An Extended Visual Intelligence Scheme for Disassembly in Automated Recycling Routines

Erenus Yildiz[1] [0000−0002−3601−7328], Erwan Renaudo[2] [0000−0003−3282−8972], Jakob Hollenstein[2] [0000−0001−5694−691X], Justus Piater[2] [0000−0002−1898−3362], and Florentin Wörgötter[1] [0000−0001−8206−9738]

[1] III.Physics Institute, Georg-August University of Göttingen, Germany
{eyildiz, worgott}@gwdg.de

[2] Department of Computer Science, University of Innsbruck, Innsbruck, Austria
erwan.renaudo, jakob.hollenstein, justus.piater}@uibk.ac.at

Abstract. The state-of-the-art of deep learning models is becoming useful in real-world tasks such as disassembly of electronic devices by robotic manipulation. We have previously proposed a visual intelligence scheme to automate the robotic disassembly of computer hard drives. This extended paper addresses the remaining problems during the scene analysis, such as encountering various screw types and wires. We propose an extension to our previously published visual scheme for disassembly of devices by robotic manipulation and implement additional supervised learning modules that utilize state-of-the-art deep learning models to solve the remaining significant problems. We evaluate these models individually and also evaluate the extended scheme's capabilities on a completely unseen device (a graphical processing unit), to evaluate the schemes generalization capability. To our knowledge, this is the first scheme to address the entire disassembly process of the chosen device including various screw types and wires.

Keywords: Wire Detection, Screw Classification, Automation, E-Waste, Recycling

1 Introduction

Due to the latest developments in information technology, the volume of obsolete electronic devices (e-waste) is growing. The composition of these products raise two main issues: firstly, electronic products (such as mobile phones, computers and televisions) contain heavy metals such as Mercury or Beryllium. The exposure of humans to these elements (e.g., destructive disassembly routines) have the potential to cause cancer [45] or pollute the environment. Secondly, electronic products contain precious materials in higher proportion than natural ore deposits (e.g. 100 times more gold in a tonne of discarded mobile phones than in a tonne of gold ore [40]). The destructive recycling process causes both health and economical dangers and thus there are both health and economical reasons to improve automation of the disassembly and recycling processes.

E-waste disassembly represents an interesting application for autonomous robotics: it is desirable to automate it due to the tedious repetitive actions required from the operator; but there is a high variability of devices or shapes within a category of devices. The latter should be addressed by using a visual scheme in such a system. This means recognizing visually slightly different devices of the same category (e.g. mobile phones) as well as generalizing the learned knowledge to unknown devices, provided similar physical features are present. In [46] we proposed a visual scheme to analyse computer hard drives (HDD) and produce a representation that can be used to disassemble them. However, it has some limitations: wires can not be detected; only the type of screws used in HDDs is detected (e.g., Torx). In this work, we address two problems: *extending the detection capabilities* of the system and assessing its *ability to generalize* to other devices (e.g., graphics processing units, GPUs). For the first problem, we introduce two modules (wire detection & screw classification) as well as a bookkeeping mechanism to better track the disassembly status and confirm removed, moved or introduced parts. For the second problem, we evaluate the performance of the system given various levels of retraining on new data. This paper is organized as follows: Section 2 gives a review of the literature on visual intelligence schemes, Section 3 presents our extended approach in detail, Section 5 presents the evaluation of the schemes capabilities on the unknown device. Finally, section 6 discusses the impact of these results and concludes.

2 Related work

Computer vision methods have already been used to automate certain processes in industrial applications in the past decades [2,29]. There is, however, a lack of vision-guided automated recycling. To use robotic manipulation in this scenario certain entities of the device (e.g., wire, screw, part) need to be identified with high accuracy. Early works show that the uncertainty problem at the operational level can be solved by an integrated sensory approach. Gil et al. [12] implemented a multi-sensorial system that combines information from a tactile sensor and a vision system in order to perform visual-servoing of the robot. The conceptual test was conducted by removing a bolt from a straight slot. They also worked on detecting partial occlusions of components using vision to simplify the disassembly task [13]. The conceptual test for this system was the detection of circuit boards.

In a 2006 survey [42], Weigl-Seitz et al. list a number of limitations found in the literature: the lack of datasets for the required tasks, the inability of existing schemes to account for more than certain number of devices and that algorithms employed were mostly inspired by classical computer vision methods with no online learning paradigm involved.

In the last decade, however, there have been plenty of works [2,39,1] focusing on detecting certain types of parts, such as screws and bolts. Most of these works either achieved only prohibitively low accuracy or they were only usable for a very narrow set of entities (e.g., only one type/size of screw). There have also been model-based methods [9,43,39,30,38,37] that were solving the problem for a specific model of screws or bolts found in/over a specific device model such as an electric motor. However, as stated in our previous work [46], assuming models of the parts to detect are available limits strongly the adaptability of the visual scheme as the variety of brands and models per device increase exponential the number of different models required. Finally, a visual scheme specifically trained for a device or a part type is also limited as the recycling disassembly process includes various type of devices. The most notable effort came in 2018 where Jahanian et al. [20] showed the disassembly of mobile phones using state-of-the-art segmentation networks. The prototype, however, was limited to work with a very limited set of mobile phones from a limited set of manufacturers. Our previous work [46] also partially suffer from this limitation as it is trained only on HDD data. This reduces the general application of these works in real recycling plants.

In addition, the original pipeline we proposed had three structural limitations: first, its inability to classify the type of detected screws causes problems on devices that contain more than one type of screw. Second, the pipeline has no means of handling wires, which is major issue as parts connected via wire can not be taken out of the device individually. Finally, the original pipeline has no awareness of its disassembly state. Thus when a part grabbed by the robot's gripper is dropped back into the scene due to a manipulation problem, the system does not know if it is the same part or not, preventing the rest of the system to evaluate the effect of the manipulation. The system needs to know on the next analysis step, that a part was moved, removed or re-introduced back into the scene.

Thus, current methods lack generalization capabilities, device and environment-independence, fault-tolerance to be used in robotic disassembly processes which involve a great degree of variance in parts. We showed that Deep Convolutional Neural Networks (DCNN) offer a powerful solution to analyze the inner structure of devices in the context of disassembly [47,46]. We further extend the capabilities of the scheme by adding two new DCNN-powered modules. Moreover, we make the extended pipeline fault-tolerant against manipulation errors to some degree. Last but not least, the proposed methods are CAD model-free, making the scheme independent of specific devices and parts. Instead, DCNNs learn to extract relevant features of the device entities (e.g., screws, wires), abstracting from manufacturing details specific to the device. End-of-Life (EOL) devices that belong to a family usually include similar entities (e.g., parts, wires, screws). Focusing on features common to these entities by employing the deep learning paradigm leads to reproducible and generalizable outputs that can be used for other devices as well.

Thus, the problem of extending the original visual intelligence scheme can be formulated as a problem where machine learning paradigms (e.g., segmentation, classification) are used in order to detect and classify the present entities of the target device, such as wires and screws, respectively. In contrast to the original scheme, the extended scheme satisfies the following requirements:

– Classification should be able to handle great variety screws types/sizes.
– Segmentation should not be affected by the great variety of physical features (e.g., color, shape) among wires.
– Moving, removing or introducing a detected part in the scene must be registered by comparing the consecutive analyses.

As we also mentioned in our original publication, many recent works [23,5,27,11,24,8] have addressed the semantic segmentation problem or pixel-wise classification problem in various domains ranging from autonomous driving to biomedical imagery.

Aforementioned work in 2018 [20] and our original visual intelligence scheme [46] are amongst the first works to utilize state-of-the-art deep learning methods, in particular convolutional networks [22] in the context automated recycling of E-Waste. Previously mentioned works preferred to use the family of R-CNN networks [15,14,31] that have been evolving recently. One of the latest R-CNN models by the time the works were published, is *Mask-RCNN* [16] which is based on the Feature Pyramid Network (FPN) [25].

Although Mask-RCNN and its ensembled models were the state of the art for instance segmentation for a while, they were outperformed by recently developed *EfficientNets* [36] models. These networks achieve much better accuracy and efficiency than previous convolutional neural networks. In particular, the EfficientNetB7, which forms the basis of our wire detection module, achieves state-of-the-art 84.3% top-1 accuracy on ImageNet. According to the authors, the networks are 8.4x smaller and inference is 6.1x faster than the best existing convolutional neural networks. It has also been shown that they transfer well and achieve state-of-the-art accuracy on CIFAR-100 (91.7%), Flowers (98.8%), and 3 other transfer learning datasets. We therefore base our modules on this family.

3 Methods

On request the extended visual scheme provides an analysis of the scene with the predictions of 4 modules: a part detection module, a screw classification module, a gap detection module and a wire detection module. The screw detection module of the original pipeline has been extended to a screw classification module that classifies the detected screws according to their type and size information. The wire detection module is employed to recognize any kind of visible wires and cables that can be found inside or around the device. Finally, a bookkeeping mechanism is in place to register every change between to consecutive frames. This is required to keep track of the disassembly sequence, as well as to gain a certain degree of fault recovery. The pseudo code for the extended pipeline is given in algorithm 1.

As mentioned earlier, the analysis of wires is a semantic segmentation problem. However, detecting the screw type and size a typical classification problem: image based classification of type and size with a deep convolutional neural network. Both problems are challenging. Wires can be any color and can occur in

Algorithm 1 Extended Perception Pipeline

$c_p, b_p, m_p :=$ [] ▷ part centers, boundaries, masks
$c_s, b_s, m_s :=$ [] ▷ screw **type/size**, centers, boundaries, masks
$c_g, b_g, v_g :=$ [] ▷ gap centers, boundaries, volumes
$c_w, b_w, v_w :=$ [] ▷ **wire centers, boundaries, masks**
$I, P := NULL$ ▷ I: Input Monocular Image, P: Input Pointcloud
$C_m, C_s := NULL$ ▷ C_m: Monocular Calibration Info, C_s: Stereo Calibration Info
predicates= []
procedure COLLECT PREDICATES
  if hddTable.State()= 0 **then**
    hddTable.changeState(angle=θ_{Stereo})
    $P \leftarrow$ getPointcloud(P)
    if $P \neq NULL$ **then**
      $c_g, b_g, v_g \leftarrow$ detectGaps(P)
    hddTable.changeState(angle=$\theta_{\text{Monocular}}$)
    $I \leftarrow$ getRGBImage(I)
    if $I \neq$ NULL & hddTable.State() = 0 **then**
      $c_p, b_p, m_p \leftarrow$ segmentParts(I)
      $c_w, b_w, m_w \leftarrow$ **segmentWires(I)**
      $c_s, b_s, m_s \leftarrow$ **detectAndClassifyScrews(I)**
  $C_m, C_s \leftarrow getCalibrationInfo()$
  predicates $\leftarrow$ mergeAllInfo($I, P, C_m, C_s, c_p,$
    $b_p, m_p, t_s, c_s, b_s, m_s, c_g, b_g, v_g, c_w, b_w, m_w$)
  bookkeep(I, predicates)
  return predicates

many shapes, including tangled wires. On top of that, there is no fixed background for wires to be used. They can be found inside or at the backside of the devices, over varying surfaces. Screws, on the other hand, are another challenge due to the high similarity between same type screws varying only in size (e.g., Torx6 and Torx7, Philips1 and Philips2). The proposed modules account for aforementioned difficulties, and execute their tasks with high accuracy.

3.1 Datasets

Datasets are at the core of any machine learning endeavour, as supervised or unsupervised machine learning algorithms heavily depend on the data available. This fact aside, even without any machine learning involved, ground truth data is required for future evaluation of any employed algorithm. In order to train the new modules of our extended pipeline, we collect and create datasets for screws and wires.

Screws are common assembling entities in electronics EOL device. Their removal is paramount to disassemble devices and access inner areas and hidden parts. A non-detected screw may hinder the entire disassembly sequence as it creates constraints between parts. Not only should screws be correctly detected, but their type must be classified so that the correct tool is selected to interacted with them.

Since we base our screw classification module on a published work [48], we use the same dataset. It was observed that most of the EOL devices considered have a certain set of screws. 12 types of screws are therefore considered: Torx 6, 7, 8, 9, Allen 2.5, 2.75, 4, Slotted 4, 6.5, 10 and Phillips 1, 2. Note that the length of the thread is irrelevant for the perception block, as they are always occluded. Therefore, any vision routine only considers the head part of the screws. Figure 1 illustrates samples from every type and size considered. In order to address detection and classification purposes, we found that 20000 positive images of screw heads are sufficient.

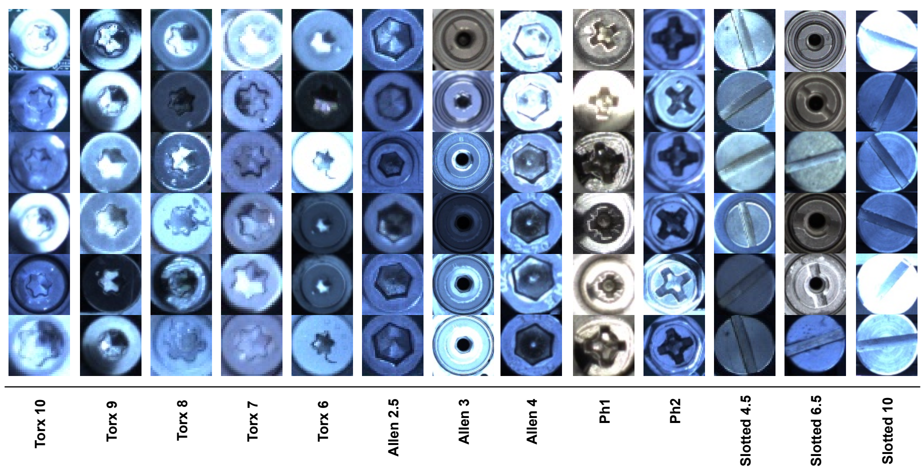


Fig. 1: Various screw types encountered during the disassembly of EOL devices, Yildiz et al.[48].

Note that the screw classification dataset does not require any negative samples, since the original pipeline already has a screw detection module, which operates to separate artefacts from screws. Therefore, it was sufficient to include the positive samples to the screw detection dataset. It must be kept in mind that in order to classify any screw, detection of that screw has to be done first. Therefore, every positive sample in the screw classification dataset, exists in the screw detection dataset as well.

All of the illustrated images were collected using the setup presented in the original paper. However, in order to account for the great variance in the dataset, the images were collected under slightly different light conditions. They were collected by the screw classification module's offline mode that was introduced in the original paper.

Wires unlike screws, do not have specific shapes (e.g., circular), making them only suitable for pixel-wise segmentation schemes. They might only appear as discontinuous segments due to occlusion in EOL devices and can be of any color. It is safe to assume that wires are the most varying entity in this domain. This makes it very difficult or even impossible to find a dedicated dataset for specific visual tasks to detect wires. Hence, one is forced to deal with limited number of annotated training samples.

3.2 Wire Segmentation

Wire detection is not a commonly studied problem in the literature. There are few notable works: The work by Madaan et al.[28] grabbed our attention, and was built on the work presented by Kasturi et al. [21], aiming to find wires in aerial navigation. This work is interesting because it is the first work to use convolutional neural networks in order to address the wire detection problem. The authors render synthetic wires using a ray tracing engine, and overlay them on 67K images from flight videos available on the internet. This synthetic dataset is used for pre-training the models before fine-tuning on real data. The work achieved 73% of precision, however, it suffers from the fact that the dataset generation requires expert knowledge in ray tracing engines. Many times these programs are not straightforward to use, making methods based on them less desirable. Additionally, the dependency on specific software, might render the data generation impossible as the required software may become unavailable in the future. The work nevertheless investigates available network architectures that could be used to detect wires, hence, it plays an important role in this research field. The authors report that dilated convolutional layers [49] were chosen since they provide a simple and effective way to gather context without reducing feature map size.

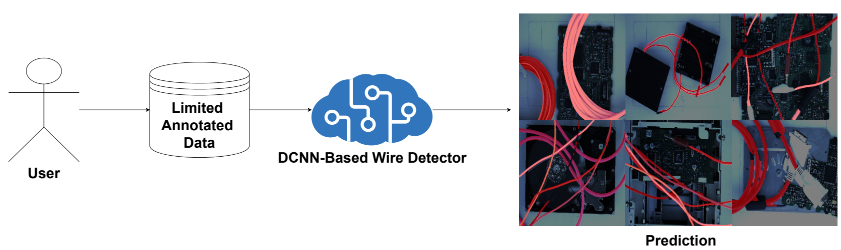


Fig. 2: We present a DCNN-Based wire detector scheme that requires limited number of annotated data from the user and delivers accurate predictions for robotic manipulation tasks.

After considering the literature, we decided to tackle the wire detection problem with DCNNs to detect, recognize and localize the wire pixels, using a paradigm called semantic segmentation. Due to the high number of available state-of-the-art networks, we train and evaluate a set of selected methods on this semantic segmentation problem: EfficientNetB7 [36], InceptionV3 [34], InceptionResnetV2 [33], Densenet201 [19]. These networks achieve over 93% top-5 accuracy [44] on the well-known Imagenet dataset [6]. There is one more criterion that we took into account, that is the number of hyperparameters. Clearly, we do not want to pick a good performing model that requires enormous amount of hyperparameters such as SENet [18]. Hence, we only evaluate the aforementioned models.

It must be mentioned that that the required training data for any DCNN should not exceed a certain amount of images, and the model should be able to generalise for all kinds of wires found in the e-waste disassembly domain. As the original visual intelligence already requires a certain amount of training data for its other modules (e.g., screw detection & classification, part segmentation), it is our intention to keep the requirement for annotated training data as low as possible. Therefore, it was decided to train any model with only 130 raw images where 100 of them would be reserved for training, 10 for validation and 20 for testing. The basic idea of our approach is shown in Figure 2.

We propose a module for the existing setup where the monocular camera faces the device's surface perpendicularly. The proposed module has the main blocks: data generation or *Datagen*, and model, as illustrated in Figure 3. The Datagen block aims to greatly augment the limited number of user-annotated raw images and generate massive amounts of augmented images along with their annotations for the training the deep neural network model. The Model block, on the other hand, aims to detect the learned features and build a segmentation map out of the input RGB image received by the monocular camera.

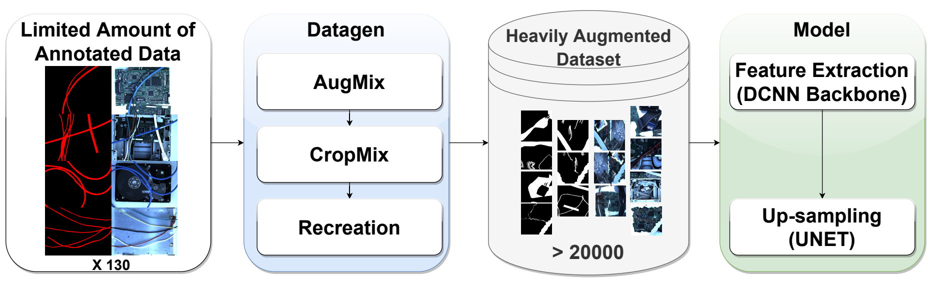


Fig. 3: Wire detection module is composed of data generation block which generates a large number of augmented images using a limited number of annotated images. An EfficientB7 [36] model trains on the massive number of generated augmented images.

Data Generation It is usually very difficult or even impossible to find a dedicated dataset for specific visual tasks such as wire detection. This inevitably forces us to deal with limited number of annotated training data. In order to succeed under this condition, one has to enrich the amount of limited annotated images. To this end, we decided to use heavy augmentation routines and generate massive amounts of annotated training data, from our existing dataset which contains only 130 annotated wire images.

For our data augmentation block, we have chosen a 3-step routine. Below, we share the augmentation library used and the operations applied. The exact parameters of the augmentation functions can be found in the published source code.

We start off by *Augmix* [17], a popular augmentation library with many options. We, however, in this step only apply shift scale rotation, blur, elastic transform, and optical distortion. In the second step, we continue with *CropMix* [35] which allows manually cropped augmentations rather than automated ones to ensure variety and mask precision based on regions. Here we go by 4-segment crop (per image), rotation, flipping along with generation ratio as 0.001 and dataset occupancy ratio (training) as 0.488. In the final step we apply our own augmentation routine. We create a new image using 4 images applying rotation, mirroring and flipping along with dataset occupancy ratio (training) as 0.416 and clustering as sample based. At the end the datagen block creates approximately 20000 images for the training data.

Model In order to conduct our investigation, we picked a use case of wire detection in digital entertainment devices such as DVD players, gaming consoles, etc. To this end, we collected 100 top-down images of open DVD players from online search engines and annotated the visible wires by hand. We then generated approximately 20000 augmented, annotated images, which served as training data for the models we investigated.

Throughout our study, we considered only two metrics to evaluate our scheme with. The standard metrics for pixel to pixel segmentation are mainly the COCO [26] average precision (AP) metrics: AP is average precision of multiple IoU's (Intersection of prediction and ground truth over Union of prediction and ground truth) values seen. The definition of IoU between a known segmentation of n pixels, Y, and a similar set of predicted segmentation, Y' (in the binary case, i.e. where Y_i, $Y_i' \in \{0, 1\}, \forall i \in [1, n]$ is as follows in Eq. 1:

$$IoU(Y, Y') = \frac{Y \cap Y'}{Y \cup Y'} = \frac{\sum_{i=1}^{n} \min(Y_i, Y_i')}{\sum_{i=1}^{n} \max(Y_i, Y_i')} \tag{1}$$

However, we decided to not to only consider IoU alone, but also the SSIM (Structural Similarity Index) [41] metric, known for measuring the objective image quality. It is based on the computation of three terms, namely the luminance term (l), the contrast term (c) and the structural term (s). The overall index is a multiplicative combination of the three terms as it is seen in Eq. 2 as follows:

$$SSIM(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} \cdot [s(x, y)]^{\gamma} \tag{2}$$

	Metric					
	SSIM			IoU/F1		
Model	**Max.**	**Mean**	**Min.**	**Max.**	**Mean**	**Min.**
EfficientNetB7	**0.956**	**0.877**	**0.761**	**0.988**	**0.952**	**0.897**
InceptionV3	0.944	0.863	0.758	0.977	0.947	0.894
InceptionResNetV2	0.941	0.859	0.743	0.983	0.943	0.886
DenseNet201	0.940	0.862	0.747	0.978	0.945	0.891

Table 1: Evaluation of the state-of-the-art models. EfficientNetB7 is proven to be the most suitable model with a high SSIM and IoU score.

where

$$l(x,y) = \frac{(2\mu_x\mu_y + C1)}{(\mu_x^2 + \mu_y^2 + C1)} \tag{3}$$

$$c(x,y) = \frac{(2\sigma_x\sigma_y + C2)}{(\sigma_x^2 + \sigma_y^2 + C2)} \tag{4}$$

$$s(x,y) = \frac{\sigma_{xy} + C3}{\sigma_x\sigma_y + C3} \tag{5}$$

where μ_x, μ_y, σ_x, σ_y, and σ_{xy} are the local means, standard deviations, and cross-covariance for images x, y. If $\alpha = \beta = \gamma = 1$, and $C_3 = C_2/2$ (default selection of C_3) the index simplifies to Eq. 6 seen below.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C1)(2\sigma_{xy} + C2)}{(\mu_x^2 + \mu_y^2 + C1)(\sigma_x^2 + \sigma_y^2 + C2)} \tag{6}$$

All of our experiments conducted were evaluated based one these two metrics.

Table 1 shows the evaluation of the state-of-the-art models based on the aforementioned metrics. One can clearly notice that the model EfficientNetB7 [36] scores slightly better in both metrics. The results clearly indicate that EfficientNetB7 is doing a better job at feature extraction from the given images (even with low resolution, low-feature conditions).

Having noticed the slightly better accuracy provided by the EfficientNetB7, we made inferences on the images of a different device, such as the XBox One gaming console. The model, although trained on DVD player wires, was able to make good predictions on the gaming console wires, marking their locations for a possible robotic manipulation action. Figure 4 illustrates this use case. We used the state-of-the-art UNET [32] model as our up-sampling backbone for all our feature extractor models.

3.3 Screw Classification

We base this module on the pipeline we inherit from a previous work [48]. As stated in that work, the module enables the user to collect training data by

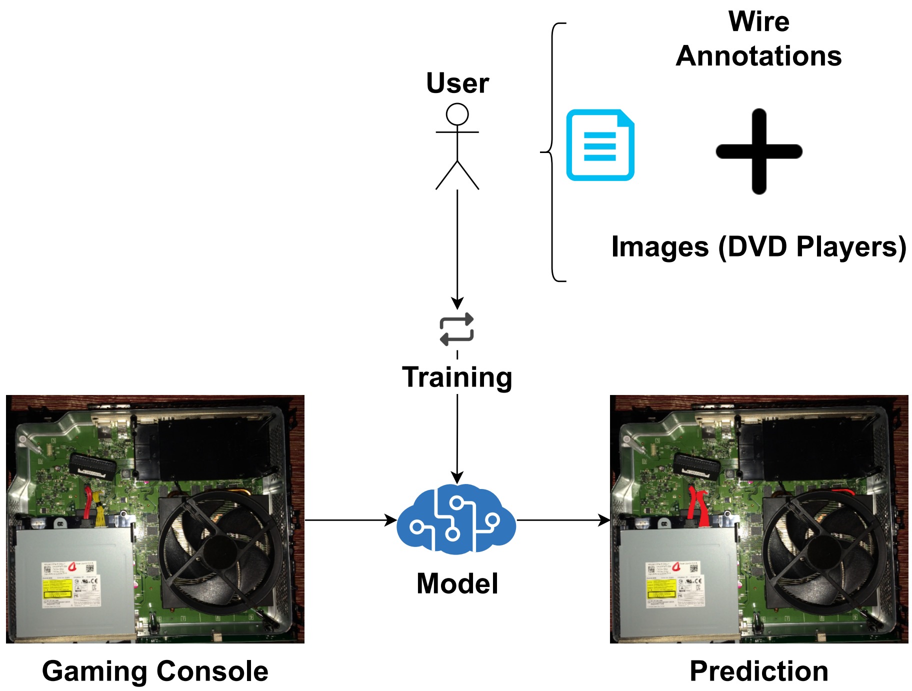


Fig. 4: We investigated the state-of-the-art models' capabilities with limited amount of data. After training the models wires of DVD players, we infer on XBox One gaming console wires. Both devices belong to the same family of devices, despite of having different inner layouts.

cropping circular candidates from the scene. The cropped circular candidates are then to be divided into their respective classes (e.g., artifact, Torx8, Ph2, Slotted6.5, Allen2.75, etc.) by a human.

First, the screw detector model is trained to classify screws from artefacts (circular non-screws structures), as explained in the work [47]. As the original paper for classification [48] instructed, the new screw data consisting of 12 different types of screws is included. This corresponds over 20000 samples, which are split into training and validation sets with the ratio of 2:1, as instructed in the original paper we base our approach on [48]. We refer the reader to his publication for the details of the training process.

The screw classification module employed marks and returns the type/size information and locations of the screws seen in the image, as illustrated in Figure 6.

The classifier accuracy reported in Table 2 is directly taken from the work [48] published. The table summarizes the experimental results with regards to accuracy of each classifier against the validation set, clearly showing that EfficientNet-

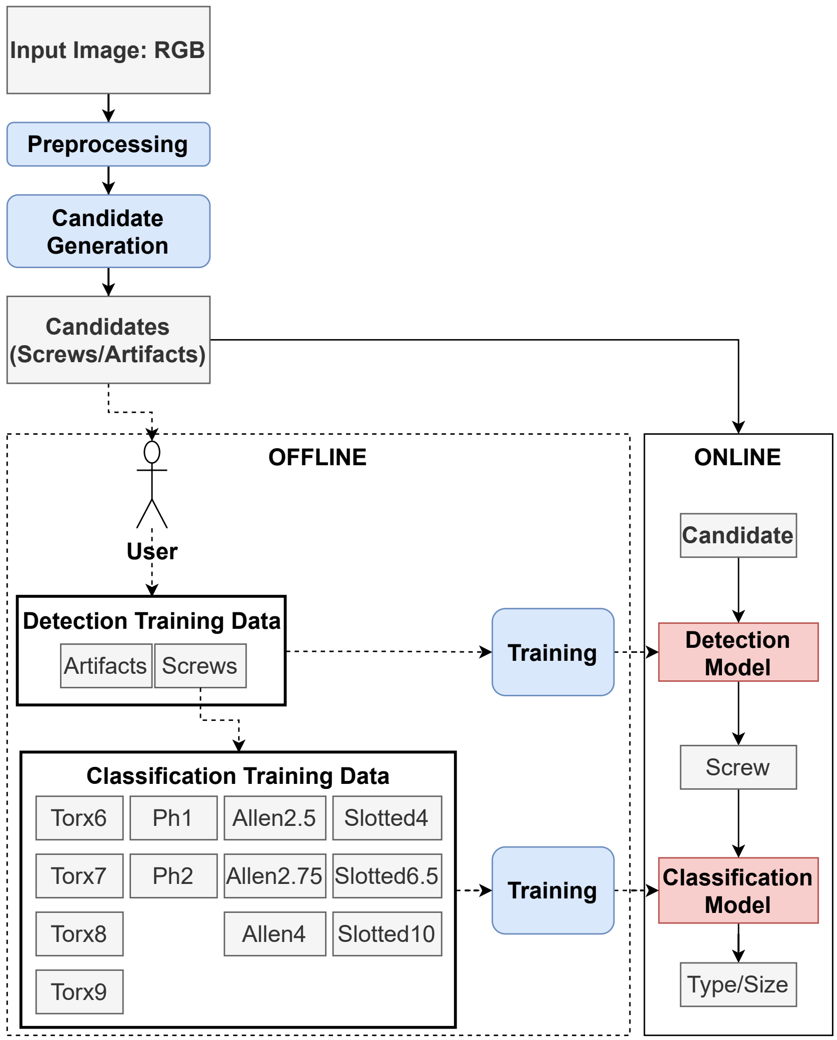


Fig. 5: Screw classification pipeline inherited from Yildiz et al. 2020 [48].

B2 re-trained on the Noisy-Student dataset with the given parameters is proven to be the best choice.

Moreover, we underline the fact that augmentation strategy plays a pivotal role in the classifier accuracy. In case of circular objects, rotation operation guarantees that the training data accounts for screws that are rotated for each angle, as reported in the original publication [48]. The Albumentations [4] library was used to apply a rotation of 360 degrees, horizontal and vertical flips, as well as brightness and contrast changes.

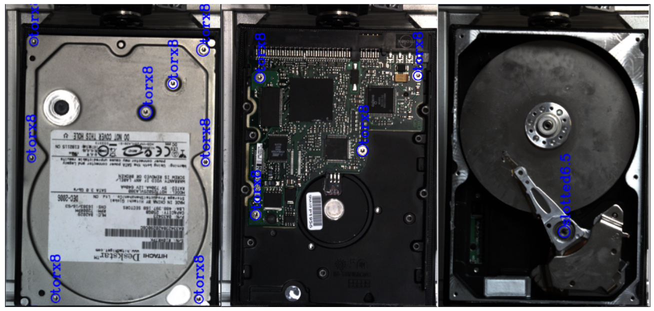


Fig. 6: Classified screw heads by the screw classification method taken from Yildiz et al. 2020 [48]

.

Table 2: Accuracy of the state-of-the-art models with huge variation of hyper-parameters. Highlighted one is the top performing one [48].

Model	Grayscale	Size	Loss	Acc.	Min. Acc.	F1	Transfer Learning
EfficientNetB2A	**No**	**256**	**0.1187**	**0.968**	**0.79**	**0.97**	**Noisy Student**
EfficientNetB2A	No	64	0.2144	0.936	0.78	0.93	ImageNet
EfficientNetB2A	No	128	0.1871	0.951	0.85	0.95	ImageNet
EfficientNetB2A	Yes	128	0.2199	0.948	0.67	0.94	ImageNet
EfficientNetB3A	Yes	64	0.2072	0.937	0.75	0.93	ImageNet
EfficientNetB3A	No	64	0.2051	0.939	0.74	0.94	ImageNet
DenseNet121	No	128	0.1415	0.961	0.81	0.96	ImageNet
DenseNet121	Yes	128	0.1489	0.957	0.74	0.95	ImageNet
DenseNet121	No	64	0.1896	0.937	0.72	0.93	ImageNet
DenseNet121	No	64	0.2306	0.934	0.71	0.93	ImageNet
DenseNet201	No	256	0.1170	0.966	0.79	0.96	ImageNet
ResNet34	No	128	0.1538	0.955	0.80	0.95	ImageNet
ResNet34	Yes	128	0.2026	0.951	0.69	0.95	ImageNet
ResNet50v2	No	256	0.1732	0.942	0.73	0.94	ImageNet

4 Bookkeeping

In the context of disassembly, a bookkeeping mechanism aims to register the status of every recognized part in the scene. This is carried out by analysing the predicted pixel-level changes between part boundaries found by the part segmentation module exists in the original scheme. The mechanism accounts for multiple situations that are explained below. It also allows user to specify the sensitivity of the change it should consider before registering. This is a required feature since, every change is detected by conducting pixel-wise comparison of part boundaries and regions in consecutive frames, meaning that a larger device (and components) may require a less sensitive analysis of changes, as a few pixels of change in large part's boundaries may not exactly mean a misplacement or failed action. If the EOL device is large, then a little touch on its part is not worth registration. On the other hand, if the EOL device is a small one (as in the case of hard drives), then a few pixels may mean more, given some parts such as a *spindle hub* are relatively small, meaning that every pixel should count towards the change threshold.

- **Difference in List of Parts:** When the lists of parts are different between consecutive frames, introduction or removal of those parts are registered.
- **Difference in Locations of Parts**: When the parts appear to be in different locations (if their center points moved more than the user-specified margin) between consecutive frames, this change is registered.

The bookkeeping mechanism has a timer, additionally allowing the system to not consider a frame that is acquired beyond the user-specified time interval. This is a needed feature as well, since the registration should not occur between a frame from a previous system run. This user-specified parameter is set to 5 minutes by default, considering a frame as consecutive if and only if the frame is acquired within this time. If not, it registers the frame as the primary frame, and starts the timer for another 5 minutes, as explained in algorithm above.

By employing the described mechanism, the extended scheme gains a certain degree of fault-tolerance and guarantees the continuation of disassembly process despite of manipulation errors.

5 Experimental Evaluation

As the original visual scheme requires two inputs (a top-down RGB image and a top-down point-cloud), we use the same setup from the original paper with a Basler acA4600-7gc monocular camera which provides images with 4608×3288 resolution and 3.5 FPS and a Nerian Karmin2 stereo camera with a depth error of 0.06cm from the minimum range of 35cm.

We then let the extended visual scheme prove its capabilities given scenes of computer hard drives (HDDs) with wires and screws. We quantify these results and additionally conduct a study to assess the generalization ability of the extended scheme.

5.1 Evaluation method

There are two new modules in the extended scheme and each of these modules has to be evaluated differently, as the paradigms running behind are different.

Wire Detection Since we have investigated the state-of-the-art models and found out that EfficientNetB7 performing the best, we decided to use a dedicated wire dataset and train the model from the scratch. To this end, approximately 130 images of wires were collected manually, using the same setup introduced in the original paper. The strategy was to use any type of wires (including connectors) and manually create occlusions with arbitrary EOL components.

Model	Metric	Min.	Max.	Mean
EfficientNetB7	SSIM	0.80	0.98	0.91
EfficientNetB7	IoU/F1	0.89	0.99	0.97

Table 3: Evaluation of our trained EfficientNetB7 model on the test data, using SSMI and IoU metrics.

As backgrounds, mostly PCBs were used, as wires are mostly found on PCBs in EOL devices. Figure 7 shows a few samples from the wire dataset acquired. Wires were later annotated with the VIA tool [7].

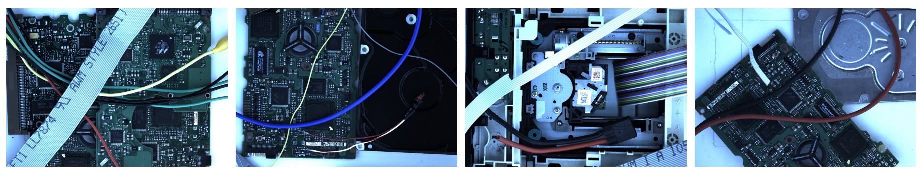


Fig. 7: Sample raw images from the wire dataset.

After choosing the model, we collected a new dataset of high (4K) resolution top-down images of wires that could be found in disassembly environments. Those are various wires taken out of devices as well as wires that connect devices to data or power supplies. As background, we chose the most commonly encountered ones such as PCBs, device lids and bays, as well as the work station surface. Having collected 130 various wires images, we split 100 of them for training, 10 for validation and 20 for testing. Below we also provide the details of the experimental evaluation and present our final model.

We use the Google provided TPU v2 for training our model via Colaboratory [3] environment which has 64 GB High Bandwidth Memory (HBM) and provides 180 TFlops computing power. The validation set was taken to be 1/10 of training dataset and training was done with early stopping enabled callbacks so that the model does not overfit. We trained our model with 4 generations of data each time taking one forth of the total data. (i.e- the augmentations are tuned in a way every time about 50 percent of training data is completely new). No transfer learning was used and the model was trained from scratch. For the final model, however, the best of these 4 weights were taken to train on the whole training dataset. As the MSE loss graph shows in Figure 8, the model reaches stable losses very quickly and converges to a final point where a plateau stage can be encountered before stopping. We summarize the experimental results with regards to performance of each classifier against the testing set in Table 1.

From the collected results in Table 3, we conclude that EfficientNetB7 outperforms any other state of the art models that are capable of semantic segmentation on this task. The model outputs show high similarity and reach a good IoU score as well.

We noticed that the resolution of the images inferred must also be of the same resolution of training data, because lowering the resolution creates completely different feature maps compared to what the model was trained on. Figure 9 illustrates some of the detections on the test set by our detector. Our scheme can handle delicate cases such as wires with tangles, and partial occlusions, which are frequently encountered cases during the disassembly of an electronic device. Our model proves to be robust, handling such delicate situations with maintaining high accuracy. We report minimum SSIM and IoU of 0.80 and 0.89, respectively.

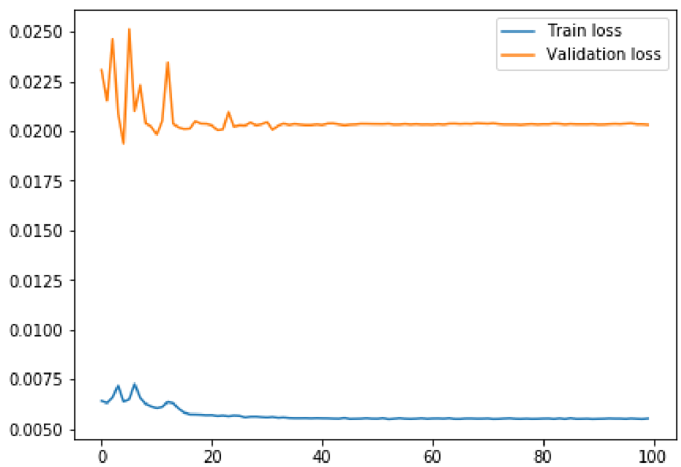


Fig. 8: Loss in MSE of our model.

Last but not least, we tested our trained model on devices that the network has never seen before, such as thermostats. Figure 9 illustrates the found wires in these devices. The results prove that our model can be used in disassembly environments where the target device was not seen before, which is usually the case.

Screw Classification Although the classifier accuracy is quite high, due to the fact that Hough circle finder method misses out finding the circles in the first place, our final average precision was found to be 80%. As the classifier expects images that are directly suggested by the Hough circle finder, any circle that is missed, is also not considered by the classifier. In other words, the classifier's ability is limited by the Hough circle finder. Therefore, the AP remains at 80%. We refer the reader to the Figure 1 to illustrate a few detection samples during the HDD disassembly sequences.

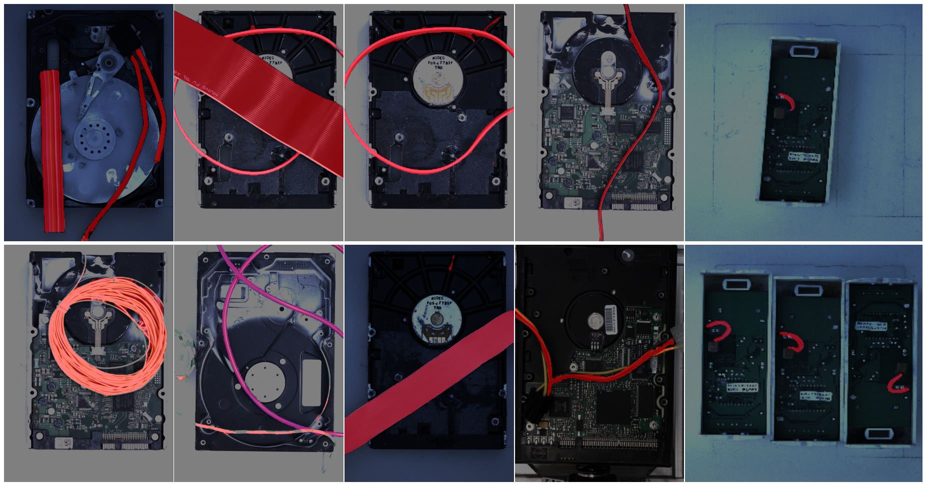


Fig. 9: Our model has a clear generalization ability since it is able to detect wires found in devices that it has not seen before. We tested it on the back side of heat allocators as well as both sides of hard drives with arbitrarily added wires on them.

As the pipeline is composed of two main blocks, namely the Hough circle detector and our classifier, EfficientNetB2, it is required to assess the detection as well as the classification abilities of it. To this end, the following strategy was pursued: First, the test images were annotated, each having only one hard drive with top-down view. These images contain drives with or without screws, by which the Hough circle finder could be assessed. These scene images were annotated by marking screws with squares, which would form the ground truth for assessing the Hough circle finder's accuracy. The standard VOC evaluation [10] was preferred and it was found out that the Hough circle detector actually works with 0.783 mean IoU (Intersection over Union) with the optimal parameters found for the IMAGINE setup. IoU here refers to what amount of screw region is correctly detected by the Hough circle finder. If the detected region for a screw is below 70%, it is bound to result in bad prediction for both detection and classification. It must be also noted that the pipeline is limited by the accuracy of Hough Transform and the screw detection previously introduced. Although the accuracy of Hough circle detection can vary depending on the parameters of the function such as min/max radius, min/max threshold, final accuracy of the pipeline is found 0.75, and calculated as follows:

$$Acc\_P = Acc\_CD * Acc\_SD * Acc\_SC$$

where Acc_P stands for the accuracy of pipeline, Acc_{CD} stands for the accuracy of the circle detector, Acc_{SD} stands for the accuracy of the screw detector, and Acc_{SC} stands for the accuracy of the screw classifier.

5.2 Generalization

Ideally, the scheme should also be evaluated on a second EOL product, so that its capabilities are proven to be robust and generalizing enough for an industrial use. For this purpose, another computer piece – GPU – was chosen. In total, 8 GPUs from various brands and models were collected. We evaluate the performance of the detection and classification of the system both on HDDs (which the visual intelligence is trained for) and GPUs (which the visual intelligence has never seen). In order to understand how fast the system can adapt, we define three experiments (E1-E3) with incremental retraining on the new device. This retraining is done with a limited training dataset of user-annotated GPUs data (see Table 4).

Module\Data	**RGB Images Existing Data (E.D.)**	**RGB Images Collected Data (C.D.)**
Component Segmentation	600	100
Screw Detection	20000	2000
Screw Classification	20000	2000
Wire Detection	130	100

Table 4: Existing data consists of RGB images of HDD images, whereas collected data consists of RGB images of GPU images. Since the modules were already trained optimally with the existing data, the collected data used was intentionally kept limited.

Experiment	**Data**	**Re-training**	**Test Device**
E1	E.D.	No	HDD, GPU
E2	E.D. + 50% C.D.	Yes	HDD, GPU
E3	E.D. + 100% C.D.	Yes	HDD, GPU

Table 5: Evaluation scheme to be used through the experiments.

Experiment E1 corresponds to the evaluation of the system trained on HDD data and without retraining on the GPU data (no annotated training data e.g., GPU screws, GPU components, GPU wires). Between the two classes of device, the most common component is the PCB (which is also the biggest entity on the GPU). Components such as bay, fan, sockets, screws and optionally wires exist in various colors and types. For instance, PCBs found in GPUs vary in colors of black, blue and green, whereas for HDDs they are green by a very large margin. Nevertheless, by conducting E1 we expect to evaluate how our approach is able to generalize to other electronic devices, i.e. whether the initial training on HDD is representative enough of commonly found parts in E-waste in general (wires, screws, PCB, etc.). Experiment E2 and E3 on the other hand, evaluate the scheme's capabilities after retraining it with 50% and 100% training data, respectively. By conducting E2 and E3, the question of "How does retraining with limited data affect the performance of the scheme on the second device?" is answered. Note that the gap detector was not evaluated on GPUs in any experiment, since there are no gaps significant to the disassembly of the device.

Table 4 shows the experimental data in numbers. For every module, the data was split into training, validation and test sets in ratios of 70%, 20% and 10%, respectively. Evaluation scheme is illustrated in Table 5. None of the training strategies is subject to change (e.g., early stopping). Weights were reset between E1, E2 and E3 experiments to prevent learning of repetitive features and introducing bias.

Experiment	**S.D.A.**	**S.C.A.**	**C.S.A.**	**W.D.A.**
E1 (no retraining)	0.91	0.94	0.71	(0.89, 0.88)
E2 (retraining with 50% GPU data)	0.99	1.0	0.78	(0.91, 0.93)
E3 (retraining with 100% GPU data)	0.99	1.0	0.84	(0.92, 0.93)

– **S.D.A.**: Screw Detection Accuracy
– **S.C.A.**: Screw Classification Accuracy
– **C.S.A.**: Component Segmentation Accuracy
– **W.D.A.**: Wire Detection Accuracy

Table 6: Accuracy of each module through experiments E1, E2, and E3.

Experiment E1 Experiment E1 is conducted on each module, testing each one's capabilities on performing visual tasks on raw HDD and GPU images without re-training. For screws, the description of the entity is largely the same (e.g., circular shape, feature in the center) therefore there is no drastic significant drop in accuracy of screw detection network, scoring 0.91. There is an insignificant drop from the original accuracy 0.99 [47] due to the fact that GPUs have black and dark gray screws which the network misses from time to time.

As for the screw classifier, the weighted average was found to be 0.94. Here as well, an insignificant drop was observed due to the aforementioned reason. Nevertheless, it was able to find what it was trained on when the learned color was present. Figure 10 illustrates such an example. The image above contains screws that have the ordinary silver-metallic color. Note that the screws in/on HDDs were of this color. Thus, the network has the learned features from the images of these HDDs. On the image below, however, it is noticeable that the detection network misses more. Since all the screws found on that particular GPU were of dark gray or black color, the accuracy is naturally lower. However, even in this case, the classifier nevertheless correctly identified the found screws as "ph1" and "torx6 as illustrated. It must be remembered that the classifier only classifies once the detector detects an instance. If there were only two classifications, it is due to the fact that there were only two screws detected.

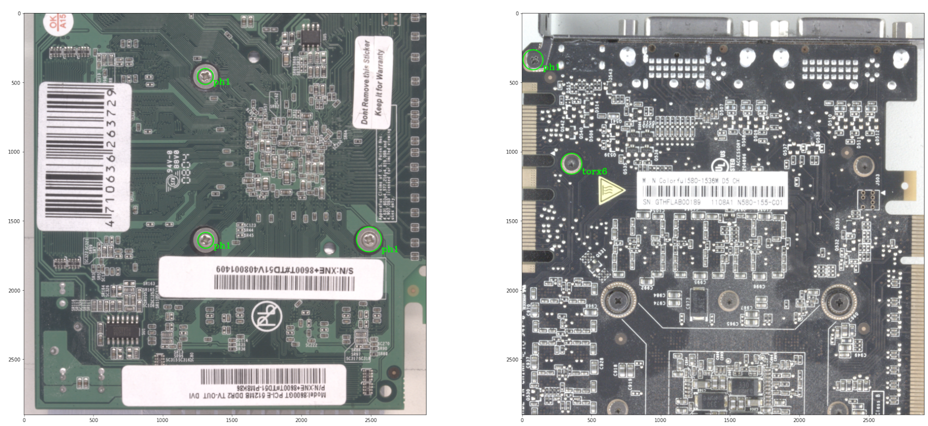


Fig. 10: Correctly detected and classified screws when the learned color is present (left), missed and incorrectly classified screws when the learned color is different (right).

Component segmentation requires specific user annotation and identification of the components of the device, which was only done for HDDs so far. E1 evaluated the segmentation module therefore, on an unseen device of GPU, and found out that the segmentation module reports 0.71 Mean-F score, which a bit lower than 0.78 (the Mean-F score calculated for the original network). Figure 11 illustrates a case where metallic bay partially occludes the PCB, thus the network is misidentifying PCB as bay as well. This is due to fact that the PCBs that were in the HDD image dataset had nothing on them, contrary to the GPU, where it is very likely to be a metallic bay and/or cooling unit over the PCB, and making PCB features less dominant. Similarly, in the same figure, the lower image shows a correctly identified and segmented PCB. Since most of the PCB was visible. The network was able to associate it with the learned PCB features.

The wire detection network was found to be the critical one here. Although it performs remarkably well on the GPU wires (see Table 6, line 1), features that resemble wires are also detected as wires. Since the context information is not there, the wire detector considers non-wire pixels as wires, as illustrated in Figure 12, where cooling pipes of the GPU are considered as wires.

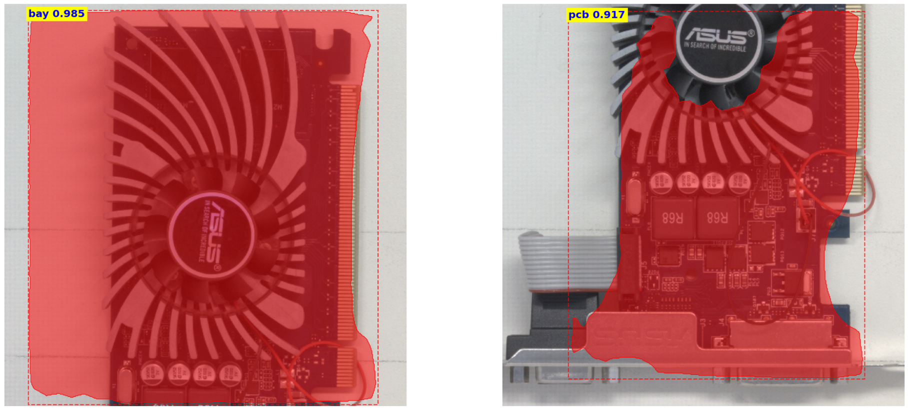


Fig. 11: Predicted mask by the component segmentation performed on a GPU. Left image depicts an incorrectly identified component, whereas the right image depicts a correctly identified one. Portion of PCB pixels play a pivotal role in segmentation of the PCB.

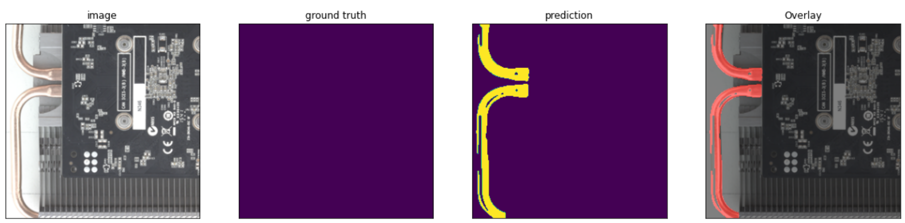
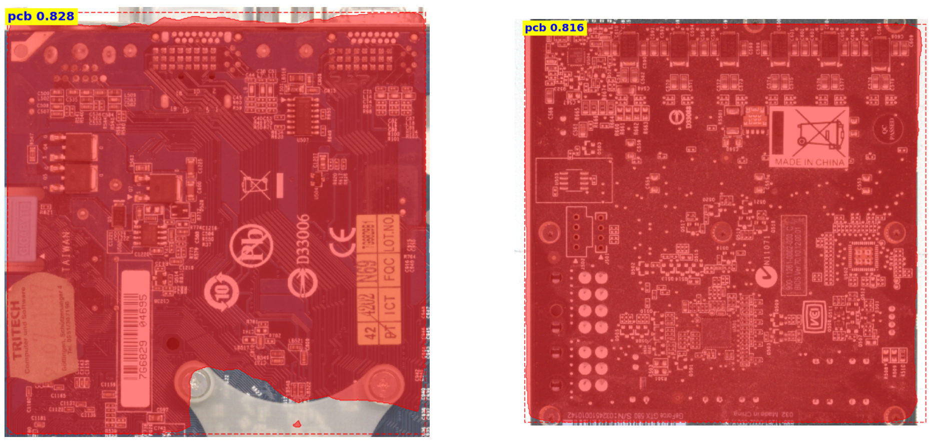


Fig. 12: Predicted mask by the wire detection, incorrectly marking wire-like objects as wires. Metallic pipes are one example of such objects.

Experiment E2 Experiment E2 aims to assess the capabilities of each module by performing visual tasks on raw HDD and GPU images with re-training involved. The training data used is set to 50% of the entire GPU training data (in addition to the existing HDD data). Table 6, line 2 reports the accuracy per module.

For screws, 50 new cropped images of screw heads belonging to GPUs were included in the dataset. The screw detector and screw classifier scores peaked, ensuring an accurate detection and classification. Therefore, it is concluded that 50 new images for screw detection and screw classification networks are sufficient for the GPU. There is a different case for the component segmentation network. This one started to predict meaningful masks for the PCB, as it was trained with extra 50 images of annotated GPU components (bay, PCB). Figure 13a shows a sample output where correctly identified PCB borders are following the correct edges of the PCB component. Note that the network is trying to avoid predicting on the irrelevant or unexpected pixels that correspond to the white plastic attachment found on the PCB. The module reports 0.78 Mean-F score, which is equal to the original 0.78 but higher than the E1 score of it, 0.71. Note that the original network was trained on 600 HDD images. Therefore, newly introduced 50 GPU images do make a difference in terms of generalizing.



(a) Predicted mask by the component segmentation performed on a GPU. The model has been trained with 50% of GPU training data.

(b) Predicted mask by the component segmentation performed on a GPU. The model has been trained with 100% of GPU training data.

Fig. 13: Results of the component segmentation in E3: the model is evaluated on GPU images with partial or complete retraining on the GPU data.

Additionally, it was observed that the wire detection network accuracy changes positively on insignificant levels.

Experiment E3 Experiment E3 differs from the previous experiment (E2) on the amount of new data for re-training. The training data used is set to 100% of the entire training data (in addition to the existing HDD data): 100 new annotated GPU images, plus 1000 screw images (cropped from these GPU images). Table 6, line 3 reports the accuracy per module. After re-training the networks, it was noted that there is a substantial improvement for the component segmentation module, where the network was observably learning the features encountered in GPUs and showing the ability to generalize. Figure 13b depicts an example where the entire PCB was correctly identified (with a prediction score of 0.846) and segmented accordingly.

Similarly to E2, highly accurate screw detection and classification abilities were observed. Features that were learnt enough for the network to capture the screws. It must be remembered that the mentioned data augmentation functions in the screw classification module generates synthetic data out of limited images and fills in the gaps in data. Therefore, it is observed that re-training the screw detection and classification networks with a small number of images is quite possible.

Wire detection network accuracy was almost the same with the previous experiment's, no change observed in behaviors either. This is due to the fact that not all GPU models had wires and thus, the newly introduced GPU images had either no wires, or wires that were very easy to detect as illustrated in Figure 14.

After conducting series of experiments to assess the generalization capability of the scheme, it can be concluded that the scheme generalises the learnt knowledge to an unseen device by acceptable margins. It was found out that visual commonalities (similar features) play a big role in generalization. Experiment E1 proved that PCB components in both GPU and HDD were mostly identified and segmented correctly, and drew the aforementioned conclusion. Some of the incorrect identifications and segments were there due to the fact that the PCBs were occluded by bays. Experiment E2 proved that introducing training data by 50% (on top of the existing data) definitely increases the accuracy of segmentation on PCBs, as illustrated in Figure 13a. Screw related capabilities were remarkably improved even in E2, with less data. Experiment E3 showed that the component segmentation is the module that reacts to the training data most. This was associated with the fact that other modules have plenty of training images, whereas the original image dataset for the component segmentation consisted of around 600 annotated images. Therefore, introducing 100 new images does make a difference for the retraining. The fine prediction of the edge features were noted as shown in Figure 13b, as well as more correct identification of components.

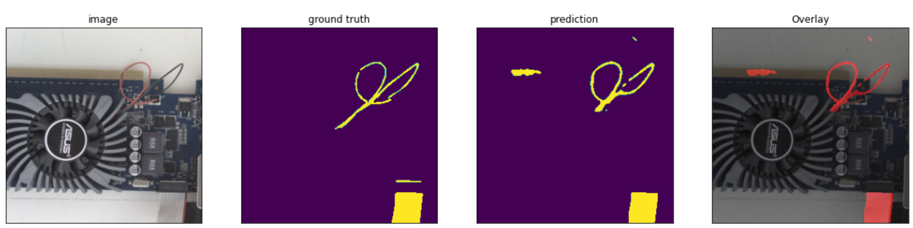


Fig. 14: Wire Detection output during the experiment E3. All wires were correctly identified.

It is acknowledged that the improvement could only be observed for each module as shown in Table 6. Wire detection proved itself to be extremely robust, scoring high in E1, and obviously in E2 and E3. Gap detection had to be skipped for the experiments involving GPUs as there is no gap entity in this EOL device. It must be also noted that not all collected images were able to contain the entire view of the GPU, since the camera lens and the setup height were initially chosen for operating with HDDs. Therefore, the acquired results are the reported predictions on images that partially contain GPUs in their view, as illustrated in the referred figures. While this is not an issue, the optimal scheme would have to operate with a view that contains the chosen EOL device from a reasonable height, proportional to the dimensions of the device.

6 Discussion and Conclusion

In this paper, we presented an extended visual intelligence scheme to analyze a disassembly scene and extract the composition of parts inside a device. We proposed new wire detection and inherited a screw classification method published [48], and, additionally a bookkeeping mechanism that compares the analysis results of consecutive scenes to find out the abnormalities that may be caused by manipulation errors (e.g, end-effector dropping the grasped PCB back into the scene).

We pointed out that the wire detection problem itself is a challenging one, since wires have variable physical properties such as geometry, color, thickness and not every electronic device has the same type of wires. We mentioned that these were the challenging features because of which the previously developed methods were not useful as a general solution to this problem. We proposed a model, which is based on the heavy augmentation and deep convolutional neural networks. The proposed model easily lets the user use the system for any device of his/her choice, as long as the user manages to collect a limited number of hand-annotated images, which we found out to be approximately 130 for accurate detection. We conducted an investigation with the-state-of-the-art models and picked EfficientNetB7 based on the results of a use case we selected for the evaluation of these models. After picking the model, we collected a limited amount of dedicated data using real wire backgrounds that can be found in disassembly environments. We generated a massive amount of data via our datagen block and trained the model from the scratch (e.g., no transfer learning), we could demonstrate that the model achieves high accuracy on scenes with random wires on the GPUs which were not seen by the network before. Our evaluation was quantified with testing images of disassembly scenes, containing different models and sizes of wires. Additionally, we pointed out that wire-looking parts such as pipes are also detected by the wire detector we propose, which we note as a limitation. Increasing the amount of annotated training data with scenes including pipes could potentially help the model.

The screw classification inherited has the default shortcomings mentioned in its original paper [48]. Replacing the initial circle detection method (Hough circle detector) with a more robust circle detector would help. Although using another DCNN is an option for circle detection, increasing the amount of required training data is not always preferable. Therefore, replacing the Hough circle detector with another method based on classical computer vision is advised. Additionally, through our experiment we discovered that color plays a pivotal role in screw classification.

The part recognition is found to be the least directly generalizing component and requires collection, annotation and retraining for other class of devices. This is a foreseeable result as it is the component that learns about the specific components in a device. However, the HDD use case offers a wide range of parts that can also be found in other devices (e.g., PCB, lid, bay), even if they have different appearances. We evaluated the performance of each module of the extended visual scheme on a second device -GPU- and quantified the results, proving that the extended scheme indeed generalizes even without any new training data. We showed the impact of the additional training data and quantified the results by conducting experiments, accordingly. The dataset as well as the implementation are going to be published to facilitate further research.

In conclusion, the extended scheme is designed to complete the required objectives. To our knowledge, it is the first visual intelligence scheme that has the demonstrated capabilities for automated disassembly. Therefore, the novel contribution of this work is promising for recycling plants that are likely to use robotic systems. As of this writing, there is a prototype developed and demonstrated [3] as one of the milestones of the IMAGINE project [4].

Acknowledgments

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 731761, IMAGINE. We would like to also express our gratitude to Maik Bergamos and Electrocycling GmbH [5] for the data and hardware used in this work as well as their expertise on the recycling domain.

References

1. Bailey-Van Kuren, M.: Automated demanufacturing studies in detecting and destroying, threaded connections for processing electronic waste. In: Conference Record 2002 IEEE International Symposium on Electronics and the Environment (Cat. No.02CH37273). pp. 295–298 (5 2002). https://doi.org/10.1109/ISEE.2002.1003283
2. Bdiwi, M., Rashid, A., Putz, M.: Autonomous disassembly of electric vehicle motors based on robot cognition. Proceedings - IEEE International Conference on Robotics and Automation **2016-June**(July), 2500–2505 (2016). https://doi.org/10.1109/ICRA.2016.7487404
3. Bisong, E.: Google colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform, pp. 59–64. Springer (2019)

[3] https://www.youtube.com/watch?v=m8aEZnSdiCA

[4] By the time this demonsration took place, the extended modules were not included yet. However, the final prototype does include wire detection and screw classification modules.

[5] http://www.electrocycling.de

4. Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A.: Albumentations: fast and flexible image augmentations. Information **11**(2), 125 (2020)
5. Dai, J., He, K., Sun, J.: Instance-aware semantic segmentation via multi-task network cascades. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3150–3158 (2016)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
7. Dutta, A., Zisserman, A.: The VIA annotation software for images, audio and video. In: Proceedings of the 27th ACM International Conference on Multimedia. MM '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3343031.3350535, https://doi.org/10.1145/3343031.3350535
8. Dvornik, N., Shmelkov, K., Mairal, J., Schmid, C.: Blitznet: A real-time deep network for scene understanding. In: Proceedings of the IEEE international conference on computer vision. pp. 4154–4162 (2017)
9. ElSayed, A., Kongar, E., Gupta, S.M., Sobh, T.: A robotic-driven disassembly sequence generator for end-of-life electronic products. Journal of Intelligent & Robotic Systems **68**(1), 43–52 (2012)
10. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International Journal of Computer Vision **88**(2), 303–338 (06 2010). https://doi.org/10.1007/s11263-009-0275-4
11. Fu, C.Y., Shvets, M., Berg, A.C.: Retinamask: Learning to predict masks improves state-of-the-art single-shot detection for free. arXiv preprint arXiv:1901.03353 (2019)
12. Gil, P., Pomares, J., Diaz, S.v.P.C., Candelas, F., Torres, F.: Flexible multi-sensorial system for automatic disassembly using cooperative robots. International Journal of Computer Integrated Manufacturing **20**(8), 757–772 (2007)
13. Gil, P., Torres, F., Ortiz, F., Reinoso, O.: Detection of partial occlusions of assembled components to simplify the disassembly tasks. The International Journal of Advanced Manufacturing Technology **30**(5-6), 530–539 (2006)
14. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
15. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
16. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
17. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781 (2019)
18. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
19. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2016)
20. Jahanian, A., Le, Q.H., Youcef-Toumi, K., Tsetserukou, D.: See the e-waste! training visual intelligence to see dense circuit boards for recycling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2019)

21. Kasturi, R., Camps, O., Huang, Y., Narasimhamurthy, A., Pande, N.: Wire detection algorithms for navigation. NASA Technical report (2002)
22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
23. Li, Y., Qi, H., Dai, J., Ji, X., Wei, Y.: Fully convolutional instance-aware semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2359–2367 (2017)
24. Lin, D., Shen, D., Shen, S., Ji, Y., Lischinski, D., Cohen-Or, D., Huang, H.: Zigzagnet: Fusing top-down and bottom-up context for object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7490–7499 (2019)
25. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2117–2125 (2017)
26. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
27. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8759–8768 (2018)
28. Madaan, R., Maturana, D., Scherer, S.: Wire Detection using Synthetic Data and Dilated Convolutional Networks for Unmanned Aerial Vehicles. International Conference on Intelligent Robots and Systems (2017)
29. Malamas, E.N., Petrakis, E.G., Zervakis, M., Petit, L., Legat, J.D.: A survey on industrial vision systems, applications and tools. Image and vision computing **21**(2), 171–188 (2003)
30. Pomares, J., Puente, S., Torres, F., Candelas, F., Gil, P.: Virtual disassembly of products based on geometric models. Computers in Industry **55**(1), 1–14 (2004)
31. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
32. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation (2015)
33. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-resnet and the impact of residual connections on learning (2016)
34. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision (2015)
35. Takahashi, R., Matsubara, T., Uehara, K.: Data augmentation using random image cropping and patching for deep cnns. IEEE Transactions on Circuits and Systems for Video Technology (2019)
36. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks (2019)
37. Tonko, M., Nagel, H.H.: Model-based stereo-tracking of non-polyhedral objects for automatic disassembly experiments. International Journal of Computer Vision **37**(1), 99–118 (2000)
38. Torres, F., Puente, S., Aracil, R.: Disassembly planning based on precedence relations among assemblies. International Journal of Advanced Manufacturing Technology **21** (02 2003). https://doi.org/10.1007/s001700300037
39. Ukida, H.: Visual defect inspection of rotating screw heads. In: SICE Annual Conference 2007. pp. 1478–1483. IEEE (2007)

40. United Nations Envrionment Programme, P., ITU, I., UNIDO, U.N.U.: A new circular vision for electronics time for a global reboot (2019)
41. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
42. Weigl-Seitz, A., Hohm, K., Seitz, M., Tolle, H.: On strategies and solutions for automated disassembly of electronic devices. International Journal of Advanced Manufacturing Technology **30**(5-6), 561–573 (2006). https://doi.org/10.1007/s00170-005-0043-8
43. Xie, S.Q., Cheng, D., Wong, S., Haemmerle, E.: Three-dimensional object recognition system for enhancing the intelligence of a KUKA robot. International Journal of Advanced Manufacturing Technology **38**(7-8), 822–839 (2008). https://doi.org/10.1007/s00170-007-1112-y
44. Yakubovskiy, P.: Segmentation models. `https://github.com/qubvel/segmentation_models` (2019)
45. Yang, J., Bertram, J., Schettgen, T., Heitland, P., Fischer, D., Seidu, F., Felten, M., Kraus, T., Fobil, J.N., Kaifie, A.: Arsenic burden in e-waste recycling workers–a cross-sectional study at the agbogbloshie e-waste recycling site, ghana. Chemosphere p. 127712 (2020)
46. Yildiz., E., Brinker., T., Renaudo., E., Hollenstein., J.J., Haller-Seeber., S., Piater., J., Wörgötter., F.: A visual intelligence scheme for hard drive disassembly in automated recycling routines. In: Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems - Volume 1: ROBOVIS,. pp. 17–27. INSTICC, SciTePress (2020). https://doi.org/10.5220/0010016000170027
47. Yildiz, E., Wörgötter, F.: 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 (2019)
48. Yildiz., E., Wörgötter., F.: Dcnn-based screw classification in automated disassembly processes. In: Proceedings of the International Conference on Robotics, Computer Vision and Intelligent Systems - Volume 1: ROBOVIS,. pp. 61–68. INSTICC, SciTePress (2020). https://doi.org/10.5220/0009979900610068
49. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)