Title:	Self-reconfiguration of a robotic workcell for the recycling of electronic waste
Acronym:	**ReconCycle**
Type of Action:	Research and Innovation Action
Grant Agreement No.:	871352
Starting Date:	01-01-2020
Ending Date:	31-03-2024

Deliverable Number:	D2.2
Deliverable Title:	Report and data format specification sheet for the semantic framework of scene description (R, M21)
Type of Deliverable:	Public
Authors:	Sebastian Ruiz, Minija Tamosiunaite, Florentin Wörgötter
Contributing Partners:	UGOE

Contractual Date of Delivery to the EC:	30-09-2021
Actual Date of Delivery to the EC:	10-11-2021

Table of Contents

1. Executive Summary

In this deliverable, we describe semantic scene components that are needed in respect to the planned robotic procedures for the generalization of the disassembly protocols. Semantic scene components include device type, parts and poses of the objects to be disassembled as well as more abstract features (like “gaps”). In the deliverable, each component is defined and we provide procedures for extracting of those components.

This deliverable is about 4 weeks delayed due to the fact the we first needed a solid disassembly protocol implemented at the robotic system from which we could make the inference about the different semantic components and their relations. This was needed to allow for defining the semantic framework of scene analysis in this deliverable.

2. Introduction

In this deliverable we specify semantic scene components relevant for the generalisation procedures of ReconCycle.

We have made a plan to develop the ReconCycle system in several steps with ever increasing demand for generalisation. We start with fixed models of non-broken heat cost allocators (HCA), go to damaged varieties and finally generalize to models of the HCA of unknown type and to different devices, too (e.g. smoke detectors). The plan for this is provided in Table 1, where in column three the relevant semantic components are named. Furthermore, we specify the descriptors and define procedures for obtaining those descriptors in separate sections below. Note that project partner ECYC is only interested in separating batteries and PCBs from the electronic devices. The rest are different plastic components. Thus, we focus on the extraction of batteries and PCBs in this document.

Table 1. Steps for the generalisation of disassembly protocols.

No of step	Planned activity	Relevant scene elements
1.	Disassembly of HCA - Kalo 1.5	Sides of HCA (front, back, left, right), PCB, battery, internal components, pose of device, pose of battery
2.	Disassembly protocol for HCA Qundis/Siemens	The same as above
3.	Disassembly of mixed pool of HCA Kalo 1.5 and HCA Qundis/Siemens	Recognition of HCA type, then same as above
4.	Disassembly of damaged HCA - Kalo 1.5	Sides of HCA (front, back, left, right), PCB, battery, gap changes in respect to known position in undamaged device
5.	Disassembly of unknown HCA	Pose of device (for grasping), gaps for levering, tool approach points for levering, batteries, PCBs
6	Disassembly of other devices	Pose of device (for grasping), gaps for levering, tool approach points for levering, batteries, PCBs.

The flow diagram given in Fig. 1 below provides an overview of the disassembly framework we plan to implement in ReconCycle. The diagram is targeted towards battery separation, but the steps for PCB separation would be similar.

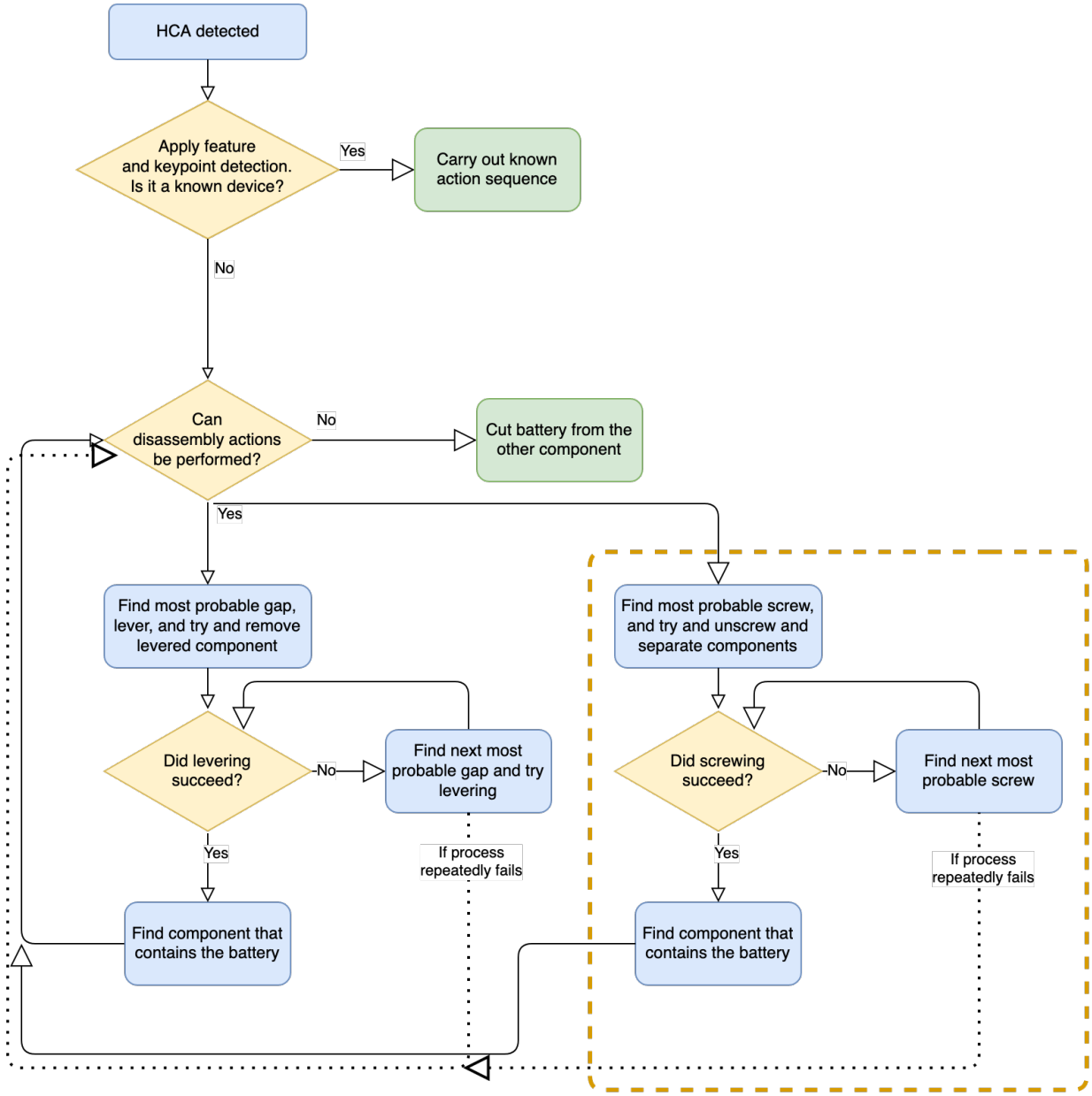

Figure 1. Flow diagram for battery disassembly for known and unknown heat cost allocators (HCAs). On the left, a levering-based approach – as currently applied – is shown. In the dashed square on the right a complementary unscrewing route is shown, which is currently not used as there are no screws in the currently processed devices.

The flow diagram begins with the HCA being detected on the work surface of the work cell. The device is detected using the above-the-scene RGB camera, and the pose of the device is determined. Using techniques described in Section 3, the device can be identified as being of the *same class* as one that has been disassembled previously, or as an examplar belonging to a *new device class*. For each device that is disassembled using this framework, the steps taken to carry out the disassembly process are stored (see Section 3 for details). If the class of device, to which the current exemplar belongs, has been disassembled before, the same disassembly procedure as before is attempted, with fall-back procedures for cases when this fails. If this device class has not been disassembled before, disassembly procedures will be attempted in an exploratory way. Currently, devices do not contain screws, so the procedures based on levering shown on the left side of the figure are carried out. Essentially, levering is performed until it is not possible to lever any more. Then the battery is cut off with material remaining around. Note, cleaner separation is more desired, but partial separation is also acceptable from business point of view of partner ECYC. Hence, the only two possible disassembly actions currently are levering and

cutting. In the future we expect to disassemble devices with the screws, too, where we then plan to employ the right part of the flow chart shown within a yellow dashed line. The procedure is similar as in the case of levering, where unscrewing is performed until components can be successfully separated.

Let us consider the disassembly based on levering actions (left side of the diagram) in more depth. First, the device is examined using the RGB camera and a depth camera. For levering actions, gaps are detected on the device. If a gap is detected for a potential levering action, then this action will be attempted. Afterwards, the success of the levering action will be determined. If more than one gap for levering is detected, then again a levering action will be attempted using those gaps. In each step of the process, the part of the device that contains the battery is of the biggest concern and shall be used for further processing. When no more levering actions can be performed, the resulting compound is used to cut the battery away from it by use of the cutter. The aim is to remove as many plastic parts as possible, however cutting away the battery only with partial success (that is some remaining plastic residue) is also acceptable in the given application.

This is the basic procedure outlined in the flow diagram. When more possible actions are added (e.g. unscrewing, as discussed above), the system will need to determine for every step the currently best possible action for disassembly. This is will be considered later in the project.

3. Specifications of Device & Component Segmentation and Pose Estimation

General definitions: The following *faces* of any device must be detected, when the device has a cuboid-like shape: front, back, left side, right side.

The following *components* must be detected: PCB, battery, plastic casing and plastic internal components.

Given an image, elements that are listed above are segmented from the background and assigned their respective label (device, battery, PCB, plastic casing). When estimating the pose of the device, first the label of the face directed towards the camera is identified. Furthermore, pose estimation uses the position and rotation of the component as given by the segmentation mask.

HCA-specific: All HCAs that we have obtained from ECYC so far are all cuboid-like in shape, and for pose estimation we recognize the front, back, left and right side. The HCA Kalo 1.5 has a lithium battery of type BR-1/2AA made by Panasonic or by Varta.

Procedures for segmentation and pose detection: Here instance segmentation is used, based on the camera image, when the camera is mounted directly above the work surface. Instance segmentation provides segments as well as segment labels. For analysis, the HCA should be placed on the work surface. The computer vision system used is called YOLACT[1], which is a neural network approach for generating real-time instance segmentation using convolutional layers. The input images are of size 1450 x 1450. YOLACT uses the ResNet-101 backbone[2].

The pose of the device is extracted using the instance segmentation results: class label for each instance as well as segment masks. We determine four class labels: front, back, left side, or right side. To get the orientation of the device on the image plane, we apply PCA to fit the smallest possible rectangle to the segmentation mask. This rectangle is called the oriented bounding box (or OBB). The orientation of the OBB is considered to be the orientation of the device. The centre point of the OBB is considered as the centre of the device. The size of the OBB is considered as the (approximate) size of the device.

In training YOLACT, the images are augmented by randomising hue, vibrance, contrast, saturation. Scaling, mirroring, flipping, and rotating are also randomly applied. The neural network has been

---

[1] Bolya, Daniel, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. 'YOLACT: Real-Time Instance Segmentation'. *ArXiv:1904.02689 [Cs]*, 24 October 2019. http://arxiv.org/abs/1904.02689.

[2] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 'Deep Residual Learning for Image Recognition'. *ArXiv:1512.03385 [Cs]*, 10 December 2015. http://arxiv.org/abs/1512.03385.

trained on a training set of 1000 annotated images of the HCAs in many different poses, configurations and states of disassembly, together with occlusions from robot arms and other systems in the setup. For each annotated instance in an image, a class label is provided for that instance. The possible class labels are: front, back, left side, right side, The neural network currently used is pre-trained on 20,000 images generated synthetically using the Nvidia Dataset Synthesizer, a plugin for Unreal Engine 4[3]. The synthetic models were created using the Artec Spider 3d scanning tool. Further testing needs to be done to assess the effectiveness of using synthetic data.

Application notes: Note that the neural network has difficulties crossing the reality gap from synthetic to real data. Using YOLACT that is first pre-trained on ImageNet and then trained on our specific dataset is also a possibility that works and does not require pre-training on synthetic data. Training only on synthetic data, however, is not good enough to detect the devices and components reliably.

Generalisation to other devices: A large training set is created of images of a variety of HCA-classes together with hand labelled segmentation masks. 15 different classes of HCA are currently available to us. For each device 100 to 200 images are taken and manually annotated. To imitate a situation of the arrival of an unseen device, we will train YOLACT on 10 classes or more and leave at least 2 classes for testing. After training, YOLACT should give reliable instance segmentation labels and masks for HCA classes that are not seen in the training data.

The images taken of each HCA class are the HCA in various states of disassembly. These images contain the device showing each face: front, back, left, and right side. Multiples of the same device can be in an image simultaneously. For creating training data, the device is manually disassembled step-by-step taking images at every step of the procedure showing the device in the corresponding hardware, such as the vice, the cutter (see Deliverable D1.1 for description of the existing ReconCycle components), and in the hand of the robot.

Batteries come in all shapes and sizes and can look completely different to batteries in this image training set. To address this, a *labelled* dataset of batteries is created, containing many batteries of different shapes, sizes and colours. For example: there are around 15 different popular coin sized batteries, which all have similar appearance, but different diameter. There are many different manufacturers of cylindrical batteries, and these usually have the brand of the manufacturer written on them. We are using the 10 most popular cylindrical batteries from ten most popular manufactures. This forms part of the training data. The same could be done for the PCBs, however there is less variation here than for the batteries.

[3] To, Thang, Jonathan Tremblay, Duncan Mckay, Yukie Yamaguchi, Kerby Leung, Adrian Balanon, Jia Cheng, William Hodge, and Stan Birchfield. *NVIDIA Deep Learning Dataset Synthesizer (NDDS)*. C++. 2018. Reprint, NVIDIA Corporation, 2021. https://github.com/NVIDIA/Dataset_Synthesizer.

4. Specifications of Device Re-identification

Types of the HCA handled: Kalo 1.5, Qundis/Siemens.

Procedures for HCA type detection. The HCA type is detected using ResNet or a similar convolutional neural network that solves the classification task on images. Given training data of images of the HCA and the corresponding type as label, the classifier learns to distinguish the HCA types.

Generalisation to other HCA types and devices. The device is detected using image segmentation as described in Section 2. From here the device needs to be identified as one that has been seen before or not seen before. At the current phase of the project, this is still work in progress. For that, we plan to use a neural network classifier trained on several classes of devices. After training on those classes, the second to last layer of the network is used as a feature vector for any image of a device provided at the input. This feature vector describes the device in a meaningful way. Using unsupervised clustering techniques such as t-SNE[4] or principle component analysis (PCA), the feature vector can be mapped to a space where both, classes of devices on which the network was trained and classes which were not participating in training, form different clusters. This way, clusters for new devices, which have not participated in network training, could be identified and labelled.

Alternatively, a convolutional autoencoder could be used for the same purpose. Trained on different classes of devices without their explicit class label, the encoding of the autoencoder provides a feature vector at its bottleneck layer for each device. This feature vector can be mapped to a lower dimensional space, again using techniques like t-SNE or PCA, where classes should form their own clusters. The autoencoder can be trained in a supervised manner by having the same image as was provided to the input as the desired output. Experimentation w.r.t. these different methods needs to be done to find the best technique for evaluation of unseen devices.

5. Specification of gaps

General definitions: Gaps are defined as an unoccupied space in a device, where around the space, there are components such as plastic parts, PCBs or batteries. The gap can be a point of disassembly. A tool end point could be inserted into the gap for the purposes of levering to separate parts of the device.

Procedures for gap detection in case HCA type is known: For a known HCA type, the gaps that lead to successful disassembly are also known, too, based on human disassembly attempts. For a known device that is broken, the known gaps can be examined using a depth camera (specifically, we use Intel Realsense D435 depth camera) to see if these gaps are still intact. In Fig. 2 B the depth image of the Kalo 1.5 HCA (panel A) is shown, where three gaps are detected by a depth camera as indicated by white ellipses in panel B. If so, the levering procedure can be attempted. The Intel Realsense D435 camera has been chosen, because it provides high resolution depth data for small scale objects. The Nerian 3D Stereo Vision camera is also being tested for how well it can detect gaps in the HCAs.

[4] Maaten, Laurens van der, and Geoffrey Hinton. 'Viualizing Data Using t-SNE'. *Journal of Machine Learning Research* 9 (1 November 2008): 2579–2605.

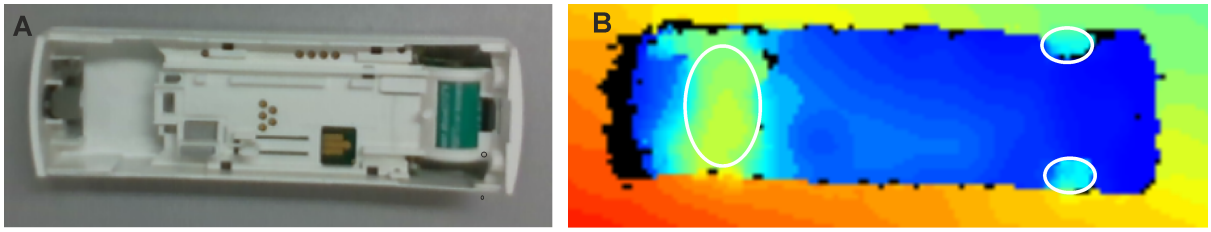

Figure 2. Intel Realsense D435 camera showing RGB image (A) and depth image (B) of the Kalo 1.5 HCA. The camera is positioned directly above the device at a distance of 20cm. A big gap on the left side as well as two smaller gaps on the right-hand side are visible in the provided heat-map (white ellipses).

Procedures for gap detection in unknown devices: If the device is unknown, then there is no data about which gaps would allow levering to lead to a successful disassembly procedure. A depth camera is used, together with classical machine learning techniques and a trained image segmentation neural network, to reliably detect gaps in the unknown device. A procedure is made for choosing the most likely gap to apply a levering action to. Gaps can be attempted in order of the decreasing probability, which would be a similar procedure to a human disassembly, too.

6. Specification of screws

General definitions: Screws are frequent part in small electronic appliances. However, HCAs are normally constructed without screws.

Procedures for screw detection: If screw detection is needed, we will re-use our previous work (from the H2020 IMAGINE project) for screw detection[5].

7. Storage and Retrieval Properties of the Disassembly Framework

General definitions: In attempting to disassemble broken devices and unknown devices, disassembly steps are carried out with varying degrees of success. The aim is to record corresponding disassembly processes in a way that allows analysis of accumulated data and re-use of that data in a situation-matched way.

Procedures for storing successful actions. We are designing a database for storing a joint description, encompassing semantic components extracted from the visual scene and corresponding components of the executed action. Semantic components of the visual scene were described in Section 3. Disassembly steps corresponding to the scene will be stored as parameterized action hierarchy. Success of the execution will be included in the database, too.

Retrieval: We will be evaluating similarity between a new scene and the stored scenes. For that, a similarity measure needs be defined. If similar enough scenes will be found, we will be executing corresponding action sequences taken from the database. For example, after gaps are detected in an unseen device, we can choose from database records where a similar gap configuration was found. Within these records, we can check in the database which of the observed gaps was allowing a successful levering action. The corresponding levering action will be attempted first, before probing other gaps. This way less probing on detected gaps will be needed before success in levering is achieved.

---

[5] Yildiz, E., & Wörgötter, F. (2019). DCNN-based screw detection for automated disassembly processes. In *2019 15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)* (pp. 187-192). IEEE.

8. Conclusion

To make use of the extracted scene semantic components, the components defined in this deliverable shall be paired with actions performed in those situations. Defining action structures that pair appropriately with extracted semantic scene components is the next required step towards generalized disassembly procedures, required in steps 4-6 defined in Table 1 above.