Abstract—The dramatic increase of electronic waste requires automatic recycling, including technologies from machine vision. A framework for segmentation and classification of THC (through-hole components) mounted on PCBAs is presented, using both RGB and depth frames from the Kinect sensor by Microsoft. A segmentation approach, combining local and global features in a flexible manner, is shown to optimize a freely definable cost function globally. We interleave segmentation and classification as we form the final components using a simple, yet robust shape model.

I. INTRODUCTION

The worldwide amount of electronic waste has been increasing dramatically since the past 20 years. Reasons for that fact are the rapid technological change and progress, resulting in ever-shorter lifespans of electronic devices ([1]). Two factors make the recycling of e-waste an important issue. Firstly, electronic devices contain a variety of materials many of which are ecologically critical. Secondly, precious metals can be found in electronic waste. The European “Waste Electrical and Electronic Equipment” (WEEE) Directive is facing the e-waste management. However, the complexity of the contained materials still poses a challenge for an appropriate recycling. Fig. 1(a) shows a draft of a recycling system for PCBAs (printed circuit board assembly). Other parts of electronic devices are not considered here. An aspired key component of such a system is a comprehensive analysis of PCBAs. Using the retrieved information, the recycling process should be optimized for the present PCBAs and the mounted components. The goal of this paper is to make a contribution to the analysis procedure.

To further confine the topic of the work at hand, we take a look at Fig. 1(b). The analysis step consists in retrieving information from sensor data, aiming at the recognition of components mounted on a PCBA. The sensor data we use are gathered employing the Kinect sensor by Microsoft ([2]). The Kinect sensor is an attractive alternative to conventional RGB cameras because it additionally offers depth information. Using structured illumination in the infrared range, the camera-to-object distance is determined. The price of the Kinect sensor is relatively low whereas the resolution of the RGB image and of the depth map as well as the imaging quality are comparatively poor for our purposes. However, we want to explore the limits of a camera and a depth sensor, respectively, from the lower price class. To get maximum benefit from the depth information, we restrict ourselves to the recognition of THC (through-hole components) that are, in comparison to SMDs (surface-mount devices), typically more elevated. Before any components can be classified, they have to be segmented at first. In fact, this task is the main topic in the following (see section II). Fig. 2 exemplarily shows some components we would like to segment.

It should be noticed that we avoid using component-specific information for segmentation. Certainly some components consist of several constituents that may strongly differ from each other visually. In this case, the component can not be segmented as a whole until the classification phase. To conjoin all of its constituents, a simple shape model is applied after each constituent has been classified (see section III).

In section IV a few results of the performed experiments are presented from which some conclusions are drawn in section V.

II. HIERARCHICAL SEGMENTATION

In this section we present a flexible framework for the segmentation of image content that is salient in contrast to the background and how to apply it to our problem, i.e. segmentation of components mounted on a PCBA. To accomplish that, several segmentation hypotheses are created...
in a multi-threshold approach first. Those hypotheses can be hierarchically arranged in a tree-like structure. In a second step we search for the best segmentation hypothesis. For this optimization step a proper cost function has to be defined. It will become apparent that in this function the depth information can be integrated easily. More importantly, it will turn out that the tree structure, resulting from the multi-threshold approach, allows us to find a globally optimal segmentation solution regarding the previously defined cost function.

One further advantage of our procedure is that it works fully automatically without any seed points or initial contours as required for example by some graph cuts-based algorithms ([3],[4]). [5] follows a similar approach like ours by resegmenting superpixels (see below). However, it does not exploit the color information thoroughly. It is not fully automatic and does not necessarily result in a global optimal solution.

A. Multi-threshold approach

1) Preprocessing: Unfortunately the quality of the depth information given by the Kinect sensor is not sufficient to directly serve as a base for the segmentation. The depth information is used to estimate the PCB colors though. For this purpose, the plane which contains the PCB is estimated first, applying the RANSAC-algorithm ([6]). The - according to the depth map and the given PCB plane - non-elevated regions are assumed to belong to the PCB and not to components.

Then we perform a k-means clustering (k equals 2) ([7]) in the CIELUV color space ([8]) on the respective regions in the color image, i.e. those regions that probably belong to the PCB. Any clustering or calculations of metrics are performed in the CIELUV color space because it is approximatively perceptually uniform.

So after those preprocessing steps there are firstly the elevation of PCBA regions with respect to the PCB and secondly the two estimated modes of the PCB color available. However, this information is not reliable enough to serve as a starting basis for segmentation directly. Hence, a color-based approach is presented in the following. To reduce the computational complexity of our algorithm and to smooth artifacts in the color image superpixels are created in advance. Superpixels are groups of neighboring image pixels that are homogeneous, in our case with respect to color. Superpixels are gained by applying the mean shift algorithm ([9]) in the CIELUV color space. The result is a tessellate color image. Now we will see how to perform multi-threshold segmentation based on superpixels.

2) Building segmentation hypotheses: The basis observation justifying a multi-threshold approach is that the vast majority of components sets itself apart from the surrounding area in terms of color. Therefore they can be segmented using a proper value in a color space as a threshold. However, the needed threshold is neither fixed nor known a priori. We give various thresholds a try, each yielding one segmentation hypothesis for the respective component.

To examine the creation of a segmentation hypothesis in detail, at first we assume that one single threshold is given. Fig. 3 illustrates the merging procedure with this threshold resulting in a segmentation hypothesis. A red component on a green PCB is sketched. For some reason, maybe inhomogeneous illumination, imaging artifacts or simply component properties, different shades of red can be observed. That is why mean shift yields not one single superpixel for the component in this example. Now we can represent the superpixels as nodes in a graph and apply the actual merging: We connect each pair of nodes by an edge whose respective superpixels are firstly neighboring in the image plane and secondly exhibit a color difference below the given threshold. We define the color difference as the Euclidean distance in the CIELUV color space. Eventually, the transitive closure of the graph is built. That is, edges are added between nodes that are already connected indirectly i.e. via further nodes. The final hypothetical component is then simply the set of connected superpixels. In the following a set of superpixels that has been merged in this manner is called a segment.

We now turn to the multi-threshold case. For the small example of Fig. 4 let us assume there are six superpixels given by mean shift and three levels. The first level belongs to a threshold equals zero, the second level to a moderate threshold and the third level to the highest one. On each level, for all superpixels the merging procedure with the respective threshold as described above for the single-threshold case
is performed. The higher you ascend in the hierarchy the more superpixels are merged. On level one the segments are the primal superpixels because the threshold equals zero, i.e. effectively no merging has been applied. Then each segment - given by the merging - is represented as a node in a tree on the level belonging to the respective threshold. We define a "mother node" on level $l$ and a "child node" on level $l-1$ as a pair of two nodes where the superpixels of the child node are included in the mother node as well. A tree-like structure as in Fig. 4 is obtained if each mother node is connected to its child node by an arrow.

As said before, each node corresponds to a hypothesis representing the desired segmentation result of a component. It is important to notice that those single hypotheses are not independent for pairs of mother and child nodes. For example if the mother node is assumed to correspond to an actual component, the child segment obviously can not be a component as well because it is part of its mother segment. To model such dependencies, we define three categories of nodes:

- $C$, a single component;
- $E$, constituent element of a component;
- $M$, remaining nodes ("mixed", i.e. combination of components and/or PCB).

Before searching for the best overall hypothesis i.e. the optimal total segmentation, it has to be pointed out that for a valid overall hypothesis there are four conditions to be met:

- children of $C$ nodes are classified as $E$,
- children of $E$ nodes are classified as $E$,
- no child of any $M$ node is allowed to be $E$,
- there is no $E$ node on the top level.

The assignment in Fig. 5 is an example of a valid overall segmentation hypothesis, in other words a valid assignment of the classes $C, E$ and $M$ to the nodes in the created tree-like structure. In the next section we deal with the problem of choosing the best out of all valid overall hypotheses.

**B. Optimization**

1) **Cost function**: To get an optimal overall segmentation result we firstly define a cost function in order to assess a given hypothesis. Let us pick up again the example of the previous section (Fig. 5). Two kinds of nodes contributing to the cost function are proposed:

- all nodes classified as $C$, constituting the set $\mathcal{C}$;
- all nodes classified as $M$ on the lowest level, constituting the set $\mathcal{M}^1$.

The consideration of the first group is obviously plausible, as we want to model the quality of the component segments. However, it is not adequate to take only the segments of components into account. We rather claim that the superpixels that are classified as $M$ on the lowest level and hence do not belong to any component, should match the assumption of being part of the PCB as well. Thereby we write for the cost of an overall segmentation hypothesis:

$$J_{\text{total}} = \sum_{i \in \mathcal{C}} A(i) \cdot J_c(\text{node}_i) + \sum_{i \in \mathcal{M}^1} A(i) \cdot J_m^1(\text{node}_i), \tag{1}$$

where $i$ denotes the node index and $A(i)$ the area of the corresponding node. Weighting the cost functions with the area of the corresponding segment is considered to be necessary. $J_c(\text{node}_i)$ models the cost of asserting that the segment belonging to $\text{node}_i$ is a component and $J_m^1$ the cost that the corresponding superpixel belongs to the PCB. In our experiments, $J_c$ included the shape of the segment (mainly the rectangularity) and the saliency in terms of color. Given a hierarchy as described above, a measure of saliency can be found easily. Segments that strongly set themselves apart from the surrounding area, remain unchanged through many levels or thresholds respectively. Hence, by subtracting the lowest threshold from the highest one which yield a certain segment, a proper criterion is established. $J_m^1$ takes into account the difference between the superpixel color and the predicted PCB color as well as the deviation from the estimated PCB plane regarding the depth information. $J_c$ assesses the corresponding segment as a whole, i.e. in a global manner, whereas $J_m^1$ considers local features. The cost functions $J_c$ and $J_m^1$ can be chosen freely and can flexibly combine local and global features as described above.

2) **Finding the optimal overall hypothesis**: Regardless of the definitions of $J_c$ and $J_m^1$, we examine the optimization of equation (1) in the following, i.e. we want to optimize the segmentation result globally. A hierarchical structure similar to Fig. 5 is taken as a given, but it is assumed that there is only one root node. This is admissible because if there are more than one top node the corresponding trees can be optimized.
separately.

At first, we introduce the definition of a more general function \( J_m \) for all \( M \)-nodes by using the function \( J_m \) that is given only for nodes on the lowest level. Let \( CH_i \) be the set of the child nodes of \( node_i \), then we write:

\[
J_m (node_i) = \begin{cases} J_m^1 (node_i), & \text{if } node_i \text{ on level } = 1, \\ J_m^{>1} (node_i) = \sum_{j \in CH_i} J (node_j), & \text{if } node_i \text{ on level } > 1, \\ \end{cases}
\]

with

\[
J (node_i) = \begin{cases} J_m (node_i), & \text{if } node_i \text{ classified as } M, \\ J_c (node_i), & \text{if } node_i \text{ classified as } C. \\ \end{cases}
\]

It should be kept in mind that the cost functions \( J_m \) and \( J_c \) are taken as a given, as we can calculate \( J_m \) for any superpixel and \( J_c \) for any segment directly. The previous definition is useful because it allows us to rewrite the total cost \( J_{\text{total}} \) of equation (1). Taking a careful look at the equations (2) and the tree-like structure in Fig. 5, it becomes apparent that the value of any node \( J_m \) or \( J_c \) respectively is handed over to the corresponding mother node. As a result, the root node at the top accumulates the total cost of its tree:

\[
J (node_{\text{root}}) = J_{\text{total}}.
\]

So far we have assumed a valid hypothesis to be given and we have expressed its total cost. Since we know that the total cost equals the cost of the root node \( J(nod_{\text{e root}}) \), we seek to minimize \( J(nod_{\text{e root}}) \) from now on. So, denoting a valid overall hypothesis by \( H \) and the one with minimal total cost by \( H^* \), the following equivalence is used:

\[
H^* = \arg\min_{H} (J_{\text{total}}),
\]

\[
H^* = \arg\min_{H} (J (node_{\text{root}})).
\]

Suppose node \( i \) is classified as \( M \) and regard it as a root node. Then the assignment of \( M, C \) and \( E \) for those nodes which belong to the subtree of node \( i \) is denoted by \( H_i \). If you optimize over \( H_i \) you get the minimal cost \( J^*_m (node_i) \) at this node. This cost value only takes lower nodes into account and can be calculated recursively for all nodes bottom-up. It is essential to bear in mind that \( J^*_m \) is only relevant if a node is classified as \( M \) because if a node is classified as \( C \), it simply carries the cost \( J_c \) without considering its children and children’s children. Formally this optimization, similar to dynamic programming, is expressed as:

\[
J^*_m (node_i) = \begin{cases} J^*_m (node_i), & \text{if } node_i \text{ on level } = 1, \\ J^*_m^{>1} (node_i), & \text{if } node_i \text{ on level } > 1, \\ \end{cases}
\]

To get the final segmentation result we run through the tree top-down and decide whether to classify nodes as \( C \) or \( M \). We start at the top node because its cost is the one we want to optimize. We classify the top node as \( C \), if and only if \( J_c (node_{\text{root}}) < J^*_m (node_{\text{root}}) \) in order to achieve the minimal \( J^* (node_{\text{root}}) \). In this case, all lower nodes are necessarily classified as \( E \). If the top node is classified as \( M \), its optimal cost consists of the minimal cost of its children. It is crucial that the children of any node in a tree establish stand-alone trees that can be optimized independently. Hence if the top node is classified as \( M \), all of its children are regarded as new root nodes of their corresponding subtrees, so that they can be treated as the top node was previously. This procedure is continued level by level until ending up on level one, having classified any node either as \( M, C \) or \( E \). The segments corresponding to nodes classified as \( C \) can be further processed in the component classification.

In our experiments some further refinements were adopted. In a few cases the procedure from above yielded fragmented segmentation results. Therefore, large segments were preferred by introducing a penalization for classifying small segments as components.

### III. Classification

As figured out before, some components require class-specific knowledge to be segmented successfully as a whole. A method used to integrate this knowledge in segmentation is a so called shape model which describes the spatial dependencies among the constituent parts of the object to be segmented ([10]). A drawback of this method is that a way has to be found to define representative constituent parts for each class. In this work, each component class is divided into ”centered” and ”non-centered” constituent parts only (Fig. 6).

During training, components are labeled as a whole containing all of its constituent parts. That means, you can easily check whether the centroid of the component as a whole lies within a certain segment or not. In the first case, we call the corresponding constituent part centered and in the second one non-centered. The classifier is trained for each class with examples of centered and non-centered segments separately. So in classification, the classifier should not only be able to determine the class a segment belongs to but also whether it is centered or non-centered. Having this information at hand for each segment gained from segmentation, we get conjoined components applying the following algorithm:
Fig. 6. Illustration of “centered” and “non-centered” segments. These definitions lead to a simple shape model.

- **Step 1**: Each centered segment initializes a component of the respective class.
- **Step 2**: Attach non-centered segments to proximate components of the same class. If there is a collision, in the sense that a segment could be attached to more than one component, choose the component with the nearest centered segment.
- **Repeat step 2** until there are no more non-centered segments left that can be attached to components.

The shape model we took as a basis here simply states that there is exactly one centered constituent part for each component. Some classifier return a probabilistic output instead of a hard decision, for example by modelling decision boundaries by means of probability density functions. Using such a classifier, we succeeded to improve the classification result by taking context information into account. That was accomplished by increasing the probability of a non-centered segment belonging to a certain class if a centered segment of the same class is around and vice versa.

First experiments with support vector machines, decision trees and a modified nearest neighbour rule were performed. Each of the three classifiers gave similar results. Admittedly relatively simple features were used, such as the quality of a fitted circle and rectangle respectively, the side lengths of the fitted rectangle, the mean of the depth values and the color difference to grey weighted by a Gaussian function. The latter feature indicates whether a component may be metallic.

### IV. Experiments

The segmentation was insensitive to variations of illumination and resolution. For our experiments a resolution of about $10 \text{ pixels/cm}$ and a PCBA-to-camera distance of 90 cm were used. If this distance exceeds a value of about 110 cm, the quality of the depth map and the resulting segmentation decreases significantly. The minimal superpixel size of mean shift was tuned to about 30, the range bandwidth to $h_r = 6$ and the spatial bandwidth to $h_s = 10$. The values of the minimal superpixel size and the spatial bandwidth were adapted proportionately to the imaging resolution. For the RANSAC procedure the value 10 was chosen for the threshold parameter to distinguish an outlier from points within the estimated plane. As far as building the described hierarchy for segmentation is concerned, the stepping of proper threshold values is not very critical. We propose to use values from 0 to about 18 with a step size of 1 or 2. If necessary, the step size in the range from 5 to 10 could be decreased down to 0.5 or even 0.25.

We used 21 PCBAs for our experiments, 9 of them being mainboards. See Fig. 7 for an example of segmentation results. To evaluate the segmentation results, the following frequently encountered component classes were defined:

1. cooler, fan;
2. DDR slot;
3. PCI slot;
4. further slot (PCI Express, AGP, CNR, AMR);
5. ATX socket, connector;
6. back panel socket (LAN, phone, USB, PS/2);
7. battery (button cell).

The segmentation results are shown in table I. As can be seen in table II, the main problem of the classification currently is to distinguish between segments belonging to the PCB and those who are components.

Fig. 7 gives an impression of the shape model’s capability.

### V. Conclusion

The fact that automatic recycling for e-waste is urgently needed, gave us stimuli for the development of a segmentation and classification framework for PCBAs. Though a RGB camera and a depth sensor with quite limited imaging capability
were used, namely the Kinect, satisfying results were obtained. Thus, further research with sensors in the lower price sector is justified. The major problem still remaining is the great number of PCB segments that are misclassified as components. More sophisticated features are to be considered for solving this problem.

The main contribution of this paper is a color-based segmentation approach that can also be used for other applications than segmentation of components on a PCB. However, our approach is particularly suitable for flexibly combining foreground and background information. Moreover, one can easily adapt the cost functions and introduce both local and global features. The presented shape model as such is robust. However, there are few cases in which non-centered segments are classified as centered ones and vice versa so that some constituent parts of a component may not be conjoined successfully. This was often observed for stickers. These stickers aren’t always placed equally on components. Therefore, they can occur both as centered and as non-centered segments.

**TABLE I**

<table>
<thead>
<tr>
<th>class</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>successfully segmented</td>
<td>16</td>
<td>20</td>
<td>34</td>
<td>17</td>
<td>31</td>
<td>33</td>
<td>5</td>
</tr>
<tr>
<td>totally available</td>
<td>18</td>
<td>25</td>
<td>34</td>
<td>24</td>
<td>50</td>
<td>40</td>
<td>9</td>
</tr>
<tr>
<td>ratio S</td>
<td>89%</td>
<td>80%</td>
<td>100%</td>
<td>71%</td>
<td>62%</td>
<td>83%</td>
<td>50%</td>
</tr>
</tbody>
</table>

**TABLE II**

<table>
<thead>
<tr>
<th>confusion matrix of classification using a decision tree.</th>
</tr>
</thead>
<tbody>
<tr>
<td>predicted class</td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>successfully segmented</td>
</tr>
<tr>
<td>totally available</td>
</tr>
<tr>
<td>ratio S</td>
</tr>
</tbody>
</table>

**REFERENCES**