Know What You See: Grounded localization of product components

Manan Soni
sonmanav@amazon.com
Amazon

Abinesh Kanagarajan
abinesk@amazon.com
Amazon

Shyam Mohan
shyaamm@amazon.com
Amazon

Abstract

Many real-world decisions about products (e.g. how they function, how they should be used) depend on their components rather than the object as a whole. Accurately identifying product component has applications like automated defect detection, visual spare-parts search, and verified assembly. However, existing object detectors treat components as isolated objects, ignoring their inherent structure. We propose Know What You See (KWYS), where we localize components by grounding them using a textual knowledge base (e.g., manuals or web descriptions). KWYS converts raw text into a hierarchical component taxonomy, which then guides an open-vocabulary object detector using a hierarchical verification algorithm. We evaluate on 1,000 product images across 5 diverse categories, improving component localization accuracy by 11% along with reducing component hallucinations by 25%.

1 Introduction

Understanding a product's functionality requires more than recognizing the product as a whole; it requires accurately localizing its individual components. This capability enables practical applications such as automatically verifying component-based attributes (e.g., whether a product includes a charger or armrests), improving visual search using components (e.g., finding laptops with a CD drive), and checking quality and compliance (e.g., detecting missing protective guards). However, this task is difficult as product components are often visually similar, small, or partially occluded. A specific part might only be distinguishable by its location on a larger assembly. Furthermore, the sheer variety of products in a real-world catalog means that relying on visual features alone is insufficient to distinguish these fine-grained details.

Current approaches generally fall into two categories. Supervised detection methods require training data with bounding box annotations for every component, making it hard to scale across the entire catalog. More recently, Open-World Object Detection (OWOD) models allow for zero-shot object detection using free-text prompts. While powerful, these models typically treat components as independent items in a flat list. They lack the context of how the product is structured. For example, an open-world detector might identify a "handle" on a background object rather than the product itself, or it might tag a "button", ignoring the fact that the specific product model does not have one.

To address these limitations, we propose KWYS (Know What You See), a framework that grounds product components by converting unstructured textual documentation into a structured, hierarchical taxonomy that guides open-vocabulary object detection. By enforcing hierarchy during inference, KWYS improves localization accuracy by 11%. We evaluate our approach on 1,000 images across five product categories and demonstrate consistent gains across all detection backbones.

2 Related Work

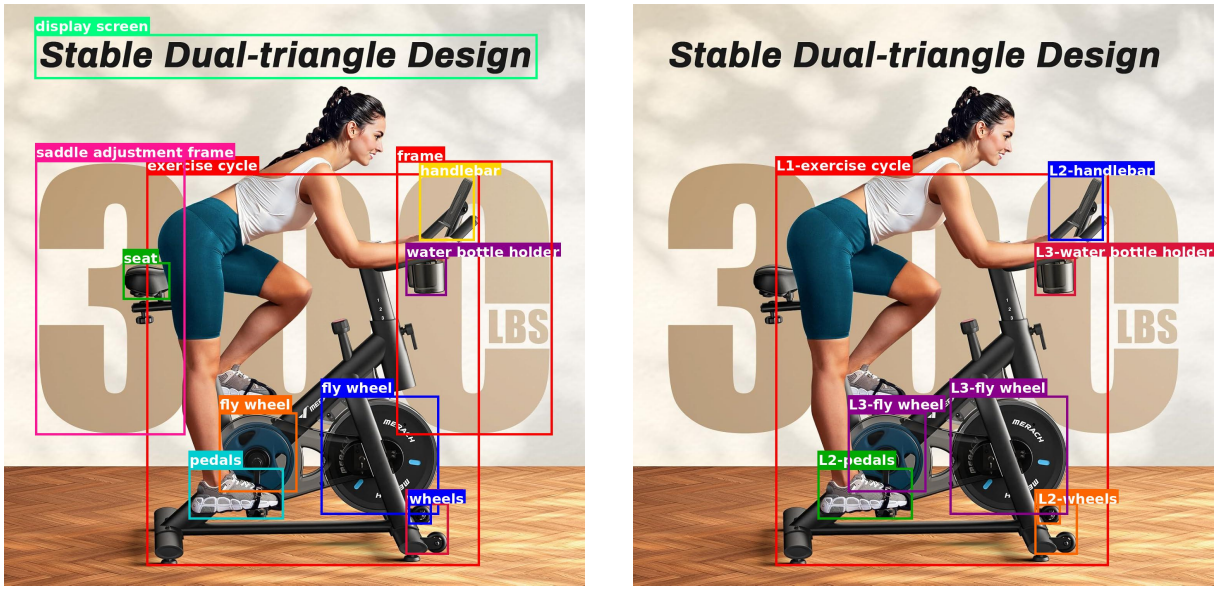
2.1 Component Localization

Previous approaches to component localization have been predominantly domain-specific, focusing on particular applications such as car parts (Pasupa et al., 2021), (Cui et al., 2024), PCB components (Li et al., 2019), and cabinet assembly (Bründl et al., 2023). While effective within their domains, these methods require extensive annotation and struggle with generalization to novel component types not seen during training.

2.2 Open World Object Detection

Open world object detection (OWOD) breaks the "closed-set" assumption of traditional detectors, allowing models to identify objects described by arbitrary text prompts. Early vision-language models like OWL-ViT (Chhipa et al., 2024) and GLIP

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pages 1077–1088



(a) Without hierarchical localization

(b) With hierarchical verification

Figure 1: Component localization for an exercise cycle (a) without and (b) with KWYS. Refer Section A.1 for corresponding the taxonomy and Section A.3 for additional qualitative examples.

v2 (Zhang et al., 2022) pioneered this by aligning image-text feature spaces through large-scale pre-training. This paradigm was advanced by Grounding DINO (1.5) (Ren et al., 2024), which uses deep cross-modality fusion to achieve state-of-the-art zero-shot performance.

These developments have significantly reduced dependence on annotated datasets, enhancing generalization across categories. However, applying OWOD to component detection is challenging due to the hierarchical nature of components, where the context of a component's parent may influence its placement and function. We build on these recent advancements, tailoring our solution specifically for component localization.

3 Problem Statement

Given a product p along with its knowledge base K_p, extract a component taxonomy $\mathcal{T}_p$ defined as:

$$\mathcal{T}_p = \left\{ c_i^{(l)} \mid l, i \in \mathbb{N}, \right\} \tag{1}$$

where $c_i^{(l)}$ denotes the i-th component at level l in the taxonomy. The taxonomy can be of arbitrary depth, with each level $l = 1, 2, 3, \ldots$ representing more granular components.

Next, given a product image I_p, and a component taxonomy $\mathcal{T}_p$, localize each component $c_i^{(l)} \in \mathcal{T}_p$ in I_p if it exists.

4 Methodology

Our goal is to find components in a product image by using the information found in K_p. We approach this in three phases: 1) knowledge base processing, 2) proposal bounding box generation, 3) hierarchical verification.

Figure 2 illustrates our three-step workflow. First, Knowledge Base Processing Section 4.1 converts unstructured text—like a PDF manual or description into a structured taxonomy. This tree define where each part is located relative to other parts (e.g., the Battery is inside the Remote).

Second, Zero-Shot Proposal Generation Section 4.2 uses an off-the-shelf open-world object detector to find all components in the image.

Finally, Hierarchy-Aware Localization Section 4.2.2 validates the raw detections against the product's taxonomy. By strictly enforcing the parent-child relationship constraints, we filter out false positives. It can guide any open-world detector to be more precise and grounded, effectively improving performance without the need for retraining or manual annotation.

All taxonomies used throughout the paper, are generated using Claude Haiku [1] as the LLM backbone and Grounding DINO (Ren et al., 2024) as the object detection backbone.

[1] https://www.anthropic.com/claude/haiku

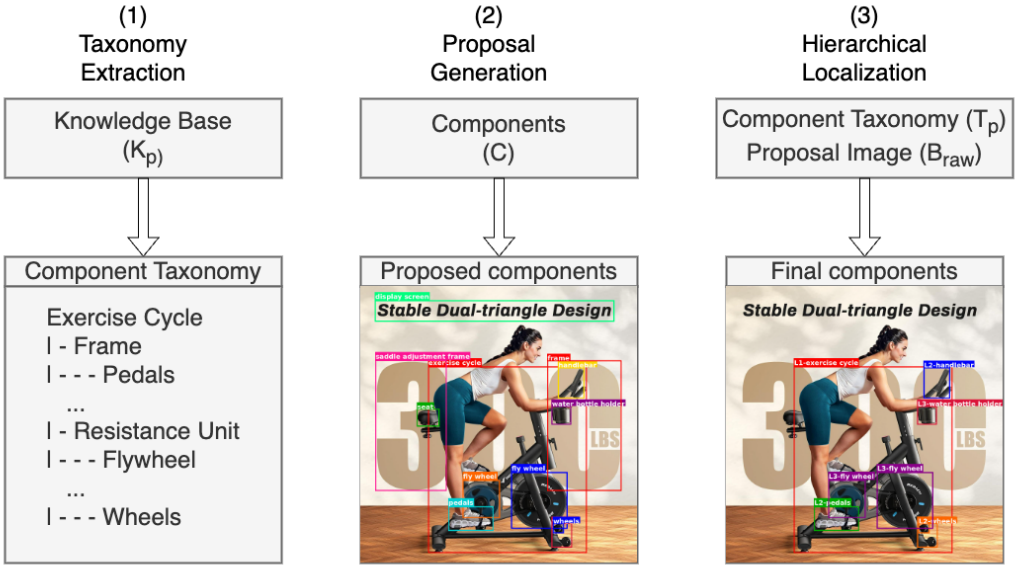


Figure 2: High level approach

4.1 Knowledge Base Processing

To ground visual localization in product knowledge, we first convert the textual source (usually unstructured) into a structured, hierarchical taxonomy. Large Language Models (LLMs) are well suited for this task due to their ability to parse complex technical jargon, resolve co-references across document sections, and infer relationships from descriptions. We decouple the process into two stages: *Component Extraction*, which mines all potential parts along with their descriptions and relationships, and *Taxonomy Creation*, which structures these into a taxonomy.

4.1.1 Component Extraction

Product documentation (E.g. PDF manuals or web descriptions) is often dense and scattered. To handle this, we process the text in overlapping chunks. For each segment, we prompt the LLM to extract a tuple t_i = (name, description, relationship) for every potential component. The *name* identifies the part (e.g., "Air Filter"), the *description* captures visual attributes (e.g., "plastic mesh grid"), and the *relationship* notes any mentioned hierarchical context (e.g., "inside the front cover"). This step yields a comprehensive but noisy list of candidates $\mathcal{C}_{raw}$, prioritizing high recall.

4.1.2 Hierarchical Taxonomy Creation

Next, we consolidate the raw tuples in $\mathcal{C}_{raw}$ into a taxonomy $\mathcal{T}_p$. In this step, the LLM acts as an intelligent filter: it de-duplicates synonymous terms (merging "Power Key" and "On/Off Switch"), removes abstract non-visual concepts (like "Warranty" or "Software Settings"), and organizes the remaining physical parts into a parent-child tree structure based on containment. For example, *Batteries* are structured as children of the *Remote Control*. Refer to Appendix section A.1 for a sample taxonomy. Empirically, we find that enforcing this structure during extraction reduces downstream visual hallucinations by 25% compared to direct taxonomy creation.

4.2 Component Localization

Once the taxonomy $\mathcal{T}_p$ is constructed, we ground these components in the product image I_p. This is a two-stage process: generating proposals and then refining them via hierarchical validation.

4.2.1 Proposed Components Generation

We first flatten the taxonomy $\mathcal{T}_p$, along with the image I, to an open-vocabulary object detector. This step uses the detector's zero-shot capabilities to generate a set of candidate bounding boxes $\mathcal{B}_{raw}$. This stage prioritizes recall, extracting all bounding boxes that the model is able to identify.

4.2.2 Hierarchical verification

We filter the proposals ($\mathcal{B}_{raw}$) using Algorithm 1. Starting from the root node (the entire image), we validate each component against its parent's bound-

ing box in a top-down manner. We enforce that a child component is retained only if it lies mostly within its parent's region using an *Intersection-over-Child* (IoC) score. Given a child bounding box b_{child} and its parent box b_{parent}, IoC is defined as:

$$\text{IoC}(b_{child}, b_{parent}) = \frac{\text{Area}(b_{child} \cap b_{parent})}{\text{Area}(b_{child})}. \quad (2)$$

A child detection is retained only if $\text{IoC}(b_{child}, b_{parent}) \geq \delta$. Detections that fall largely outside the parent's region are discarded.

Note that if an intermediate component is not present, it inherits its parent's box. This ensures the search space is preserved for deeper leaf nodes (e.g., finding a "Button" inside a missed "Remote"). However, this inheritance is used solely for search propagation. These inherited boxes are excluded from the final output to ensure that only detected components are finally surfaced.

Algorithm 1 Hierarchical Verification Algorithm

Require: Image I, Taxonomy $\mathcal{T}_p$, Threshold $\delta = 0.8$
  function DETECT(I, c)
    return bounding box b of component c in I
  end function
  $b_{root} \leftarrow \text{ImageBounds}(I)$
  $\mathcal{B} \leftarrow \{(c_{root}, b_{root})\}$ ▷ Initialize with Root
  for level $l = 0$ to L **do**
    for $(c_{parent}, b_{parent}) \in \mathcal{B}$ **do**
      for $c_{child} \in \text{Children}(c_{parent})$ **do**
        $b_{child} \leftarrow \text{DETECT}(I, c_{child})$
        if $b_{child} \neq \varnothing$ **then**
          if $\frac{\text{Area}(b_{child} \cap b_{parent})}{\text{Area}(b_{child})} \geq \delta$ **then**
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{(c_{child}, b_{child})\}$
          else ▷ Inconsistent: Discard
            continue
          end if
        else ▷ Missed: Inherit
          $\mathcal{B} \leftarrow \mathcal{B} \cup \{(c_{child}, b_{parent})\}$
        end if
      end for
    end for
  end for
  return $\mathcal{B}$

5 Experimental Design

5.1 Dataset

To evaluate fine-grained component localization, we curate a dataset derived from the e-manuals corpus (Nandy et al., 2021), augmenting its PDF manuals with product images from the Flipkart dataset (PromptCloud, 2017). We sample 250 products across five categories - Household Appliances, Multimedia, Furniture, Fitness, and Tools - selected to maximize diversity (e.g., Household Appliances spans washing machines and vacuum cleaners; Multimedia spans televisions and laptops). Each product is paired with up to 4 images capturing varied angles and real-world backgrounds, yielding 1,000 images in total.

Ground Truth Generation. Since the e-manuals corpus lacks fine-grained visual annotations, ground truth bounding boxes were established via manual annotation. Domain experts annotated each image using candidate bounding boxes as reference, producing validated component boxes used for mAP evaluation. While the dataset scale (250 products) is intentionally scoped for controlled evaluation, products with sparse or incomplete documentation represent an open challenge we discuss in Section 8.

5.2 Evaluation Metrics

We evaluate the textual and visual stages of our pipeline separately. To assess the quality of knowledge extraction, we measure the **Taxonomy Hallucination Rate (THR)**. For each component node in the generated taxonomy $\mathcal{T}_p$, we retrieve the top-5 most relevant text chunks from the manual. An auditor then validates the node against this context using a 3-point Likert scale: 0 if there is an exact textual match, 0.5 if the component is inferable but not explicit, and 1.0 if there is no supporting evidence. The final THR is calculated as the mean across all components.

For visual localization, we report **Mean Average Precision (mAP@50)** following standard detection protocol (Everingham et al., 2010). Since the raw dataset lacks fine-grained annotations, we manually annotate ground-truth bounding boxes for our dataset. We evaluate AP for the specific components defined in each product's taxonomy, considering a prediction a True Positive (TP) only if it overlaps a ground-truth box of the same class with IoU ≥ 0.5.

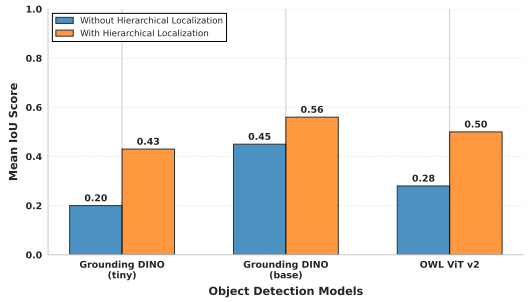


Figure 3: Performance comparison of Grounding DINO and OWL-ViT backbones, with and without hierarchical localization.

6 Experiments

6.1 Impact of Hierarchical Validation

We evaluate Algorithm 1 on three state-of-the-art open-world object detection backbones: Grounding DINO (Tiny), Grounding DINO (Base), and OWL-ViT (v2).

Figure 3 reports localization accuracy (mAP@50) for each model with and without hierarchical validation. Enforcing hierarchical constraints yields consistent gains across all architectures, with an average improvement of 18.6%. Notably, even the strongest backbone, Grounding DINO (Base), benefits from hierarchical validation, achieving an 11% improvement in localization accuracy. We use Grounding DINO (Base) as the default backbone for all experiments and figures. These consistent gains across multiple backbones indicate that KWYS is largely detector-agnostic and can be integrated with different OWOD architectures.

6.2 Taxonomy Hallucination Analysis

To evaluate effectiveness in reducing hallucinations, we compare it against two baselines: standalone LLM inference (P_{base} A.2.3) and retrieval augmented generation (RAG) over the product manual (P_{RAG} A.2.4).

As shown in Table 1, KWYS approach outperforms both baselines, reducing the Taxonomy Hallucination Rate (THR) by 25%.

6.3 Sensitivity to Containment Threshold (δ)

Algorithm 1 relies on the containment threshold δ to distinguish valid child components from spatially invalid detections. We study the effect of varying δ on detection performance (mAP@50) using the Electronics subset.

Table 1: Taxonomy extraction performance. We compare the average number of extracted components and the Taxonomy Hallucination Rate (THR) against baselines.

Method	**Avg. Components**	**THR**
Standalone LLM (P_{base})	19.3	30%
RAG (P_{RAG})	7.2	20%
Structured Extraction (KWYS)	60.7	**5%**

Threshold (δ)	0.2	0.4	0.6	**0.8**	0.9	1.0
mAP@50	0.46	0.49	0.51	**0.56**	0.54	0.52

Table 2: Impact of spatial containment threshold δ on localization accuracy.

As shown in Table 2, increasing δ improves performance, with accuracy peaking at $\delta = 0.8$. When $\delta \to 0$, hierarchical validation behaves like a vanilla object detector. Conversely, overly strict thresholds ($\delta \to 1$) reject valid components due to minor overflow, or irregular shapes, leading to degraded performance.

This sensitivity highlights a key limitation of the bounding-box-based containment assumption, which may fail for detached, elongated, or obliquely oriented components.

6.4 Ablation: Impact of Knowledge Base Source

Industrial products are typically accompanied by multiple textual knowledge sources. We analyze how the choice of source - PDF user manuals versus web product descriptions - affects KWYS. We were able to get $\tilde{5}0$ products per category where both the web description and the PDF manual were available.

Knowledge Base Source	**Components**	**Depth**
Product Descriptions	2.4	1.3
User Manuals	51.9	2.5

Table 3: Average impact of knowledge base source on taxonomy extraction across all product categories.

As shown in Table 3, product descriptions produce sparse and shallow taxonomies (2.4 components, depth 1.3), reflecting their marketing-oriented focus on a small number of high-level features. In contrast, user manuals yield significantly richer and deeper hierarchies (51.9 components, depth 2.5), as they are designed to support usage and maintenance and therefore describe fine-grained components and relationships.

7 Conclusion

Building on recent advances in open-world object detection, we introduce KWYS, a framework for zero-shot component localization in product images that requires no manual annotation. By grounding visual components in unstructured product knowledge such as product manuals, KWYS improves end-to-end localization accuracy by 11% while reducing component hallucinations by 25%.

We also show the real-world applicability of KWYS through evaluation on diverse product images, including challenging cases with background clutter, occlusions, and varying angles. Qualitative results across product categories, with representative examples provided in Appendix A.3, show that the proposed approach remains robust under these conditions.

8 Limitations and Future Work

Despite the relative gains achieved by KWYS, absolute localization accuracy remains modest, reflecting the intrinsic difficulty of fine-grained component detection in open-world settings. Current OWOD models are pretrained primarily on coarse objects and lack supervision for granular, product-specific components, which are rarely localized in large-scale vision-language corpora.

While KWYS mitigates this limitation by injecting hierarchical structure derived from product knowledge, the hierarchical containment assumption (*child* $\subset$ *parent*) may fail when components appear outside their assembly context (e.g., detached accessories or exploded views), leading to valid detections being discarded. Additionally, reliance on bounding boxes limits precision for irregularly shaped components (see Appendix A.4).

Extending the framework to segmentation masks is a promising direction, enabling finer-grained containment validation and more robust handling of irregular component geometries. Additional directions include improving robustness to incomplete or noisy knowledge base text, exploring prompt tuning strategies, and benchmarking against vision-language models such as GPT-4o and Gemini.

9 Ethics Statement

The data used in this work consists of product manuals and images collected from publicly available sources. We do not claim ownership of the dataset. The data is used solely for research and evaluation purposes.

References

Patrick Bründl, Benedikt Scheffler, Micha Stoidner, Huong Nguyen, Andreas Baechler, Ahmad Abrass, and Jörg Franke. 2023. Semantic part segmentation of spatial features via geometric deep learning for automated control cabinet assembly. *J. Intell. Manuf.*, 35(8):3681–3695.

Prakash Chandra Chhipa, Kanjar De, Meenakshi Subhash Chippa, Rajkumar Saini, and Marcus Liwicki. 2024. Open-vocabulary object detectors: Robustness challenges under distribution shifts. *arXiv preprint arXiv:2405.14874*.

Jiahui Cui, Zhongcheng Wu, and Xinkuang Wang. 2024. Vehicle appearance parts identification based on instance segmentation algorithms. In *Proceedings of the International Conference on Algorithms, Software Engineering, and Network Security*, ASENS '24, page 306–311, New York, NY, USA. Association for Computing Machinery.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *Preprint*, arXiv:0909.5206.

Jing Li, Jinan Gu, Zedong Huang, and Jia Wen. 2019. Application research of improved yolo v3 algorithm in pcb electronic component detection. *Applied Sciences*, 9:3750.

Abhilash Nandy, Soumya Sharma, Shubham Maddhashiya, Kapil Sachdeva, Pawan Goyal, and NIloy Ganguly. 2021. Question answering over electronic devices: A new benchmark dataset and a multi-task learning based QA framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4600–4609, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kitsuchart Pasupa, Phongsathorn Kittiworapanya, Napasin Hongngern, and Kuntpong Woraratpanya. 2021. Evaluation of deep learning algorithms for semantic segmentation of car parts. *Complex Intelligent Systems*, 8.

PromptCloud. 2017. Flipkart products. `https://www.kaggle.com/datasets/PromptCloudHQ/flipkart-products`. Accessed: 2026; Dataset with 20k Flipkart product listings including descriptions and images.

Tianhe Ren, Qing Jiang, Shilong Liu, Zhaoyang Zeng, Wenlong Liu, Han Gao, Hongjie Huang, Zhengyu Ma, Xiaoke Jiang, Yihao Chen, and 1 others. 2024. Grounding dino 1.5: Advance the" edge" of open-set object detection. *arXiv preprint arXiv:2405.10300*.

Haotian Zhang, Pengchuan Zhang, Xiaowei Hu, Yen-Chun Chen, Liunian Harold Li, Xiyang Dai, Lijuan Wang, Lu Yuan, Jenq-Neng Hwang, and Jianfeng Gao. 2022. Glipv2: unifying localization and vl understanding. In *36th Conf. Neural Inf. Process. Syst. NeurIPS*.

A Appendices

A.1 Sample component taxonomy

We present the generated taxonomy for the MERACH Exercise Bike from figure 1.

```
[L1] Exercise Cycle
  [L2] Frame
    [L3] Main Frame
    [L3] Handlebar Post
    [L3] Seat Post
    [L3] Front Stabilizer
    [L3] Rear Stabilizer
    [L3] Base
  [L2] Saddle
    [L3] Saddle Adjustment Structure
      [L4] Saddle Adjustment Frame
      [L4] Saddle Adjustment Bracket
      [L4] Saddle Bar
      [L4] Saddle Lock Knob
    [L3] Seat Adjustment Mechanism
      [L4] Seat Adjustment Slider
  [L2] Handlebar
    [L3] Handlebar Stem
    [L3] Handlebar Mount
    [L3] Handlebar Adjustment Wheel
    [L3] Handlebar Adjuster
    [L3] Height Adjustment Knob
  [L2] Seat
  [L2] Pedals
  [L2] Wheels
  [L2] Resistance Unit
    [L3] Fly wheel
    [L3] Resistance Adjustment Knob
    [L3] Tension Adjustment Knob
    [L3] Belt
  [L2] Console
    [L3] Display Screen
    [L3] Monitor
    [L3] Heart Rate Sensor
      [L4] Heart Rate Sensor Cable
    [L3] Speed Sensor
  [L2] Accessories
    [L3] Water Bottle Holder
    [L3] Tablet Mount
```

A.2 Prompts

This section defines all prompts used throughout the paper. Contains prompts for component extraction, taxonomy creation, baseline taxonomy creation and RAG-based taxonomy creation.

A.2.1 $P_{extract}$ - Component extraction prompt

```
You will analyze product manual text to extract
    components of product. Here is the product
    information:

<product_info>
Category: {PRODUCT_CATEGORY}
Type: {PRODUCT_TYPE}
Name: {PRODUCT_NAME}
Variant: {PRODUCT_VARIANT}
</product_info>

Here is the manual text chunk to analyze:
<manual_text>
{MANUAL_CHUNKED_TEXT}
</manual_text>

Analyze the text to extract physical components
    of the product. A component must be a
    physical, tangible part of the product
    itself

For each component identify:
1. the exact snippet from the text that mentions
     it in <snippet> tags
2. Explain why you consider it a component in <
    reasoning> tags
3. Tell its relationship with overall product
    such as where it is attached to or its
    parent component in <relationship tags>.
4. List the components in <component> tags

If the manual text appears unrelated to the
    product or you are unable to identify any
    valid components, respond with: <
    no_components>No physical product components
     could be identified in this text chunk.</
    no_components>

Format your complete response inside <analysis>
    tags.

Example response format:
<analysis>

<component_extraction>
<snippet>"...text mentioning component..."</
    snippet>
<reasoning>Explanation of why this is a physical
     product part</reasoning>
<relationship>information on parent component or
     part</relationship>
<component>Component Name</component>
</component_extraction>

<!-- Repeat for each component found seperated
    by newline -->
</analysis>

Begin your analysis now
```

A.2.2 P_{tax} - Component taxonomy creation prompt

```
You are a precision-focused technical
    documentation expert. You will receive a
    list of components related to a flat screen
    TV user manual. Your task is to create a
    FLEXIBLE hierarchical organization where:

1. CRITICAL RULES:
   - Each component MUST appear EXACTLY ONCE in
    the hierarchy
   - ALL components MUST be placed in the
    hierarchy
   - NO new components can be created
   - NO component can be repeated
   - Each branch can vary in depth from 1 to 7
    levels
   - Branches do not need to have the same depth
   - A component can be a leaf node at any level
     (1-7)
```

```
2. Organization Guidelines:
   - More general/inclusive components should be
     higher in hierarchy
   - More specific/detailed components should be
     lower in hierarchy
   - Each component must logically belong under
     its parent
   - Branch depth should be determined by
     natural component relationships
   - Don't force unnecessary levels if logical
     grouping is complete

3. Output Format:
   [L1] Component A
     [L2] Component B
       [L3] Component C (can be leaf)
   [L1] Component D
     [L2] Component E (can be leaf)
   [L1] Component F (can be leaf)

4. Quality Checks (run these before finalizing):
   - Count input components vs. placed
     components (must match)
   - Verify each component appears only once
   - Confirm logical parent-child relationships
   - Ensure no branch exceeds 7 levels
   - Verify each branch has at least one
     component

Please create a flexible hierarchical structure
     for the following components, allowing
     natural depth progression while adhering to
     ALL rules above:
{subtopics}
```

A.2.3 P_{base} - Baseline LLM prompt

```
You will analyze product information to identify
     physical components and create a
     hierarchical taxonomy of these components.
Here is the product information to analyze:

<product_info>
Category: {PRODUCT_CATEGORY}
Type: {PRODUCT_TYPE}
Name: {PRODUCT_NAME}
Variant: {PRODUCT_VARIANT}
</product_info>

First, analyze the product information in your
     scratchpad. Break down the information
     systematically:

1. Identify:
   - Understand the product and visualize it
   - Think about its possible parts at high
     level to granular most

2. For each potential component, validate
     whether it meets ALL these criteria:
   - Is it a tangible, physical part?
   - Is it visible/observable?
   - Is it actually part of the product itself?

3. Explicitly exclude these types of elements:
   - Features or functionalities (e.g., "
     automatic shut-off")
   - Properties (e.g., "waterproof", "durable")
   - Software or digital elements
   - Materials (e.g., "plastic", "metal")
   - Positions or locations (e.g., "front", "
     back")
   - Systems or mechanisms (e.g., "cooling
     system")

4. Consider hierarchical relationships between
     components

Create a hierarchical taxonomy of the validated
     components using these rules:
- Use L1, L2, L3, etc. to indicate hierarchy
     levels
- It can have any number of any levels
- Each subsequent level (L2, L3, etc.)
     represents components that are part of the
     parent level
- Create as many levels as needed based on
     component relationships
- Each component should only appear once in the
     hierarchy

Format your output like this example:
<taxonomy>
[L1] Television
  [L2] Display Panel
    [L3] LCD Screen
    [L3] Backlight
  [L2] Remote Control
    [L3] Buttons
      [L4] Power Button
      [L4] Volume Buttons
    [L3] Battery Compartment
</taxonomy>

Think deep and provide hierarchical taxonomy in
     taxonomy xml.
```

A.2.4 P_{RAG} - Baseline LLM prompt with RAG

```
You will analyze product knowledge base to
     extract components of product.
Also , You will be creating a dynamic-level
     hierarchical taxonomy for product components
     .

Here is the product information:

<product_info>
Category: {PRODUCT_CATEGORY}
Type: {PRODUCT_TYPE}
Name: {PRODUCT_NAME}
Variant: {PRODUCT_VARIANT}
</product_info>

Here is the manual text chunk to analyze:
<knowledge_base>
{KNOWLEDGE_BASE}
</knowledge_base>

Analyze the text in knowledge base to extract
     physical components of the product. A
     component must be:
- A tangible, physical part of the product
     itself
- Visible/observable
```

```
For each component you identify, list the
    component name in <component> tags

Follow these steps to create the taxonomy:

Remove duplicate and standardize components:
- Components with the same function but
    different names should be consolidated
- Components that are part of the same larger
    component should be grouped
- Consider hierarchical relationships between
    components

Create a hierarchical taxonomy of the validated
    components using these rules:
- Use L1, L2, L3, etc. to indicate hierarchy
    levels
- It can have any number of any levels
- Each subsequent level (L2, L3, etc.)
    represents components that are part of the
    parent level
- Create as many levels as needed based on
    component relationships
- Each component should only appear once in the
    hierarchy
- Do not return anything in parentheses if you
    are standardizing something.

Format your output like this example:
<taxonomy>
[L1] Television Unit
  [L2] Display Panel
    [L3] LCD Screen
    [L3] Backlight
[L1] Additional Accesories
  [L2] Remote Control
    [L3] Buttons
      [L4] Power Button
      [L4] Volume Buttons
    [L3] Battery Compartment
</taxonomy>

Begin your response now
```

A.3 Sample inference images

We present qualitative inference results across five product categories. Figures 4, 5, 6, and 7 show representative examples from the household, multimedia, furniture, and fitness categories, respectively.

A.4 Failure Case Analysis

We analyze the two most common failure modes of KWYS in Figures 8 and 9. For clarity, we visualize only incorrect bounding boxes, omitting correct detections where necessary.

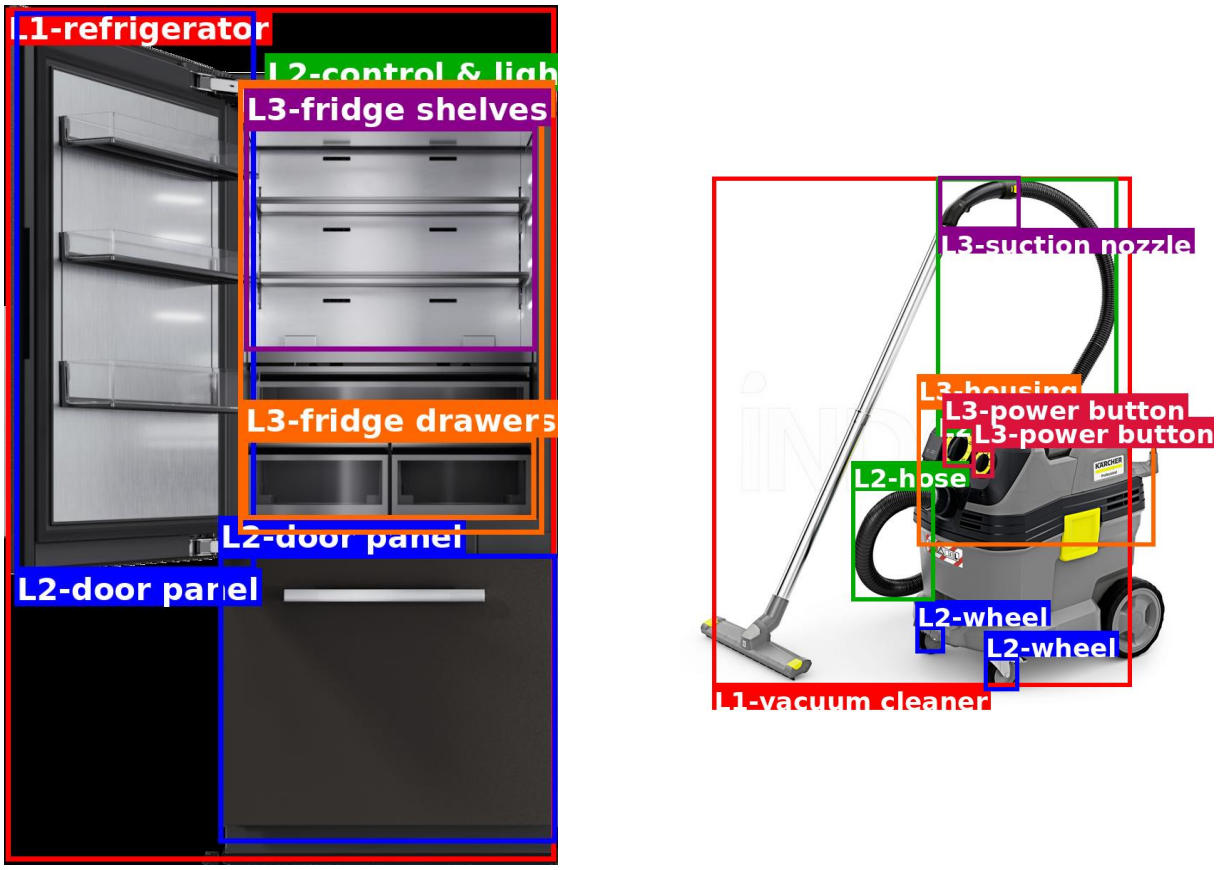


Figure 4: left: refrigerator, right: vacuum

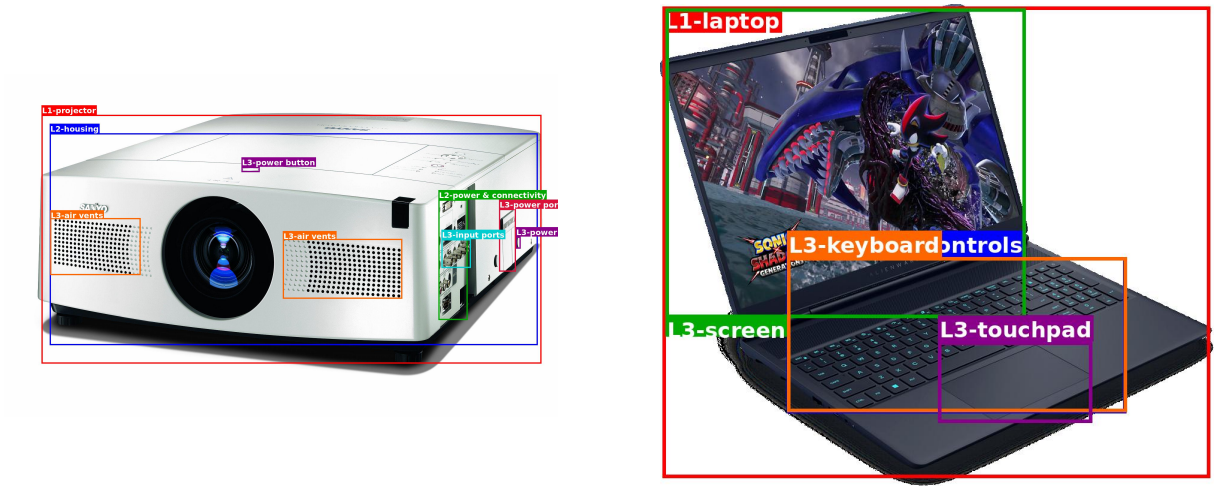


Figure 5: left: projector, right: laptop

Figure 6: Figure shows component extraction of the same product from different angles

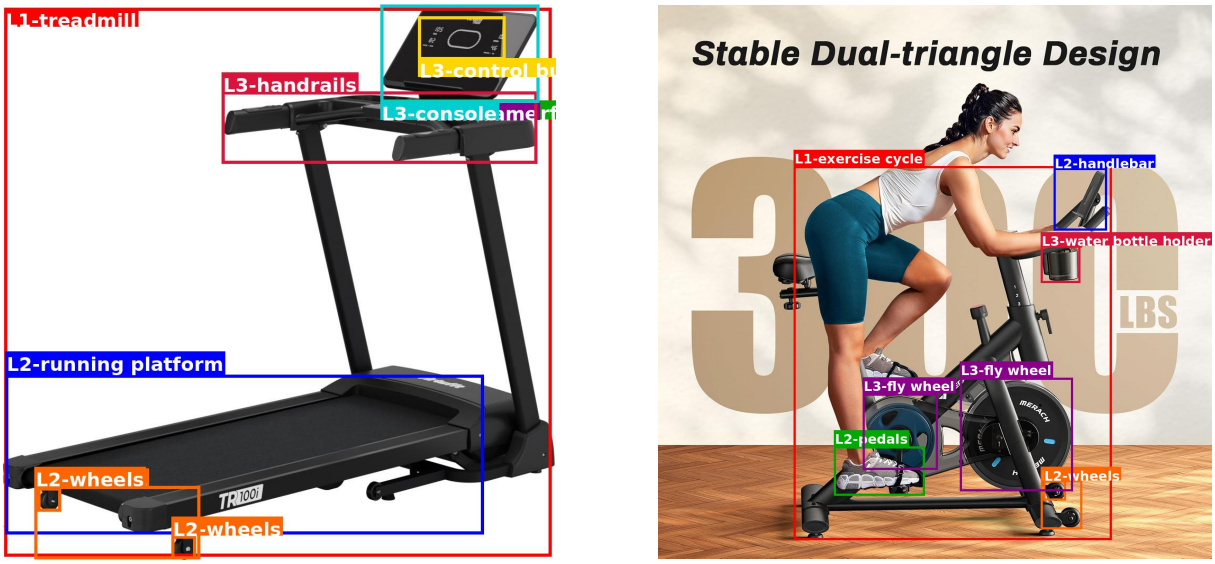


Figure 7: left: treadmill, right: exercise cycle

Figure 8: **Failure Case 1: Cascading errors from parent misclassification**. When the model misclassifies an L1/L2 to a small bounding box, child components like door (left) and seat (right) are omitted due to hierarchical validation.

Figure 9: **Failure Case 2: Oblique or elongated components.**. The wood piece is classified as a child component despite being outside the bed, due to the bed being oblique.