two graph views.

Semantic plus spatial graph. Brand concept nodes cluster instances and spatial edges capture adjacency.

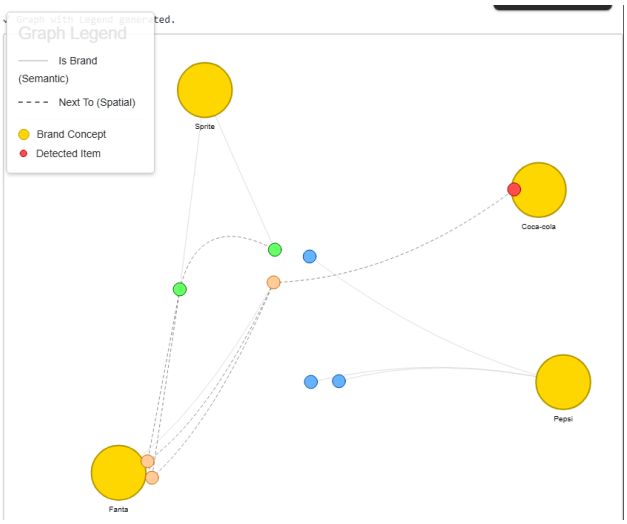

Figure 10. Example semantic plus spatial graph output for a shelf image.

Semantic Planogram flow graph. A left to right view is created using `left_of` edges so the result looks closer to a planogram ordering.

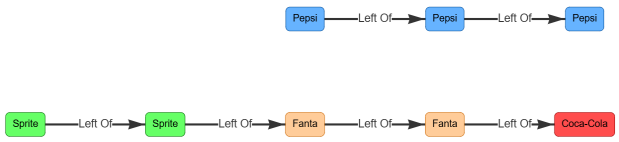

Figure 11. Example planogram flow graph using directed `left_of` relations.

8. Discussion

A key design choice is keeping recognition modular. Detection solves the dense localization problem and recognition solves the semantic labeling problem. The CLIP anchor method works well for a small number of target brands and avoids training a dedicated classifier head. It also makes the system easy to extend by adding more anchor images.

At the same time, the approach depends on anchor coverage. If a brand has packaging styles that are missing from the anchor set, the similarity score can drop and the crop may become `UNKNOWN`. In practice, we found that a conservative threshold reduces incorrect labels and keeps the graph cleaner, but it increases the unknown rate.

9. Limitations

Our system is built to be practical and easy to debug, but it has limitations:

- Spatial relationships are computed from 2D geometry and can be affected by perspective distortion.
- Recognition quality depends on the diversity and quality of anchor images.
- The end to end pipeline testing is currently qualitative and uses a small image set, so it is better viewed as a functional validation.

10. Software and Data

We implement the full pipeline in Python using Ultralytics for detection, Hugging Face Transformers for CLIP, and NetworkX plus PyVis for graph generation. The Streamlit interface is used to visualize intermediate outputs and final graphs. The system uses SKU-110K for detection training, Roboflow beverage datasets for anchor construction, and Kaggle shelf image datasets for pipeline testing.

Impact Statement

This project targets retail auditing automation. A potential positive impact is reducing manual planogram checks and enabling faster shelf analysis. Potential risks include biased performance across store types due to domain shift and privacy concerns if deployed in real stores. Our current work is limited to research and class project experimentation with public datasets.

References

ChocoWu. Awesome-scene-graph-generation. GitHub Repository, 2025. URL https://github.com/ChocoWu/Awesome-Scene-Graph-Generation. A curated collection of scene graph generation literature and resources. Retrieved December 5, 2025.

cold drinks. cold drinks dataset. https://universe.roboflow.com/cold-drinks-cftkw/cold-drinks-qki32, may 2023. URL https://universe.roboflow.com/cold-drinks-cftkw/cold-drinks-qki32. visited on 2025-12-5.

Fang, Y., Kuan, K., Lin, J., Tan, C., and Chandrasekhar, V. Object detection meets knowledge graphs. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 1661–1667, 2017. doi: 10.24963/ijcai.2017/230. URL https://doi.org/10.24963/ijcai.2017/230.

Faseeh001. Cold drinks inventory dataset. Kaggle Dataset, 2022. URL https://www.kaggle.com/datasets/faseeh001/cold-drinks-inventory-dataset. Retrieved December 5, 2025.

Ferré, A. Shelf product identifier. GitHub Repository, 2019. URL https://github.com/albertferre/shelf-product-identifier. Retrieved December 5, 2025.

Goheer, A. Beverage detection dataset. Kaggle Dataset, 2021. URL https://www.kaggle.com/datasets/ayeshagoheer/beverage-detection-dataset. Retrieved December 5, 2025.

Goldman, E., Herzig, R., Eisenschtat, A., Ratzon, O., Levi, I., Goldberger, J., and Hassner, T. Precise detection in densely packed scenes. *arXiv preprint arXiv:1904.00853*, 2019. URL https://arxiv.org/abs/1904.00853.

Jocher, G. and Qiu, J. Ultralytics yolo11, 2024. URL https://github.com/ultralytics/ultralytics.

Li, R., Zhang, S., and He, X. SGTR: end-to-end scene graph generation with transformer. *CoRR*, abs/2112.12970, 2021. URL https://arxiv.org/abs/2112.12970.

Lv, W., Xu, S., Zhao, Y., Wang, G., Wei, J., Cui, C., Du, Y., Dang, Q., and Liu, Y. Detrs beat yolos on real-time object detection, 2023.

Neau, M., Santos, P. E., Bosser, A.-G., Buche, C., and Sugimoto, A. React: Real-time efficiency and accuracy compromise for tradeoffs in scene graph generation, 2025. URL https://arxiv.org/abs/2405.16116.

Ninja, D. Visualization tools for sku110k dataset. https://datasetninja.com/sku110k, dec 2025. URL https://datasetninja.com/sku110k. visited on 2025-12-14.

Pannoy, N., Nonsiri, S., and Makdee, S. Object detection for retail product recognition. In *2024 9th International Conference on Business and Industrial Research (ICBIR)*, pp. 1547–1552, 2024. doi: 10.1109/ICBIR61386.2024.10875711.

Pfitzner, F., Braun, A., and Borrmann, A. From data to knowledge: Construction process analysis through continuous image capturing, object detection, and knowledge graph creation. *Automation in Construction*, 164: 105451, 2024. ISSN 0926-5805. doi: https://doi.org/10.1016/j.autcon.2024.105451. URL https://www.sciencedirect.com/science/article/pii/S0926580524001870.

Project1. Refrescos-detector dataset. https://universe.roboflow.com/project1-w4jns/refrescos-detector-2napt, aug 2025. URL https://universe.roboflow.com/project1-w4jns/refrescos-detector-2napt. visited on 2025-12-5.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

sj moon. pepsi dataset. https://universe.roboflow.com/sj-moon-drdsa/pepsi-gmqid, dec 2022. URL https://universe.roboflow.com/sj-moon-drdsa/pepsi-gmqid. visited on 2025-12-5.

thedatasith. Sku-110k annotations. Kaggle Dataset, ? 2023. URL https://www.kaggle.com/datasets/thedatasith/sku110k-annotations. Retrieved December 5, 2025.

Zhu, G., Zhang, L., Jiang, Y., Dang, Y., Hou, H., Shen, P., Feng, M., Zhao, X., Miao, Q., Shah, S. A. A., and Bennamoun, M. Scene graph generation: A comprehensive survey. *CoRR*, abs/2201.00443, 2022. URL https://arxiv.org/abs/2201.00443.