Self-learning Pipeline for 3D Object Detection, Acquisition and Recognition in Autonomous Systems

Researcher:
Dimitrij-Marian Holm
University of Applied Science Munich
Departement of Electrical Engineering and Information Technology
Lothstraße 64, 80335 Munich
Email: d.holm@hm.edu

Supervisor:
Prof. Dr. Alfred Schöttl
University of Applied Science Munich
Departement of Electrical Engineering and Information Technology
Lothstraße 64, 80335 Munich
Email: alfred.schoettl@hm.edu

CONTENTS

I. NOTATION

$\Delta R \in \mathbb{N}^+$: Resolution Steps, Size to quantize
$k \in \mathbb{N}^+ | k \leq \Delta R$: current resolution step
$n_{max} \in \mathbb{N}^+$: maximal required poses
$n \in \mathbb{N}^+ | n \leq n_{max}$: current pose step
$i \in \mathbb{N}^+$: maximal points in point cloud
$j \in \mathbb{N}^+ | j \leq i$: current point in point cloud
pc : point cloud with included points, $pc \in \mathbb{R}^{i,\times n,3}$
$p \in \mathbb{R}^3$: point within the corresponding pc, $p_{j,k,n} \in pc_{k,n}$

II. Introduction to the self-learning pipeline (SLP)

One of the key functions of an autonomous system is the interaction with the environment. In view of the possibility for collaborating with other systems or humans like service robots are doing it is obvious that the system needs an understanding of the environment. This means that environmental information, especially spatial information, which are often represented by point clouds provided by sensors have to be analyzed and classified. This allows autonomous systems to learn more about the surrounding. One Example of a point cloud is shown in figure 1, p. 3. Analyzing and classifying is in general called object recognition. For this two definition are necessary. The first one is called objects. PC contains different objects but they can exist several times (e.g. two cups, five pencils, ...). objects belongs to classes, objects (for example two different cups) are instances of a class (different cups are still cups). We can separate object recognition into two research fields.

The first field is called supervised object recognition. It uses a data set of point cloud data and associated labels [1] to provide information and try to find these classes in a given PC which represents the current environment. Despite the long time this field has been studied [2] it has one big disadvantage. The recognition and the creation task are completely separated, the recognizable object data must be known a priori before the object recognition is possible. This leads to external interactions like 3D-modelling of objects which is not desired for an autonomous system. The target is to combine the creation and recognition into a single algorithm. This is provided by the unsupervised object recognition.

But this requires a new task called detection. This task finds unknown objects in the current environment and is the base for following procedures. Based on this two research fields the paper presents a complete pipeline combining the tasks of detecting unknown objects in the environment, creating a sustainable geometric model what can be used for object recognition solutions. The focus is not to improve or assemble algorithms for recognition or the detection task. But is is a new approach to connect existing solutions to a complete pipeline called SLP (Self-learning pipeline), including all steps from the raw point cloud to a 3D-mesh usable for object recognition or other purposes in which 3D-models are needed. For most of the pipeline parts we used algorithm from the Point Cloud Library (PCL, [3]) which is developed and supported from dozens of organizations to provide state-of-the-art algorithms for point cloud processing ([4]) to connect the necessary computation steps. But for the point cloud registration part we developed a complete new approach to force the challenges occurring (see IV-C1, p. 10) after data acquisition.

III. Related Work

A lot of approaches exist for the task of detecting known objects in a PC. Most 3D-based approaches uses local features in small patches arround keypoints to represent the underlying 3D surface to find similarities between the given 3D model and the input data (PC) to identify objects and calculate pose [2], [5]. The main difference between approaches based on local features are the representation of the information. In [6], a 3D tensor descriptor is generated by creating a local 3D grid over the PC. Other approaches transform the shape of a 3D model into a probability distribution [7] or calculating an Feature Histogram based on the Viewpoint [8]. Exploiting geometrical relationships like distance

between detected 3D keypoints are used in [9].

On the other hand unsupervised object detection is a field of research where in comparison to object recognition relatively little work has been done [10] so far. Nevertheless a lot of remarkable approaches were proposed. In [11], every PC based on a scene is segmented and weighted by a saliency approach to extract objects. Unknown objects which fit several conditions are linked with locally-consistent labels. To find these labeled objects in other scenes a class graph is created which enables mapping local labels in different scenes to global labels which are created by affinity propagation. The condition that for object detection at least two instances of an class needed to appear in a scene and several objects must be discovered before the object recognition task can work constrains this approach. Based on the Latent Dirichlet Allocation introduced in [12] the PC is split into local segments extracting the local features by spin-images techniques. With this features several Dirichlet distributions are calculated. This allows the algorithm to discover new objects and registry already discovered objects. The approach similar to other algorithms based on unsupervised techniques are not able to automatically determine the number of classes. Regardless of the research done at this two fields it seems there is a consequent distinguish between them. While the first field uses complete a-prior created 3D-models to be as flexible as possible in recognition, the second suppress the benefit of the high degree of freedom but let the system gather information by itself. To combine the general meaning of this areas and profit from the advantages it leads us to develop the SLP.

IV. SLP - Theory

Based on the different fields and the resulting research as mentioned in III, page 1 our approach unites the separated view to a full system to achieve the skills an autonomous system needs to understand its surrounding environment. We assume that the operation field of the considered autonomous systems allows the manipulation of the environment allowing to change the pose of objects (fig. 3, p. 5). This enables a much wider range of possibilities (see IV-C, p. 10). We introduce a dynamically growing object database which overcomes the limitation of a fixed number of recognizable object classes. We assume that the recognized as well as the unrecognized objects are medium sized household items like bottles, to be located on a plane underground and have a suitable distance from each other to prevent wrong segmentation. Figure 2, page 4 depicts the steps as well the data flow of the whole pipeline. As following the structure of the pipeline the report is split into two subcategories, Detection and Acquisition.

A. Detection

As the input for the whole pipeline is a raw point cloud (PC) the SLP - Detection is used for filtering the raw PC, gathering necessary information about the environment, extracting possible objects and provide the possibility to track resulting parts of the PC.

1) Point Cloud:

Input of the pipeline is the raw point cloud as shown in figure 4 on page 5. In our approach the pixels of a image are expanded with spatial information x, y, z, relative to the sensor device, leading to a point cloud. The spatial information is used for the next steps, including cloud processing, data storage

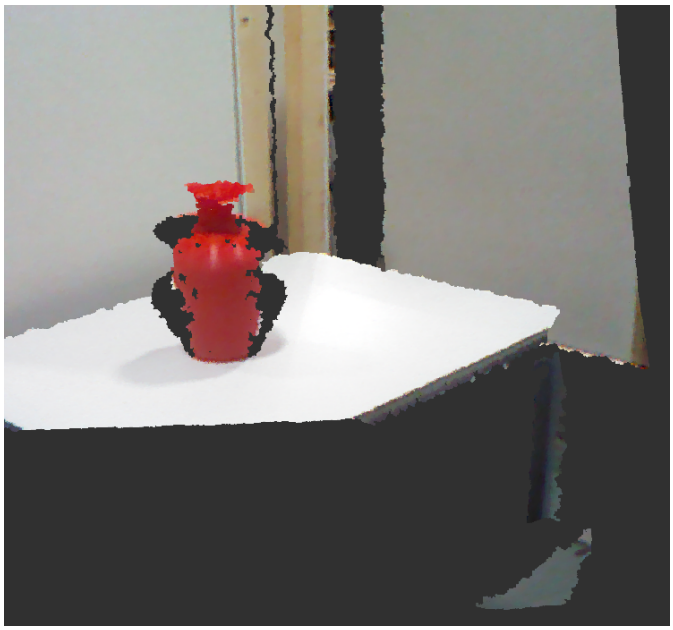


Fig. 1: Snapshot of a point cloud as used for the pipeline. This shows the current environment of the investigated approach.

and object generation. Figure 5 on page 6 shows the underlying coordinate system and the three views (front, top, side) which are occurring.

2) Preprocessing:

This task removes outliers, downsizes, smooths and removes planes like walls and big tables from the incoming raw PC to obtain segmentable data as shown in figure 6, p. 3.

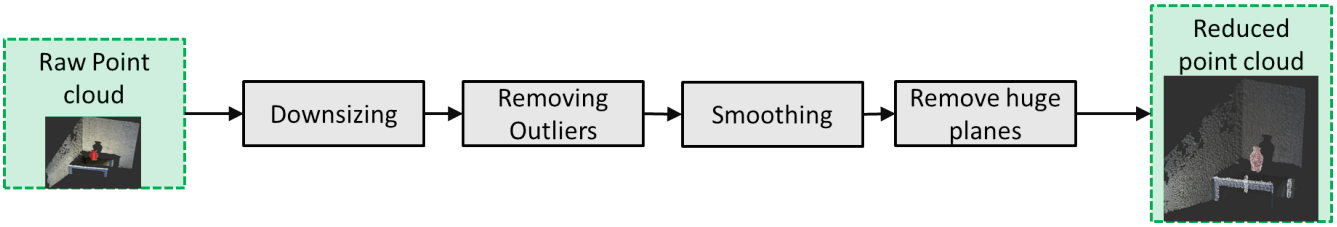


Fig. 6: The pipeline for the preprocessing. Input is the raw PC. After several operations we obtain a usable data for the next tasks.

The first step is to downsize the PC. As the raw PC may contain a lot of information it is necessary to reduce this amount of point in order to decrease the calculation time for the autonomous system. The point cloud is separated into small voxels (small 3D-boxes). After that the centroid of all points within this box is calculated and will replace the original points inside the box. To decrease the noise inside the measured points a statistical outlier-removal is done. It calculates the mean distances of all

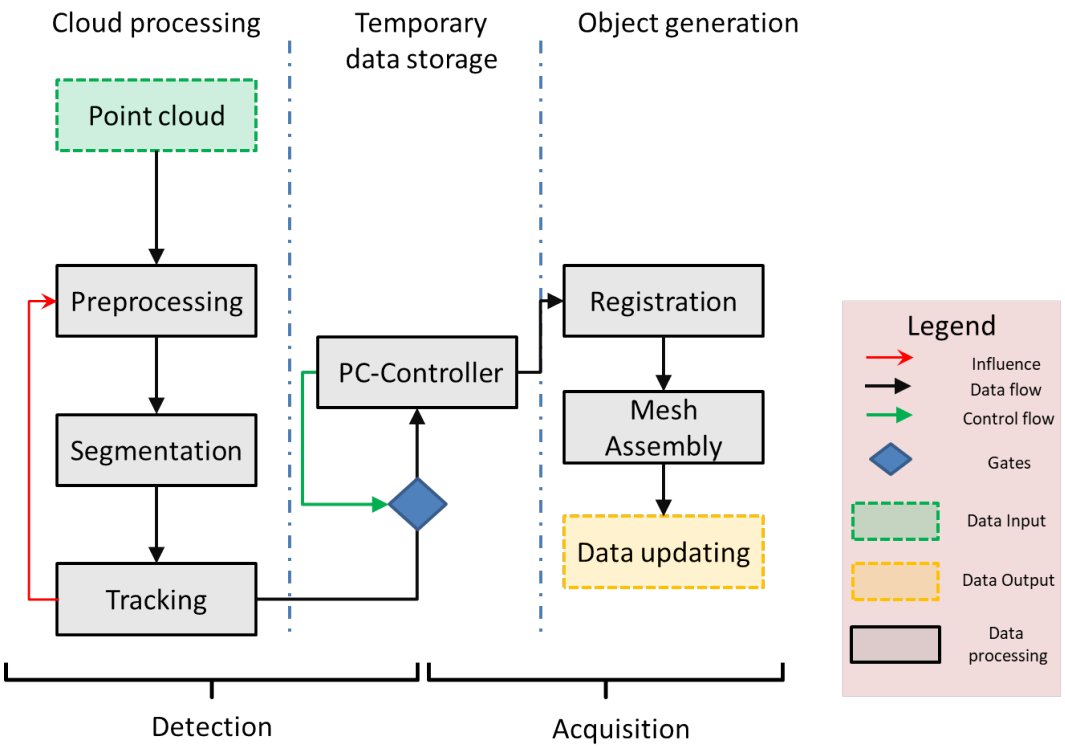


Fig. 2: The general overview of the Pipeline. As the Input a raw point cloud is used. The PC passes the Preprocessing, Segmentation and Tracking Tasks. If the Data-Controller open the Data Gate a snapshot of the current processed PC is stored. After a certain amount of PCs is available in the storage of the Data-Controller the PCs are transmitted to the object generation step. This includes PC-registration and mesh creating. Finally the result is uploaded in the database of the object recognition algorithm

neighbours for all given points in the PC. With the assumtion the distance distribution is Gaussian, all points whose mean undercut the standart deviation are removed from the dataset. The algorithm is closer elucidated in [13]. Smoothing the surfaces enhances the efficiency subsequent task, huge plane removal. An effective method smoothing surfaces is using the Moving Least Square (MLS) algorithm as explained detailed in [14]. To distinguish between objects of interest and insignificant parts like walls, tabletops or other big objects we use the SACSegmentation algorithm which is based on RANSAC. It seeks for one plane structure in the PC data and removes the inherent points from the PC data set. This is repeated until the containing amount of points located in the current plane structure underlines a certain threshold or no plane structure is found (see table I on page 7).

A good explanation how RANSAC is used for finding plane structures is given in [16]. More information about the SACSegmentation provided by the PCL library, is able at [15]. After that last task in the preprocessing step, the remaining PC is passed to the Segmentation part. The result, the

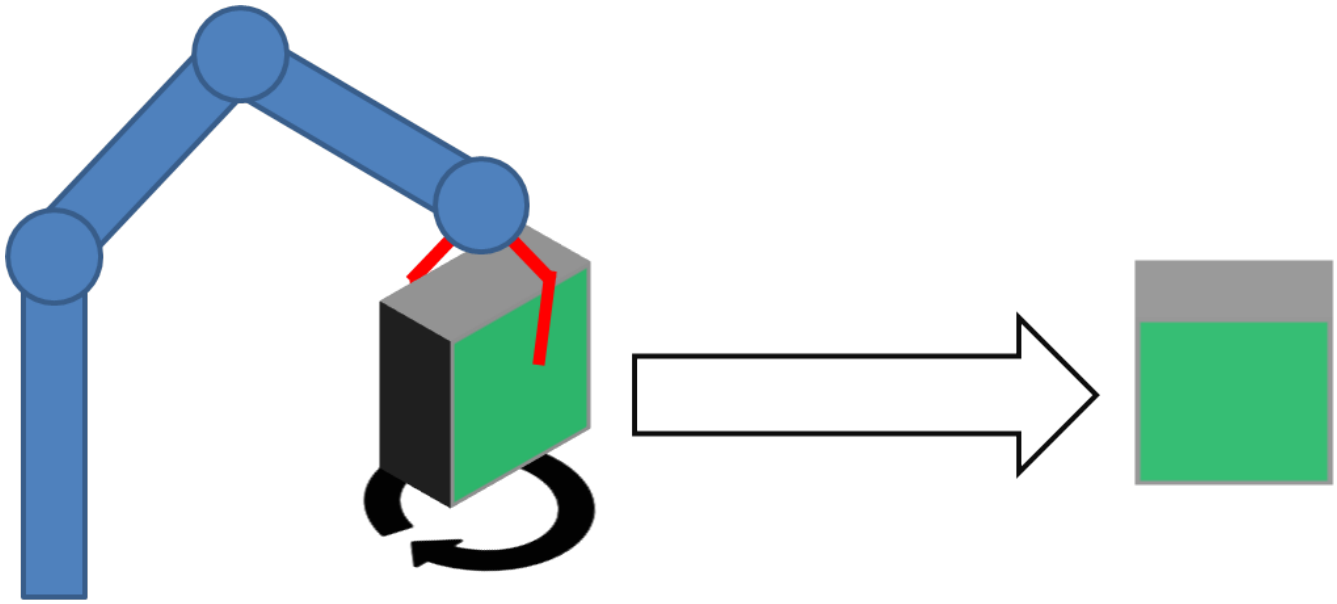


Fig. 3: The autonomous system able to manipulate the environment, change the pose of objects. The base idea is that the manipulator rotates an object around its axis of rotation orthogonal to the surface where the object is deposited to gain new poses. In our approach this is defined by the y-axis.

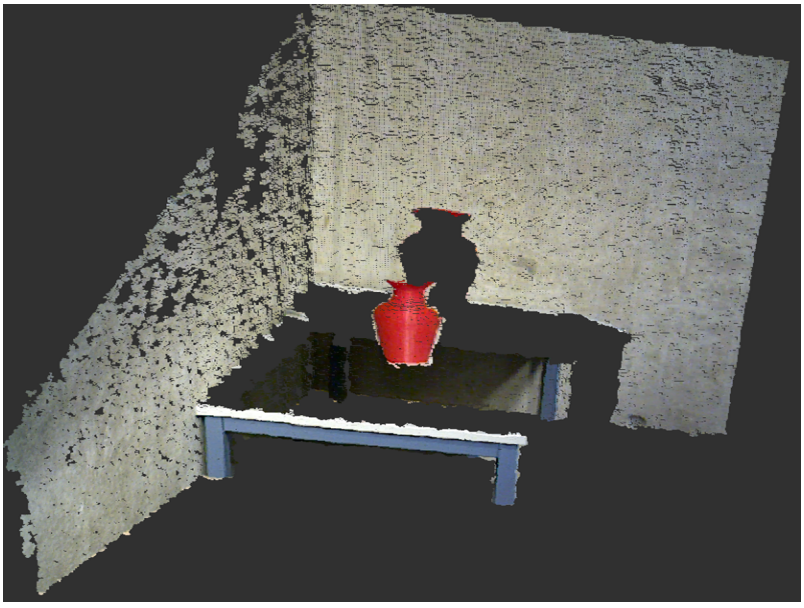


Fig. 4: A raw PC as the Pipeline received as data input. It contains the pixel from the camera and the depth information about every pixel.

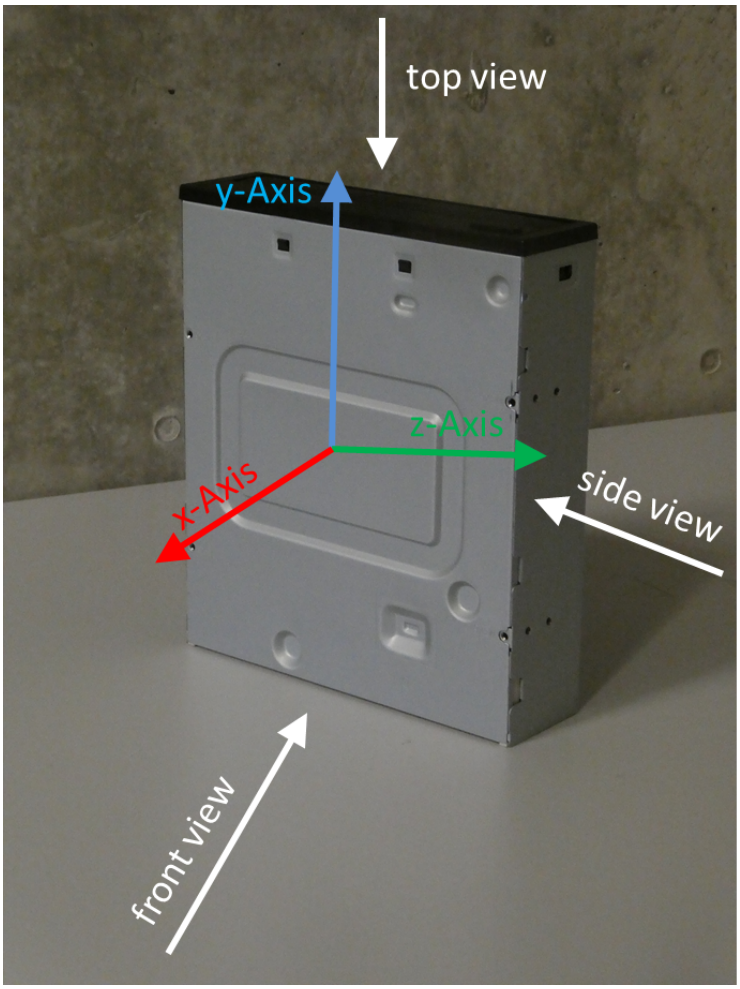


Fig. 5: The coordinate system definition (x-Axis =red arrow, y-Axis = blue arrow, z-Axis = green Arrow and the three views (front, top, side view = white arrows) used for this approach. The views describe from which position on the axis the object is observed.

remaining PC, is shown in figure 7 (p. 8).

3) Segmentation:

This step, shown in 8 on page 8, uses Euclidean Cluster Extraction (ECE) disjoint in [17] to segment the remaining PC into a list of sub PCs. This algorithm, similar to the previous task (remove huge planes), has two thresholds, $minClusterSize, maxClusterSize$. The ECE extracts possible objects with the condition that the amount of points inside the object relies in between $minClusterSize$ and $maxClusterSize$. This prevents the algorithm to extract too tiny objects or measurement uncertainties which does not fit the requirements of finding household sized objects. PCs that fit the requirements

```
1  Remove plane structures (PC, threshold)
2  stopPlaneRemoval = false
3  while !stopPlaneRemoval do
4      extracted_plane_PC = extractPlane(PC)
5      if sizeOf(extracted plane PC) < threshold then
6          stopPlaneRemoval = true
7      end
8      else
9          removePlaneStructure(PC, extracted plane PC)
10     end
11 end
12 return PC
```

TABLE I: Algorithm for removing plane structures. For the plane extraction in line 4 a SACSegmentation algorithm is used as described in [15]. It picks the points inside the PC which create a plane structure. As long as the amount of points is above a threshold, the points are removed from the remaining point cloud with removePlaneStructure() in line 9. Is the threshold is undershot the remaining PC is passed out. SizeOf() counts the points inside a point cloud.

are stored in a list for further processing. Figure 9 on page 9 shows the different colored objects which fit the requirements within a PC.

4) Tracking:

A successful tracking task is required for further steps of the pipeline. It is needed to concentrate exactly one target object. The robot picks one of the items in the PC list generated before and tracks it. For the tracking the centroid is determined by equation (1) and stored. To hold the focus while the robot is moving for every new PC in the list the centroids are calculated and listed.

$$p^i_{center}(\vec{x}) = \begin{pmatrix} \frac{1}{n}\sum_{j=1}^{n} x_j \\ \frac{1}{n}\sum_{j=1}^{n} y_j \\ \frac{1}{n}\sum_{j=1}^{n} z_j \end{pmatrix} \tag{1}$$

With the assumption that the moving speed of the system is much slower than the PC update rate we can find the previous focused object by searching the nearest neighbour of the tracked centroid within the centroid list and a maximal radius around the tracked centroid. If a neighbour is found, the new centroid replaces the previous to hold the track while moving. When no tracking is possible, a new object is tracked. Figure 11 shows a tracked vase. Turquoise represents the detected unknown objects, green the tracked object and the yellow dot shows the calculated center point of the object point cloud. For comparison figure 1 is the raw point cloud as input.

5) Tracking - Preprocess influence:

As mentioned in IV-A4 we use a certain radius for observing the centroids. This means that there is no need searching for other possible objects as long as we track our selected object. We exploit this to reduce the calculating afford in general by removing all points inside the PC which exceed a certain

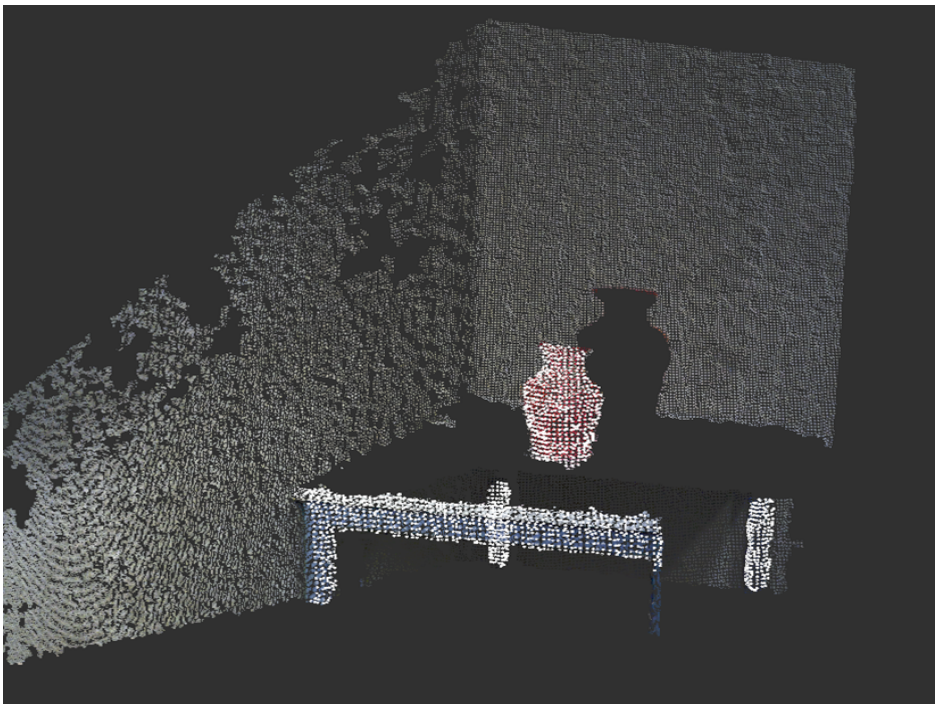


Fig. 7: A preprocessed PC. The white dots are marking the remaining points after preprocessing. As the initial PC has a very high resolution the coarse resolution afterwards is visible. In addition the surrounding walls were removed from the PC.

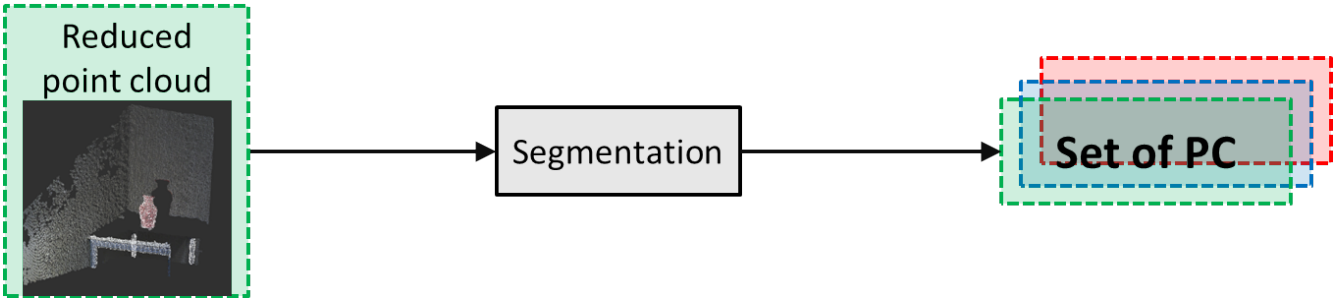


Fig. 8: The pipeline for segmentation. Input is the raw PC. After several processing operations we obtain a set of point clouds for the next task.

distance to our tracked centroid before the preprocessing. This means we reduce the amount of points in the raw PC (Fig. 1, page 3) as is used as input. The reduced raw PC is shown in figure 12 on page 11.

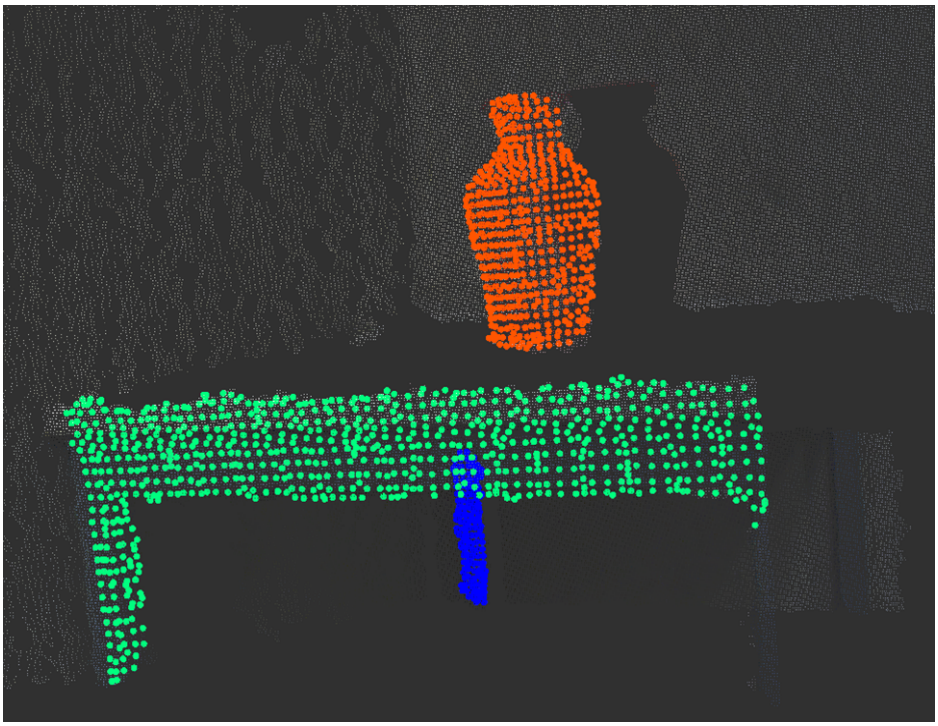


Fig. 9: A point cloud after preprocessing and segmentation. The remaining PCs which may be objects of interest are colored. Red = Vase, green = front of a desk, blue a table leg.

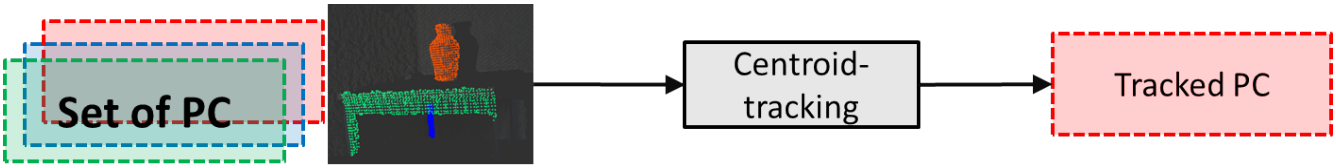


Fig. 10: The pipeline for tracking. The Input is the PC list as generated from segmentation (IV-A3). After choosing a centroid the corresponding PC is used as the output.

B. PC-Controller

The PC-Controller is the interface between Detection and Acquisition. The first task is to store the different point clouds according to the tracked object. The controller is continuously delivered with the tracked PC by the detection and is stored when the PC-Controller is triggered by an external source. This means the controller saves an snapshot of the current PC after triggering. A completed pose change of the object is the necessary source for triggering the storage event. After that the pose changing process is started again until a certain amount of poses to describe the full body of the object are stored as a PC (Table II, p. 12). As the pose of the object changes and not the camera the stored

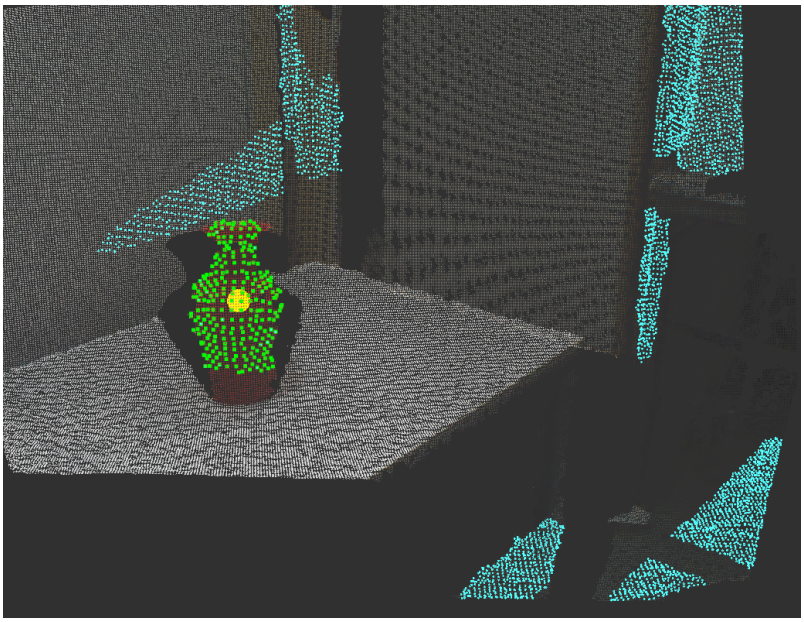


Fig. 11: Example of a PC with a tracked objects (green) and the centroid of the tracked object (yellow) after the preprocessing and the tracking. The turquoise PCs are possible objects stored in the PC list but don't belong to the tracked object.

PCs are superimposing as shown in figure 13 on page 13 where the results of the storage process for a cup and a CD-ROM drive from the top view is shown. Figure 14, p. 14 explains the reason for the superimpose effect.

C. Acquisition

After we gained enough information about our object of interest we still have unregistered and superimposed point clouds (Fig. 13, p. 13) which must be converted to a full 3D-mesh. The Acquisition part of the pipeline provides a task to register the unregistered point clouds to a full point cloud object and with the mesh assembly the pipeline is able to assembly the necessary mesh out of the point cloud to upload it in the database of the object recognition.

1) Registration:

As mentioned we need to register incoherent point clouds before we can create a 3D-mesh. As result of the approach our only information is the angle of every rotation of the object by calculating $\alpha = \frac{2\pi}{n_{max}}$ with $n = amount\ of\ different\ poses$ and the spatial information (x, y, z) of the entries in the point cloud. There is no camera rotation or other external improvements as markers in this scenario.

The challenge of registration in this approach is the missing information about the relationship between the different stored point clouds except that they describe the body of an object. Depending on the amount of poses recorded the PCs have overlapping sections but not how much. This means the regular

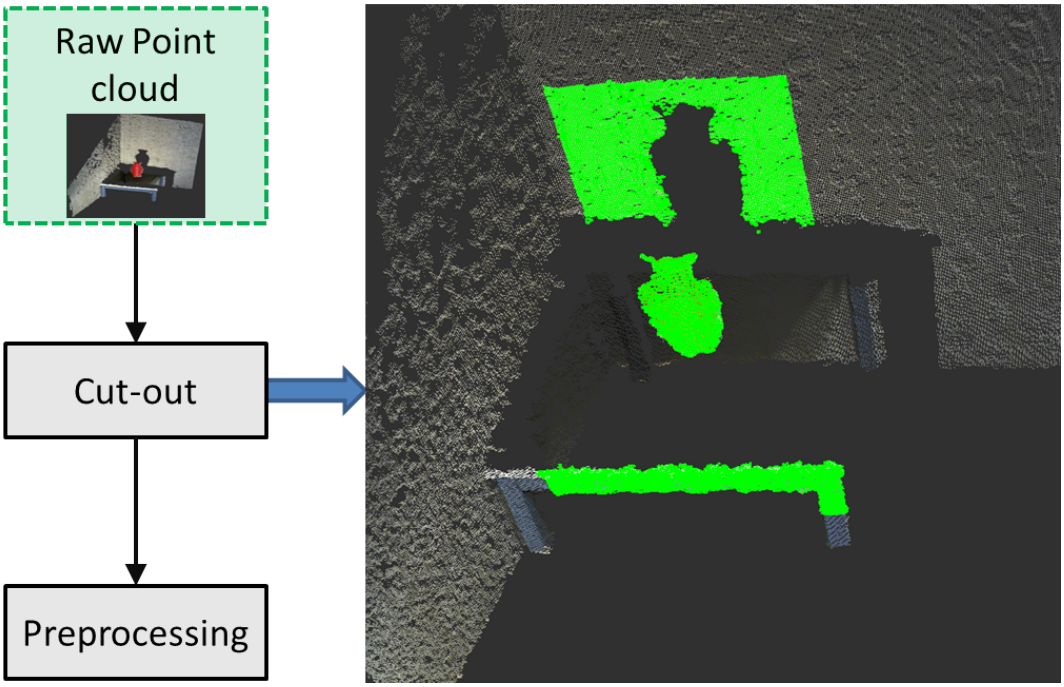


Fig. 12: The pipeline for reducing the points inside the raw PC when a centroid is tracked (here: cubic excluding). The green area marks the remaining points after removal which is passed to preprocessing.

Iterative Closest Point algorithm (ICP) is not working as it needs nearly full overlapping point clouds as described in [18]). Even expanded approaches with the possibility to handle partial overlapping point clouds as [19] are unable to find a suitable solution for the registration problem when one point one pose of the object is part of another pose while most ICP algorithms are unable to detect multiple local minima ([20]). This means the first point cloud fits in the other one resulting in several local solutions where the algorithm is not able to decide which one would be the best. Other algorithms based on keypoint detectors and feature description like mentioned in [20] are not able to find a correlation as there are no unique features in some objects, a vase as rotation invariant for example has the same features from different poses. In addition this approach has no difficulty with the loop closure problem as often occurs when it comes to a registration of point clouds, especially in the scientific field of SLAM.

This difficulties leads to an algorithm solving the registration based on the available information and occurring challenges. The advantage of this approach is that there is no demand for a good initialization, a robustness against measurement noise, including compensating translation and rotation variance while pose changing procedure, and a fast computation time.

The main idea behind the algorithm describes that the distance of the surface of an ideal round object compared to its center of gravity is always the same if we exclude one dimension, reducing the object

```
PC-Controllers (n_max)
get current PC frame
store current PC frame at position 0
n = 1
while n <= n_max do
    change pose of object
    if pose is changed then
        get current PC frame
        store currentPC frame at position n
        n + 1
    end
end
return all stored PCs
```

TABLE II: The algorithm of the PC-Controller. As input n is the amount of required poses. The algorithm takes the current available point cloud, stores it and increases the counter until all necessary poses are stored.

to a circle (see fig. 15b, p. 14), hence the accumulated euclidean difference between the distances is zero (eq. 2, p. 12), describing a coherence between distances and best fit registration. This means to a registration of partial components from a circle the optimum is found when all parts fits together with full overlapping but no overcrossing of the parts.

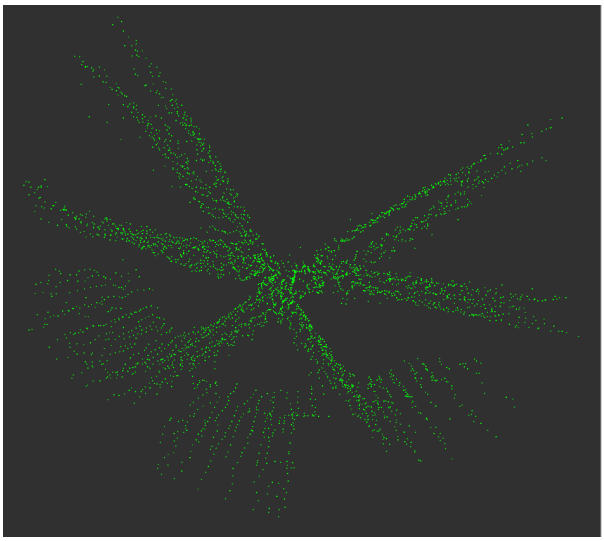
$$\sum_{j=1}^{i} \|v_j(x,y,z)\|_2 \stackrel{!}{=} 0 \quad with\ v_j(x,y,z) = p_{cog,init}(x,y,z) - p_j(x,y,z) \tag{2}$$

To extend this coherence for every object we manipulate them with different settings, collapsing or expanding the point clouds, to vary the distance from the points to the center of gravity as seen in figure 15on page 14. This leads to different distributions of distances for every setting. If we weight the curvature of this distribution we get an indicator for the quality Q of the current setting. The goal is to find the global minimum of Q which belongs to the setting parameter:

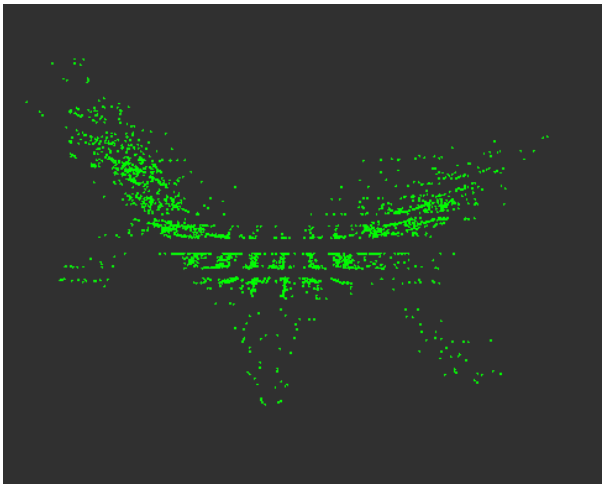
$r(k) \in \mathbb{R}$ within a setting interval

$r(k) =]0, max(\|p_j(x,y,z) - p_{cog}(x,y,z)\|_2)]$

Based on this theory we started with the initial system. Our available information is the point of the camera $p_{cam}(x,y,z)$, in general with the coordinates $(0,0,0)$, and the original center of gravity $p_{cog,init}$



(a) unregistered CD-ROM drive: top view



(b) unregistered cup: top view

Fig. 13: The visualization of the stored PCs after the PC-Controller gains enough information. Because of the rotation of the object the point clouds superimposing each other. This is the unregistered state of the point clouds, shown for a CD-ROM drive and a cup.

based on every available points within all superimposed point clouds (eq. 3, p. 13).

$$p_{cog}(x,y,z) = \frac{1}{i}\sum_{j=1}^{i} p_j(x,y,z) \tag{3}$$

$$with\ p_j \in pc_{init}\ and\ pc_{init} = \{pc_1,\ pc_2,\ ...,\ pc_{n_{max}}\}$$

Next we calculate the vector $v_{init}(x,y,z)$ between the known points (eq. 4, p. 13).

$$v_{init}(x,y,z) = p_{cog,init}(x,y,z) - p_{cam}(x,y,z) \tag{4}$$

The setting parameter $r(k)$ is the maximal euclidean distance between the center of gravity $p_{cog}(x,y,z)$ and all the points p_j within pc_{init} divided by the current setting step $k \cdot \Delta R$. Only the dimension of x,z are used for calculating the distances. This means $r(k)$ and therefore Q depends on the current step $k \in \mathbb{N}^+$ with the interval $k = [1, \Delta R]$. $r(k)$ is used to transform the vector v_{init} to v_{trans} (eq. 5, p. 13).

$$v_{trans,k}(x,y,z) = (r(k)) \cdot v_{init}(x,y,z)\ with\ r(k) = \frac{max(\|p_j(x,z) - p_{cog}(x,z)\|_2)}{\Delta R} \cdot k \tag{5}$$

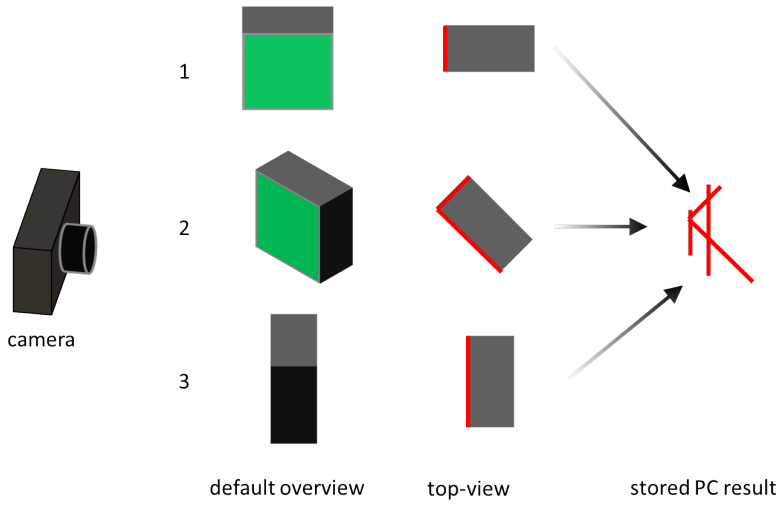
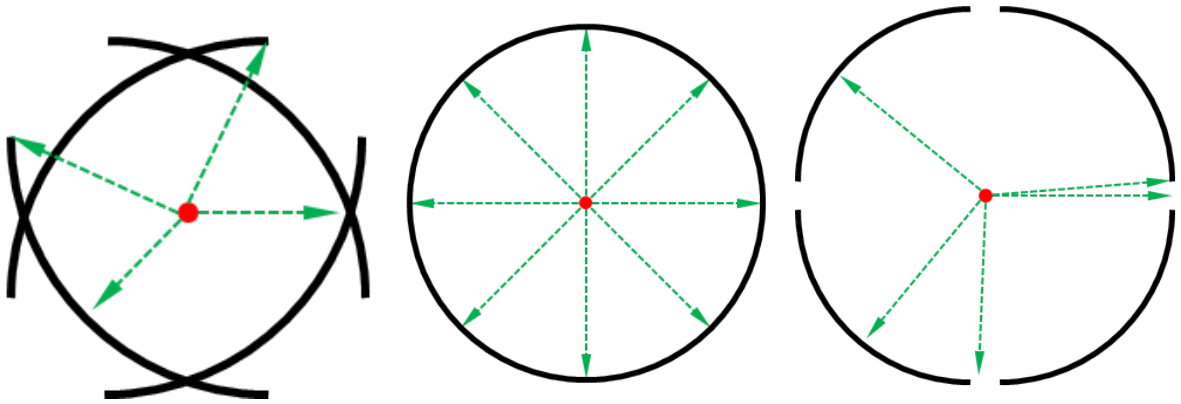


Fig. 14: While the objects change their pose the camera keep its position (1, 2, 3). The red lines on the top-view visualize the part of the object the camera is able to capture as seen from the top. The combined result are superimposed PCs which are stored separately.



(a) collapsed circle *(b) perfect shaped circle* *(c) expanded circle*

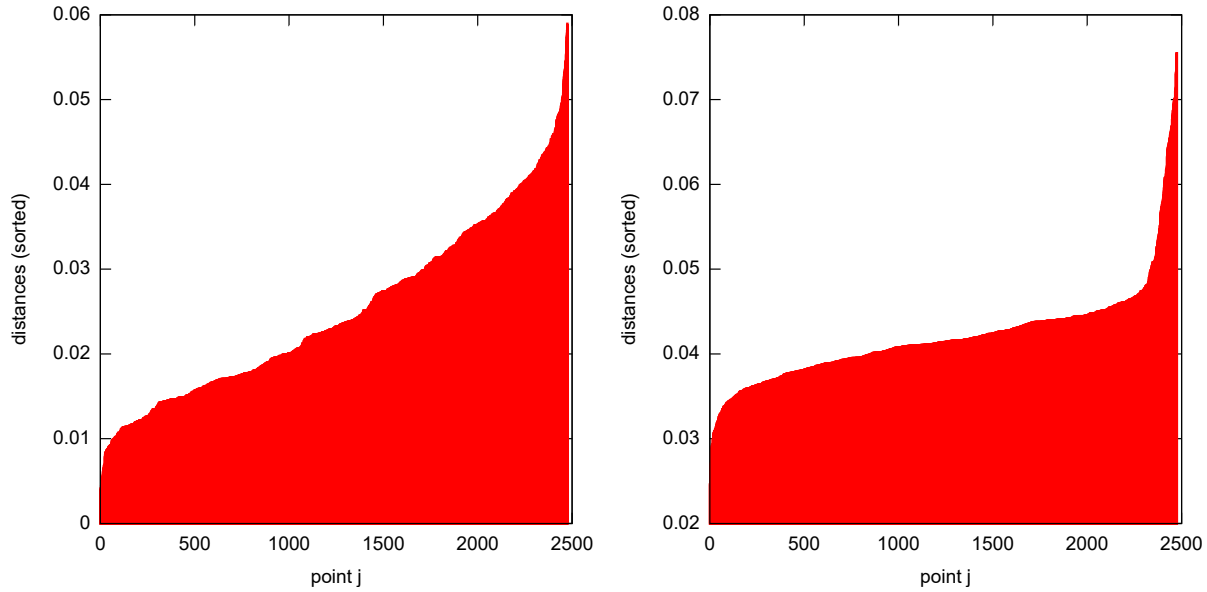
Fig. 15: three possible states of a circle. The distances (green arrows) between the center of gravity (red dot) and the surface (black) in (a) and (c) are not equal. In (b) the distances are equal and therefore represents the best registration.

By adding $v_{trans,k}$ to our original center of gravity $p_{cog,init}$ we gain our new center of gravity $p_{cog,k}$ for the current step k, changing the euclidean distances between $p_{cog,k}$ and all existing points (eq. 6, p. 14).

$$p_{cog,k} = p_{cog,init} + v_{trans}(x, y, z) \tag{6}$$

Before determining the quality Q for k we calculate the changed distances between the points $p_{k,j} \in pc_k$ included in $pc_k = \{pc_{k,1},\ pc_{k,2},\ ...,\ pc_{k,n_{max}}\}$; $pc_k \in \mathbb{R}^{i \cdot n \times 3}$ and $p_{cog,k}$ plus sorting them from the smallest to the largest element (eq. 7, p. 15) as seen in figure 16 on page 15. Due to the later rotation around the y axis we only calculate the distances in the x and z space.

$$f_k(j) = sort(\|p_{k,j}(x,z) - p_{cam}(x,z)\|_2)\ ;\ sort := \ sorting\ from\ lowest\ to\ highest \tag{7}$$



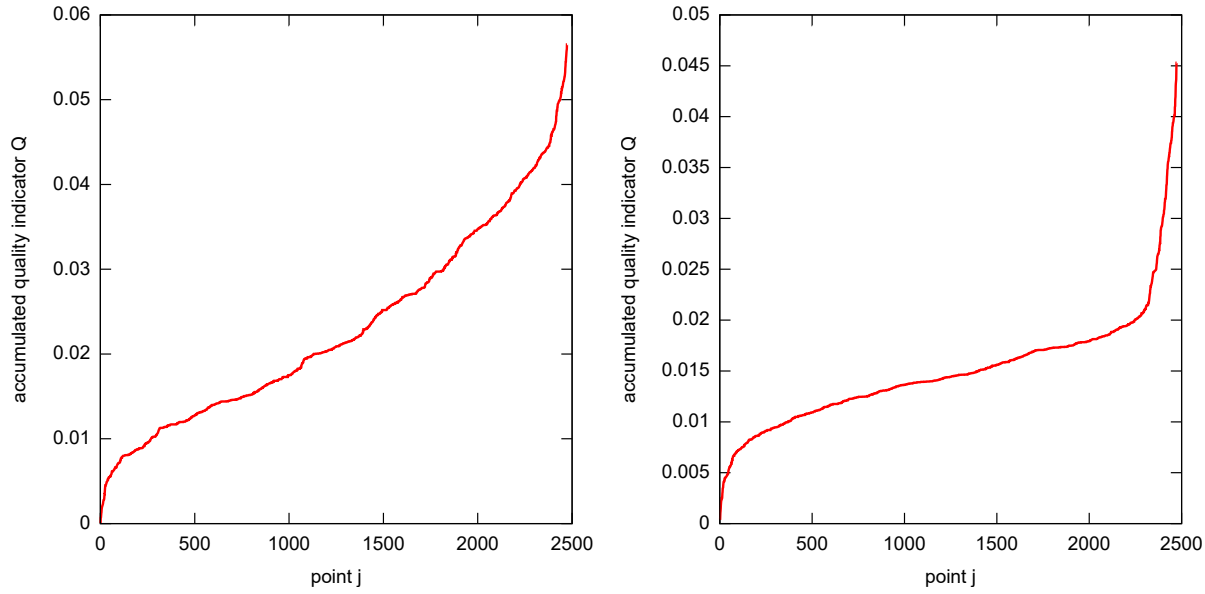
(a) sorted distances from cup example 1

(b) sorted distances from cup example 2

Fig. 16: An example for a result of the function $f(j)$ (eq. 7) for step k. The graph shows sorted distance distribution. While the distances of example 1(a) seems rough and the curvature is always changing strongly (going up and down), the curvature of example 2 (b) only has remarkable changes at the two curves at j 150 and j 2300. For example 1 and 2 we used $r_{example\ 1} = 1.002925$ and $r_{example\ 2} = 1.030420$.

After sorting we get an ascending distribution of distances. Q_k for the current step as our quality Q is computed by punishing the curvature of the distribution, more curvature means a worse solution, since the curvature represents a deviation from the real shape of an object. The result is equation 8 on page 15. It takes the accumulated absolute change of the curvature from $f_k(j=1)''$ till $f_k(j=i)''$. The result is shown in figure 17 on page 16.

$$Q_k = \sum_{j=1}^{i} |f_k(j-1)'' - f_k(j)''| \tag{8}$$



(a) inferred quality from cup example 1 *(b) inferred quality from cup example 2*

Fig. 17: Results of the quality function Q_k for the step k based on the sorted distances in figure 16 for a cup record. In example 1 (a), $Q_1 = 0.56$ which is much higher than $Q_2 = 0.455$ in example 2 (b).The parameter $r_{example\ 2} = 1.030420$ fits better for registration than $r_{example\ 1} = 1.002925$.

If the quality Q_k is calculated for all steps k, the minimal value of Q_{opt} over all Qs represents the optimal possible registration based on the quality function Q_k.

$$Q_{opt} \stackrel{!}{=} min(Q)\ with\ Q = \begin{pmatrix} Q_{k=1} \\ Q_{k=2} \\ \vdots \\ Q_{k=n} \end{pmatrix} \tag{9}$$

As a conclusion follows that one point cloud pc_opt and the center of gravity $p_{cog,opt}$ which relies to the global minimum Q_{opt} describes the best parameter $r(k)$ for a registration based on this approach. k is now the step in which Q_{opt} was approaching.

$$pc_{opt} = \{\exists!\ pc_k | pc_k \Rightarrow Q_{opt}\} \tag{10}$$

$$p_{cog,opt} = \{\exists!\ p_{cog,k} | p_{cog,k} \Rightarrow Q_{opt}\} \tag{11}$$

To reconstruct the rotation of the object and gather a full point cloud describing the whole object a rotation around $p_{cog,opt}$ as the pivot point and y as rotation axis is performed creating the new point

cloud pc^{α}_{opt} out of pc_{opt} containing all point clouds $pc_{opt,1},\ ...,\ pc_{opt,n_{max}}$ and the points within (eq. 12, p. 17).

$$pc^{\alpha}_{opt} = \sum_{n=1}^{n_{max}} pc_{n,opt} \cdot rot(n \cdot \alpha, y, p_{cog,opt})\ with\ \alpha = \frac{2\pi}{n_{max}} \tag{12}$$

$$and\ rot() := \text{rotation matrix arround } y - axis \text{ and } p_{cog,opt} \text{ as pivot point}$$

As a current constraint for a good registration the rotation steps of the object while gaining the point clouds (see IV-B, p. 9) should be as defined in equation 12, page 17.

This creates the full representation of an object merging all points within one point cloud. Several registration results are shown in the section VII, Attachment. To show the work flow of the algorithm it is displayed in III on page 18.

As the calculation time of the registration approach is relied on two input parameters, the amount of the incoming points i within the combined point clouds pc_{init} (which is depending on the resolution of the preprocessing step, see IV-A2, page 3, plus the number of poses n_{max}) and the resolution steps ΔR leading to:

$$estimated\ computation\ time = i \cdot \Delta R \cdot computation\ time\ per\ single\ run \tag{13}$$

the $computation\ time\ per\ run$ contains the time needed for calculating one run, e.g. $n_max = 1$ making this approach very fast in calculation and easy to adjust for the target system. The final step is to transform the current point cloud to a seamless 3D mesh. In addition this algorithm is very easy for parallel processing since the different computation steps k are independent from each other.

2) Mesh Assembly:

The mesh assembly generates a 3D-mesh based on the remaining points inside the registered PC. To create a mesh the point cloud registration technique kd-tree nearest neighbor search provided by FLANN [21] is used. [22] gives an overview of advantages using FLANN. After creating the mesh is uploaded to the data base and can be used for the object recognition or other tasks

PC-Registration $(n_{max}, \Delta R, pc_1, pc_2, ..., pc_{n_{max}})$

calculate initial center of gravity p_{cog} (3)

calculate initial vector v_{init} (4)

$Q_{opt} = 0$

for $k = 1; k <= \Delta R; l++$ **do**

  $sum = 0$

  calculate translation vector v_{trans} (5)

  $p_{cog,k} = p_{cog,init} + v_{trans}$ (6)

  for $j = 1; j <= i; j++$ **do**

    $dst[j] = \|p_{k,j} - p_{cam}\|_2$ (7)

  end

  sort dst ascending to gain $f_k(j)$ (7)

  for $j = 1; j <= i; j++$ **do**

    $sum = \|f_k/j - 1)'' - f_k(j)''\|_2 + sum$ (8)

  end

end

$Q_k = sum$ (8)

if $Q_k < Q_{opt} || Q_{opt} == 0$ **then**

  $Q_{opt} = Q_k$ (9) $_{p_cog,opt} = p_{cog,k}$ (11)

end

for $n = 1; j <= i; j++$ **do**

  $pc^{\alpha}_{opt} += rotate(n \cdot \alpha, pc_{n,opt})$ (10

)

end

return pc^{α}_{opt}

TABLE III: The registration algorithm based on the equations 3 to 10.

V. RESULTS

In this section we will demonstrate the pipeline results based on several real objects. The system specification on which the computations are running is a Intel Core i7-4790 with 3.60 GHz base clocking, 16 GB of RAM with Ubuntu 14.04 as the operating system. To gather the pixel with depth information we used a Asus Xtion Pro camera. The following items are our test objects (original pictures, see figure 21 in section Attachment):

- bottle of coke
- cup
- robot gripper
- vase
- CD-ROM drive

The goal is to start from trivial geometrical objects e.g. the CD-ROM drive as a cuboid or rotation symmetrical object like the the vase up to more complex objects like the robot gripper. The Rotation of the objects to gain a new view was done manually. It means we changed the view by hand while the preprocessing step was concurrently active, sending the processed clouds to the PC-Controller, leading in additional translation and rotation discrepancies. The rotation done by hand has always inevitable variant around the mean angle. By picking up and putting down objects as needed for rotation a small translation is not preventable. This increases the challenge for the approach, proving the robustness of the algorithm. We changed the poses seven times to create eight views, so $n_{max} = 8$. For the PC registration we used a step resolution of $\Delta R = 100$.

1) Preprocessing & Segmentation:
After the Preprocessing and Segmentation the resulting points stored inside the pcs pc_{opt} by the PC-Controller is shown in table IV on page 19.

object	total points
bottle of coke	3162
cup	2476
robot gripper	2728
vase (low quality)	3506
CD-ROM drive (low quality)	3755
vase (high quality)	8730
CD-ROM drive (high quality)	6649

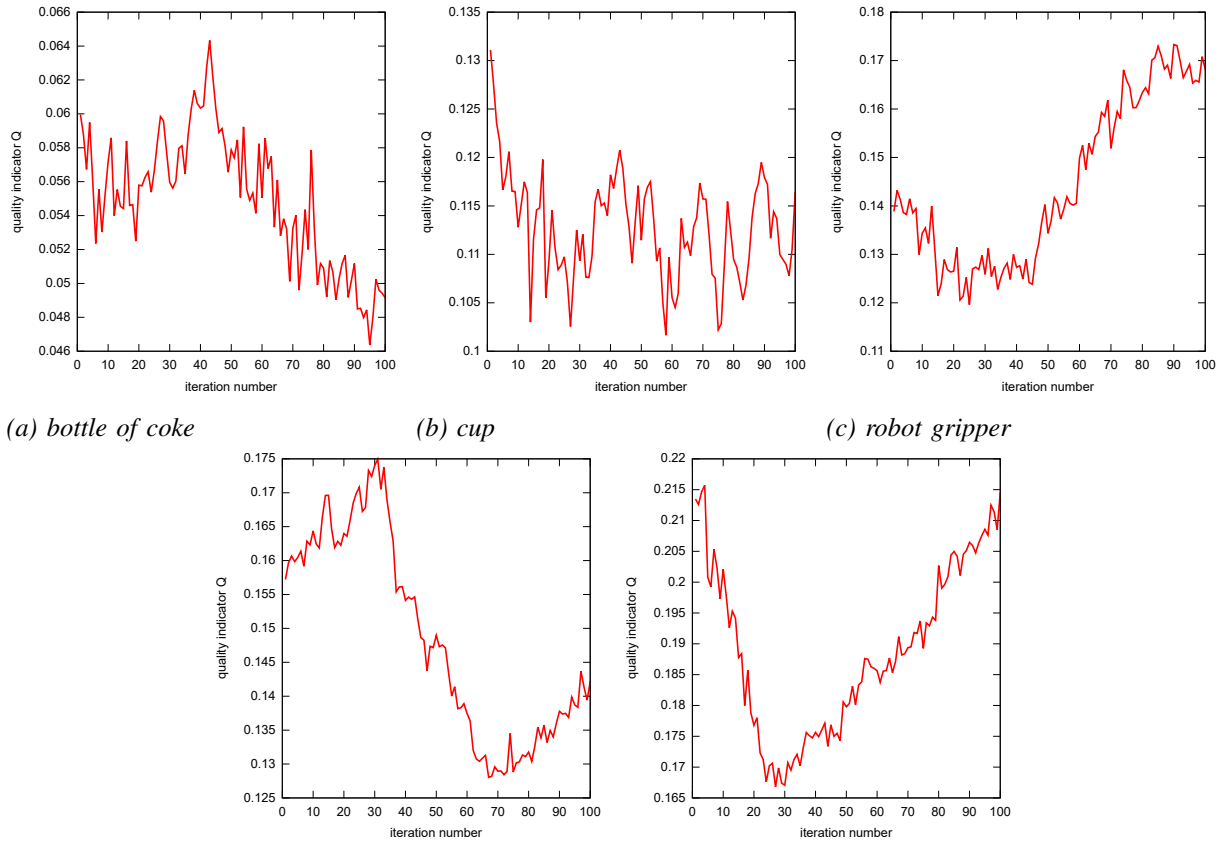
TABLE IV: The used objects and the corresponding points p_{opt} within pc_{opt} based on Q_{opt}. For the vase and CD-ROM drive we used different preprocessing resolutions to see the dependency between the point cloud density and the 3D-mesh.

2) Quality Function & Registration:
As mentioned we use the PC-Controller point cloud storage for the registration and mesh assembly. The registration is based on Q which is calculated as described in chapter IV-C,Acquisition, page 10. Figure 18 on page 20 presents the evolution of the quality Q in comparison with step k.
After analyzing the graphs for the Quality Q we found the best value for Q_{opt} and the corresponding r_{opt} (table VI, page 21). Using this parameters we can generate our registrated point cloud.

object	total points	quality Q_{opt}	r_{opt}
bottle of coke	3162	0.046361	1.032026
cup	2476	0.101656	1.033930
robot gripper	2728	0.119601	1.020574
vase (low quality)	3506	0.07420	1.038492
CD-ROM drive (low quality)	3755	0.123773	1.025508
vase (high quality)	8730	0.128040	1.039870
CD-ROM drive (high quality)	6649	0.166807	1.024720

TABLE V: The best quality Q and the corresponding r_{opt} settings for the point cloud registration.

In view of the fact that we had eight views $n_{max} = 8$ our current angle $\alpha = \frac{2\pi}{8}$. With the information



(a) bottle of coke *(b) cup* *(c) robot gripper*

(d) vase *(e) CD-Rom drive*

Fig. 18: The result of the quality Q based on the step k. k reaches from 1 to $\Delta R = 100$. For (c), (d) and (e) the global minimum is easily obtained. (b) results in four possible minimas where higher resolution steps ΔR should be used to gain a higher probability of finding the optimum (see fig. 19, p. 21 for the quality result of $\Delta R = 1000$). (a) reaches its optimum close at the border of r.

of the angle and the calculated parameters r_{opt} we can execute our point cloud registration. The results of the registration algorithm is show in 22 in the section VII, Attachment.

3) Real Object and created Mesh:

After the successful registration the last part of the pipeline is creating a seamless mesh. The quality of the mesh strongly depends on the resolution of the point clouds. For the vase and the CD-ROM drive the resolution of the preprocessing step was changed once. This gives a good comparison between a low point density and a high density of points within the point cloud influencing the quality of the created mesh (figure 20, p. 23).

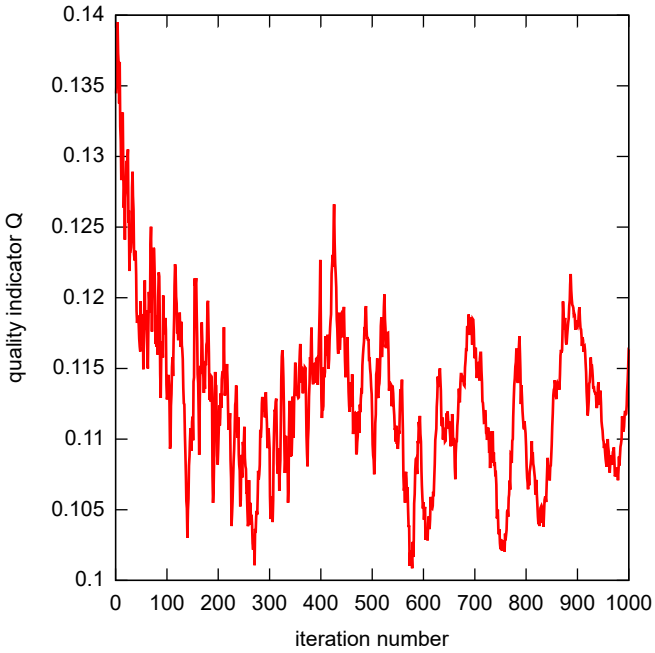


Fig. 19: The result of the quality Q based on the step k. k reaches from 1 to $\Delta R = 1000$. With this resolution the error calculating the singularity of the minimum is reduced far enough to distinguish them.

object	# measure	real	mesh	variance in %
bottle of coke	1	6.5	7.8	16.6
cup	1	11.7	11.2	4.46
	2	8.2	8.8	6.81
robot gripper	1	7.8	7.4	5.41
	2	14.5	14.7	1.36
vase	1	5.2	5.2	0
	2	11.9	12.1	1.65
CD-ROM drive	1	4.1	3.9	5.12
	2	14.5	14.7	1.36
	3	16.9	14.9	13.42

TABLE VI: The real objects compared to the created mesh. The measuring points in # measure, is shown in figure 28 in chapter VII, Attachment. All measure units are in centimeter. The variance gives the inaccuracy between real and mesh measurement in percent.

To validate the precision of the created meshs we defined some points and measure the direct distance between them, on the real object and the mesh. The result is shown in VI on page 21. The vase bottleneck (measure # 1) in the mesh is, based on a coarse measure resolution, very close to the original, for example. Only the CD case measurement # 3 is 13.42 percent smaller than the original. This is due to the algorithm of removing huge planes. Sometimes it cuts the bottom of objects. The

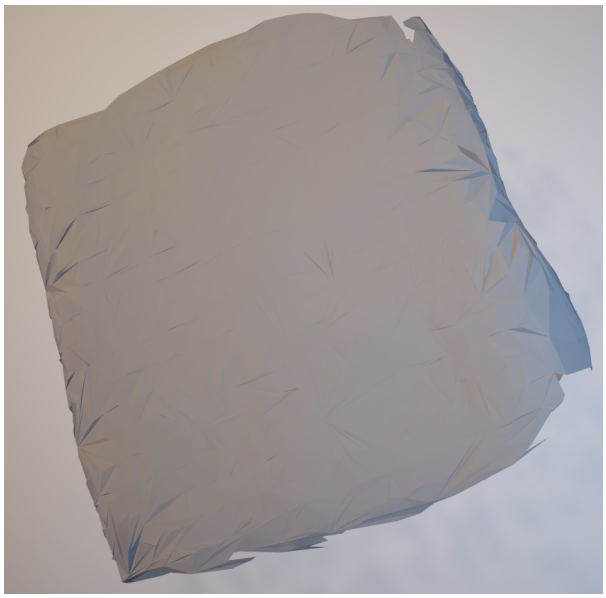
only not satisfying result is the bottle of coke with a 16.6 % bigger diameter than the real. Despite of this the remaining objects reach a satisfying level of accuracy, mostly around 3 percent.
All generated meshs, from the coke up to the CD-ROm drive, are shown in the section VII, Attachment. These meshs can now used for any task which needs a 3D-Mesh of an object, including object recognition. With this result the pipeline is able to make the object detection, data acquisition, 3D-Model generation tasks by itself in order to automate object recognition (with constraint to the manual pose changing).

VI. Conclusion & Outlook

We described the different tasks necessary for a curious autonomous system starting with pre-processing up to the acquisition and recognition task and the connection to each other. The pipeline replaces the manual data creation task as is common for supervised learning and solves the occurring problems in unsupervised approaches. In addition were also able to force the problems of point cloud registration as it is needed for 3D mesh creating. To force the mesh deformation we have to improve the density of our point clouds. A further step is to add a local point cloud registration algorithm like ICP to improve the point cloud registration accuracy after the global registration. With this additional implementation we create a second factor of quality. we can optimize. Finally it is necessary to replace the manually rotation of the objects by the manipulation system to create a fully autonomous system and remove the constraint needing a rotation angle α which should be the same for every rotation at the moment. Beside further improvements the current state of the approach shows the huge possibilities and the satisfying results in this early stage of development.

(a) Mesh of vase with 3506 points

(b) Mesh of vase with 8730 points



(c) Mesh of CD-ROM drive with 3755 points

(d) Mesh of CD-ROM drive with 6649 points

Fig. 20: Different meshs depending on the resolution. The vase in (a) seems smaller than vase (b), the bottom is cut and the top of (a) is not fully reconstructed. Compared to (c) we can see the the contours much more better (the circular indentation) than in (c).

REFERENCES

[1] B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, vol. 9912.

[2] Yulan Guo, M. Bennamoun, F. Sohel, Min Lu, and Jianwei Wan, "3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2270–2287, Nov. 2014.

[3] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.

[4] "About - Point Cloud Library (PCL)," online; accessed 02-01-2018. [Online]. Available: http://pointclouds.org/about/

[5] X. Song, D. Muselet, and A. TréMeau, "Affine transforms between image space and color space for invariant local descriptors," *Pattern Recogn.*, vol. 46, no. 8, pp. 2376–2389, Aug. 2013.

[6] A. S. Mian, M. Bennamoun, and R. Owens, "Three-dimensional model-based object recognition and segmentation in cluttered scenes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 10, pp. 1584–1601, 2006.

[7] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 4, pp. 807–832, 2002.

[8] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference On*. IEEE, 2010, pp. 2155–2162.

[9] S. A. A. Shah, M. Bennamoun, and F. Boussaid, "Keypoints-based surface representation for 3D modeling and 3D object recognition," *Pattern Recognition*, vol. 64, pp. 29–38, Apr. 2017.

[10] M. Ruhnke, B. Steder, G. Grisetti, and W. Burgard, "Unsupervised learning of 3d object models from partial views," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference On*. IEEE, 2009, pp. 801–806.

[11] J. Shin, R. Triebel, and R. Siegwart, "Unsupervised 3D Object Discovery and Categorization for Mobile Robots," in *Robotics Research*, H. I. Christensen and O. Khatib, Eds. Cham: Springer International Publishing, 2017, vol. 100, pp. 61–76.

[12] F. Endres, C. Plagemann, C. Stachniss, and W. Burgard, "Unsupervised discovery of object classes from range data using latent Dirichlet allocation." in *Robotics: Science and Systems*, vol. 2. Seattle, Washington;, 2009, p. 113120.

[13] R. B. Rusu, N. Blodow, Z. Marton, A. Soos, and M. Beetz, "Towards 3D object maps for autonomous household robots," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference On*. IEEE, 2007, pp. 3191–3198.

[14] "Rendering techniques 2001: proceedings of the [12th] eurographics workshop [on rendering] in london, united kingdom, june 25 - 27, 2001 ; eurographics," OCLC: 248344663.

[15] Documentation - point cloud library (PCL). Online; accessed 17-11-2017. [Online]. Available: http://pointclouds.org/documentation/tutorials/planar_segmentation.php#planar-segmentation

[16] M. Y. Yang and W. Förstner, "Plane detection in point cloud data," in *Proceedings of the 2nd int conf on machine control guidance, Bonn*, vol. 1, pp. 95–104.

[17] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, October 2009.

[18] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992. [Online]. Available: http://dx.doi.org/10.1109/34.121791

[19] D. Chetverikov and D. Stepanov, "Robust euclidean alignment of 3d point sets," in *First Hungarian Conference on Computer Graphics and Geometry*, 2002, pp. 70–75.

[20] H. Lei, G. Jiang, and L. Quan, "Fast Descriptors and Correspondence Propagation for Robust Global Point Cloud Registration," *IEEE Transactions on Image Processing*, pp. 1–1, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/7918612/

[21] "FLANN - Fast Library for Approximate Nearest Neighbors : FLANN - FLANN browse," http://www.cs.ubc.ca/research/flann/, online; accessed 02-05-2017.

[22] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, 2014.

[23] "Object Recognition Kitchen — object_recognition_core," http://wg-perception.github.io/object_recognition_core/#, online; accessed 05-05-2017.

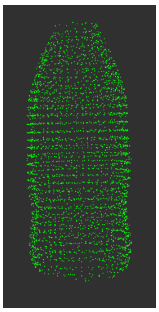
VII. ATTACHMENT

(a) bowl *(b) cup* *(c) robot gripper*

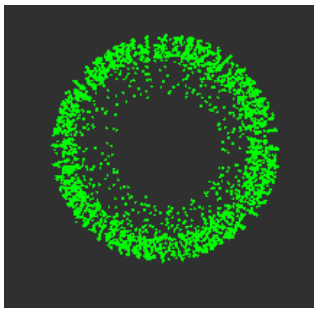
(d) vase *(e) CD-Rom drive*

Fig. 21: The five different object used for the approach

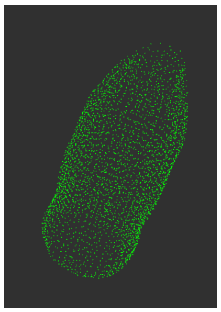
.



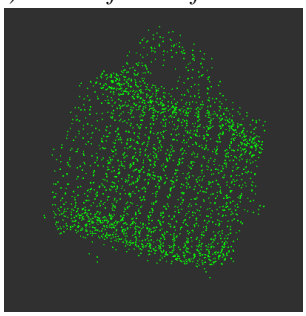
(a) bottle of coke: front view



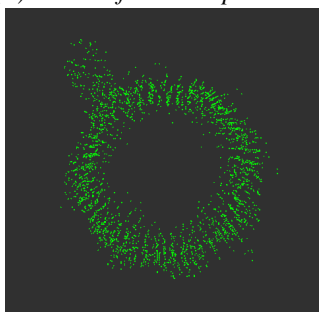
(b) bottle of coke: top view



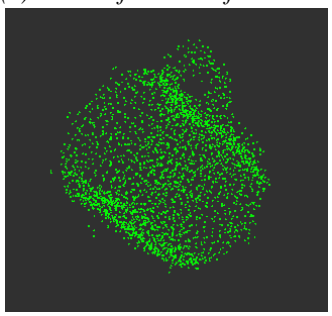
(c) bottle of coke: default view



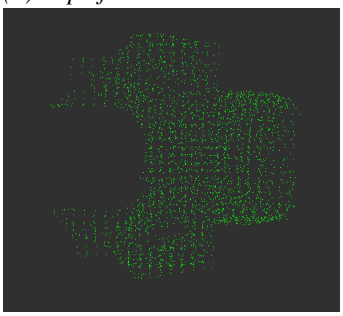
(d) cup: front view



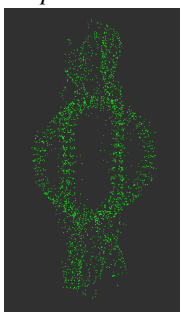
(e) cup: top view



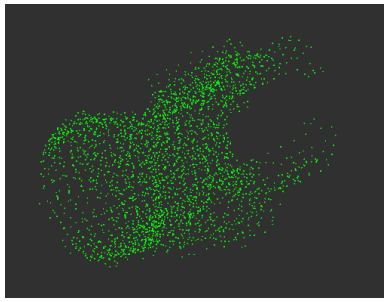
(f) cup: default view



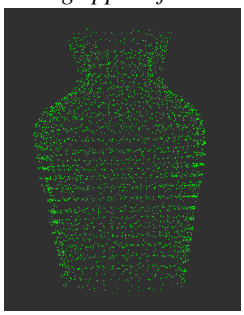
(g) robot gripper: front view



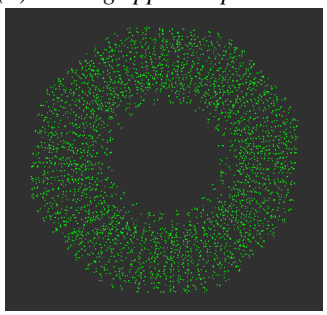
(h) robot gripper: top view



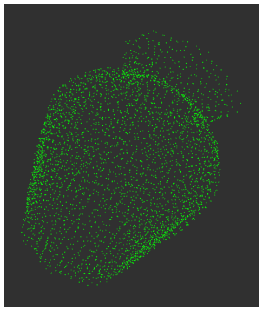
(i) robot gripper: default view



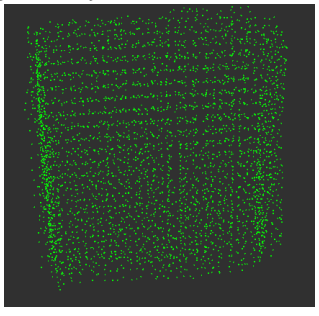
(j) vase: front view



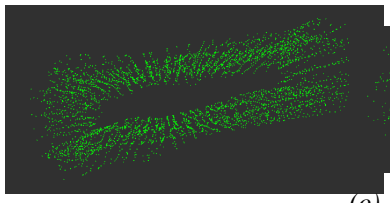
(k) vase: top view



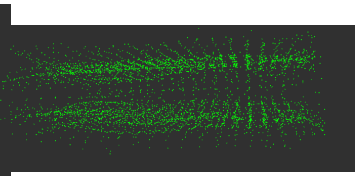
(l) vase: default view



(m) CD-ROM drive: front view



(n) CD-ROM drive: top view



(o) CD-ROM drive: side view

Fig. 22: different point clouds of objects from the tests after the point cloud registration algorithm

Fig. 23: Mesh: bottle of coke

Fig. 24: Mesh: cup

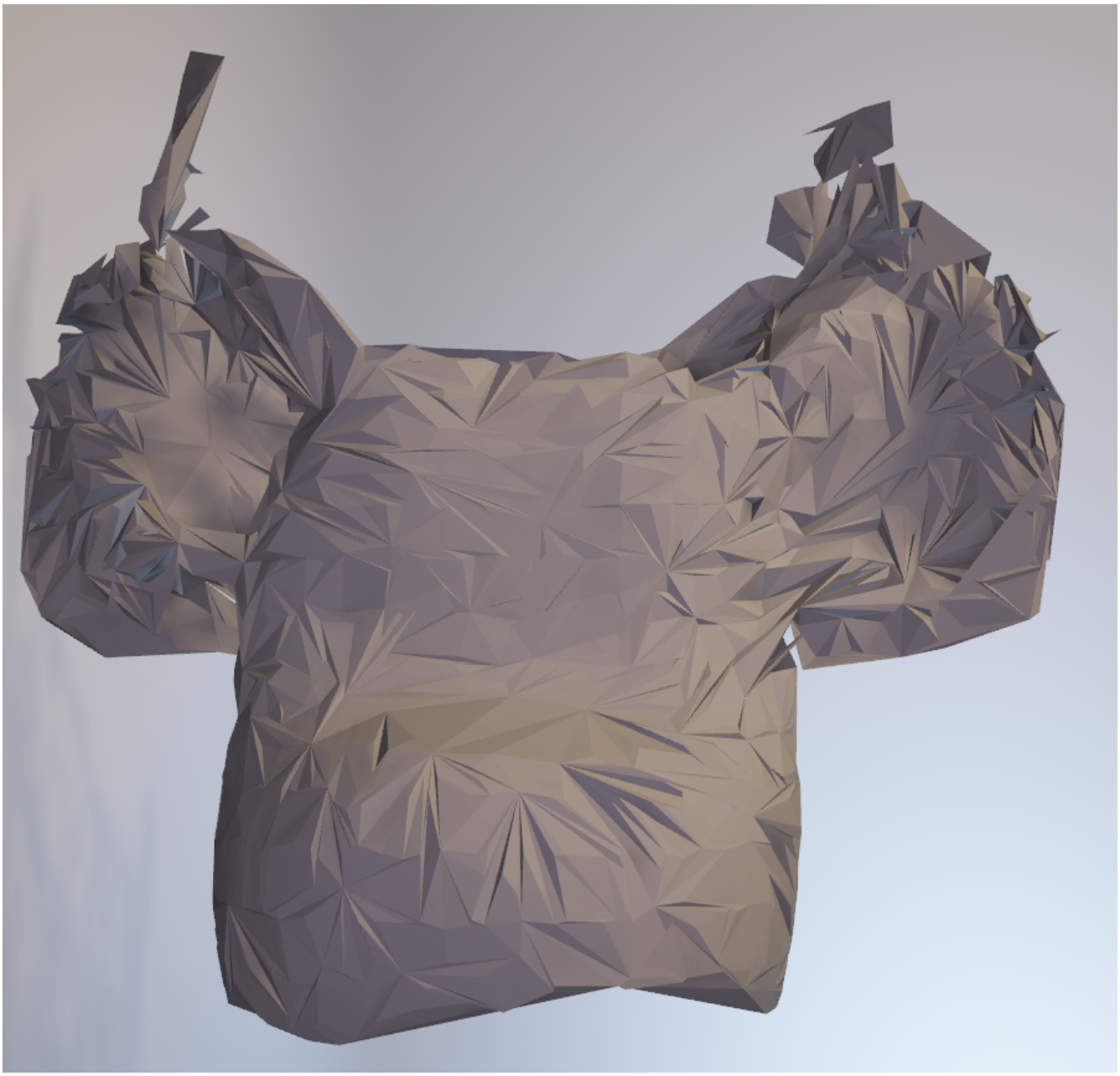


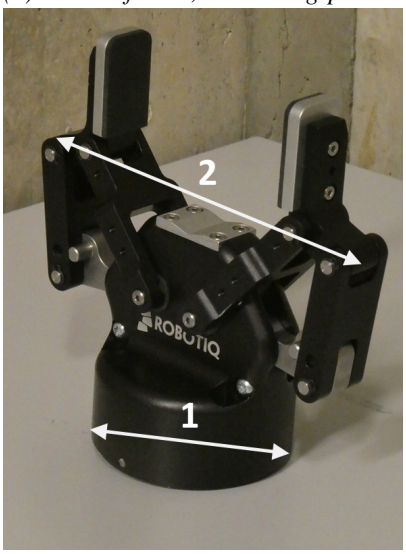
Fig. 25: Mesh: robot-gripper

Fig. 26: Mesh: vase

Fig. 27: Mesh: CD-ROM drive

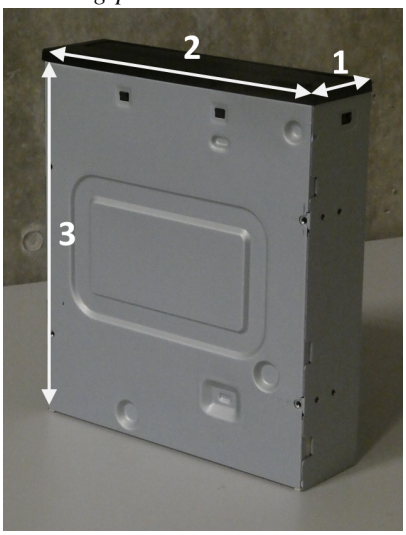
(a) bottle of coke, measuring points

(b) cup, measuring points



(c) robot gripper, measuring points

(d) vase, measuring points



(e) CD-ROM drive, measuring points

Fig. 28: The measurements done for validating the precision of the pipeline.