SoftwareX 36 (2026) 103173

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

HILIPS: A human-in-the-loop image labeling pipeline for visual-semantic ambiguity

Taehyung Noh, Beejin Son, Junghyun Kim, Haein Yeo, Kyungsik Han*

Hanyang University, Seoul, Republic of Korea

ARTICLE INFO

Keywords:
Image labeling
Human-in-the-loop
Visual-semantic ambiguity
Multimodal large language model
Interactive segmentation
Industrial object detection

ABSTRACT

Image labeling in industrial environments frequently encounters visual-semantic ambiguity, where visually similar objects cannot be distinguished by appearance alone. We present *HILIPS* (Human-In-the-Loop Image Labeling Pipeline for Semantic ambiguity), a three-stage framework that integrates multimodal large language models (MLLMs) with interactive segmentation and lightweight object detection. In the Cold-start Labeling stage, an MLLM and SAM2 generate initial labels without requiring training data. In the Label Transfer stage, the labels generated by the MLLM and reviewed by the user are used as ground truth to train a deployable lightweight YOLO model through supervised label transfer. In the Iterative Refinement stage, the trained detector labels new images, and users review only the detections below the auto-approval threshold and retrain the model with the corrected annotations. A user study involving 10 participants showed consistent reductions in perceived workload (NASA-TLX) and labeling time, alongside improved system usability (SUS) as the level of automation increased. *HILIPS* is released as open-source software with a modular architecture that allows for independent replacement of MLLMs, segmentation models, and detection components to suit diverse industrial requirements.

Metadata

Code metadata.

Nr.	Code metadata description	Metadata
C1	Current code version	1.0.0
C2	Permanent link to code/repository used for this code version	https://github.com/yestaehyung/HILIPS
C3	Permanent link to Reproducible Capsule	None
C4	Legal Code License	MIT License
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	Python, TypeScript, FastAPI, Next.js, React, PyTorch
C7	Compilation requirements, operating environments & dependencies	Python ≥ 3.10, Node.js ≥ 18, CUDA ≥ 11.8 (GPU); Docker (optional)
C8	If available Link to developer documentation/manual	https://github.com/yestaehyung/HILIPS#readme
C9	Support email for questions	yestaehyung@hanyang.ac.kr

* Corresponding author.
Email addresses: yestaehyung@hanyang.ac.kr (T. Noh), beejinson@hanyang.ac.kr (B. Son), junghyuning@hanyang.ac.kr (J. Kim), haeinyeo@hanyang.ac.kr (H. Yeo), kyungsikhan@hanyang.ac.kr (K. Han).

https://doi.org/10.1016/j.softx.2026.103173
Received 4 May 2026; Received in revised form 1 October 2026; Accepted 21 October 2026
Available online 27 October 2026
2352-7110/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Software metadata.

Nr.	(Executable) software metadata description	Metadata
S1	Current software version	1.0.0
S2	Permanent link to executables of this version	https://github.com/yestaehyung/HILIPS/releases/tag/v1.0.0
S3	Permanent link to Reproducible Capsule	None
S4	Legal Software License	MIT License
S5	Computing platforms/Operating Systems	Linux, macOS, Microsoft Windows (Docker supported)
S6	Installation requirements & dependencies	Python ≥ 3.10, Node.js ≥ 18, CUDA ≥ 11.8 (optional)
S7	If available, link to user manual	https://github.com/yestaehyung/HILIPS#readme
S8	Support email for questions	yestaehyung@hanyang.ac.kr

1. Motivation and significance

As collaborative robots become increasingly common in industrial environments [1], they are taking on tasks that directly involve interaction with human work environments, including control panel operation, equipment inspection, and maintenance support. Reliable object recognition is essential for these tasks [2], yet building high-quality training data [3] that reflects the diverse environmental conditions and object variety found in industrial environments remains a significant challenge.

Image labeling is often the primary bottleneck in training data construction, requiring substantial time and manual effort [4]. This challenge is amplified in industrial images, which differ from general object images in important ways [5]. Specifically, objects that appear visually similar but serve distinct functions are common, creating a phenomenon we define as *visual-semantic ambiguity*. In these cases, appearance alone is insufficient for classification; instead, contextual cues such as embedded text or spatial relationships are required to assign the correct labels. Unlike standard fine-grained recognition [6] benchmarks (e.g., CUB-200 [7], Stanford Cars [8]), where the difficulty lies in detecting fine-grained visual features (e.g., beak shape), industrial environments present a qualitatively different challenge in which buttons on a machine control panel may be identical and distinguished only by printed labels or coordinates.

Current annotation tools provide different levels of assistance for image labeling. Traditional tools such as LabelMe [9] and VGG Image Annotator [10] mainly support manual annotation, while more recent platforms such as CVAT [11] and Label Studio [12] can integrate SAM [13] for boundary extraction. However, semantic label assignment still requires user input. Grounded SAM-2 [14] combines text-guided grounding based on Grounding DINO [15] with SAM2 [16]. The present study does not directly compare HILIPS with Grounded SAM-2 in the target industrial setting.

This visual-semantic ambiguity creates a semantic judgment burden that affects two groups differently. Domain experts understand the target objects but are unfamiliar with structured annotation workflows, while annotation workers are proficient with labeling tools but lack domain-specific knowledge. In either case, each image requires the worker to (1) understand the overall scene structure, (2) identify spatial and functional relationships among objects, and (3) integrate textual or domain cues into their labeling decision. Workers without a domain background must repeatedly refer to reference materials, and extended sessions lead to decision fatigue [17] that degrades labeling consistency. Because the labeling process is cognitively demanding, automation that supports semantic judgment is required.

Multimodal large language models (MLLMs) [18,19] offer a promising direction by jointly processing images and text to interpret complex visual scenes based on natural-language domain descriptions [20]. Their reasoning ability is particularly useful in cold-start scenarios, where no training data exist, and in industrial environments that require rapid adaptation to new equipment or control panels [21]. This flexibility significantly exceeds what task-specific vision models or fixed rule-based systems can provide, though the high inference cost and latency of MLLMs [22] make their direct deployment in real-time industrial environments impractical.

To address these limitations in both semantic ambiguity and MLLM inference overhead, we present *HILIPS*, a Human-in-the-Loop framework that integrates human judgment into the pipeline to handle cases that automated models cannot reliably resolve, while minimizing the overall annotation burden.

The framework proceeds in three stages:

- ***Cold-start Labeling*** combines an MLLM with SAM2 [16] to generate high-quality initial labels when no training data is available.
- ***Label Transfer*** involves training a lightweight object detection model on the MLLM-generated labels to produce a model capable of running on edge devices.
- ***Iterative Refinement*** deploys the trained model back into the labeling pipeline, where users review only low-confidence predictions and the accumulated corrections are used to retrain the model, progressively raising the automation rate.

A user study with 10 participants showed lower perceived workload and labeling time under conditions with higher levels of assistance, providing evidence that the workflow can reduce labeling effort.

The key contribution of this paper is a three-stage workflow (cold-start labeling, label transfer, and iterative refinement), where the MLLM generates candidate bounding boxes and class labels from an image and the user's prompt, the user reviews and corrects these candidates, and the reviewed labels are used to train a lightweight detection model. *HILIPS* is released as open-source software with a modular design, ensuring that the MLLM backend, segmentation model, and detection model can each be swapped independently. The current evaluation is limited to industrial control panels. Applying the pipeline to other domains that involve visual-semantic ambiguity remains to be validated.

2. Software description

2.1. Software architecture

HILIPS consists of three sequential stages (Table 1, Fig. 1, Fig. 2): (1) *Cold-start Labeling*, (2) *Label Transfer*, and (3) *Iterative Refinement*. Each stage consumes the output of its predecessor, and stages 2 and 3 can be repeated to increase the automation rate over successive iterations. The system is implemented as a web application with a Next.js 15, React 19, TypeScript frontend and a FastAPI backend.

Table 1
Overview of each stage in the *HILIPS* pipeline.

Stage	Main Function	User Involvement	Input	Output
Cold-start Labeling	Generate initial labels using MLLM and SAM without training data	Prompt writing, result review	Raw images, user prompts	Reviewed initial label dataset
Label Transfer	Train a lightweight model on user-reviewed labels through supervised label transfer	Hyper-parameter tuning, model selection	Reviewed initial label dataset	Trained lightweight model
Iterative Refinement	Re-inject the lightweight model to increase automation rate	Review of low-confidence objects	New images, lightweight model	Expanded dataset, improved model

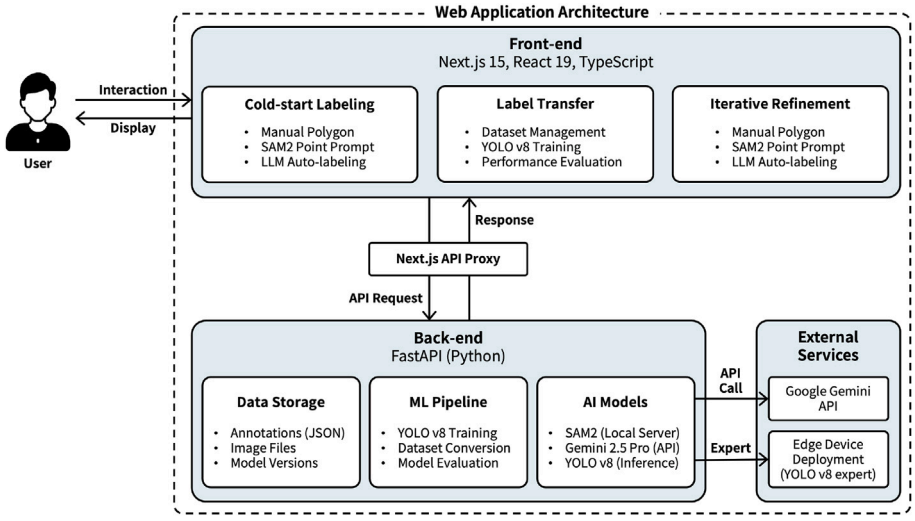

Fig. 1. Overview of the *HILIPS* web application architecture. The system consists of a three-stage frontend (*Cold-start Labeling, Label Transfer, Iterative Refinement*) built with Next.js 15 and React 19, a FastAPI backend managing data storage and ML pipelines, and external AI services (Google Gemini 2.5 Pro API, edge device deployment via YOLOv8).

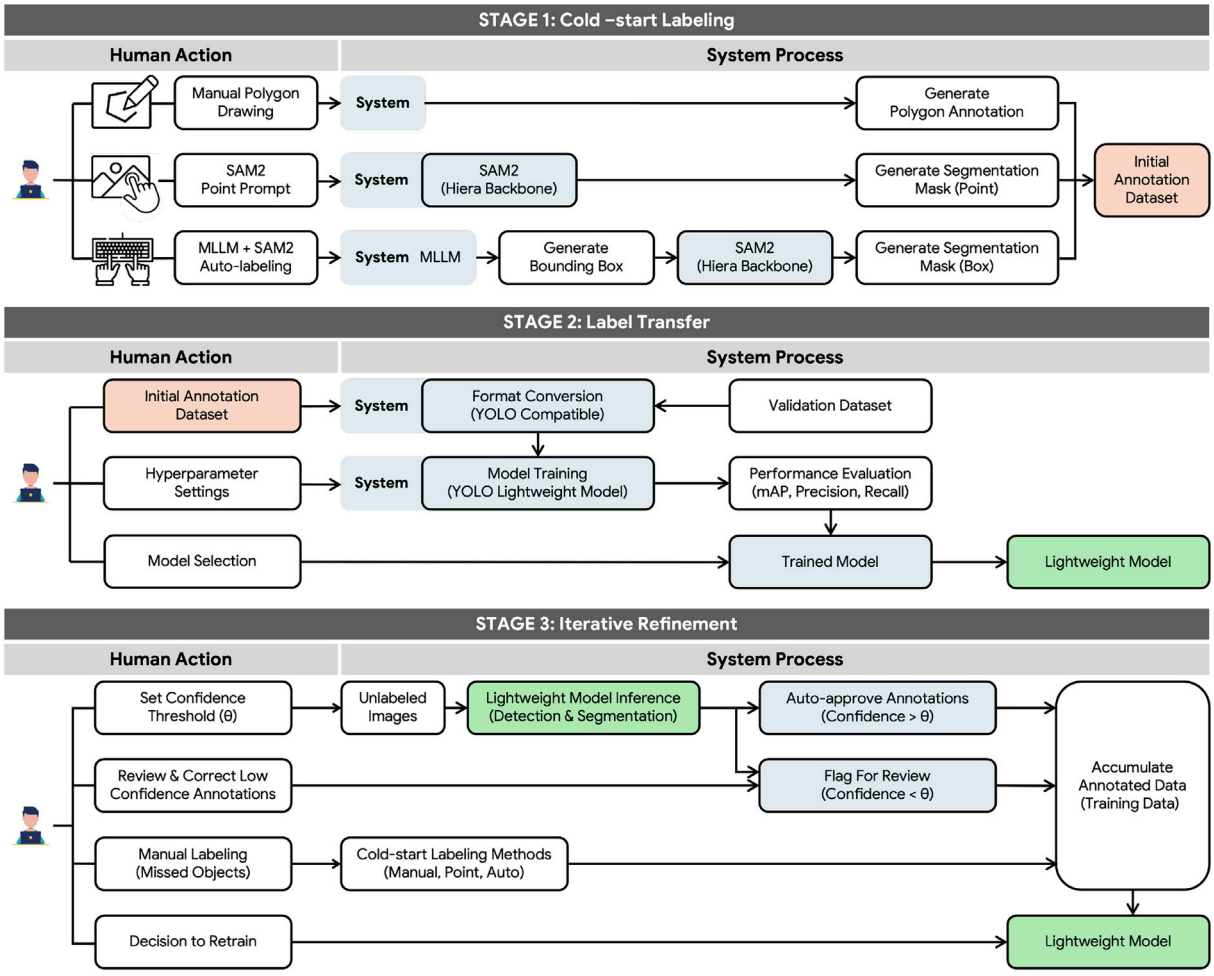

Fig. 2. Overview of the *HILIPS* three-stage pipeline. Each stage distinguishes between human actions on the left and system processes on the right. In stage 1, an initial annotated dataset is generated using manual, point-based, or MLLM-assisted labeling. Stage 2 involves training a lightweight YOLOv8 model based on those initial annotations. Finally, stage 3 applies the model iteratively, routing high-confidence detections to auto-approval while directing low-confidence detections to human review, with subsequent retraining performed on the accumulated data.

Cold-start Labeling. This stage produces initial labels without any training data by pairing an MLLM with a segmentation model. We use the Google Gemini 2.5 Pro [23] API for semantic reasoning: given an image and a prompt, the model returns bounding boxes and class labels for the described objects. Gemini 2.5 Pro was chosen for its stable API availability and cost effectiveness, both of which facilitate reproducibility. The predicted bounding boxes are then passed as box prompts to SAM2 [16], which produces precise segmentation masks. SAM2 improves on the original SAM [13] in both accuracy and speed and runs locally, which removes the dependency on an external segmentation API. The Supplementary Material reports the prompt structure in Table S1 and the MLLM configuration in Table S2, and it discusses the cost, privacy, confidentiality, and connectivity considerations of the external API in Section S9.

Label Transfer. In this stage, the labels generated during the cold-start stage, reviewed and corrected by the user are used as ground truth to train a lightweight detection model through supervised learning. This process is called supervised label transfer, in which the reviewed labels are directly used as training data to train a lightweight model for efficient local inference. We adopt YOLOv8 [24] as the lightweight detector because it offers a favorable accuracy–speed trade-off and can be deployed on edge devices. Users review and correct the polygon and bounding-box coordinates generated in the Cold-start Labeling stage. The system then saves the corrected annotations in COCO format. Each annotation contains the class label, polygon coordinates, and bounding-box coordinates of the object. The Label Transfer stage uses these user-reviewed annotations as ground truth to train the lightweight detection model through supervised label transfer. Users can set hyperparameters and start training through the web interface.

Iterative Refinement. The trained YOLOv8 model is deployed back into the labeling pipeline. For each new image, the model detects objects and assigns a confidence score to each detection. The system uses two values: a detection-confidence cutoff of 0.50 and the auto-approval threshold θ. The system does not display detections below the detection-confidence cutoff. The system initializes θ to 0.80, so users can start labeling without setting a value, and users can also change θ from 0.50 to 0.95 in the interface. Among the displayed detections, the system automatically approves those at or above θ and displays them in green, and it displays those below θ in yellow and routes them to the user for review. Users can manually add missed objects with the three methods of the Cold-start Labeling stage. Corrected annotations are appended to the training set, and the user can retrain the model at any point. With each iteration, the fraction of auto-approved detections increases, reducing the manual review burden. The Supplementary Material reports the full threshold configuration in Table S3. All three core components—MLLM backend, segmentation model, and detection model—are designed as independently swappable modules, ensuring that upgrading any single component does not necessitate changes to the rest of the pipeline.

2.2. Software functionalities

HILIPS provides the following key features (Table 2).

2.2.1. Cold-start labeling

The *Cold-start Labeling* stage provides three annotation methods that can be mixed within a single image:

- Manual Polygon Drawing: the user traces the boundary of objects by hand. This method is suitable for complex or irregular shapes where automatic segmentation algorithms may fail.
- Point-based Labeling: the user clicks a point on a target object, and SAM2 automatically generates a corresponding segmentation mask.
- Prompt-based Auto-labeling: the user describes target objects in natural language. An MLLM interprets the request and returns bounding boxes and class labels, which are then refined by SAM2 into precise segmentation masks. This method is designed to label targets that involve visual-semantic ambiguity, such as identical-looking buttons on a control panel that can only be distinguished by their printed text.

Table 2
Summary of *HILIPS* software functionalities.

Category	Feature	Description
Annotation Methods	Manual Polygon Drawing	User draws polygon boundaries to define objects
	Point-based Auto-labeling	Automatic object segmentation via SAM point prompts
	Prompt-based Auto-labeling	Natural-language-based auto-labeling combining MLLM and SAM2
Model Training	Dataset Management	Annotation storage and training set construction
	YOLOv8 Training	Hyper-parameter configuration and model training interface
	Performance Evaluation	mAP, precision, recall, and other evaluation metrics
Iterative Refinement	Auto-labeling	Object detection and segmentation using the trained model
	Confidence-based Review	Filtering review targets by confidence score
	Data Accumulation & Retraining	Accumulating reviewed data and retraining on the full dataset
Data Management	Multi-format Export	Support for COCO, YOLO, and Pascal VOC formats
	Version Control	Per-version model storage and performance comparison
	Annotation History	Labeling history tracking and prompt reuse

Table 3
User-configurable hyper-parameters for YOLOv8 training.

Parameter	Range
Epochs	1–1000
Batch size	8 / 16 / 32
Image size	640 / 1280
Learning rate	0.001–0.01

The annotations generated by the MLLM in the cold-start stage are reviewed and corrected by the user before being stored in the dataset. The labels reviewed and corrected by the user are then used as the training targets in the Label Transfer stage.

2.2.2. Label transfer

In the *Label Transfer* stage, the system uses the accumulated user-reviewed annotations to train a YOLOv8 model. Users configure hyperparameters such as the number of Epochs and Batch size, through the training interface, as listed in Table 3. After training, the system reports key performance metrics including mAP (mean average precision), precision, and recall on a held-out validation set. To facilitate experimentation, each trained model is saved with a version tag, allowing users to compare different runs and select the best checkpoint for the next stage.

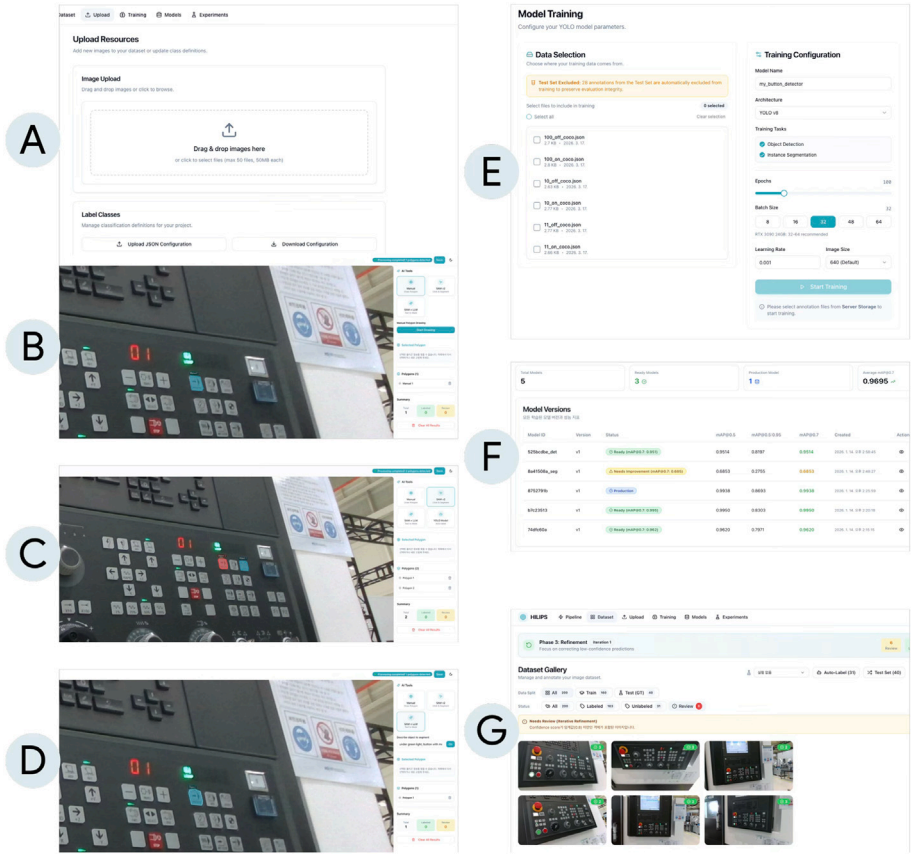

Fig. 3. Overview of the *HILIPS* workflow. *Cold-start Labeling*: (A) image upload and label class definition, (B) manual polygon drawing, (C) SAM2 point prompting, and (D) MLLM-based auto-labeling. *Label Transfer*: (E) training data selection and hyperparameter configuration, and (F) model version comparison using mAP metrics. *Iterative Refinement*: (G) automatic detection on new images with review-flagged items highlighted.

2.2.3. Iterative refinement

When users upload new images during the *Iterative Refinement* stage, the trained YOLOv8 model performs detection upon a user request. The system does not display detections below the detection-confidence cut-off of 0.50. Among the displayed detections, the system automatically approves those at or above θ and displays them in green, and it displays those below θ in yellow and routes them to the user for review. Any objects missed by the model can be manually added using the three methods described in the *Cold-start Labeling* phase (Section 2.2.1). Once reviewed, the corrected data are appended to the training set, and the user can trigger a retraining cycle at any time. Through successive rounds of this confidence-based review and retraining, the fraction of auto-approved detections progressively increases, reducing the overall manual workload.

3. Illustrative examples

Fig. 3 illustrates the *HILIPS* workflow. In the *Cold-start Labeling* stage, the user (A) uploads an image and (B) chooses from manual polygon drawing, (C) SAM2 point prompting, or (D) MLLM-based auto-labeling. With the MLLM approach, auto-generated labels are overlaid on the image; the user corrects or adds to them and saves. In the *Label Transfer* stage, the user selects a dataset and hyperparameters (E), runs training, checks the resulting metrics, and picks a model version (F). In the *Iterative Refinement* stage, the user uploads new images and the trained model detects objects automatically (G). High-confidence detections (green) and review-required ones (yellow) are visually distinguished; the user inspects only the flagged items and can add missed objects via Cold-start methods. Retraining on the accumulated data raises the automation rate for subsequent rounds.

Fig. 4 shows industrial CNC control panels and labeling results from the *Cold-start Labeling* stage. The four upper photographs show the panel layout from different viewpoints, and the two lower photographs show labeling results for `DOOR`, `MEMORY`, and `JOG`. On a CNC control panel, buttons such as `MEMORY` and `JOG` are visually similar but differ in function, and their printed text or position on the panel distinguishes them. *HILIPS* labels these buttons using the user's natural-language description, without prior training data. The user reviews these labels before *HILIPS* uses them in the subsequent stages.

4. Impact

HILIPS targets users who need to build training data in industrial settings but lack ML teams or annotation workers. It combines MLLM-driven semantic judgment, supervised label transfer to a lightweight model, and confidence-based iterative refinement in a single workflow. This covers the full path from cold-start annotation to a deployable detector. To validate whether this workflow reduces the labeling burden, this section evaluates *HILIPS* through a user study, measuring both efficiency (time, interactions) and overall user experience (workload, usability) across three levels of automation.

4.1. Experimental procedure

Ten participants took part in the user study. They had varied levels of basic experience with object-detection or labeling tools and were not image-annotation experts. This participant group was selected to examine whether *HILIPS* can reduce labeling effort for users who are not experts. The experiment followed a within-subjects design, requiring every participant to use all three labeling methods in a randomized order



Fig. 4. Representative labeling results on industrial CNC control panels. The four upper photographs show the panel layout from different viewpoints. In the two lower photographs, colored bounding boxes indicate objects labeled by *HILIPS*: `DOOR` (red), `MEMORY` (yellow), and `JOG` (orange). The close-up views show visually similar buttons that differ in their printed text or positions on the panel. The user reviews these labels before *HILIPS* uses them in the subsequent stages of the pipeline. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

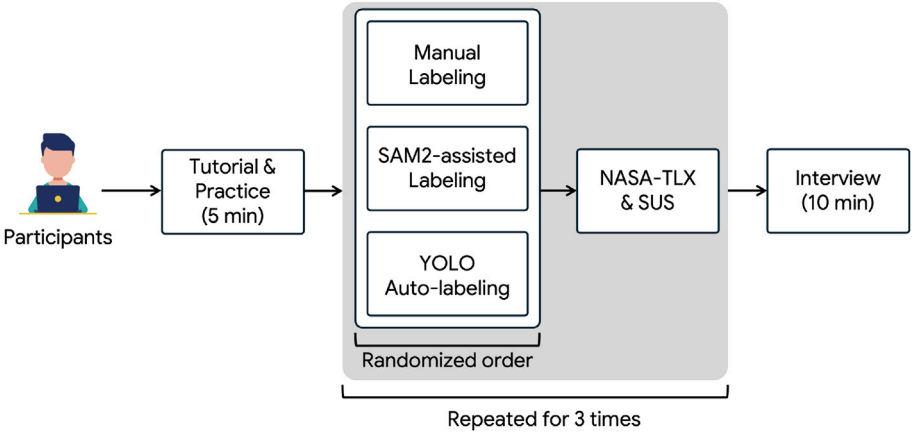

Fig. 5. Within-subjects design. Participants completed a 5-minute tutorial and practice session, and then performed three labeling methods (Manual, SAM2-assisted, YOLO Auto-labeling) in a randomized order to mitigate learning effects. After each method, participants completed the NASA-TLX and SUS surveys. The session concluded with a semi-structured interview.

to mitigate potential learning effects. Each participant spent approximately 50 min completing the tasks. Fig. 5 illustrates the overall procedure. Participant and study details are provided in the Supplementary Material.

Following a tutorial and a 5-minute practice session, participants labeled 10 images per method in a randomized order. To prevent data leakage, test images were held out from the YOLOv8 training set used in the third method (M3). The three methods were:

- **M1 (Manual)**: participants drew polygon boundaries and assigned class labels entirely by hand.
- **M2 (SAM2-assisted)**: SAM2 generated segmentation masks from point clicks while class labels were assigned manually.
- **M3 (YOLO auto-labeling)**: the trained YOLO model automatically detected objects and proposed labels for participants to review and correct as needed.

After each method, participants completed the NASA-TLX [25] workload questionnaire and SUS (System Usability Scale) [26] survey. The evaluation concluded with 10-minute semi-structured interviews. The system also logged objective metrics per session, including time per image, interaction count, user-added objects, and total detected objects.

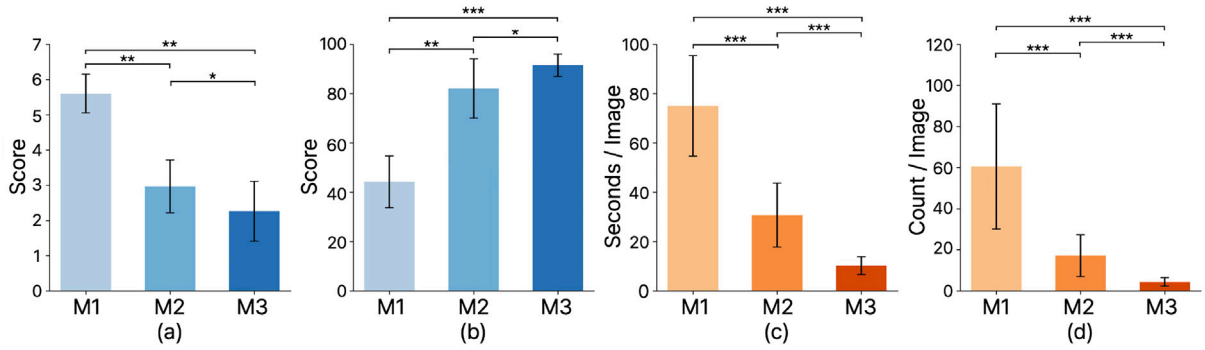

Fig. 6. User study results across M1 (Manual), M2 (SAM2-assisted), and M3 (YOLO Auto-labeling). (a) NASA-TLX workload and (b) SUS usability scores show consistent improvement with increasing automation. (c) time per image and (d) interaction count both decreased significantly across all method pairs (***p <.001). Error bars represent standard deviation.

Because the study involved repeated measures with a small sample, NASA-TLX, SUS, time per image, and interaction count were analyzed using Friedman tests followed by post-hoc Wilcoxon signed-rank tests. Direct comparative preference ratings were analyzed separately using Wilcoxon signed-rank tests against the neutral midpoint (4).

4.2. Results

As shown in Fig. 6(a), the perceived workload (NASA-TLX) decreased across the three methods (Friedman $\chi^2 = 16.22$, p <.001, $W = .901$), with all pairwise differences reaching statistical significance (p <.05). The mental demand subscale showed the same pattern. M3 received the lowest mental demand score (mean 1.33 on a 7-point scale), compared with M1 (6.00) and M2 (2.33) (Friedman $\chi^2 = 15.70$, p <.001, $W = .872$). Participants rated the automatically proposed labels as helpful (mean 6.67 out of 7).

Usability scores (SUS) followed a similar positive trend as illustrated in Fig. 6(b) ($\chi^2 = 14.89$, p <.001, $W = .827$). On the adjective scale [27], M1 (Mean = 46.7) was rated "Not Acceptable," while both M2 (Mean = 86.7) and M3 (Mean = 91.7) were "Excellent." These results indicate that either assisted method provides a qualitatively better user experience than fully manual labeling.

Objective system logs reinforced these subjective findings (Fig. 6(c) and (d)). Time spent per image significantly dropped from 37.5 s (M1) to 15.4 s (M2) to 5.1 s (M3) (Friedman $\chi^2 = 18.000$, p <.001, $W = 1.000$). Similarly, the interaction count fell from 60.6 (M1) to 16.0 (M2) and 4.4 (M3) ($\chi^2 = 18.000$, p <.001, $W = 1.000$), with every participant showing a consistent decrease.

In M3, no participant found it necessary to add objects manually; the YOLO model detected all targets, allowing participants to focus solely on verification and approval. This confirms the intended system shift from a labeling-centric to a review-centric workflow, validating the core principle of our Human-in-the-Loop design. Furthermore, the total number of detected objects remained comparable across all three methods, suggesting that increased automation did not compromise detection completeness.

Direct preference surveys confirmed that participants perceived both M2 and M3 as more efficient, accurate, and convenient than M1 (all p <.05; Table 4). While a direct comparison between M2 and M3 was not explicitly included in the survey, the perceived advantage of the assisted methods over manual labeling is clearly reflected in the standardized instruments and objective logs.

The measures above characterize labeling effort and user experience, but they do not establish that the resulting annotations are correct. We therefore evaluated the transferred YOLOv8 detector on a held-out test set of 100 images excluded from training, using human-reviewed ground truth that the detector did not contribute to. Precision was 99.12% and recall was 94.03% overall, and mAP@0.50 was 97.34% (Table 5). These results show high downstream object-detection performance after supervised label transfer. The Supplementary Material reports the detector training configuration in Table S4, the hardware and software environment in Table S5, and the composition of the held-out evaluation set in Table S6.

Table 4
Direct preference survey results comparing manual labeling (M1) with assisted methods (M2, M3). Ratings were collected on a 7-point comparative scale (1 = strongly prefer M1, 4 = no difference, 7 = strongly prefer the assisted method). Results are reported as median [IQR], Wilcoxon signed-rank statistic (W), p-value, and effect size (r) from Wilcoxon signed-rank tests against the neutral midpoint (4).

Comparison	Metric	Mdn [IQR]	W	p	r
M2 vs. M1	Efficiency	7.0 [6.0–7.0]	0.0	.008	.940
	Accuracy	5.0 [5.0–6.0]	0.0	.016	.914
	Ease of use	7.0 [7.0–7.0]	0.0	.008	.940
M3 vs. M1	Efficiency	7.0 [7.0–7.0]	0.0	.008	.940
	Accuracy	7.0 [6.0–7.0]	0.0	.008	.940
	Ease of use	7.0 [7.0–7.0]	0.0	.008	.940

Table 5
Held-out detection performance of the transferred YOLOv8 detector on 100 images excluded from training, measured using box IoU against human-reviewed ground truth.

Metric	Overall	DOOR	JOG	MEMORY
Precision	99.12%	98.45%	100.00%	98.92%
Recall	94.03%	91.84%	97.66%	92.60%
mAP@0.50	97.34%	95.39%	98.63%	98.00%
mAP@0.50:0.95	84.66%	81.58%	87.04%	85.35%

We also examined how the auto-approval threshold θ affects the automation rate and labeling quality. We applied each θ from 0.50 to 0.95 to the confidence scores logged for the 30 detections in M3. At the default $\theta = 0.80$, the system would auto-approve 29 of the 30 detections (96.7%) and route one to the user. The automation rate would decrease to 83.3% at $\theta = 0.90$ and to 23.3% at $\theta = 0.95$. Participants accepted all 30 detections, including the one with the lowest score (0.798), so labeling quality was 100% at every θ. In this task, a higher θ would increase the review workload without improving labeling quality. Supplementary Table S9 reports the results for all θ values.

Semi-structured interviews complemented the quantitative results by highlighting specific labeling challenges, such as visual ambiguity and interaction difficulty. One participant reported issues with interaction consistency: *"It was hard to click precisely; the buttons weren't consistent."* (P01). Some participants noted low-contrast stimuli as a barrier, *"The slightly blurry buttons were difficult."* (P02), and highlighted both physical

and cognitive strain from manual labeling, *"It was physically tiring, and the classes and buttons got confusing."* (P04).

Feedback regarding automation was positive. Participants attributed lower cognitive effort specifically to the reduced need for class judgment in M3: *"Method 3 was the most comfortable; I didn't get confused about which class to pick. I just had to check whether the segmentation was right."* (P01). Another participant contrasted the experience directly: *"The automatic detection was helpful. With Method 2, I had to figure out what to label from scratch, but Method 3 handled the mapping too."* (P02). One participant summarized the efficiency gain concisely: *"Method 3. Just one button."* (P04). These remarks align with the quantitative finding that no manual object additions were needed in M3.

Despite the overall positive reflections, participants noted certain limitations, particularly regarding the need for precise interaction even within automated workflows (P02). Concerns about reliability in high-stakes contexts were also raised, with one participant expressing skepticism about using the system without oversight in situations where accuracy is critical (P04). These insights suggest that future research should focus on improving preprocessing quality and establishing better trust calibration for users.

5. Related work

Prior work on human-AI annotation spans human-in-the-loop interfaces, AI-assisted segmentation, and annotation supported by language and multimodal models. Desmond et al. [28] proposed an AI-assisted labeling interface that recommends likely labels and reduces the candidate label space presented to users. Garzarella et al. [29] presented middleware that transfers the outputs of visual deep-learning models to annotation tools for review and enrichment by domain experts. Zhang et al. [30] assigned object selection to human annotators and label generation to a large multimodal model, and Grounded SAM-2 [14] supports text-guided grounding and segmentation. These systems assist individual annotation steps, whereas *HILIPS* connects user-reviewed annotations to local detector training and routes subsequent predictions for automatic approval or human review through a confidence threshold with a default value of 0.80. *HILIPS* does not perform a separate confidence-calibration procedure; instead, the threshold allocates the review effort within the workflow. Supplementary Table S8 compares the workflow components of these approaches.

6. Conclusions

In this paper, we presented *HILIPS*, a three-stage human-in-the-loop pipeline for industrial image labeling where visual-semantic ambiguity exists. *HILIPS* incorporates human semantic judgment into the labeling process and gradually increases the use of automated labeling. The *Cold-start Labeling* stage pairs an MLLM with SAM2 to bootstrap annotations without training data; the *Label Transfer* stage uses the reviewed annotations as ground truth to train a lightweight YOLO model through supervised label transfer; and the *Iterative Refinement* stage deploys the model back into the pipeline, progressively replacing manual effort with automated detection. Together, the three stages address both labeling efficiency and model construction in a single workflow. A within-subjects user study confirmed that increased automation reduces workload and labeling time, and allows users to complete tasks through review alone. This validates the design goal of shifting the user's role from active labeling to verification. Several limitations remain. The participant pool was small, and a broader study is planned. This work does not include a direct component-level comparison between the MLLM-assisted cold-start annotations and end-to-end methods such as Grounded SAM-2; our evaluation instead reports the downstream detection performance obtained after supervised label transfer. Extending the evaluation to other industrial domains, benchmarking the cold-start stage directly, and integrating active learning are left for future work. Adding image-preprocessing modules could further improve robustness. *HILIPS* is released as open-source software and is available for use and extension by the research community.

CRediT authorship contribution statement

Taehyung Noh: Writing – review & editing, Writing – original draft, Software, Project administration, Methodology. **Beejin Son:** Writing – original draft, Software, Methodology. **Junghyun Kim:** Software, Methodology. **Haein Yeo:** Project administration, Methodology, Investigation. **Kyungsik Han:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used Grammarly's AI writing suggestions in order to improve the language and readability of the manuscript. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of the publication.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. RS-2020-II201373, the Artificial Intelligence Graduate School Program (Hanyang University)) and the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korean government (MOTIE) (No. RS-2025-02633871, Development of automated data pre-processing technology for industrial AI).

Appendix A. Supplementary data

Supplementary data to this article can be found online at doi:10.1016/j.softx.2026.103173.

References

[1] Villani V, Pini F, Leali F, Secchi C. Survey on human-robot collaboration in industrial settings: safety, intuitive interfaces and applications. Mechatronics 2018;55:248–66. https://doi.org/10.1016/j.mechatronics.2018.02.009
[2] Andrés del Valle AC, Fernández JG. Human-robot perception in industrial environments: a survey. Sensors 2021;21(5):1571. https://doi.org/10.3390/s21051571
[3] Northcutt C, Jiang L, Chuang I. Confident learning: estimating uncertainty in dataset labels. J Artif Intell Res 2021;70:1373–411. https://doi.org/10.1613/jair.1.12125
[4] Roh Y, Heo G, Whang SE. A survey on data collection for machine learning: a big data-ai integration perspective. IEEE Trans Knowl Data Eng 2021;33(4):1328–47. https://doi.org/10.1109/TKDE.2019.2946162
[5] Czimmermann T, Ciuti G, Milazzo M, Chiurazzi M, Roccella S, Oddo CM, Dario P. Visual-based defect detection and classification approaches for industrial applications - a survey. Sensors 2020;20(5):1459. https://doi.org/10.3390/s20051459
[6] Wei X-S, Xie C-W, Wu J, Shen C. Mask-CNN: localizing parts and selecting descriptors for fine-grained bird species categorization. Pattern Recognit 2018;76:704–14. https://doi.org/10.1016/j.patcog.2017.10.002
[7] Wah C, Branson S, Welinder P, Perona P, Belongie S. The Caltech-UCSD Birds-200-2011 dataset (CNS-TR-2011-001). Tech. Rep California Institute of Technology; 2011.
[8] Krause J, Stark M, Deng J, Fei-Fei L. 3D object representations for fine-grained categorization. In: Proceedings of the IEEE international conference on computer vision workshops (ICCVW); 2013. p. 554–61.
[9] Russell BC, Torralba A, Murphy KP, Freeman WT. LabelMe: a database and web-based tool for image annotation. Int J Comput Vis 2008;77(1–3):157–73. https://doi.org/10.1007/s11263-007-0090-8
[10] Dutta A, Zisserman A. The VIA annotation software for images, audio and video. In: Proceedings of the 27th ACM international conference on multimedia. ACM; 2019. p. 2276–9. https://doi.org/10.1145/3343031.3350535
[11] Sekachev B, Manovich N, Zhiltsov M, Zhavoronkov A, Kalinin D, Ben-Shahar O, Sidnev D, et al. opencv/cvat: v1.1.0; 2020. https://doi.org/10.5281/zenodo.4009388., Zenodo.
[12] Tkachenko M, Malyuk M, Holmanyuk A, Liubimov N. Label studio: data labeling software; 2020–2025. https://github.com/HumanSignal/label-studio., Open source software.

[13] Kirillov A, Mintun E, Ravi N, Mao H, Rolland C, Gustafson L, et al. Segment anything. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV); 2023. p. 4015–26. https://openaccess.thecvf.com/content/ICCV2023/html/Kirillov_Segment_Anything_ICCV_2023_paper.html.
[14] Ren T, Liu S, Zeng A, Lin J, Li K, Cao H, et al. Grounded SAM: assembling open-world models for diverse visual tasks, arXiv preprint arXiv:2401.14159. 2024.
[15] Liu S, Zeng Z, Ren T, Li F, Zhang H, Yang J, et al. Grounding DINO: marrying DINO with grounded pre-training for open-set object detection, arXiv preprint arXiv:2303.05499. 2023.
[16] Ravi N, Gabeur V, Hu Y-T, Hu R, Ryali C, Ma T, et al. SAM 2: segment anything in images and videos, arXiv preprint arXiv:2408.00714. 2024.
[17] Baumeister RF, Vohs KD. Self-regulation, ego depletion, and motivation. Soc. Personal. Psychol. Compass 2007;1(1):115–28.
[18] OpenAI. GPT-4V(ision) system card. Technical Report OpenAI; 2023.
[19] Team G, Anil R, Borgeaud S, Wu Y, Alayrac J-B, Yu J, et al., Gemini: a family of highly capable multimodal models, arXiv preprint arXiv:2312.11805. 2023.
[20] Liu H, Li C, Li Y, Lee YJ. Improved baselines with visual instruction tuning. In: 2024 IEEE/CVF conference on computer vision and pattern recognition (CVPR). IEEE; 2024. p. 26286–96.
[21] Madan A, Peri N, Kong S, Ramanan D. Revisiting few-shot object detection with vision-language models. In: Advances in neural information processing systems (NeurIPS); 2024. arXiv:2312.14494.
[22] Zhao WX, Zhou K, Li J, Tang T, Wang X, Hou Y, et al., A survey of large language models, arXiv preprint arXiv:2303.18223. 2023.
[23] Google DeepMind. Gemini 2.0: our new AI model; 2024. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/., Google AI Blog.
[24] Jocher G, Chaurasia A, Qiu J. Ultralytics YOLOv8; 2023. https://github.com/ultralytics/ultralytics.
[25] Hart SG, Staveland LE. Development of NASA-TLX (task load index): results of empirical and theoretical research. In: Advances in psychology, vol. 52. North-Holland; 1988. p. 139–83. https://doi.org/10.1016/S0166-4115(08)62386-9
[26] Brooke J. SUS: a “quick and dirty” usability scale. In: Jordan PW, Thomas B, Weerdmeester BA, McClelland IL, editors. Usability evaluation in industry. Taylor & Francis; 1996. p. 189–94.
[27] Bangor A, Kortum PT, Miller JT. Determining what individual SUS scores mean: adding an adjective rating scale. J Usability Stud 2009;4(3):114–23.
[28] Desmond M, Muller MJ, Ashktorab Z, Dugan C, Duesterwald E, Brimijoin K, Finegan-Dollak C, Brachman M, Sharma A, Joshi NN, Pan Q. Increasing the speed and accuracy of data labeling through an AI assisted interface. In: Proceedings of the 26th international conference on intelligent user interfaces. ACM; 2021. p. 392–401. https://doi.org/10.1145/3397481.3450698
[29] Garzarella S, Stacchio L, Cascarano P, De Filippo A, Cervellati E, Marfia G. Preserving and annotating dance heritage material through deep learning tools: a case study on rudolf nureyev. In: Proceedings of the international workshop on artificial intelligence and creativity, vol. 3810. CEUR Workshop Proceedings; 2024. p. 88–97.
[30] Zhang H, Fu X, Carroll JM. Augmenting image annotation: a Human-LMM collaborative framework for efficient object selection and label generation, arXiv preprint, 2025. https://doi.org/10.48550/arXiv.2503.11096