Computer Vision for Automated Robotic Device Disassembly: Object Detection, Pose Estimation, and Action Prediction

Sebastian Ruiz[1†], Boris Kuster[2], Aleš Ude[3], and Florentin Wörgötter[4]

Computational Neuroscience Group, Georg-August-Universität[1,4]
(E-mail: sebastian.ruiz@uni-goettingen.de[1], worgott@gwdg.de[4])
Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute[2,3]
(E-mail: Boris.Kuster@ijs.si[2], ales.ude@ijs.si[3])

Abstract: The increasing demand for electronic products has led to a surge in end-of-life devices, making efficient recycling crucial for minimizing environmental impact. However, current recycling processes are often tailored to specific models, making adaptation to varying devices complex and costly. This paper addresses the challenge of automating electronic device disassembly using computer vision and action prediction methods. The research explores key components of a robotic disassembly system, including pose estimation, device classification, rotation estimation, gap detection, and action prediction. High accuracy is achieved using segmentation models and supervised learning for known devices, while zero-shot classification and data-driven approaches show promise for handling unseen devices. A large language model (LLM) is introduced for action prediction, demonstrating its ability to adapt to diverse disassembly tasks with 91% accuracy. The results indicate that generalization across device models is possible but varies by method. This study provides a framework for developing flexible and robust robotic systems, paving the way for more sustainable and scalable electronic recycling solutions.

Keywords: Computer vision, instance segmentation, LLMs for action prediction, robotic device disassembly

1. INTRODUCTION

The rapid growth of population and technological advancement has significantly increased the demand for electrical products, while simultaneously their life cycles have become shorter [18, 3]. This trend has led to an increase in end-of-life (EOL) products, with an estimated 44.7 million tons generated in 2016 alone [1]. Efficient recycling of these EOL products is crucial to mitigate their environmental impact; however, it is rarely considered during product design. Disassembly, a critical step in recycling, often proves economically unfeasible due to high labour costs, particularly in developed countries [14].

Robotic automation has the potential to address these economic challenges, making disassembly both feasible and efficient. Yet, the disassembly of electronic devices presents considerable technical challenges. The variability in device design, even within the same product family, requires systems capable of handling diverse disassembly processes. A robust, automated system must be adaptable to this high variability to maximize its impact on sustainability and recycling rates.

This paper examines to what extent the computer vision and action prediction within the ReconCycle system [27, 23, 22, 10], can generalize to new unseen devices with minimal or no supervision. Here, *generalizability* refers to the ability of the system to disassemble new devices within a specific product family, even when these devices have not been encountered previously. In contrast to conventional industrial disassembly systems, which are tailored for specific devices, the ReconCycle system aims to enable flexible reconfiguration to accommodate new devices.

Effective disassembly relies on understanding the physical properties of the target device. This research focuses on four critical computer vision methods to obtain this information: pose estimation, device classification, component detection, and gap detection. A knowledge dataset is developed, encompassing similar devices with annotated descriptions of disassembly actions and reasoning written in human language descriptions. The dataset is used for improved action prediction using a large language model (LLM), where the LLM predicts high-level actions.

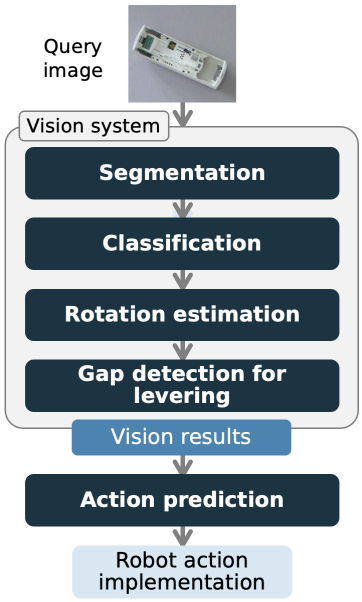


Fig. 1: Computer vision workflow for action prediction, with the goal of device disassembly, to be performed by the ReconCycle workcell.

† Sebastian Ruiz is the presenter of this paper.

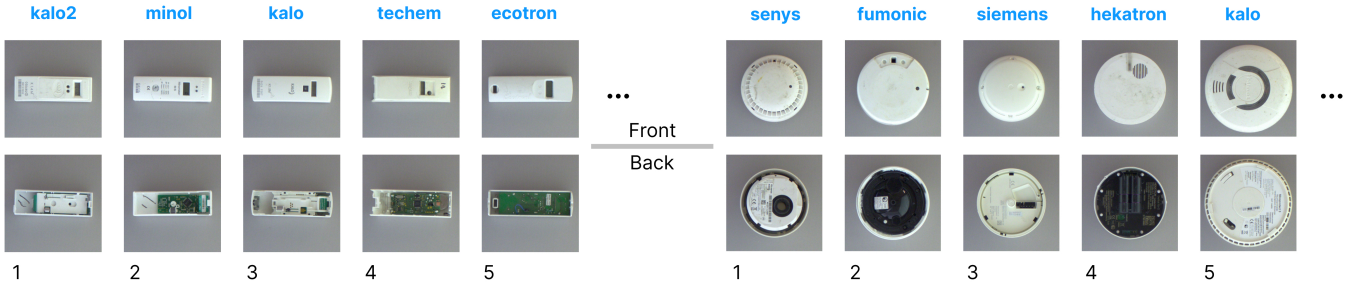


Fig. 2: Examples of HCAs and smoke detectors (excluding variations). Our dataset contains 13 HCAs and 10 smoke detectors.

2. RELATED WORK

Recent research has explored various aspects of disassembly automation, but large-scale industrial adoption remains limited due to challenges such as managing diverse product variants and material conditions. Apple's robotic system, Liam [19], uses specialized automation, efficiently disassembling iPhone 6 components for material recovery but lacking flexibility for broader device handling.

We use action prediction to determine the next disassembly step. To predict actions we need a scene description of the device and its current state of disassembly. For this we use computer vision methods in pose estimation, classification, and gap detection.

NVIDIA's Deep Object Pose Estimation (DOPE) [25] is a one-shot neural network for 6D pose estimation. However, DOPE faces challenges in bridging the synthetic-to-real-world gap due to its reliance on entirely synthetic datasets. Therefore, we research image segmentation models, such as YOLOv8 [7], in combination with rotation estimation to provide the necessary pose information, since it offers a straightforward path for data annotation and training.

We use ResNet-50 [5] for the device classification task, and we compare it to CLIP [16], trained on 400 million image-text pairs for zero-shot classification without fine-tuning. Radford et al. [16] show that CLIP outperforms ImageNet trained models in classification tasks, but struggles with fine-grained distinctions, such as differentiating between visually similar object instances.

Since segmentation models are not able to estimate the rotation of devices, we research models to solve this problem. For rotation estimation, classical feature matching methods, such as SIFT [12] have been used, but are outperformed by modern learning-based approaches like SuperGlue [20]. End-to-end CNN-based models have been applied to rotation estimation problems, typically using regression or cross-entropy losses for tasks involving natural scenes [8, 26]. We compare featuring matching and end-to-end CNN models on our rotation estimation problem.

Determining gaps in a device is useful for removing internal components by levering them out, possibly once the screws holding down the components have been removed. The work of Yildiz et al. [29] on hard drive recycling used HDBSCAN for clustering point cloud data to find gaps. We extend on this work and compare it with a segmentation model on the depth and image data.

Recently, LLMs such as GPT-4 [15] have been used for action prediction, following from their reasoning ability. For instance, prompting LLMs with reasoning steps, as proposed by Wei et al. [28], enhances their ability to generate complex plans. Shentu et al. [21] propose methods to convert natural language tasks into executable actions, using strategies like generating Python function calls, parsing natural language solutions, or embedding outputs as policy inputs. LLMs have been combined with classical planners, as in Liu et al. [11], where LLMs translate tasks into PDDL for classical solvers, offering a hybrid approach. We investigate LLMs for action prediction of disassembly actions using retrieval augmented generation of examples from our knowledge dataset.

3. METHODS

The vision system consists of the vision modules: pose estimation in Section 3.1, device classification in Section 3.2, rotation estimation in Section 3.3, and gap detection for levering in Section 3.4. The focus of these methods is to provide a generalizable approach, such that the methods perform well on new devices with none or minimal retraining. These methods provide a scene description that is used for action prediction in Section 3.5, providing a high level action, that can be implemented by the ReconCycle workcell.

3.1. Device and Components Segmentation

We collected images of 13 HCA types and 10 smoke detector types, see Figure 2, with a total of 4458 labelled images taken using a combination 2900×2900 px RGB images, 1280×720 px RGBD images. We created a *Synthetic Dataset Generator Tool* to augment our dataset of 4458 images, to a dataset of 20,000 images.

The images are annotated with masks with the following labels: HCA (front, back, back empty), smoke detector (front, back, insides, insides empty), PCB, PCB covered, battery, battery covered, internals, plastic clips, screws, and wires. *PCB covered* and *battery covered* indicate partial obstruction by plastic while remaining visible.

YOLOv8-Seg is an instance segmentation model built upon the YOLOv8 object detection framework, sharing a similar architectural structure. The model demonstrates state-of-the-art performance across object detection and segmentation benchmarks, while maintaining real-time processing capabilities [24].

The pose estimation problem is constrained to scenarios where the camera is positioned above the scene, observing objects of interest on a surface that is perpendicular to the

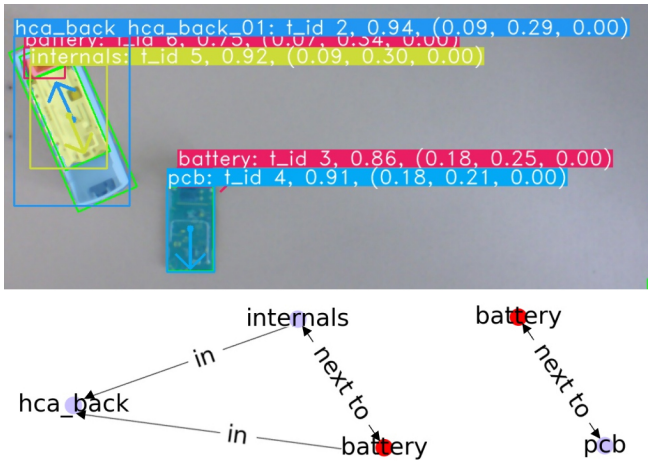


Fig. 3: Example of relations graph for an HCA and a PCB with battery. The natural language text representation is: Component 1: The battery is next to the PCB. Component 2: the internals are next to the battery. The internals and battery are in the HCA (back).

camera's view.

The relation graph represents devices and components within the scene, along with the positional relationships between them, as shown in Figure 3. This graph is constructed based on the results of pose estimation. In Section 3.5, the relation graph, translated into natural language text, is utilized as a component of the prompt for the LLM used in action prediction.

3.2. Device Classification

The segmentation model from Section 3.1, learns the device family, but not the specific device type, such that the segmentation is able to detect unseen devices from the same family, without needing to retrain the model. Determining the device type remains a critical requirement; therefore, this work explores methods for device classification. From the HCAs and smoke detectors, see Figure 2, and a subset of the images in Section 3.1, we create an image dataset for the classification task consisting of 61 classes, including front and back for each device, with a total of 1694 images.

For device classification, a ResNet-50 convolutional neural network (CNN) [5] is employed. The performance of the CNN model is compared to that of the CLIP model [16], which is utilized for zero-shot image classification.

3.3. Rotation Estimation

The device position is found using the segmentation model, as described in Section 3.1. The rotation can be found from the segmentation mask, up to its rotation invariances. The correct rotation from the invariances can be chosen if there are other features found by the segmentation model, but when this is not the case, the device rotation needs to be estimated separately, see examples in Figure 4.

We use the same dataset as used in the classification task. We compare an end-to-end CNN model with SuperGlue [20], a feature matching method, for rotation estimation. To train our methods on the task of rotation estimation, we generate

Fig. 4: Examples of rotation estimation for smoke detectors.

pairs of images with a ground truth rotation between them, see Figure 5. We generate these pairs using homographies as used in SuperPoint [4], but with random rotations in range $[-180°, 180°)$ instead of $[-25°, 25°)$ rotations [9].

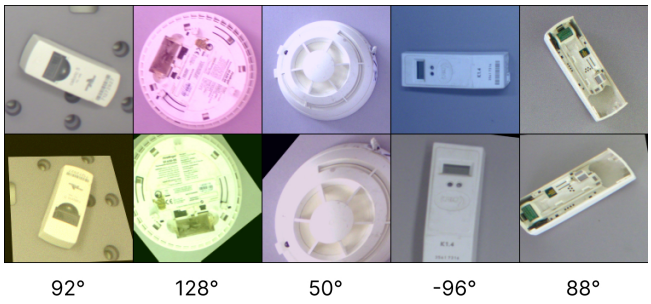


Fig. 5: Example of pairs for rotation estimation with ground truth rotation.

Our methods take a pair of images, and estimates the rotation between them. The CNN model takes two images and extracts feature vectors using ResNet-50 [5] with shared weights, the output is concatenated, and then passed through 3 linear layers with ReLU. We compare two different loss functions for learning the rotation angle: regression loss and binary cross entropy by discretising the space into 36 bins. We found that by using more bins, the accuracy in rotation estimation drops.

The SuperGlue model uses SuperPoint [4] as a feature extractor, and SuperGlue learns to match the features between the two images. The weights of SuperPoint are frozen, and we train SuperGlue on our dataset of homography pairs. From the feature matches on an image pair, we recover the rotation by fitting a rotation transform to the matched points.

3.4. Gap Detection for Levering

Certain devices require a levering operation for disassembly, which involves inserting a levering tool into a gap and leveraging the fulcrum effect on the housing to extract internal components, such as the PCB. A key challenge in this process is the detection of gaps, which is complicated by their variable appearance and the need for precise localization. To address this, a RealSense camera [17] is utilized to capture both RGB and depth information, see Figure 6.

We compare two methods for gap detection: a clustering algorithm, and an instance segmentation model. The first method to gap detection employs the HDBSCAN clustering algorithm [2] to segment the depth image into clusters based on the depth profile. The second method is training the YOLOv8 segmentation model on labelled images of gaps.

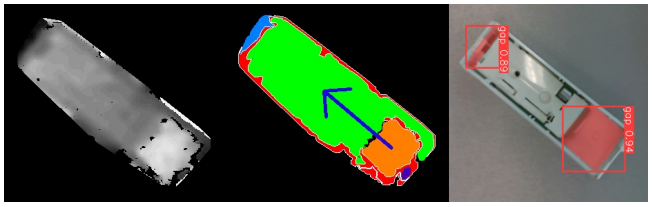


Fig. 6: Example of gap detection. Left: depth image, centre: HDBSCAN with filtering to find gaps with levering action (blue arrow), right: YOLOv8 segmentation model.

3.5. Action Prediction

An LLM is used for action prediction, given a query image and results from the vision system. The LLM uses a data-driven approach to predict the actions, using information from a knowledge tree for retrieval augmented generation (RAG).

The workflow for action prediction using an LLM is shown in Figure 9 (appendix). Given a query image, the vision system generates a scene description, which serves as input for the LLM. The LLM is provided with an objective, a set of available tools, background information, examples of similar disassembly actions, and the current disassembly state. The full prompt is detailed in Appendix C.

To supply the LLM with relevant examples of similar disassembly actions, the input image is compared to images stored in a knowledge tree. The k-nearest neighbour images, under the CLIP embedding and the L_2 distance, and their corresponding question-answer (QA) pairs are retrieved and presented to the LLM. This process enables the LLM to predict disassembly actions for devices exhibiting similarities to those stored in the knowledge tree.

As outlined by Wei et al. [28], incorporating reasoning steps enhances the ability of LLMs to make accurate predictions. The disassembly action examples provided to the LLM include reasoning to justify the suggested actions. Furthermore, the LLM is instructed to first reason about the appropriate action and then provide the action in its output. This approach improves performance compared to scenarios where the model is first asked to provide an action and subsequently justify it.

The prompt structure for the LLM follows a format similar to that proposed by Shentu et al. [21]. It includes the objective, a list of available actions, relevant background information, examples retrieved from the knowledge tree, and question-answer (QA) pairs related to the query image that the model is tasked with completing. The possible disassembly actions are: turn, move, lever, cut (cutter), and cut (CNC mill).

In Section 4.5, we compare the LLM-based approach to a baseline model that uses k-nearest neighbour majority voting. The comparison evaluates accuracy based on the proportion of correctly predicted actions from a designated test set.

4. RESULTS

The previous section explained the methods for the vision and action prediction models. In this section, we present the corresponding results.

4.1. Segmentation

When using image size input of 1280 px and using the YOLOv8 P2 model, compared to the 640 px model, the results for screw class improve from mask mAP@.50-.95. of 0.18 to 0.47. The results of the YOLOv8 1280 px P2 model are shown in Table 1. For the HCAs and smoke detectors, the box mAP@.50-.95. is above 0.93, detecting them effectively. On the components including internals, battery, PCB, and plastic clip the box mAP@.50-.95. is above 0.88.

Table 1: YOLOv8 1280 px P2 results on test set.

Class	Instances	**Box mAP @.50-.95**	**Mask mAP @.50-.95**
all	7209	**0.903**	**0.84**
HCA front	429	**0.978**	**0.958**
HCA back	414	**0.962**	**0.749**
HCA back empty	342	**0.94**	**0.909**
smoke det. front	396	**0.958**	**0.947**
smoke det. back	416	**0.936**	**0.915**
smoke det. insides	393	**0.947**	**0.829**
smoke det. insides empty	371	**0.945**	**0.927**
internals	530	**0.952**	**0.927**
battery	995	**0.88**	**0.823**
battery covered	302	**0.917**	**0.875**
PCB	862	**0.893**	**0.853**
PCB covered	398	**0.935**	**0.917**
plastic clip	366	**0.946**	**0.869**
wires	593	**0.822**	**0.598**
screw	378	**0.576**	**0.468**

Table 2: YOLOv8 640 px results on seen and unseen test set. The seen test set contains classes that are not present in the unseen set. For a fair comparison, the results shown are constrained to the classes that are in the unseen test set.

	Box mAP		Mask mAP	
	@.50	@.50-.95	@.50	@.50-.95
all	0.94	0.88	0.93	0.79
hcas_excl8-10	**0.97**	**0.92**	**0.96**	**0.81**
unseen	0.91	0.84	0.91	0.73
classes: HCA front, HCA back, HCA back empty, battery, internals, PCB, PCB covered, plastic clip				
hcas_excl11-13	**0.93**	**0.85**	**0.90**	**0.75**
unseen	0.72	0.67	0.71	0.56
classes: HCA front, HCA back, battery, internals, PCB, PCB covered, plastic clip, HCA back empty, screw				
smoke_excl7,8	**0.86**	**0.72**	**0.79**	**0.65**
unseen	0.72	0.62	0.70	0.58
classes: smoke detector front, smoke detector back, screw				
smoke_excl9,10	**0.96**	**0.94**	**0.96**	**0.93**
unseen	0.93	0.90	0.93	0.88
classes: smoke detector front, smoke detector back, screw				

To assess the model's generalization capability, multiple datasets are generated with specific device types excluded. This approach allows for evaluating the model's generalization performance to unseen in-distribution devices. The *all* dataset contains all the devices, see Table 2. The *hcas_excl(8-10/11-13)* are the datasets with HCAs 8-10 and 11-13 excluded respectively. Similarly, the *smoke_excl(7,8/9,10)* excludes smoke detectors 7,8 and 9, 10 respectively.

Overall, see Table 2, the YOLOv8 segmentation model

demonstrates the ability to detect unseen devices that share similarities with those in the training set. Across all experiments, performance degradation on unseen devices ranges between 10% and 20%.

4.2. Device Classification

We train the ResNet-50 model from Section 3.2 on the dataset of devices described, with train/val/test split 1328/183/183. By evaluating the trained model on the test set we obtain a 99.1% overall accuracy.

We present the results of CLIP for zero-shot classification. All images from the training set are projected onto their corresponding image embeddings, with their associated ground truth labels. For each test set image, an image embedding is generated, and the k-nearest neighbours are retrieved from the training set embeddings. The top-k accuracy is defined as the proportion of test cases where the correct classification is included among the retrieved k-nearest neighbours. In Table 3 the top-1, top-3 and top-5 accuracy results using CLIP are shown.

Table 3: Top-k accuracy for all classes, using CLIP

		Smoke det.		HCA	
Top-k	All classes	back	front	back	front
Top-1	89%	100%	91%	89%	75%
Top-3	98%	100%	100%	99%	90%
Top-5	98%	100%	100%	99%	96%

From the results above, the CNN model outperforms the CLIP model, but it must be trained, whereas the CLIP model, with an accuracy of 89% shows promise for zero-shot methods for classification, especially since the top-3 result gives an average accuracy over all classes of 98%.

4.3. Rotation Estimation

The CNN model and SuperGlue model are trained on the same dataset as used for device classification. The seen/unseen split is odd numbered/even numbered devices, as in Figure 2.

First we evaluate both models on our generated homographies, see Table 4, and second we evaluate the SuperGlue model on real world rotated devices, see Table 5.

The results in Table 4 show that the SuperGlue model trained on our dataset of devices with homographies including random rotations in range $[-180°, 180°)$ outperforms the pretrained SuperGlue indoor model by orders of magnitude.

Table 4: Rotation error of CNN baseline and SuperGlue models.

	Seen test		Unseen test	
Models	Mean	std.	Mean	std.
CNN rotloss	31.2°	24.6°	31.4°	25.7°
CNN binloss (36 bins)	8.2°	18.2°	14.4°	29.9°
SuperGlue indoor	43.9°	65.9°	40.3°	58.4°
SuperGlue (our dataset)	**0.40°**	**0.73°**	**0.46°**	**0.95°**

To evaluate real-world performance, we further test on real-world perspective and rotation changes, see Table 5. Poor performance was observed in cases where devices have many symmetries with little to no features that can guide SuperGlue to the correct orientation. The improved performance on the back of devices is attributed to the presence of numerous features that facilitate feature matching.

Table 5: Rotation error of SuperGlue model (our dataset) on real-world rotated devices. Rotation correct if it is correct to $\pm 10°$.

Setting	Accuracy
60 images of smoke detectors	83%
...ignoring devices with little to no features	92%
...only images with backs of devices	100%

4.4. Gap Detection for Levering

We compare the two models, HDBSCAN and YOLOv8, on a dataset of RGBD images of HCAs with a 136/61 train/-val split, with no test set because of the small size of the dataset. The results of the HDBSCAN clustering method and the YOLOv8 model on the validation set are shown in Table 6. The HDBSCAN clustering method is shown to perform best on large gaps, however it is generally outperformed by the YOLOv8 segmentation model.

Table 6: Accuracy for detecting gaps using the HDBSCAN clustering method and the YOLOv8 segmentation model.

	HCA Device				
Model	1	3	5	7	8
HBDSCAN	86%	100%	50%	42 %	58%
YOLOv8 RGBD	100%	100%	100%	92%	100%

To evaluate the unseen performance of the YOLOv8 model, we use a dataset of 161 images consisting of HCAs 1, 2.1, 3, 4, 5, 6, 7, 8, 9, and 11. We exclude two devices in cases A, B and C, see Table 7, to evaluate the unseen performance of detecting gaps. We find that there is a significant performance drop in mAP@.50 for the unseen classes, but note that the performance drop is much worse for case C than for A and B. We reason that, with only 8 devices to train on, the model has only few examples of what constitutes as a gap, especially since the gaps vary much between each device, and this is what leads to such differing results.

Table 7: YOLOv8 RGBD model on comparison with seen and unseen devices.

		Box	
Dataset	seen/unseen	mAP@.50	mAP@.50-.95
All	seen	**0.995**	**0.919**
case A	seen	**0.978**	**0.726**
excl. HCA 1, HCA 2.1	unseen	0.612	0.36
case B	seen	**0.995**	**0.849**
excl. HCA 3, HCA 4	unseen	0.633	0.291
case C	seen	**0.981**	**0.739**
excl. HCA 5, HCA 6	unseen	0.202	0.115

4.5. Action Prediction

This section presents the results of an LLM for action prediction using the knowledge tree as part of a retrieval-augmented generation (RAG) approach. To evaluate the LLM's performance, three models are compared: *majority vote*, *LLM with relevant examples*, and *LLM with non-relevant examples*. The evaluation is conducted on a test set comprising disassembly steps for four HCA devices, HCA 1 (kalo2), 2 (minol), 5 (ecotron), and 8 (exim), none of which are present in the knowledge tree. The knowledge tree contains 25 disassembly steps of 4 devices (2 HCAs, 2 smoke detectors). The test set is made up of 19 disassembly queries of 4 unseen devices (2 HCAs, 2 smoke detectors) for disassembly action prediction. By unseen devices, we mean that they are not present in the knowledge tree. As described in Section 3.5, the possible actions to be predicted are: turn, move, lever, cut (cutter), or cut (CNC mill). For each image in the test set, the model predicts the corresponding disassembly action.

The baseline model takes a *majority vote* from the k-nearest neighbour examples, where $k = 3$, to determine the predicted action. We compare the baseline model with two cases of the LLM model, in the first case, *LLM with 3 relevant examples*, the LLM model is given 3 examples from the knowledge tree, retrieved using CLIP for nearest neighbour search with the query image. In the second case, *LLM with 3 non-relevant examples*, the LLM is provided with 3 examples where none of the examples contain the correct action of the query image. The reasoning for the *LLM with 3 non-relevant examples* approach is two-fold. First, it aims to evaluate the LLM's ability to predict actions based on the provided prompt information and contextual understanding of the problem, rather than relying solely on replication of similar examples. Second, it assesses the performance of the method in scenarios where the knowledge tree lacks examples similar to the query device.

The LLM generates not only the predicted action but also accompanying reasoning, tool arguments, and the designation of the new module within the workcell on which the device is located. However, for the results presented in Table 8, only the predicted action is evaluated. Although the LLM's reasoning responses offer valuable insights, they are not included in the quantitative analysis.

Table 8: Results of models for action prediction using knowledge tree, averaged over 10 repetitions with GPT-4o.

Model	Avg. acc.	Std.
Majority vote w/ 3 relevant examples	0.42	0
LLM w/ 3 non-relevant examples	0.64	0.052
LLM w/ 3 relevant examples	0.91	0.034

The results presented in Table 8 demonstrate that the *LLM with 3 relevant examples* achieves the highest accuracy, at 91%, while the *LLM with 3 non-relevant examples* attains an accuracy of 64%. These findings highlight the critical role of relevant examples in accurately predicting disassembly actions. The results are averaged over 10 repetitions using the OpenAI GPT-4o model [15].

Table 9: Action prediction given one example of each disassembly action, averaged over 10 repetitions with GPT-4o.

Model	Avg. acc.	Std.
Majority vote	0.21	0
LLM	0.91	0.065

In Table 9 we provide the LLM with one example of each disassembly action. The LLM is able to correctly predict the correct disassembly action with 91% accuracy, similar to the result of the LLM with 3 relevant examples.

Previous action prediction methods, such as using PDDL [13], require all disassembly scenarios to be programmed. Meaning, actions cannot be predicted for scenarios that are not specifically accounted for. Furthermore, the PDDL code can become unmanageable with many device families. The LLM with the knowledge tree for action prediction has the advantage of being a data-driven method. New actions can be added by providing relevant examples to the knowledge tree and adding the action description to the prompt. The disadvantage of this method is that if the LLM predicts the wrong action, given relevant examples, it is not always transparent as to why the wrong action is being predicted. The generated reasoning response is however useful to determine how the prompt or examples can be improved. The objective and action description text may need to be rephrased, or the text in the examples may need to be improved.

5. DISCUSSION

We developed a vision-based device recognition and action prediction system designed to generalize to unseen devices. For pose estimation, the YOLOv8-based segmentation model achieved high performance on seen devices, with an average mask mAP@.50-.95 of 84% across classes, though certain classes like wires and screws scored lower. Performance on unseen devices dropped by 10-20%, highlighting the need for additional data and improved methods. The supervised CNN device classifier achieved 99% accuracy, outperforming the zero-shot CLIP-based method at 89%. For gap detection, a segmentation model achieved up to 100% accuracy for seen devices, whereas a clustering approach struggled with smaller gaps. Action prediction using an LLM with retrieval-augmented generation (RAG) demonstrated promising results, achieving 91% accuracy. However, achieving industrial-level reliability requires further refinement, for example in the integration of detailed scene descriptions and using multimodal, image and text, models.

This research establishes a robust baseline for automated disassembly of electronic devices, demonstrating the potential for both vision-based scene description and data-driven action prediction. For industrial viability, future work should address challenges with unseen devices by expanding datasets, refining LLM prompts, and incorporating richer scene representations. By examining the generalization capabilities of the vision and action prediction methods, this work paves the way for economically viable recycling solutions, contributing to improved material recovery and sustainability.

REFERENCES

[1] Sabah M. Abdelbasir et al. “Status of electronic waste recycling techniques: a review”. In: *Environmental Science and Pollution Research* 25.17 (June 1, 2018), pp. 16533–16547.

[2] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. “Density-Based Clustering Based on Hierarchical Density Estimates”. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Jian Pei et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 160–172.

[3] Peter Dauvergne. “The Problem of Consumption”. In: *Global Environmental Politics* 10.2 (May 1, 2010), pp. 1–10.

[4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. *SuperPoint: Self-Supervised Interest Point Detection and Description*. Apr. 19, 2018.

[5] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: arXiv:1512.03385 (Dec. 10, 2015).

[6] Geoffrey Hinton and Sam Roweis. “Stochastic Neighbor Embedding”. In: (2002).

[7] Glenn Jocher and RangeKing. *Brief summary of YOLOv8 model structure · Issue #189 · ultralytics/ultralytics*. GitHub. Jan. 10, 2023. URL: https://github.com/ultralytics/ultralytics/issues/189 (visited on 02/27/2024).

[8] Ujash Joshi and Michael Guerzhoy. “Automatic Photo Orientation Detection with Convolutional Neural Networks”. In: *2017 14th Conference on Computer and Robot Vision (CRV)*. 2017 14th Conference on Computer and Robot Vision (CRV). Edmonton, AB: IEEE, May 2017, pp. 103–108.

[9] Goutham Kumar. *SuperGlue training*. original-date: 2021-04-09T20:39:44Z. May 17, 2021.

[10] Boris Kuster, Mihael Simonič, and Aleš Ude. “Adaptive Robotic Levering for Recycling Tasks”. In: *Advances in Service and Industrial Robotics*. Ed. by Tadej Petrič, Aleš Ude, and Leon Žlajpah. Cham: Springer Nature Switzerland, 2023, pp. 417–425.

[11] Bo Liu et al. *LLM+P: Empowering Large Language Models with Optimal Planning Proficiency*. Sept. 27, 2023.

[12] D.G. Lowe. “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Proceedings of the Seventh IEEE International Conference on Computer Vision. Kerkyra, Greece: IEEE, 1999, 1150–1157 vol.2.

[13] Drew McDermott et al. “PDDL-the planning domain definition language”. In: 1998.

[14] Geraldo Cardoso de Oliveira Neto, Auro de Jesus Cardoso Correia, and Adriano Michelotti Schroeder. “Economic and environmental assessment of recycling and reuse of electronic waste: Multiple case studies in Brazil and Switzerland”. In: *Resources, Conservation and Recycling* 127 (Dec. 1, 2017), pp. 42–55.

[15] OpenAI et al. *GPT-4 Technical Report*. Mar. 4, 2024.

[16] Alec Radford et al. *Learning Transferable Visual Models From Natural Language Supervision*. Feb. 26, 2021.

[17] Intel Realsense. *IntelRealSense/librealsense: Intel® RealSense™ SDK*. 2016. URL: https://github.com/IntelRealSense/librealsense (visited on 03/25/2024).

[18] Julio L. Rivera and Amrine Lallmahomed. “Environmental implications of planned obsolescence and product lifetime: a literature review”. In: *International Journal of Sustainable Engineering* 9.2 (Mar. 3, 2016), pp. 119–129.

[19] Charissa Rujanavech et al. “Liam - An Innovation Story”. In: (2016).

[20] Paul-Edouard Sarlin et al. *SuperGlue: Learning Feature Matching with Graph Neural Networks*. Mar. 28, 2020.

[21] Yide Shentu et al. *From LLMs to Actions: Latent Codes as Bridges in Hierarchical Robot Control*. May 8, 2024.

[22] Mihael Simonič et al. “An Application Study on Reconfigurable Robotic Workcells and Policy Adaptation for Electronic Waste Recycling”. In: *2024 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE International Conference on Robotics, Automation and Mechatronics (RAM)*. 2024 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE International Conference on Robotics, Automation and Mechatronics (RAM). ISSN: 2326-8239. Aug. 2024, pp. 180–186.

[23] Mihael Simonič et al. “Modular ROS-based software architecture for reconfigurable, Industry 4.0 compatible robotic workcells”. In: *2021 20th International Conference on Advanced Robotics (ICAR)*. 2021 20th International Conference on Advanced Robotics (ICAR). Dec. 2021, pp. 44–51.

[24] Juan Terven and Diana Cordova-Esparza. “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS”. In: *Machine Learning and Knowledge Extraction* 5.4 (Nov. 20, 2023), pp. 1680–1716.

[25] Jonathan Tremblay et al. *Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects*. Sept. 27, 2018.

[26] Daniel Sáez Trigueros. *Correcting Image Orientation Using Convolutional Neural Networks - Daniel Sáez*. Jan. 12, 2017.

[27] Aleś Ude et al. *ReconCycle Project*. 2020. URL: https://reconcycle.eu/ (visited on 11/21/2023).

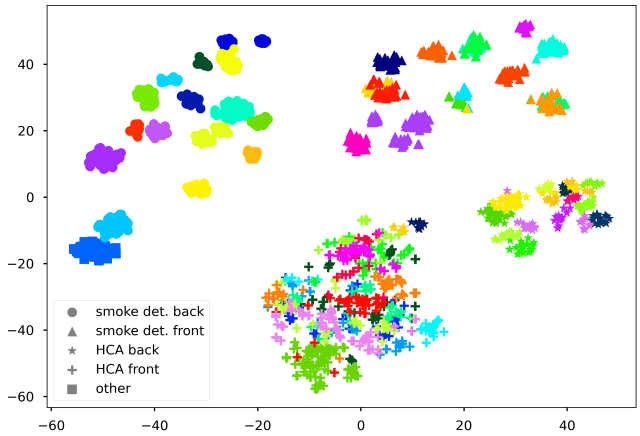
[28] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. Jan. 10, 2023.

[29] Erenus Yildiz et al. “A Visual Intelligence Scheme for Hard Drive Disassembly in Automated Recycling Routines”. In: Nov. 4, 2020.

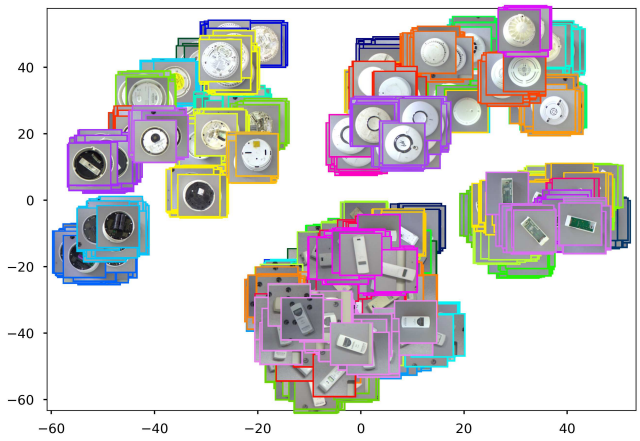
APPENDIX

A. ZERO-SHOT DEVICE CLASSIFICATION

The CLIP image embeddings are represented in a 2D projection in Figure 7 using t-distributed stochastic neighbour embedding (t-SNE) [6], where the colours represent the ground-truth class.



(a) The colours represent the ground truth classes, with the image representation shown below.



(b) The images are the ground truth classes. Each class has a unique colour.

Fig. 7: CLIP image embedding of the images from Figure 2, with 2D projection using t-SNE. The colours represent the ground truth classes.

Figure 7 presents two visualizations: one using only colours to represent different classes, and another that includes both colours and device images. The colour-only version makes the clusters easier to distinguish, as the device images in the second version can obscure them.

In the embedding space, the smoke detector back classes exhibit the clearest clustering, with well-separated groups and minimal overlap. The smoke detector front classes show slightly more overlap, indicating less distinct separation between them. The HCA back classes are positioned closer together, resulting in greater overlap, while the HCA front classes show the most overlap of all, despite the clusters being spread out. Overall, the t-SNE visualization aligns with the classification performance reported in Table 3.

B. ROTATION ESTIMATION USING THE SUPERGLUE MODEL

An example comparison of the SuperGlue Indoor model, vs. the SuperGlue model trained on our data with extended augmentations to 360° rotations is shown in Figure 8.

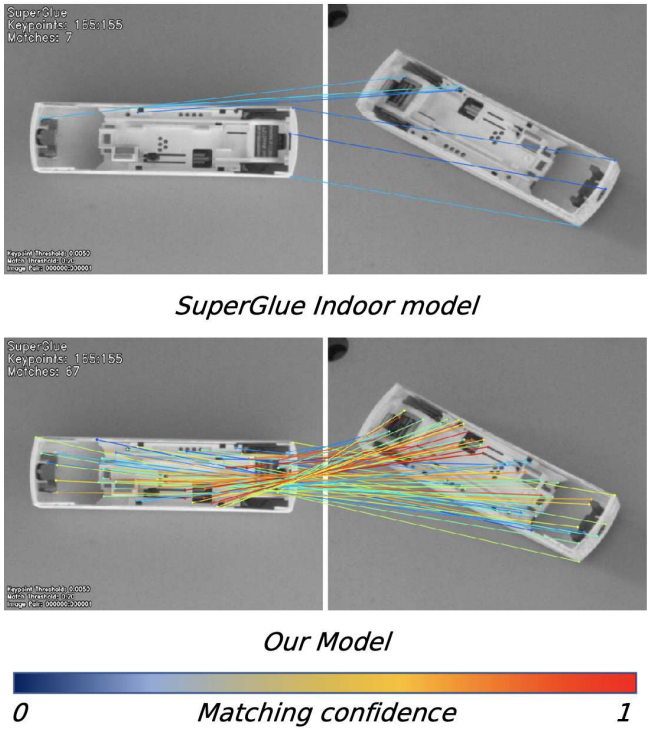


Fig. 8: Example of SuperGlue Indoor Model compared to our trained model

C. LLM PROMPT AND WORKFLOW FOR ACTION PREDICTION

An example prompt is shown in Listing 1, where the LLM is provided with the objective, available actions, background information, examples of question-answer pairs, the query. The query is to be completed by the LLM in JSON format. It should give reasoning for the next disassembly step, the tool to use, and further parameters.

```
You are an agent controlling a robotic workcell.
Objective:
Disassemble visible electronic devices to separate
     out the batteries for recycling. To do this,
    provide reasoning, select a tool and tool
    argument, i.e. predict the disassembly action.

Available actions:
turn: if the front of the device is showing, the
    device can be turned to show the back.
move: pick up an object from the work surface and
    place somewhere. For example a device can be
    picked up and put in the vise for levering.
lever: given a gap, lever out internal components
    of a device. The module must be vise to use
    this tool.
cut: given a device, cut it in two pieces, either
    using the cutter or the CNC mill.
unscrew: unscrew a screw from a device.
push: can be used to push out a plastic_clip from
    a device.
```

```
Background information:
If the module is vision you can perform the 'move'
     action.
If the module is vise you can perform the actions
    'lever', 'move', 'push', and 'cut'. Lever can
    only be used when the device is on the vise
    module. The battery can be cut from the PCB if
     it is has been removed from the device.

Example 1:
<Example QAs>

Example 2:
<Example QAs>

Example 3:
<Example QAs>

End of examples. You are given these facts about a
     query:
Q: What are the object relations?
A: at_location(table_vision). Device 1: internals
    in hca, battery in hca, battery next to
    internals.

Q: What are the object positions?
A: hca back (kalo2) at (1519, 1734), internals at
    (1519, 1676), battery at (1522, 1845).

Q: What is the JSON formatted disassembly step?
    Use the format:
{
reasoning: [explain you reasoning for tool choice
    ],
tool: [tool name],
tool_arguments: [arguments to tool function],
current_module: vision,
new_module: [the new module the device is on after
     performing the action]
}
```

Listing 1: LLM prompt providing objective, available actions, background information, and examples. The LLM is queried with the next disassembly step to be given in JSON format.

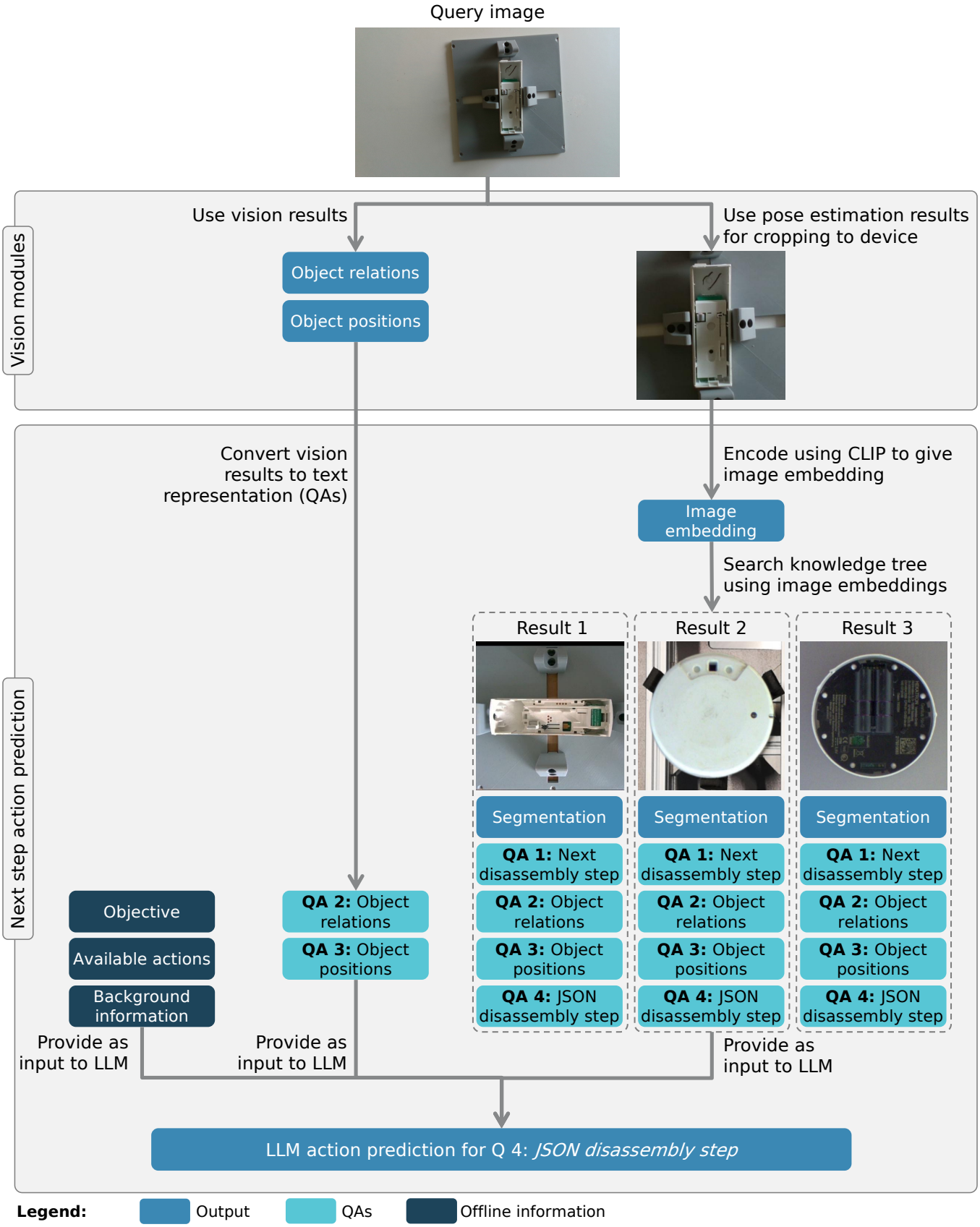


Fig. 9: Workflow of the LLM for action prediction. Question-answer pairs are: **QA 1**: What is the next disassembly step?, **QA 2**: What are the object relations?, **QA 3**: What are the object positions?, **QA 4**: What is the JSON formatted disassembly step?