Liquid Segmentation in 3D CT Baggage Scans

Bachelor Thesis

Jihao Andreas Lin
Visual Inference Lab
Department of Computer Science

Jihao Andreas Lin

2416278

B.Sc. Computer Science

Bachelor Thesis

Title	Liquid Segmentation in 3D CT Baggage Scans
	Flüssigkeitssegmentierung in 3D CT Gepäckscans
Date of Submission	October 16, 2018
Examiner	Prof. Stefan Roth, Ph.D.
Supervisor	M.Sc. Faraz Saeedan

Prof. Stefan Roth, Ph.D.

Visual Inference Lab

Department of Computer Science

Hochschulstraße 10

Technische Universität Darmstadt

64289 Darmstadt

Germany

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Jihao Andreas Lin, die vorliegende Bachelor-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§38 Abs.2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird.

Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Datum: Unterschrift:

Declaration of Authorship

I hereby formally declare that I, Jihao Andreas Lin, have written the submitted thesis independently pursuant to § 22 section 7 of APB TU Darmstadt. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all the literature and all the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.

I am aware that in case of an attempt at deception based on plagiarism (§ 38 Abs. 2 APB), the thesis would be graded with 5.0 and counted as one failed examination attempt.

The thesis may only be repeated once.

In the submitted thesis the written copies and the electronic version for archiving are identical in content pursuant to § 23 section 7 of APB TU Darmstadt.

Date: Signature:

Abstract

Semantic segmentation recently received major performance improvements in the domain of two-dimensional images, mainly due to the increasing availability of ground truth data and the success of fully convolutional neural networks. However, recent methods are not applicable to dense, high-resolution, three-dimensional data due to cubically increasing computational costs of 3D convolutions. We propose two different approaches for liquid segmentation in dense, high-resolution 3D CT baggage scans: The first approach combines traditional region growing with an automatic seed point selection algorithm that exploits liquid-specific properties. The second approach is based on an efficient deep neural network, namely *PointNet* [1]. To deal with the overwhelmingly voluminous data, we propose thresholding and sampling techniques, converting dense volumes into sparse point clouds before processing. Experiments show that, in terms of segmentation results, the traditional approach performs significantly better than the deep learning-based approach. However, the deep learning-based approach is significantly faster. Segmentation performances of both approaches are only slightly influenced by varying input point cloud sizes. Furthermore, we analyze shortcomings of our deep learning-based approach, providing theoretical discussions and empirical studies to support our hypotheses.

Contents

I. List of Figures

II. List of Tables

III. List of Equations

IV. List of Algorithms

V. List of Abbreviations

2D		two-dimensional
3D		three-dimensional
4D		four-dimensional
APB	*Allgemeine Prüfungsbestimmungen*	general exam regulations
ARG		Automatic Region Growing
CAD		computer-aided design
ca.	*circa*	approximately
cf.	*confer*	compare
CNN		convolutional neural network
CPU		central processing unit
CRF		conditional random field
CT		computed tomography
et al.	*et alia*	and others
etc.	*et caetera*	and the others
e.g.	*exempli gratia*	for example
GT		ground truth
GPU		graphics processing unit
i.e.	*id est*	in other words
IoU		Intersection over Union
max		maximum
MiB		mebibyte (2^{20} bytes)
min		minimum
MLP		multilayer perceptron
MRI		magnetic resonance imaging
Ph.D.	*Philosophiae Doctor*	Doctor of Philosophy
PN		PointNet
RGB		red, green, blue color model
s		second
SI	*Système international d' unités*	International System of Units
TU	*Technische Universität*	Technical University
TV		television
vs.	*versus*	against

1 Introduction

Semantic segmentation is an important pixel-level computer vision task. The goal is to precisely understand images by assigning semantic categories, *e.g.* human, street, building, tree or dog, to each pixel in an image. Possible application areas include autonomous driving, to perceive the surrounding traffic; medical image analysis, such as tumor localization; and satellite image analysis, separating roads, forests, waters, *etc*.

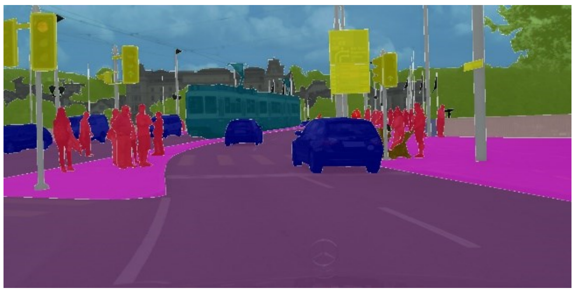


Figure 1: 2D street scene semantic segmentation example from the *Cityscapes* [10] dataset

Traditional techniques for image segmentation include region growing algorithms [2, 3, 4, 5], the watershed transformation [6, 7] and clustering algorithms [7, 8]. They are simple, efficient and do not require ground truth data, but results are typically mediocre.

Recent research significantly improved semantic segmentation performance, mainly by utilizing available ground truth data [9, 10] with deep learning, particularly fully convolutional neural networks and receptive field increasing techniques [11, 12, 13, 14]. However, these methods strictly require annotated data and tremendous computing power.

Besides regular 2D RGB images, nowadays, other types of data become increasingly available. In particular, 3D data types have gained a lot of attention, especially in the context of autonomous driving and medical image analysis. Sometimes, deep learning algorithms use multiple 2D projections or slices to represent the 3D data [15], because direct processing with *e.g.* 3D convolutions is infeasible due to immense computational costs. However, three dimensions are inherently more expressive than two because the extra dimension can capture complex 3D poses and depth information, whereas projection to two dimensions forfeits both, resulting in occlusions and ambiguities. It is therefore beneficial to directly process 3D data.

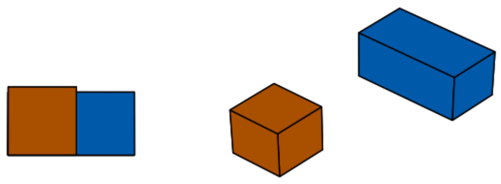


Figure 2: Depending on the point of view, a 2D projection can be highly ambiguous, discarding valuable information.

Three-dimensional data is typically available as either dense volumes or sparse point clouds:

Dense volumes are the 3D equivalent of regular 2D images, represented by a 3D grid filled with intensity values. Their regular shapes, namely cuboids, facilitate the usage of native memory arrays and operations like convolutions. However, cubically increasing computational costs can quickly become a bottleneck. Dense volumes are prominently used for computed tomography (CT) scans and magnetic resonance imaging (MRI).

Point clouds are collections of coordinates that sparsely resemble an arbitrary, and therefore irregular, shape together. This sparse representation lowers computational costs and memory requirements but makes native processing intractable, such that specific point cloud algorithms were developed, involving voxelization, projection and direct processing [1, 16, 17, 18, 19]. Point clouds are typically created using a laser rangefinder.

In this thesis, we face the task of liquid segmentation in high-resolution 3D CT baggage scans. In particular, the goal is to separate liquids from the background by labelling every voxel as either liquid or background. Accomplishing this task opens new avenues for airport security checks and gathers insight about dealing with high-resolution 3D CT data.

Computed tomography scans are commonly used in the field of medical image processing, where images usually depict human organs or body parts, such as brains or joints. Our 3D CT baggage scans include the bags, suitcases, *etc.* and miscellaneous travel items, such as clothing, laptops or bottles. They were provided by *Smiths Detection* and created using their screening systems. Notably, a single scan contains more than 180,000,000 voxels, which is more than 20x the pixel count of 4K ultra-high-definition images. While high resolutions are widely preferred in *e.g.* smartphone cameras or TV monitors, computer vision algorithms suffer from escalating computational costs and memory requirements. Finding a way to deal with these extremely voluminous images is therefore an important subtask.

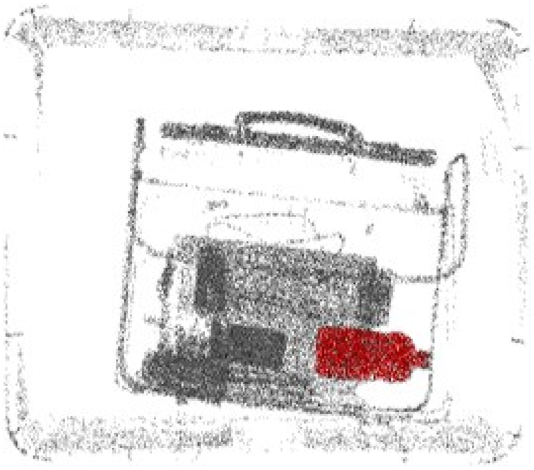


Figure 3: Point cloud version of a 3D CT briefcase scan with liquid annotation.

To identify and compare applicability, performance and shortcomings of traditional and deep learning-based semantic segmentation approaches in the novel context of high-resolution 3D CT baggage scans, we chose one representative from either category to perform the task of liquid segmentation.

For the traditional method, we designed a simple seeded region growing algorithm. It is complemented with an automatic seed point selection algorithm. Both algorithms exploit liquid-specific properties, namely spatial coherence and intensity continuity. We call the combined procedure *Automatic Region Growing* (*ARG*).

For the deep learning-based method, it is infeasible to extend popular CNN-based 2D semantic segmentation techniques to 3D due to overwhelming computational costs of 3D convolutions. Instead, we chose the very popular, simple and efficient *PointNet* (*PN*) architecture, introduced by Qi *et al.* [1], because it performed well on 3D point cloud classification and segmentation tasks. We incrementally introduce slight extensions to adapt *PointNet* to the foreign task of liquid segmentation in 3D CT baggage scans. Furthermore, we provide theoretical discussions about shortcomings and conduct empirical studies to support our hypotheses.

To deal with dense high-resolution 3D data, we propose a two-step procedure to convert the volumes into sparse point clouds. In the first step, a threshold is applied to remove irrelevant voxels. In the second step, sampling creates sparse point clouds, establishing manageable input data sizes with a fixed number of points. Both segmentation algorithms receive a sampled point cloud as their input.

Our contribution can be summarized as follows:

We present two fundamentally different approaches to liquid segmentation in 3D CT images. The approaches are analyzed with individual experiments and evaluated by subsequent comparison. Furthermore, we provide theoretical and empirical studies about shortcomings of the deep learning approach. Additionally, techniques to handle voluminous 3D CT data are discussed and applied.

In the following sections, we firstly review related work about traditional and deep learning-based segmentation and deep learning with three-dimensional data. Secondly, we present our liquid segmentation algorithms and explain how we convert dense high-resolution 3D CT scans into sparse yet meaningful point clouds. Thirdly, we showcase experiments and results for both liquid segmentation algorithms individually. Subsequently, we compare them in terms of segmentation performance, inference speed and memory requirements. Lastly, we discuss shortcomings of our deep learning-based approach, supporting our hypotheses with previously showcased experiments.

2 Related Work

2.1 Traditional Image Segmentation

While traditional methods were often used for generic image segmentation, algorithms can be adapted to perform semantic or at least foreground-background segmentation. Although deep learning approaches significantly advanced semantic segmentation, traditional, non-learning techniques can still be useful because they are simple and do not require annotated training data. Gathering enough training data can be incredibly difficult *e.g.* in medical situations [20].

One of the oldest image segmentation techniques, commonly used for x-ray, MRI or CT scans, is the concept of region growing whose basic objective is to join adjacent pixels to form larger, coherent regions. Various approaches were proposed many years ago [2]:

Muerle and Allen [3] divide images into small cells and compute a statistical descriptor, *e.g.* intensity histogram, for each cell. Starting from an initial cell, adjacent cells are appended to form a region if their descriptors are similar enough. Brice and Fennema [4] improve intensity-based results by merging adjacent regions if large portions of the shared boundary separates similar intensities or if the shared boundary makes up a large percentage of the union of both boundaries. Horowitz and Pavlidis [5] iteratively merge and split regions of an arbitrarily initialized segmentation by computing a node cutset of a pyramidal tree data structure with its root being the entire image and its leaves being individual pixels.

Another traditional segmentation technique is the watershed transformation. It treats intensity values as elevations and images as topographic reliefs to separate regions as if these were drainage basins [6].

Clustering is a versatile data processing technique with many different types of algorithms [21], used in various situations, including medical image segmentation. The shared underlying idea of clustering is the division of data into groups of similar entities. These groups are then referred to as clusters. Algorithm-specific rules are used to define similarity.

In medical image segmentation, clustering has been used for various purposes:

Ng *et al.* [7] use *k-means* [22] clustering to preprocess MRI scans. The clustered images then serve as input for their segmentation pipeline which involves edge detection and their improved watershed transformation. Li *et al.* [8] segment medical images in two stages: Firstly, histogram equalization is performed to enhance the images. Secondly, their unsupervised, iterative, fuzzy clustering algorithm minimizes a Euclidean distance-based energy function by assigning labels to every pixel, thereby generating the segmentation. Moriya *et al.* [23] propose an unsupervised feature representation learning algorithm to segment 3D medical images. The algorithm starts by feeding patches of 27 x 27 x 27 voxels to a convolutional neural network. Afterwards, the computed feature representations are extracted and clustered using *k-means* [22]. Finally, the generated clusters serve as supervisory signals to train the neural network.

In Section 3.1, we propose a simple region growing algorithm for liquid segmentation in 3D CT baggage scans. While we do not directly employ clustering, our seed point selection algorithm uses clustering-inspired ideas, such as centroid localization and density awareness.

2.2 Deep Learning-based Semantic Segmentation

Recently, semantic segmentation performance has been improved by a cascade of CNNs:

Long *et al.* [11] repurposed image classification CNNs, such as *VGG* [24] or *GoogLeNet* [25], for semantic segmentation tasks by replacing fully connected layers with convolutional layers, creating fully convolutional networks. CNNs use mechanisms, such as pooling or strided convolution, to aggregate features and to increase the receptive field of neurons. Thereby, the resolution of computed feature maps decreases. A simple way to create a semantic segmentation result that has the same size as the input image is bilinear or bicubic interpolation of the low-resolution output. However, this yields coarse results without precise object boundaries.

Therefore, Long *et al.* [11] introduce skip connections which combine shallow features with upsampled deep features to generate more detailed segmentation results. Deep features enable classification because they represent global structure due to larger receptive fields, while shallow features improve localization and segmentation details due to smaller receptive fields. In this regard, the deep features are upsampled by transposed convolution layers within the network. These layers are initialized to perform bilinear interpolation and learn upsampling patterns during training.

With a relative improvement of 20% compared to the former state-of-the-art in terms of mean IoU on the *PASCAL VOC 2012* challenge [9], Long *et al.* [11] popularized fully convolutional networks for per-pixel tasks. Additionally, skip connections are also used by *PointNet* [1] to address semantic segmentation.

Leveraging the success of Long *et al.* [11], Chen *et al.* [12, 13] introduce *DeepLab*, combining a fully convolutional network with conditional random field postprocessing. Instead of regular convolutions, Chen *et al.* [12, 13] employ dilated convolutions [26] with different rates to explicitly control receptive field sizes and feature scales. This enables *DeepLab* to capture global context information without extensively forfeiting resolution. In contrast to Long *et al.* [11], *DeepLab* computes meaningful class scores at 1/8 of the original image resolution, while the network of Long *et al.* [11] computes coarse results at a scale of 1/32. Therefore, Long *et al.* rely on learned transpose convolution layers, whereas Chen *et al.* [12, 13] use simple bilinear interpolation to upsample to the original image size.

Chen *et al.* [12, 13] postprocess the upsampled CNN output using a fully connected CRF model that defines an energy based on predicted class scores and pairwise terms for each pair of pixels. These pairwise terms contribute to the total energy through a weighted Potts model. The weights are a sum of Gaussian kernels which depend on pixel positions and color intensities. Chen *et al.* [12, 13] emphasize that approximate probabilistic inference of this model can be expressed as convolutions and thus computed efficiently.

Additionally, Chen *et al.* [12, 13] propose a collection of parallel dilated convolutional layers, namely *Atrous Spatial Pyramid Pooling*, to explicitly extract features among various scales. By using different sampling rates for individual layers within the collection, the convolutions obtain distinct field-of-views and therefore distinct scale-sensitivity. The computed features of all parallel layers are combined to create a scale-aware feature vector.

Peng *et al.* [14] propose a convolutional network architecture, called *Global Convolutional Network*, that uses large kernel sizes, almost covering whole feature maps, to increase the valid receptive field, which benefits classification. Symmetric, separable kernels are used to keep computational costs feasible. To improve localization, Peng *et al.* [14] build their network fully convolutional, adopting the concept of Long *et al.* [11]. Boundary alignment is improved by *Boundary Refinement* blocks which learn convolutions to predict residuals of the coarse output.

Recent work in the field of semantic segmentation relies heavily on convolutions for feature extraction and classification. However, the computational costs of convolutions increase cubically for 3D volumes, with respect to the number of voxels, such that application to high-resolution 3D CT scans becomes infeasible. In addition to overwhelming computational costs, dense 3D convolutions are also inefficient in terms of feature extraction from 3D CT baggage scans. Despite being dense volumes, the contained information is sparse due to large amounts of air occupying most of the space.

Therefore, we propose techniques to transform dense 3D CT images into sparse point clouds, while keeping relevant information, such that we can apply efficient algorithms that operate on point clouds (*cf.* Section 3.3).

Very recently, specific semantic segmentation algorithms for 3D point clouds were proposed:

Tchapmi *et al.* [18] pass voxelized 3D point clouds through a 3D fully convolutional neural network to predict coarse semantic labels. To obtain the final segmentation, this coarse result is upsampled using trilinear interpolation and refined with a 3D fully connected CRF, which is implemented as recurrent neural network to make the complete architecture, called *SEGCloud*, end-to-end trainable.

Landrieu and Simonovsky [19] separate the initial point cloud into geometrically homogeneous pieces, called superpoints, via energy minimization using an iterative graph-cut algorithm. *PointNet* [1] is used to compute a descriptor for every superpoint. A superpoint graph is created by connecting superpoints. Two superpoints are connected if their cartesian product contains a pair of points that is connected in the relative neighborhood graph of the initial point cloud. Eventually, a novel graph convolution architecture is used to compute the result.

Although these approaches yield promising results, their methods are too involved and out of scope of this thesis.

2.3 Deep Learning with 3D Data

Common approaches to working with 3D data apply 3D convolutional neural networks on volumetric representations, obtained by voxelization, or typical 2D convolutional neural networks on multiple 2D views of the 3D data [27, 15].

Qi *et al.* [27] propose volumetric CNNs for 3D shape classification with subvolume classification as auxiliary tasks to prevent overfitting or anisotropic probing to create 2D features for a 2D CNN. They are provided with voxelized 3D CAD models at a resolution of 30 x 30 x 30 to match computational costs of 2D CNNs.

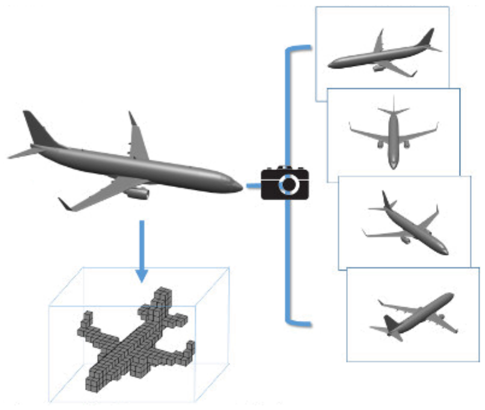


Figure 4: Volumetric representation (left) vs. multi-view rendering (right), visualization from Qi *et al.* [27]

Su *et al.* [15] use multiple 2D projections to represent 3D shapes, motivated by the fact that humans have an astonishing grasp of the 3D world, although only perceiving 2D views of the 3D world. Su *et al.* apply multi-view convolutional neural networks to their 2D projections to extract features and learn feature aggregation from different views into a single 3D shape descriptor.

Qi *et al.* [1] propose an efficient and effective approach to process 3D point clouds. Their deep neural network, called *PointNet*, consumes unordered, raw point clouds and addresses the tasks of 3D object classification, part segmentation and semantic scene segmentation. The novelty of *PointNet* lies in removing the necessity of intermediate 3D data representations, such as voxel occupancy grids or collections of rendered 2D views, thus also removing the loss of information due to quantization or projection. Invariance to input order permutations is obtained by using the symmetric max-pooling operation to aggregate features on a global scale. To become invariant to point cloud transformations, such as rotations and translations, simplified *Spatial Transformer Networks* [28], called *T-Nets*, learn to predict affine transformation matrices which align the input to a canonical space.

Qi *et al.* [16] introduce *PointNet++*, extending *PointNet* by improving the feature aggregation mechanism: The single max-pooling operation is substituted with hierarchical point set feature learning, consisting of multiple *set abstraction* stages. Every stage reduces the number of points but maintains clusters to be robust against non-uniform densities and outliers. In this regard, density adaptive *PointNet* layers are used to extract multi-scale features.

While it seems natural to extend 2D convolutional neural networks to 3D, the approach is limited to low resolutions due to cubically increasing computational costs of 3D convolutions. Additionally, it suffers from high amounts of sparsity and quantization artifacts, in case of point cloud voxelization.

Although our 3D CT baggage scans are inherently voxelized, tremendous computational costs prevent us from directly applying 3D convolutions. Downsampled representations are too coarse by the time that computational costs become feasible. Representing the data with 2D views is also infeasible: Projections create locally overcrowded images which discard valuable intensity information due to occlusions, while cross sections are too inefficient because large subsets of the volumes are empty.

Therefore, we create sparse, yet meaningful point clouds from the 3D CT data (*cf.* Section 3.3). We chose *PointNet* over *PointNet++* to process those point clouds because empricial studies show that performance differences are minor [16]. By using *PointNet*, we keep computational costs feasible and the overall architecture simple.

3 Segmentation and Data Processing

In this section, we propose two different approaches to liquid segmentation in high-resolution 3D CT baggage scans. The first approach combines traditional region growing with automatic seed point selection. The second approach is based on *PointNet* [1] and involves deep learning.

Additionally, we present a sequence of thresholding and sampling to convert overwhelmingly high resoluted 3D CT scans into sparse yet feature-preservering point clouds, such that computational costs become feasible. These point clouds serve as input for both algorithms.

3.1 Automatic Region Growing

Our first segmentation approach employs traditional, non-learning techniques.

In the following section, we explain our region growing algorithm, discussing assumptions and configurations. Afterwards, we propose and discuss an energy function that exploits liquid-specific properties for automatic seed point selection.

3.1.1 Region Growing

The input for our algorithm consists of the image in volume format, a previously selected seed point α and a predefined, instance-independent threshold σ^2. The liquid segmentation is returned as a binary map.

Initially, the seed point is marked as liquid and already checked. Its neighbors are appended to a queue. Then, the following procedure is repeated until the queue is empty: Dequeue a point from the queue. If this point has been checked already, skip it. Otherwise, compute the squared difference of its intensity and the mean intensity of its neighbors. If this difference is smaller than σ^2, classify the point as liquid and enqueue its neighbors.

Firstly, we assume that all parts of the liquid are connected by nearby points that belong to the liquid itself. This property is exploited by the nature of the algorithm because regions can only grow to neighbors of the initial seed or the region itself.

Secondly, we assume intensity continuity within liquids, such that neighboring points within a liquid have similar intensities. Substantial changes in intensity indicate the end of the liquid. This assumption is implemented by the classifier.

The region is grown within the original volume itself, because the algorithm requires the data to be laid out in a grid, such that each entry has defined neighbors. In our case, every voxel is an entry with the upper, lower, left, right, front and back voxels being the neighbors.

A judiciously chosen seed point is required to initialize the algorithm. The result is highly dependent on it because it defines the spatial origin and the range of relevant intensities. It is possible to use multiple seed points to be more robust against noise or to create multiple distinct segments. However, for the sake of simplicity, we are only using a single seed point.

The threshold is used to classify neighboring points as either liquid or background. We selected our threshold based on the intensity variance of liquids throughout the data. The intention is to estimate an upper bound for intensity variation, such that the algorithm can evaluate the changes in intensities and use the threshold to decide if a certain point belongs to the segment.

Although, from a mathematical point of view, the squared Euclidean distance $\|\cdot\|_2^2$ is not a metric, because it does not satisfy the triangle inequality, we used it to penalize intensity variations quadratically instead of *e.g.* linearly, which is justified by the extremely low liquid intensity variances. If the squared difference is lower than σ^2, the change in intensity is assumed to be reasonable and the candidate point will be included in the segment.

After analyzing the ground truth data, we chose our threshold as the 95th percentile of the liquid intensity variance distribution, which yields $\sigma^2 = 0.0275$, where the total range of intensities is $[0, 1]$. We avoid using the maximum variance or higher percentiles, instead of the 95th percentile, to be more robust against noise and imperfect annotations.

	Intensity variance
Maximum	0.0483
Minimum	0.0004
Mean	0.0074
95th percentile	**0.0275**

Table 1: Liquid intensity variances

Algorithm 1: Region growing

```
1:  function grow_region(Image, $\alpha$; $\sigma^2$):
2:      seg = zeros_like(Image); seg[$\alpha$] = 1
3:      checked = zeros_like(Image); checked[$\alpha$] = 1
4:      queue = neighbors($\alpha$)
5:      while not empty(queue):
6:          p = dequeue(queue)
7:          if checked[p] == 1:
8:              continue
9:          if (Image[p] – mean_intensity(neighbors(p))$^2$) < $\sigma^2$:
10:             seg[p] = 1
11:             enqueue(queue, neighbors(p))
12:     return seg
```

3.1.2 Seed Point Selection

Our region growing algorithm requires a previously selected seed point that is within the region that we want to grow. Therefore, in our case, the seed point should be a liquid voxel. Selecting such a seed point usually requires human interaction, but we propose a novel energy function whose argument of the minimum is a suitable seed point for liquid detection.

In Equation 1, α defines the liquid seed point, given a point cloud and parameters κ and λ.

$$\alpha(S;\kappa,\lambda) = \underset{p}{\operatorname{argmin}}[\lambda D(S,p;\kappa) + (1-\lambda)V(S,p;\kappa)] \tag{1}$$

In this respect, S denotes a point cloud sample, taken from the thresholded volume, while p denotes a single 4D point (x,y,z,i) from that sample. D and V are energy functions: D computes Euclidean distances between p and its neighbors, V estimates the local intensity variance. κ defines the number of neighbors used for the local estimations, λ trades off between D and V.

From a high-level point of view, D represents the spatial coherence of liquids. Usually stored in bottles or jar-like containers, liquids generate clusters and bulks of points when being scanned by a CT machine. By minimizing D, we maximize the local point cloud density which ensures that the surrounding space is dense. Dense clusters of points are more likely to be part of a liquid because thresholding keeps liquid clusters intact, while removing or sparsening non-liquid objects.

Cooperatively, V models the intensity continuity of liquids. Since the measured intensities correspond to the liquid's density, intensity values of points belonging to the same liquid are constant if we disregard noise and assume a consistent temperature and pressure during the scan. Therefore, minimizing V, *i.e.* the local intensity variance, increases the likelihood that a point belongs to a liquid because non-liquid objects typically do not have continuous densities with low variances.

To compute the seed point, the following procedure is repeated for every point in the sample:

Firstly, the Euclidean distances between a point p and all other points from the sample S are computed. Instead of computing the distances for every individual point, it is possible to avoid redundancy by calculating the triangular distance matrix once and using it as a lookup table.

Secondly, the κ closest neighbors are selected by sorting the distances.

Thirdly, the energy terms D and V are computed for every point: $D(S,p;\kappa)$ is defined as the sum of the Euclidean distances between point p and its κ closest neighbors. $V(S,p;\kappa)$ is defined as the intensity variance of the κ closest neighbors. Both terms are normalized to [0, 1] to ensure numerical stability and comparability.

Lastly, the combined energy is computed by trading off between D and V using λ. The point that minimizes this energy is returned as the selected seed point α.

Algorithm 2: Seed point selection

```
1:  function select_seed_point($S$; $\kappa$, $\lambda$):
2:      for p in $S$:
3:          distances = euclidean_distances($S$, p)
4:          sorted_indexes = argsort(distances)
5:          neighbors = sorted_indexes[0:$\kappa$]
6:          $D$[p] = sum(distances[neighbors])
7:          $V$[p] = intensity_variance($S$[neighbors])
8:      normalize($D$); normalize($V$);
9:      energy = $\lambda D + (1 - \lambda) V$
10:     $\alpha$ = argmin(energy)
11:     return $\alpha$
```

Choosing κ too low makes the local estimations imprecise due to the randomness of point cloud sampling. Choosing κ too high causes imprecision because the estimations start to incorporate non-liquid voxels. Therefore, we analyzed how many closest neighbors a liquid voxel has before the next closest neighbor becomes the first non-liquid voxel.

N_{sample}	10,000	25,000	50,000
Mean	394.74	904.45	1705.03
Minimum	36	70	133
1st percentile	43.93	85.96	160.78
5th percentile	62.80	114.95	191.90

Table 2: Average number of liquid neighbors before the next closest neighbor becomes non-liquid

We chose κ corresponding to the lower end of this distribution, such that most of the time all κ neighbors of a liquid voxel, which are considered by the energy function, are also liquid voxels.

For our experiments we used 25, 50, 100 and 200 for κ, in combination with different sample sizes, and 0, 0.25, 0.5, 0.75 and 1 for λ to investigate different weightings of D and V.

3.2 PointNet

The unmodified *PointNet* [1] consumes an arbitrary number of 3D points, each represented by three Euclidean coordinates, to predict either a global class label or pixel-wise scores for segmentation tasks. All n input points are processed by a cascade of shared MLPs to extract features of size n x 1024. Using the max pooling operation, these extracted features are aggregated as a global feature vector of constant length 1024. This global feature vector is then used by another shared MLP to compute the output classification scores for different classes. For segmentation tasks, the global feature vector is combined with earlier features, creating a matrix of size n x 1088. This matrix is used by shared MLPs to compute the segmentation. Combining features from different depths benefits semantic segmentation [11], because deep features allow classification and shallow features improve localization.

Because the input point clouds are a subset from Euclidean space, Qi *et al.* [1] emphasize that *PointNet* must deal with three properties of such point clouds:

Firstly, the neural network must treat point clouds as unordered, *i.e.* be invariant to all data ordering permutations. *PointNet* meets this requirement by using multilayer perceptrons as identical and independent feature extracting operations and max pooling, involving the symmetric maximum function, to aggregate features.

Secondly, the feature extractor and the classifier must be invariant to transformations, such as rotations and translations. To achieve this invariance, simplified versions of *Spatial Transformer Networks* [28], namely *T-Nets*, are employed to predict affine transformation matrices which align point clouds to a canonical space, thus negating the effects of possible prior rotations and translations. Using this technique, all input points and extracted intermediate features are transformed.

Lastly, Euclidean space has a defined distance metric. Therefore, coordinates (x, y, z) of different points should not be treated as independent variables. Instead, multiple points with low Euclidean distances to eachother, *i.e.* local clusters of points, usually represent meaningful subsets, *e.g.* segments or objects, within the whole point cloud. Although Qi *et al.* [1] mention this property, *PointNet* does not exploit it.

Using the segmentation mode of the unmodified *PointNet* [1] as a baseline, we incrementally introduce novel extensions to adapt *PointNet* to our task:

For our first extension, we append a fourth input dimension to *PointNet* to incorporate the measured intensity values. Thereby, the input-transforming *T-Net* is modified to predict a 4x4 matrix instead of a 3x3 matrix and the first shared MLP is modified to accept an input shape of nx4 instead of nx3. We call the resulting architecture *PointNet 4D* (*PN4D*).

Our second extension introduces loss weighting to increase the loss contribution of liquid voxels during training. We observed that the training process is disturbed by an imbalance between liquid and background instances, *i.e.* the network prefers to predict background labels because it sees background voxels significantly more often than liquid voxels. This imposes problems when combined with the cross entropy loss function, which is commonly used for classification and segmentation tasks:

$$L_i(pred_i, gt_i) = \begin{cases} -\log(1 - pred_i)\,, & gt_i = 0 \\ -\log(pred_i) \quad , & gt_i = 1 \end{cases} \tag{2}$$

The sum of this binary cross entropy loss function, in which every point contributes equally to the sum, is minimized by predicting the correct label for as many points as possible. Ignoring this imbalance promotes a lazy network that prefers trivial solutions: By predicting 0, *i.e.* background, for every point, a lazy network achieves a small total loss value because most of the points are indeed background and only a minor number of points are liquid.

We address this problem by weighting the loss, such that the few liquid occurances have a more significant contribution to the overall loss and are not overwhelmed by the vast amount of background points. Therefore, we introduce hyperparameters γ_c, where c denotes the class label, to scale the loss values of individual points within a single prediction. The parameters γ_c are chosen antiproportional to the frequency of the corresponding class c, such that the relative per-class loss contribution is distributed more evenly among classes.

The weighted binary cross entropy loss for a single prediction becomes:

$$\hat{L}(pred, gt) = \frac{1}{N_{sample}} \sum_{i=0}^{N_{sample}} \gamma_{gt_i} * L_i(pred_i, gt_i) \tag{3}$$

This method is extendible to multiple tasks and classes by adding appropriate parameters for every additional class. For example, Eigen *et al.* [29] used this technique to jointly predict depth, surface normals and semantic labels.

Our third extension intends to counteract overfitting. We apply data augmentation techniques in form of random rotations. In particular, we distinguish between random 3D rotation and random azimuth rotation: The former rotates an object in 3D space, whereas the latter rotates an object in the 2D plane that is orthogonal to the direction of gravity. For example, if the z-axis denotes the direction of gravity and we rotate a vector that is pointing in the direction of the x-axis, a random 3D rotation results in a vector with arbitrary direction in 3D space, while a random azimuth rotation confines the vector to a random direction within the xy-plane. Liquids that are stored in bottles and affected by gravity assume irregular shapes with a plane orthogonal to the direction of gravity, such that azimuth rotations keep that plane parallel whereas 3D rotations do not.

Lastly, we observed that large sample sizes perform worse than expected and hypothesize that insufficient feature space and feature learning are the cause. Therefore, we insert 3 additional shared MLP(64, 64) after the feature transform to increase the number of parameters and the network's depth. Increasing the number of parameters extends the network's feature space, while increasing the network's depth improves feature learning [24, 25, 30].

Further discussions and analyses about these extensions are conducted in Section 5.

Experimental setup and results for all incremental extensions are presented in Section 4.2.

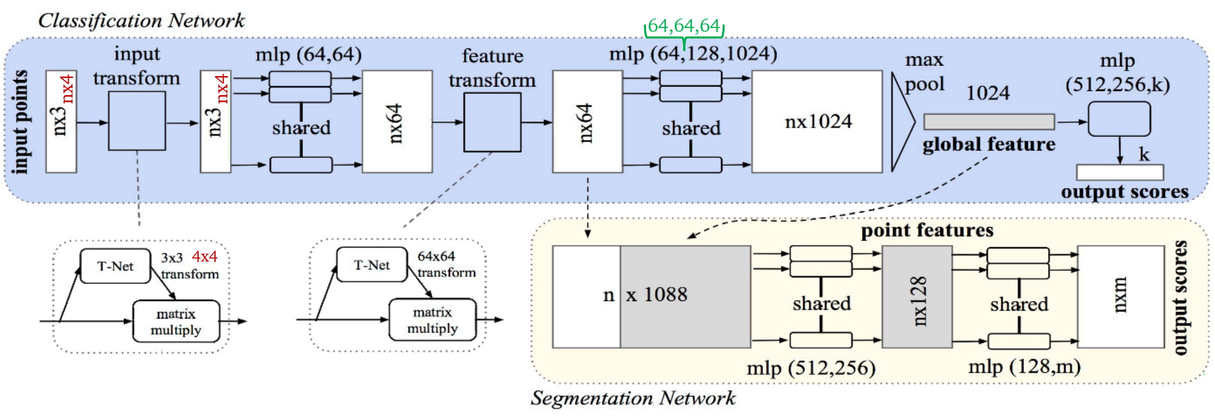


Figure 5: The *PointNet* architecture introduced by Qi *et al.* [1]

(*PointNet 4D* alterations in red, additional layers in green)

3.3 Creating Sparse Point Clouds from Dense 3D CT Volumes

The main issue with processing dense, high-resolution 3D CT scans is the overwhelming number of voxels per image. Popular computer vision tools, particularly convolutional neural networks, cannot handle such images efficiently due to memory limits and immense computational costs.

A popular solution is to downsample the data before processing to match available resources. However, for our 3D CT scans, the number of voxels would have to be reduced by a factor of *ca.* 1000 to match common input sizes. Therefore, downsampling is infeasible because essential image contents and features would diminish significantly and eventually vanish.

We propose a two-step process, namely thresholding and sampling, to create sparse point clouds from dense, high-resolution 3D CT scans while preserving relevant image contents, discarding irrelevant information and offering the flexibility of choosing the resulting point cloud size as a trade-off between resource requirements and representativeness.

These point clouds serve as computationally feasible inputs for our segmentation algorithms.

3.3.1 Thresholding

We applied a threshold to the data by utilizing the measured intensity values, exploiting the property that liquids have similar intensities. Thereby, the 3D CT baggage scan is trimmed to a relevant intensity range, while irrelevant components, such as air and clothes, are discarded.

Therefore, we defined an interval $[\rho_{min}, \rho_{max}]$, that captures the majority of measured liquid intensities. By removing all data points with intensity values outside of this interval, the dense voxelgrid is transformed into a sparse point cloud for liquid segmentation.

After analyzing the training data, we obtained the 1st and 99th percentile of the liquid intensity distribution. Therefore, we cover 98% of the training data by using the following thresholds.

$$[\rho_{min} = 1700, \rho_{max} = 2700] \tag{4}$$

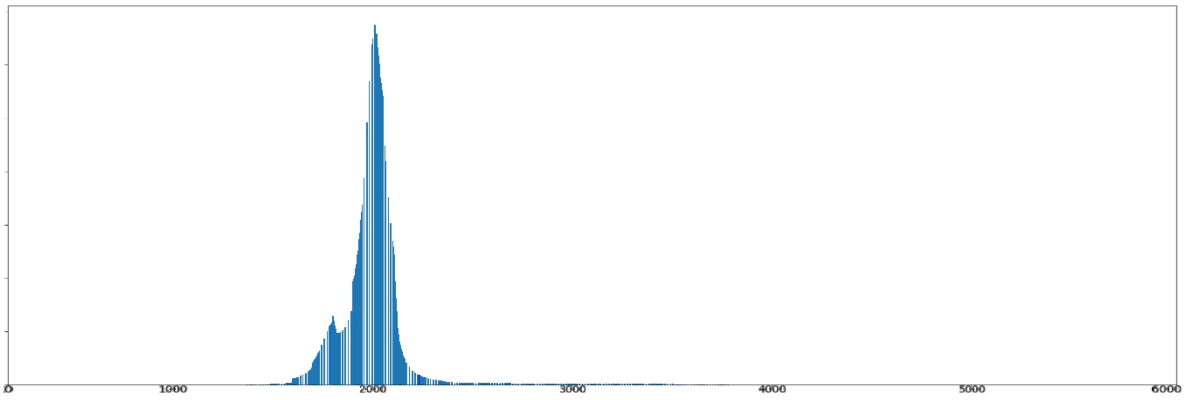


Figure 6: Histogram plot of the training data's liquid intensity distribution

The absolute intensity values correspond to the densities of scanned objects. However, they do not represent the SI unit of density but rather a proprietary density unit. Additionally, we normalize all intensities within this interval to $[0, 1]$ after thresholding to ensure numerical stability during computation.

Removing all voxels with intensities outside of this interval reduces the number of data points per CT image by a substantial amount. Fortunately, we thereby lose almost no relevant information because the removed voxels are mostly air. But unfortunately, the remaining data is still very voluminous and therefore cannot be processed efficiently due to limitations in memory size and computational power.

Average number of voxels per image	Before thresholding	After thresholding	Per annotation
Absolute	188,173,584	4,288,160	120,984
Relative	4388.21%	100%	2.82%

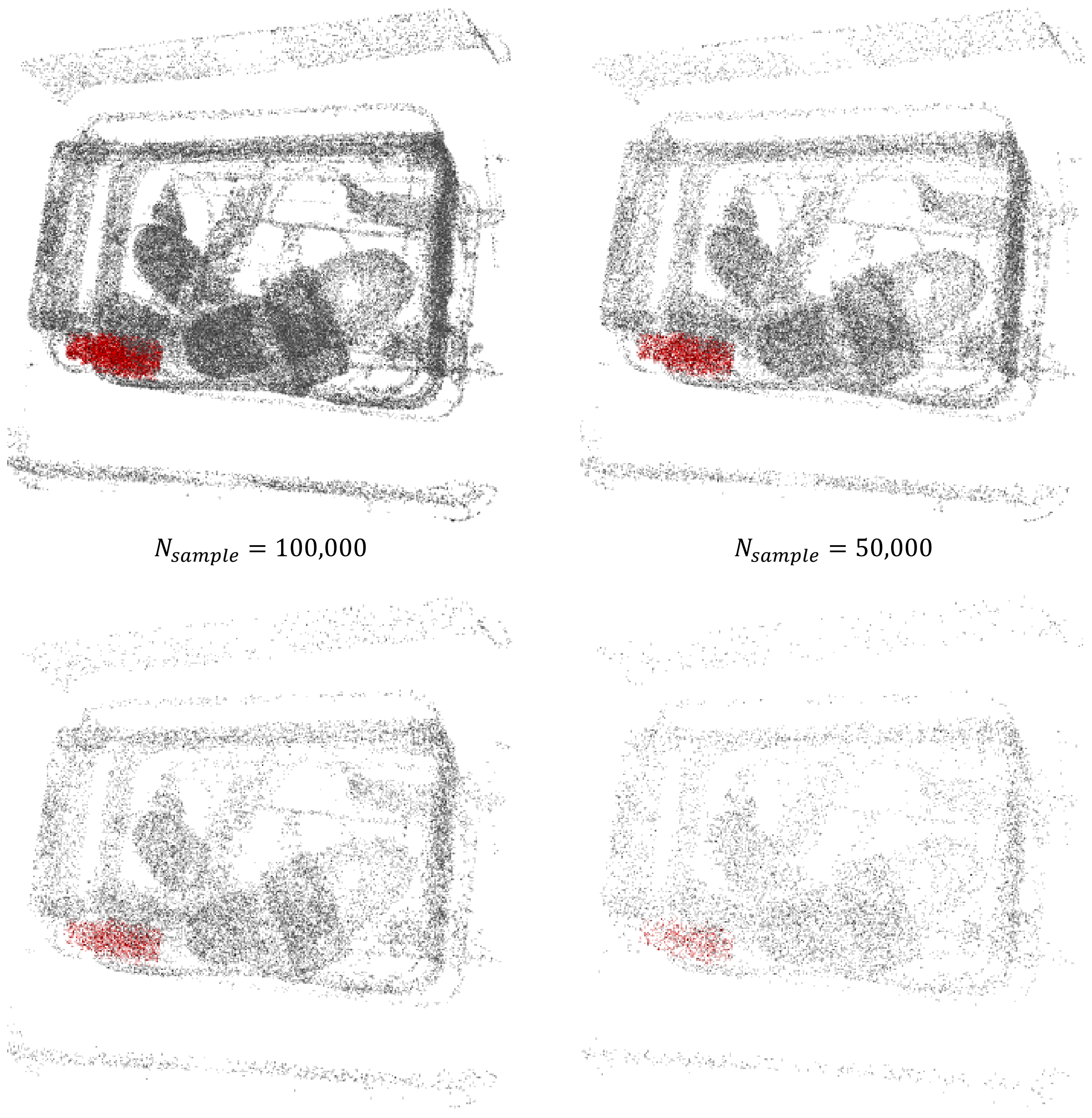
Table 3: Average voxel numbers of liquid annotations and CT images, before and after thresholding

3.3.2 Sampling

To further reduce computational costs and memory requirements, we reduce the number of voxels by representing a thresholded CT image with a sample of N_{sample} points. Samples are taken uniformly at random after thresholding, such that only relevant data is sampled.

Figure 7 illustrates different sample sizes. The samples serve as input for both *PointNet 4D* and the automatic seed point selection algorithm. To analyze benefits and drawbacks of smaller and larger sample sizes, we chose sample sizes of 10,000, 25,000 and 50,000 for our experiments.

Although Qi *et al.* [1] emphasize that *PointNet* can process arbitrarily sized inputs, the input dimension is invariable for a single training instance. In other words, we can choose our sample size N_{sample}, but we cannot change it depending on the current CT image or inbetween training iterations because choosing a different input size N_{sample} changes the dimensions of the first network layer, such that its weights must be retrained. Therefore, all consecutive layers, *i.e.* the whole network, must be retrained because they depend on the outputs of the first layer.



$N_{sample} = 100{,}000$ $N_{sample} = 50{,}000$

$N_{sample} = 25{,}00$ $N_{sample} = 10{,}000$

Figure 7: Visualization of different point cloud sample sizes

4 Individual Experiments and Comparison

To analyze the segmentation performance of the proposed liquid segmentation approaches, we first present and discuss individual experiments using different parameters and configurations. Subsequently, we compare the best results of both approaches. In particular, we compare the effects of different sample sizes in terms of segmentation performances, inference speed and memory requirements to identify strengths and weaknesses of both approaches.

The input for both algorithms is a sparse point cloud sample taken from the original volume using our proposed thresholding and sampling techniques. Both algorithms return the liquid segmentation as a binary mask, classifying every point within the point cloud sample.

All segmentations are evaluated using the mean Intersection over Union score, which is also known as the Jaccard index. The score for a single prediction is computed by comparing the predicted binary mask with the ground truth annotation and dividing the number of correctly predicted liquid voxels (intersection) by the number of liquid voxels which are either predicted or annotated (union). The final score is obtained by taking the mean over all individual scores, like the name implicates.

We divided a total of 712 images into a training set of 512 and a testing set of 200 images. For every CT image, there is one corresponding liquid annotation given as a binary mask that labels every voxel as either 1 for liquid or 0 for background.

For all experiments, we used a *Skylake i7* CPU and a *GeForce GTX Titan X* GPU.

4.1 Automatic Region Growing

We tested *Automatic Region Growing* with different parameter configurations and sample sizes. Although *Automatic Region Growing* does not involve deep learning, we kept the training data and testing data separated to compare with *PointNet 4D* results. Ground truth annotations were only used for statistical analyses and evaluation purposes.

Parameter configurations are investigated by applying grid searches with $\kappa \in \{25, 50, 100, 200\}$ and $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$. A full grid search is applied separately for training and testing data and for all sample sizes, *i.e.* $N_{sample} \in \{10{,}000; 25{,}000; 50{,}000\}$.

First, we compute a suitable seed point using the input point cloud and parameters. This seed point is then used to grow the liquid region. Notably, the sample size only influences the automatic seed point selection directly. The region growing algorithm receives the original volume, selected seed point and parameters as input.

	Automatic Region Growing Training Data, $N_{sample} = 10{,}000$				
	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1$
$\kappa = 25$	24.03%	55.02%	56.79%	56.66%	42.81%
$\kappa = 50$	22.72%	54.90%	56.63%	**57.12%**	45.12%
$\kappa = 100$	27.24%	52.00%	55.15%	55.97%	47.35%
$\kappa = 200$	26.12%	47.06%	51.90%	54.28%	48.03%

Table 4: *Automatic Region Growing* results on the training data with $N_{sample} = 10{,}000$

	Automatic Region Growing Training Data, $N_{sample} = 25{,}000$				
	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1$
$\kappa = 25$	26.59%	55.81%	57.19%	57.51%	37.83%
$\kappa = 50$	26.33%	54.89%	57.36%	**57.69%**	43.71%
$\kappa = 100$	23.77%	53.37%	55.75%	56.45%	48.35%
$\kappa = 200$	27.36%	51.01%	54.87%	55.85%	50.46%

Table 5: *Automatic Region Growing* results on the training data with $N_{sample} = 25{,}000$

	Automatic Region Growing Training Data, $N_{sample} = 50{,}000$				
	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1$
$\kappa = 25$	26.88%	55.76%	57.01%	56.18%	34.33%
$\kappa = 50$	26.10%	54.53%	56.70%	**57.52%**	42.30%
$\kappa = 100$	25.75%	54.30%	56.93%	57.49%	47.96%
$\kappa = 200$	23.24%	54.07%	56.16%	57.32%	49.20%

Table 6: *Automatic Region Growing* results on the training data with $N_{sample} = 50{,}000$

Automatic Region Growing Testing Data, $N_{sample} = 10{,}000$					
	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1$
$\kappa = 25$	22.88%	54.16%	55.91%	57.35%	44.44%
$\kappa = 50$	22.85%	52.92%	55.41%	**57.77%**	47.06%
$\kappa = 100$	27.87%	54.14%	55.81%	55.62%	46.30%
$\kappa = 200$	32.94%	48.94%	52.38%	53.51%	47.78%

Table 7: *Automatic Region Growing* results on the testing data with $N_{sample} = 10{,}000$

Automatic Region Growing Testing Data, $N_{sample} = 25{,}000$					
	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1$
$\kappa = 25$	24.90%	56.16%	57.12%	57.92%	36.87%
$\kappa = 50$	24.17%	56.02%	57.58%	**58.31%**	47.33%
$\kappa = 100$	22.91%	55.08%	55.51%	56.57%	50.14%
$\kappa = 200$	27.67%	52.98%	55.56%	55.86%	52.00%

Table 8: *Automatic Region Growing* results on the testing data with $N_{sample} = 25{,}000$

Automatic Region Growing Testing Data, $N_{sample} = 50{,}000$					
	$\lambda = 0$	$\lambda = 0.25$	$\lambda = 0.5$	$\lambda = 0.75$	$\lambda = 1$
$\kappa = 25$	23.85%	54.09%	56.50%	58.31%	33.62%
$\kappa = 50$	22.36%	56.09%	57.83%	**59.23%**	41.03%
$\kappa = 100$	24.13%	54.39%	56.67%	57.64%	47.00%
$\kappa = 200$	23.30%	53.75%	55.43%	56.57%	48.88%

Table 9: *Automatic Region Growing* results on the testing data with $N_{sample} = 50{,}000$

Our experiments clearly show that solely using D ($\lambda = 1$) or V ($\lambda = 0$) to compute a seed point is insufficient. Combining both energies significantly increases the segmentation performance. Experiments with different ratios show that it is favorable to take D more into account than V. However, performance differences are not as substantial, if compared to sole usage of D or V. The best ratio yielded a relative performance increase of *ca.* 10%, compared to the worst ratio.

Additionally, our results also illustrate that there is a correlation between segmentation performance and κ, the considered number of closest voxels. In most cases, increasing κ beyond 50 worsened the performance among all sample sizes. However, smaller sample sizes lost more performance. This observation is reasonable because we chose κ by analyzing the number of closest liquid voxels, which will be smaller for small sample sizes. Therefore, increasing κ will eventually force the seed point selection to take non-liquid voxels into account, resulting in performance drops. Choosing $\kappa = 25$ performs slightly worse than $\kappa = 50$ which is expectable because less information is used.

The sample sizes are only slightly influencing the segmentation performances because only the seed point selection utilizes the sampled representations. However, inference speed and memory requirements are strongly affected by the sample size.

The best parameter configuration among our grid search is $\kappa = 50, \lambda = 0.75$. The corresponding results will be used for comparison in Section 4.3.

Automatic Region Growing			
N_{sample}	10,000	25,000	50,000
Runtime	3.71 s	12.15 s	28.85 s
Memory	$\sim$ 2750 MiB	$\sim$ 2750 MiB	$\sim$ 2750 MiB

Table 10: Inference speed and memory requirements of *Automatic Region Growing*

Although GPU parallelization is used for seed point selection, *Automatic Region Growing* is extremely slow, due to the sequential nature of region growing. Even for our smallest sample size of 10,000 points several seconds are required to compute the segmentation. Computing the segmentation for large sample sizes of 50,000 points takes up to half a minute.

Automatic Region Growing requires a convenient amount of memory among all sample sizes. The region growing step accounts for most of the memory requirements because it uses the original volumes. Therefore, all sample sizes require the same amount of memory.

4.2 PointNet

We used the officially recommended third-party *PyTorch* implementation [31] of *PointNet* [1] for our experiments. Since Qi *et al.* [1] do not mention optimizer configurations, we used the settings provided by their official implementation: 250 training epochs of *Adam* [32] with default configuration for β_1, β_2 and ε, learning rate 0.001 and exponential learning rate decay. Due to the high memory requirements of the data, we used a minibatch size of only 2.

PointNet denotes the unmodified segmentation setup that only processes Euclidean coordinates.

PointNet 4D incorporates intensity values in the input: First, we create 3D point cloud samples. Afterwards, we retrieve the corresponding intensities from the volume using the sampled coordinates. Lastly, we concatenate these intensity values to the coordinates to create 4D inputs.

"+ loss weighting" signals that we increased the loss contribution of liquid voxels by multiplying their loss values with a weight. For our experiments, we chose a loss contribution ratio of 1:3 respectively for background and liquid voxels.

"+ 3D rotation" indicates that we applied data augmentation in form of random rotations. We used the Fast Random Rotation Matrices [33] algorithm to compute uniformly distributed three-dimensional rotations.

"+ azimuth rotation" also implies data augmentation via random rotations. Here however, we rotated about the vertical axis by choosing angles $\theta \in [0, 2\pi)$ uniformly at random.

"+ additional layers" denotes the three additional shared MLP(64, 64). They extend the feature space by adding more parameters and improve feature learning by deepening the network.

Segmentation performances are displayed in Table 11 on the next page.

Our experiments show that the unmodified *PointNet* performs poorly. Including intensity values with *PointNet 4D* significantly increases segmentation performances. Introducing loss weighting during training further improves the results.

At this stage, large differences between training and testing performances indicate overfitting. Our data augmentation techniques alleviate this issue. Notably, azimuth rotations yield better performances than full 3D rotations. However, the overall performance is reduced.

Up to this point, sample sizes of 25,000 points perform reasonably better than sample sizes of 10,000 points, while sample sizes of 50,000 points perform unexpectedly worse than both smaller sample sizes. Experiments with additional layers result in increased performance for sample sizes of 50,000 points.

PointNet

N_{sample}	10,000		25,000		50,000	
Data	Train	Test	Train	Test	Train	Test
PointNet	14.45%	12.03%	21.97%	17.77%	11.97%	10.37%
PointNet 4D	32.02%	22.78%	38.79%	25.58%	20.80%	16.17%
PointNet 4D + loss weighting	**47.32%**	**29.95%**	**57.76%**	**31.85%**	**46.84%**	28.58%
PointNet 4D + loss weighting + 3D rotation	15.93%	14.73%	14.48%	14.58%	15.69%	16.36%
PointNet 4D + loss weighting + azimuth rotation	24.77%	22.70%	28.73%	25.04%	27.26%	23.91%
PointNet 4D + loss weighting + additional layers	41.63%	29.53%	45.61%	30.85%	41.84%	**30.58%**

Table 11: *PointNet* results for different sample sizes and configurations

PointNet

N_{sample}	10,000	25,000	50,000
Runtime	0.16 s	0.42 s	0.97 s
Memory	~ 2150 MiB	~ 4500 MiB	~ 8500 MiB

Table 12: Inference speed and memory requirements of *PointNet*

In terms of inference speed, *PointNet* is extremely fast due to the efficient neural network architecture and massive computational parallelism. Even for our largest sample size of 50,000 points, computing the segmentation only takes a single second. For smaller sample sizes, it is even faster.

PointNet requires substantial amounts of memory to build the network architecture and store all necessary parameters. Therefore, our thresholding and sampling techniques are necessary to create sparse representations of the original volumes, such that memory capacities of modern GPUs suffice. For dense volumes, memory requirements would be far beyond limits.

4.3 Comparison and Evaluation

In terms of segmentation performances, *Automatic Region Growing* beats *PointNet 4D* by a large margin, achieving almost twice as high mean IoU. Surprisingly, performances are similar among sample sizes for both approaches. Increasing the sample sizes yield relative mean IoU increases of only 2.79% for *ARG* and 6.34% for *PN4D*.

In terms of inference speed, *PointNet 4D* dominates *Automatic Region Growing*. Even our largest samples are processed by *PN4D* in less than a second, while *ARG* takes more than three seconds to process a small sample. Therefore, *ARG* is infeasible if fast inference speed is required, *e.g.* for airport security checks, whereas *PN4D* would be fast enough.

Automatic Region Growing requires similar amounts of memory among sample sizes, while *PointNet 4D* memory requirements increase significantly as the sample size increases. However, this can be disregarded if larger sample sizes do not substantially improve mean IoU.

Visual results illustrate that *ARG* predictions are sometimes wrong, but overall human-like and reasonable, while *PN4D* predictions are often scattered and incomprehensible (*cf.* Figure 8).

Comparison of both approaches			
N_{sample}	10,000	25,000	50,000
Mean IoU			
Automatic Region Growing	**57.62%**	**58.31%**	**59.23%**
PointNet 4D + loss weighting	29.95%	31.85%	28.58%
Runtime			
Automatic Region Growing	3.71 s	12.15 s	28.85 s
PointNet 4D + loss weighting	**0.16 s**	**0.42 s**	**0.97 s**
Memory			
Automatic Region Growing	∼ 2750 MiB	**∼ 2750 MiB**	**∼ 2750 MiB**
PointNet 4D + loss weighting	**∼ 2150 MiB**	∼ 4500 MiB	∼ 8500 MiB

Table 13: Comparison of *PointNet* and *Automatic Region Growing* performances

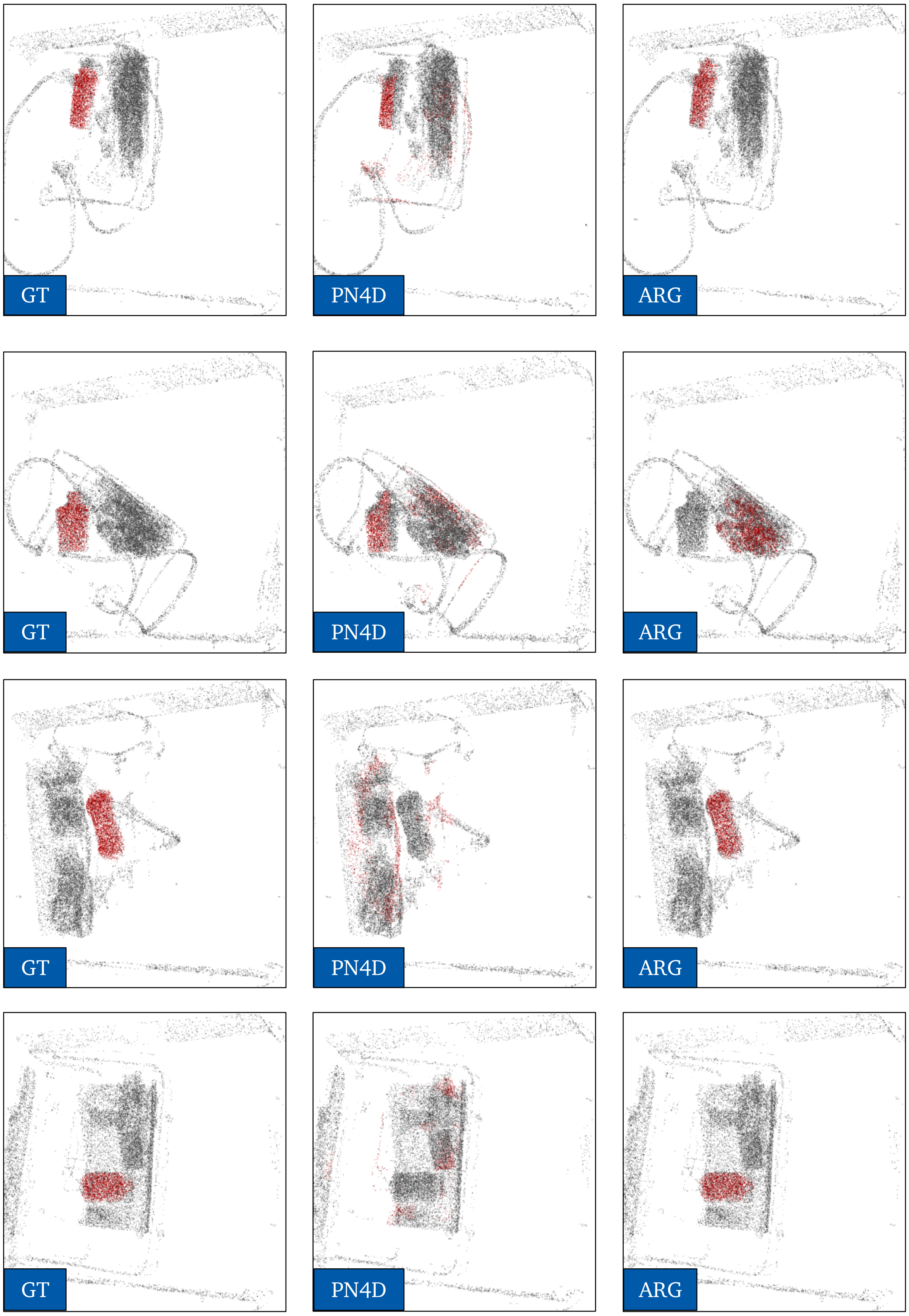


Figure 8: Ground truth (left) *vs. PointNet 4D* (middle) *vs. Automatic Region Growing* (right)

5 Analysis and Discussion of PointNet Shortcomings

In this section, we analyze shortcomings of our deep learning-based approach by presenting hypotheses, theoretical discussions and empirical studies. In particular, we examine the effects of liquid-background class imbalance, scene complexity of 3D data and gravitational influence on liquids and their shapes. Furthermore, we discuss why our largest sample size performs worst and why annotated 3D CT data is scarce but necessary for deep learning.

5.1 Liquid-Background Imbalance

To provide empirical evidence for our claim that class imbalance is a disturbing factor during training of our neural network and to find out which ratio is appropriate for our task, we experimented with different loss weighting ratios. We compare the performance in terms of training mean IoU, because, in this case, we are dealing with an optimization problem rather than a generalization problem.

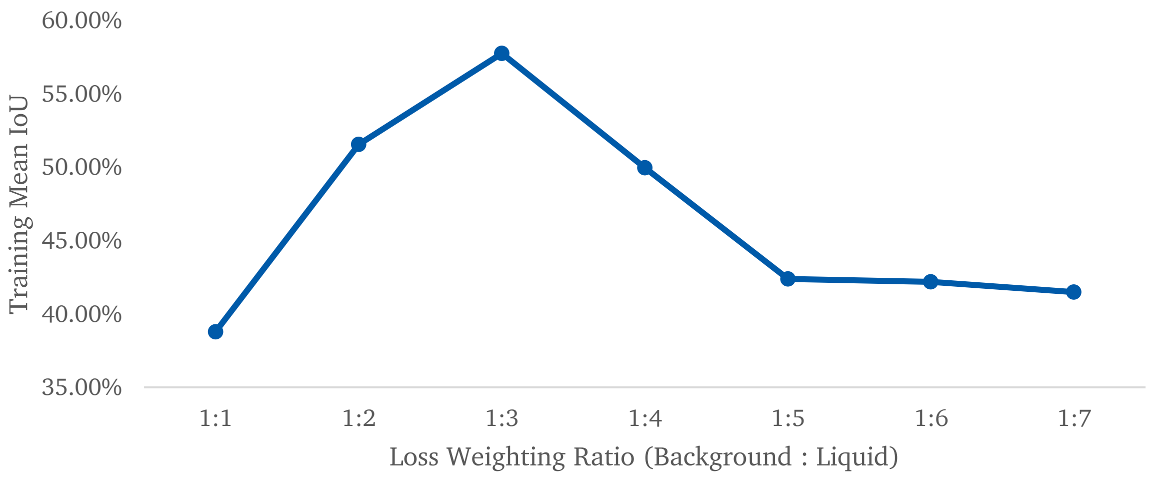


Figure 9: Weighting class specific loss improves the performance

Our experiments show that 1:1 weighting, *i.e.* no loss weighting, performs worst. Increasing the loss contribution of liquid voxels increases the network's performance until ratios become too high. We observed that ratios of 1:5 and higher cause unstable training. In these cases, we trained the network multiple times and report the average mean IoU among training instances.

Based on our results, we chose $\gamma_0 = 1$ and $\gamma_1 = 3$, where 0 represents the background class and 1 represents the liquid class, to increase the loss contribution of liquid points.

Considering the actual class distributions (*cf.* Table 3), the exact liquid frequency might suggest significantly higher ratios, *e.g.* $\gamma_0 = 1$ and $\gamma_1 = \frac{1}{0.0282} \approx 35$. However, when using such values, we observed that the network becomes lazy, predicting liquid for the whole sample.

5.2 Scene Complexity of 3D Data

Qi *et al.* [1] provide empirical results showing that *PointNet*, despite being simple, performs well on 3D object classification, part segmentation and semantic scene segmentation.

To train their network for object classification, they used the *ModelNet40* dataset [34], which is a collection of 3D CAD models of different object categories. Points are sampled randomly from these models to create point clouds for training and testing.

For part segmentation, they used the *ShapeNet* part data set [35], consisting of 3D CAD models with labelled object parts, *e.g.* wings, body, tail and engine in case of an airplane. Again, training and testing inputs are generated by random sampling.

The semantic segmentation has been trained and evaluated using the *Stanford 3D Semantic Parsing* dataset [36]. It contains 3D point cloud scans of different rooms and from various points of view. At training time, points are sampled, whereas the whole image is used at testing time.

Notably, the number of points used by Qi *et al.* [1] to represent a single 3D image is much lower than the number of points we must use to reliably represent a single 3D CT scan. Our smallest sample size of 10,000 points barely creates discriminative representations.

	Sample size of representations
ModelNet40 [34]	1024
ShapeNet [35]	1024
Stanford 3D [36]	4,096
Smiths Detection (ours)	10,000+

Table 14: Representing *Smiths Detection*'s 3D CT images requires larger sample sizes

Small sample sizes of 1,024 suffice to represent *ModelNet40* [34] and *ShapeNet* [35] images because they only depict single, isolated objects without background. Due to being 3D CAD models, the data is also noise free. However, Qi *et al.* [1] add small amounts of Gaussian noise during training to improve robustness.

Additionally, the points are sampled from the surfaces of the 3D models, such that the samples become more descriptive in terms of object shapes. This sampling strategy can be interpreted as simulating a laser depth scanner.

The *Stanford 3D Semantic Parsing* dataset [36], used by Qi *et al.* [1] for semantic segmentation, features real office and class room scenes instead of isolated objects. However, the diversity and complexity of these room scenes is limited in a sense that ordinary rooms have a common layout. They usually have planar and orthogonal walls, chairs and desks stand on the ground, blackboards and windows are on the walls and lamps are on the ceiling.

With our 3D CT data, we cannot create samples based on mesh surfaces because we do not know where the surfaces are. Furthermore, our data is not noise free and single scans do not depict isolated objects. Instead, most scans contain tightly packed diverse baggage contents, the baggage itself and background, such as plastic containers.

Additionally, there is no fixed set of objects or common layout like in offices or class rooms. Diverse baggage contents can be arranged arbitrarily inside of various kinds of baggages, such as bags, suitcases, briefcases or backpacks.

Therefore, 3D CT data is much more complex than the data used by Qi *et al.* [1] to train and evaluate *PointNet*. We believe that this complexity causes the disappointing performance.

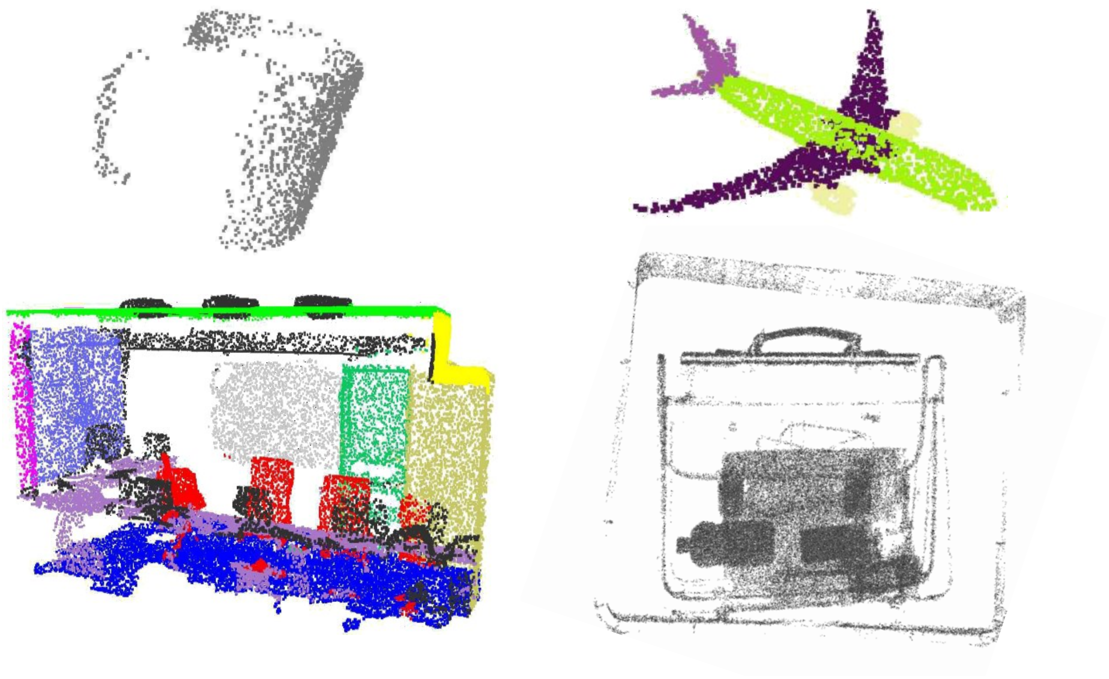


Figure 10: Visual comparison of our 3D CT scans (bottom right) and 3D data used by Qi *et al.* [1]: *ModelNet40* [34] (top left), *ShapeNet* [35] (top right), *Stanford 3D* [36] (bottom left)

To support our hypothesis, we divided the set of 200 testing images into three subsets of different scene complexities and applied our algorithm separately on each individual subset to evaluate and compare the performance among varying scene complexities.

In this regard, we determined the complexity of every scene by means of human perception, *i.e.* we looked at each image and decided whether it is *easy*, *medium* or *difficult* for us to tell where the liquid is, judging based upon the crowdedness of the scene and the amount of confusion caused by multiple liquid-like objects.

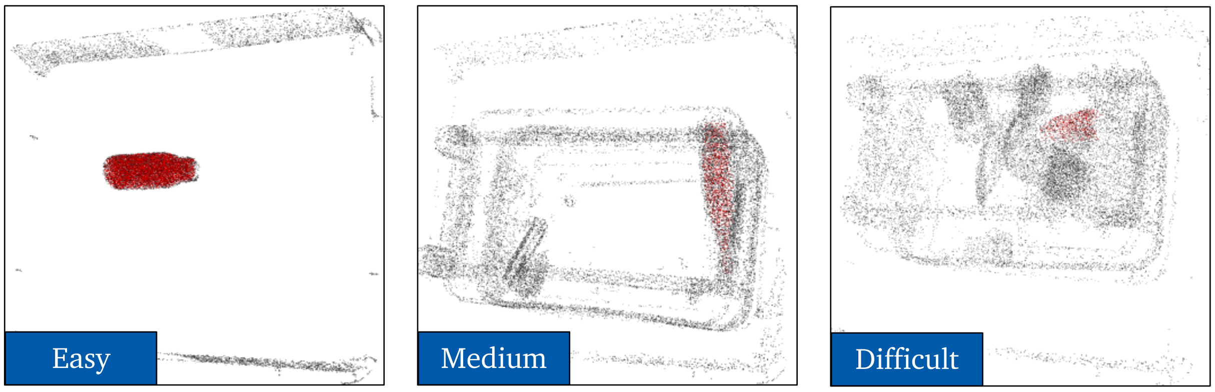


Figure 11: Ground truth examples of 3D CT scans with different human-perceived scene complexity

Segmentation performances for different scene complexities
Testing Data, $N_{sample} = 25{,}000$

Scene complexity	Easy	Medium	Difficult
Number of images	21	35	144
Automatic Region Growing	79.47%	61.34%	54.49%
PointNet 4D + loss weighting	82.72%	33.19%	24.11%
Random	30.94%	13.62%	7.50%

Table 15: Human-perceived scene complexity correlates with algorithm performance

This experiment shows that *PointNet* performs very well on easy scenes, slightly better than *Automatic Region Growing*, but, as the scene complexity increases, segmentation performance drops significantly.

In difficult scenes, we observed that *PointNet 4D* tends to predict segments with multiple or incorrect dense, liquid-like patches, resulting in low IoU. Unfortunately, most of our 3D CT scans provide difficult scenes, reducing *PointNet 4D*'s mean IoU among all scene complexities.

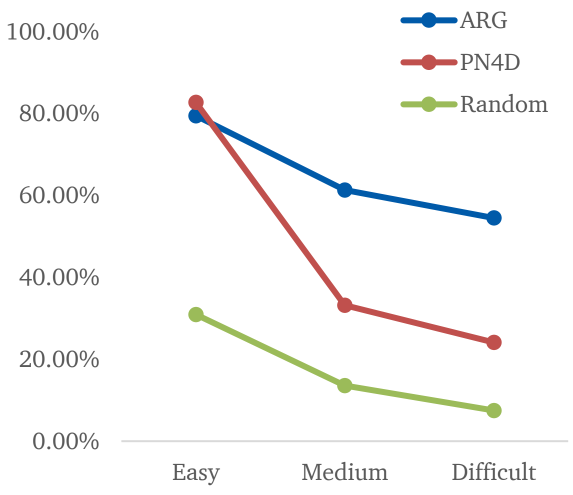


Figure 12: Visualization of performance drops due to scene complexity

Increasing the scene complexity also decreases the performance of *Automatic Region Growing*, but the mean IoU remains reasonably high for difficult scenes because the algorithm has no global awareness: Seed points are selected based on local point cloud features and liquids are segmented using region growing based on the seed point and its neighborhood.

The performance of *Random* also decreases as the scene complexity increases which indicates that human-perceived scene complexity is an adequate measurement.

5.3 Liquid Shapes Under Gravity

Our 3D CT baggage scans depict liquids stored in containers, such as bottles, jugs and flasks. The corresponding ground truth annotations for our liquid segmentation task are given as binary masks. Notably, only liquid voxels, neither containers nor contained air, are annotated. Additionally, the containers have different sizes, shapes and orientations, and the amounts of contained liquid vary. Combined with gravity, the liquids assume irregular shapes:

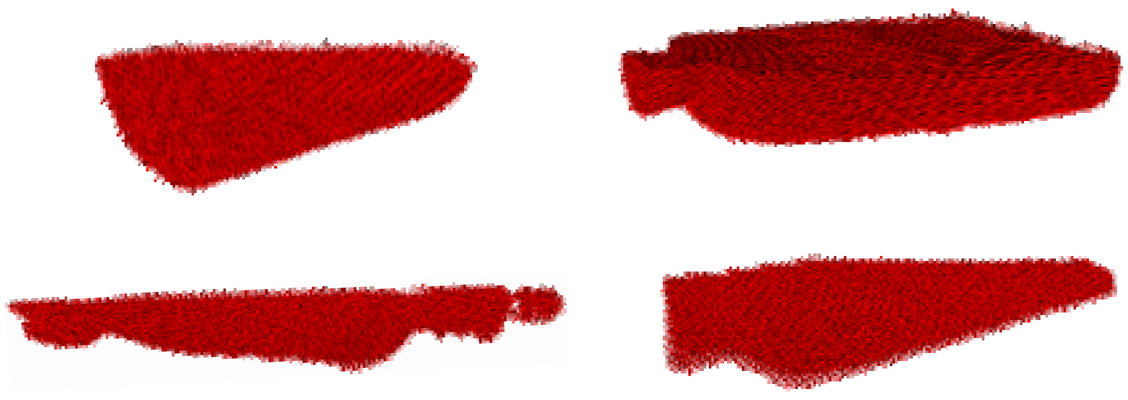


Figure 13: Examples of irregularly shaped liquid annotations

According to Qi *et al.* [1], *PointNet* "learns to summarize a shape by a sparse set of key points". However, the liquids from our 3D CT baggage scans do not have representative shapes which could be learned. Comparing the performance of the unmodified, solely shape-aware *PointNet* with *PointNet 4D* clearly illustrates the necessity of incorporating crucial intensity information and the deficiency of *PointNet*'s shape-aware learning in the context of liquid segmentation.

Unmodified *PointNet vs. PointNet 4D*

N_{sample}	10,000		25,000		50,000	
Data	Train	Test	Train	Test	Train	Test
PointNet	14.45%	12.03%	21.97%	17.77%	11.97%	10.37%
PointNet 4D	32.02%	22.78%	38.79%	25.58%	20.80%	16.17%

Table 16: Comparing *PointNet* with *PointNet 4D* illustrates the importance of intensity information

Liquids from our dataset have one common shape regularity that *PointNet* could exploit: At the transition between liquid and air inside a container, liquids end with a plane that is orthogonal to the direction of gravity. Our random azimuth rotation data augmentation keeps these planes orthogonal, whereas our random 3D rotation data augmentation does not.

Random azimuth rotations vs. full 3D rotations

N_{sample}	10,000		25,000		50,000	
Data	Train	Test	Train	Test	Train	Test
PointNet 4D + loss weighting + 3D rotation	15.93%	14.73%	14.48%	14.58%	15.69%	16.36%
PointNet 4D + loss weighting + azimuth rotation	24.77%	22.70%	28.73%	25.04%	27.26%	23.91%

Table 17: Random azimuth rotations keep planes at liquid ends parallel, outperforming 3D rotations

Our experiments with both data augmentation techniques show that preserving those parallel liquid planes by applying random azimuth rotation instead of random 3D rotations yields better results. This indicates that the network is taking orthogonal liquid planes into account because they are the only shape regularity that the liquids from our dataset can offer.

5.4 Insufficient Feature Space and Feature Learning

Common sense dictates that increasing the number of points of any point cloud representation increases its expressiveness. Therefore, large samples should contain more information and yield higher segmentation performances than small samples. Our experiments confirm that 25,000 points perform better than 10,000 points. However, 50,000 points performed worse than 10,000 points, although it is expectable that 50,000 points perform better than or at least as well as 25,000.

To explain the unexpected, poor results of using 50,000 points, we state two hypotheses:

Firstly, *PointNet*'s feature space is insufficient. It is already satiated with the information from 25,000 points and not capable of representing the additional information from 50,000 points.

However, if this were the sole reason, 50,000 points would still perform as well as 25,000 points. Therefore, secondly, *PointNet*'s feature learning mechanism is insufficient. The network is overwhelmed by 50,000 points and cannot learn features effectively.

To address both problems at the same time, we inserted additional layers into *PointNet 4D*: Adding parameters increases the feature space and deepening the network improves the feature learning mechanism [24, 25, 30]. In Table 18, we display relative segmentation performances, comparing 25,000 and 50,000 to 10,000 to illustrate the relative performance gain or loss by increasing the sample size. Experiments with *PointNet 4D* show that inserting additional layers to deepen the network improves the performance for large sample sizes of 50,000 points.

Relative performances of *PointNet* variants

N_{sample}	10,000		25,000		50,000	
Data	Train	Test	Train	Test	Train	Test
PointNet	±0.00%	±0.00%	+7.52%	+5.74%	-2.48%	-1.66%
PointNet 4D	±0.00%	±0.00%	+6.77%	+2.80%	-11.22%	-6.61%
PointNet 4D + loss weighting	±0.00%	±0.00%	+10.44%	+1.90%	-0.48%	-1.37%
PointNet 4D + loss weighting + additional layers	±0.00%	±0.00%	+3.98%	+1.32%	+0.21%	+1.05%

Table 18: Relative *PointNet* segmentation performances of larger sample sizes compared to 10,000

5.5 Ground Truth Scarcity

Supervised deep learning techniques, such as most modern computer vision algorithms, require human-annotated or otherwise measured ground truth data. However, acquiring ground truth data to train a deep neural network can be time consuming and difficult.

Training a network for 3D CT liquid segmentation requires voxel-accurate annotations of 3D CT scans. This type of ground truth data is particularly difficult to obtain due to several reasons:

Firstly, the raw data itself is very uncommon due to limited access to 3D CT machines. In contrast to common 2D images that everyone can take using any camera, 3D CT images can only be generated by the corresponding machines, which are usually only available at *e.g.* the airport or the doctor's office and may only be used under strict security and privacy regulations.

Secondly, 3D data is inherently more convoluted than regular 2D images due to the additional dimension enabling complex 3D poses and compositions instead of simpler projections and occlusions.

Thirdly, manual annotation of 3D data is extremely difficult because monitors cannot display 3D data neatly. Instead, they can only display projections or translucent representations of the original data, making navigation and selection very challenging.

Lastly, most humans can recognize a dog in an image or draw a bounding box around a person, but in the case of annotating liquids in 3D CT scans, sometimes not even humans can estimate the correct segmentation due to the unusual data format. According to *Smiths Detection*, some customers declined to purchase advanced 3D CT machines and opted for traditional 2D x-ray scanners instead because they have no trained operators who can interpret 3D CT images.

Semantic segmentation datasets for other data types, such as 2D RGB, offer substantially larger numbers of images because they are more popular and easier to obtain and annotate.

	Number of images	Data type
PASCAL VOC 2012 [9]	9,993	2D RGB
Cityscapes [10]	5,000 (fine)	2D RGB
Mapillary Vistas [37]	25,000	2D RGB
ModelNet40 [34]	12,311	3D CAD
ShapeNet [35]	16,881	3D CAD
Stanford 3D [36]	1,626	3D point cloud
Smiths Detection	712	3D CT

Table 19: Semantic segmentation dataset sizes and image types: 3D CT ground truth data is rare

Without enough ground truth data, neural networks overfit to the known training data, *i.e.* they try to learn the data by heart. In this case, the trained model will perform significantly worse on unknown testing data, whereas a successfully trained, generalizable model will have similar performances on training and testing data.

Commonly, data augmentation is used to reduce overfitting: Input and corresponding ground truth are transformed using random rotations, translations, mirroring, scaling, *etc.*, such that the input is diversified, making it more difficult for the network to memorize instance-specific details. Instead, it learns more task-specific features during training, improving generalization to unknown data.

We applied data augmentation in form of random 3D and azimuth rotations. While the overall segmentation performance did not increase, the gap between training and testing performance was reduced significantly, *i.e.* the network's ability to generalize to unknown data improved. This indicates that more training data would benefit generalization.

Difference between training and testing performance

N_{sample}	10,000	25,000	50,000
PointNet 4D + loss weighting	17.37%	25.91%	18.26%
PointNet 4D + loss weighting + 3D rotation	1.20%	0.10%	0.67%
PointNet 4D + loss weighting + azimuth rotation	2.07%	3.69%	3.35%

Table 20: Data augmentation reduces overfitting

6 Conclusion

Recent semantic segmentation approaches rely heavily on deep convolutional neural networks. While these approaches perform incredibly well on 2D images, they are inapplicable for dense, high-resolution 3D CT images due to enormous computational costs and memory requirements.

In this thesis, we proposed two fundamentally different approaches for liquid segmentation in dense, high-resolution 3D CT baggage scans and techniques to handle voluminous data. Our traditional approach performs reasonably well in terms of segmentation results, but it is slow and specifically designed for the task. Our deep learning-based approach is extremely fast and generalizable to other tasks, if ground truth data is available, however, segmentation results are insufficient. We identified, analyzed and discussed shortcomings of our deep learning-based approach by explaining hypotheses and conducting supporting empirical studies.

Both proposed methods are not ready for real-world applications. Therefore, further research is necessary to improve segmentation performances. In particular, we suggest replacing uniform sampling with smart, feature-aware sampling to improve sparse representations and piecewise processing of images to ensure scene simplicity. Furthermore, using sophisticated intermediate representations, *e.g.* demonstrated by Landrieu and Simonovsky [19], might be beneficial.

VI. Bibliography

[1] C. R. Qi, H. Su, K. Mo and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[2] S. W. Zucker, "Region Growing: Childhood and Adolescence," in *Computer Graphics and Image Processing*, 1976.

[3] J. L. Muerle and D. C. Allen, "Experimental Evaluation of Techniques for Automatic Segmentation of Objects in a Complex Scene," in *Pictorial Pattern Recognition*, 1968.

[4] C. R. Brice and C. L. Fennema, "Scene Analysis Using Regions," in *Artificial Intelligence*, 1970.

[5] S. L. Horowitz and T. Pavlidis, "Picture Segmentation by a Directed Split-and-Merge Procedure," in *Proceedings of the International Joint Conference on Pattern Recognition*, 1974.

[6] L. Vincent and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1991.

[7] H. P. Ng, S. H. Ong, K. W. C. Foong, P. S. Goh and W. L. Lowinski, "Medical Image Segmentation Using K-Means Clustering and Improved Watershed Algorithm," in *2006 IEEE Southwest Symposium on Image Analysis and Interpretation*, 2006.

[8] X.-W. Li, Y.-X. Kang, Y.-L. Zhu, G. Zheng and J.-D. Wang, "An Improved Medical Image Segmentation Algorithm Based on Clustering Techniques," in *10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI 2017)*, 2017.

[9] M. Everingham, L. van Gool, C. K. Williams, J. Winn and A. Zisserman, *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results,* 2012.

[10] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] J. Long, E. Shelhamer and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[12] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[13] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," in *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2016.

[14] C. Peng, X. Zhang, G. Yu, G. Luo and J. Sun, "Large Kernel Matters -- Improve Semantic Segmentation by Global Convolutional Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] H. Su, S. Maji, E. Kalogerakis and E. Learned-Miller, "Multi-view Convolutional Neural Networks for 3D Shape Recognition," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[16] C. R. Qi, L. Yi, H. Su and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[17] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[18] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Y. Gwak and S. Savarese, "SEGCloud: Semantic Segmentation of 3D Point Clouds," in *International Conference on 3D Vision (3DV)*, 2017.

[19] L. Landrieu and M. Simonovsky, "Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[20] J. Hagerty, R. J. Stanley and W. V. Stoecker, "Medical Image Processing in the Age of Deep Learning - Is There Still Room for Conventional Medical Image Processing Techniques?," in *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, 2017.

[21] X. Dongkuan and T. Yingjie, "A Comprehensive Survey of Clustering Algorithms," in *Annals of Data Science*, 2015.

[22] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[23] T. Moriya, H. R. Roth, S. Nakamura, H. Oda, K. Nagara, M. Oda and K. Mori, "Unsupervised Segmentation of 3D Medical Images Based on Clustering and Deep Representation Learning," in *SPIE Medical Imaging 2018: Biomedical Applications in Molecular, Structural and Functional Imaging*, 2018.

[24] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going Deeper with Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[26] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[27] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan and L. J. Guibas, "Volumetric and Multi-View CNNs for Object Classification on 3D Data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] M. Jaderberg, K. Simonyan, A. Zisserman and K. Kavukcuoglu, "Spatial Transformer Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[29] D. Eigen and R. Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[30] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Rcognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] "PointNet PyTorch," [Online]. Available: https://github.com/fxia22/pointnet.pytorch. [Accessed April 2018].

[32] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proceedings of the International Conference for Learning Representations (ICLR)*, 2014.

[33] J. Arvo, "Fast Random Rotation Matrices," in *Graphics Gems III*, 1992.

[34] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao, "3D ShapeNets: A Deep Representation for Volumetric Shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[35] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer and L. Guibas, "A Scalable Active Framework for Region Annotation in 3D Shape Collections," in *ACM Transactions on Graphics (TOG)*, 2016.

[36] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer and S. Savarese, "3D Semantic Parsing of Large-Scale Indoor Spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[37] G. Neuhold, T. Ollmann, S. R. Bulò and P. Kontschieder, "The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

VII. Appendix

A. Additional Result Visualizations

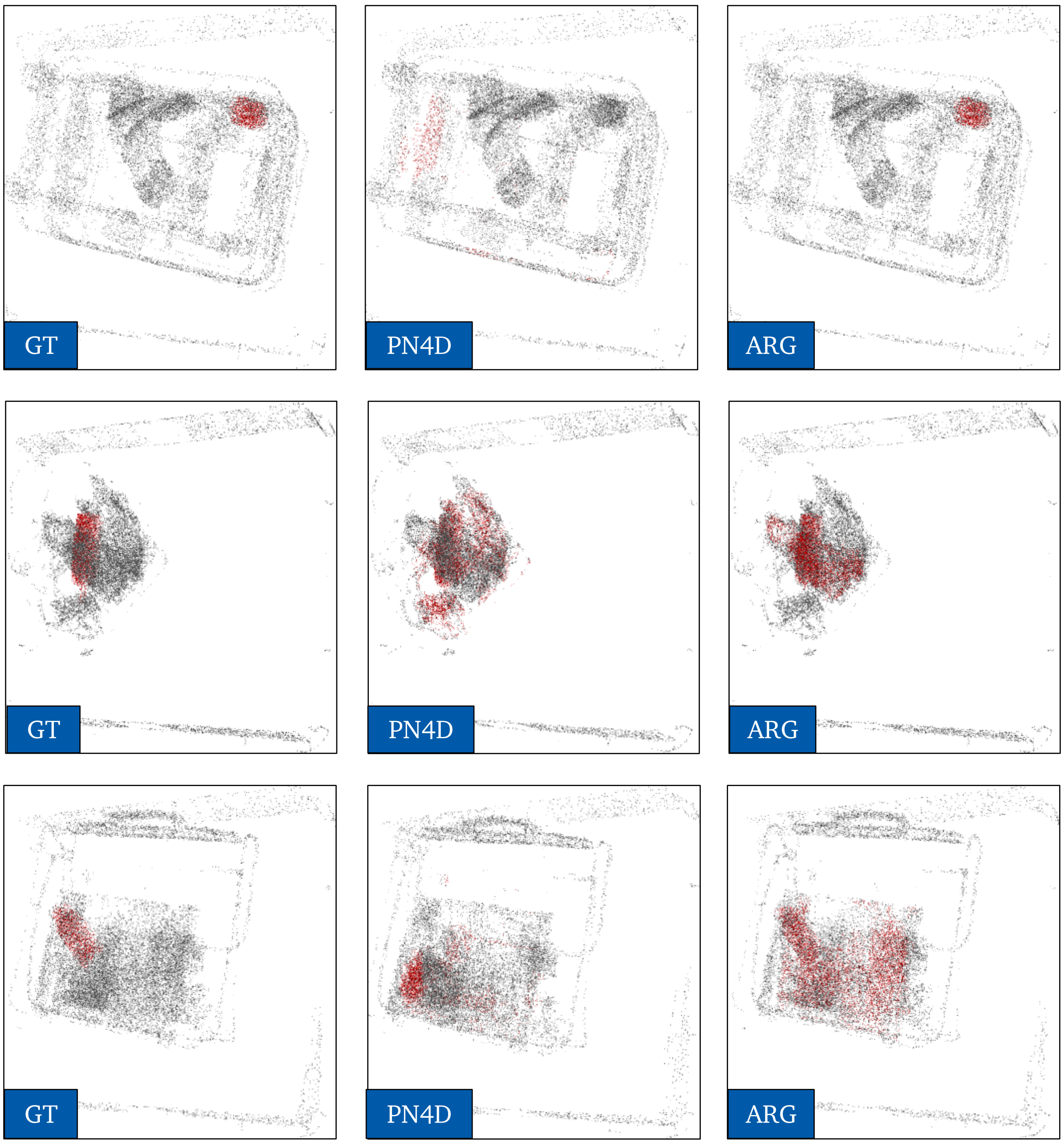


Figure 14: Ground truth (left) *vs. PointNet 4D* (middle) *vs. Automatic Region Growing* (right)

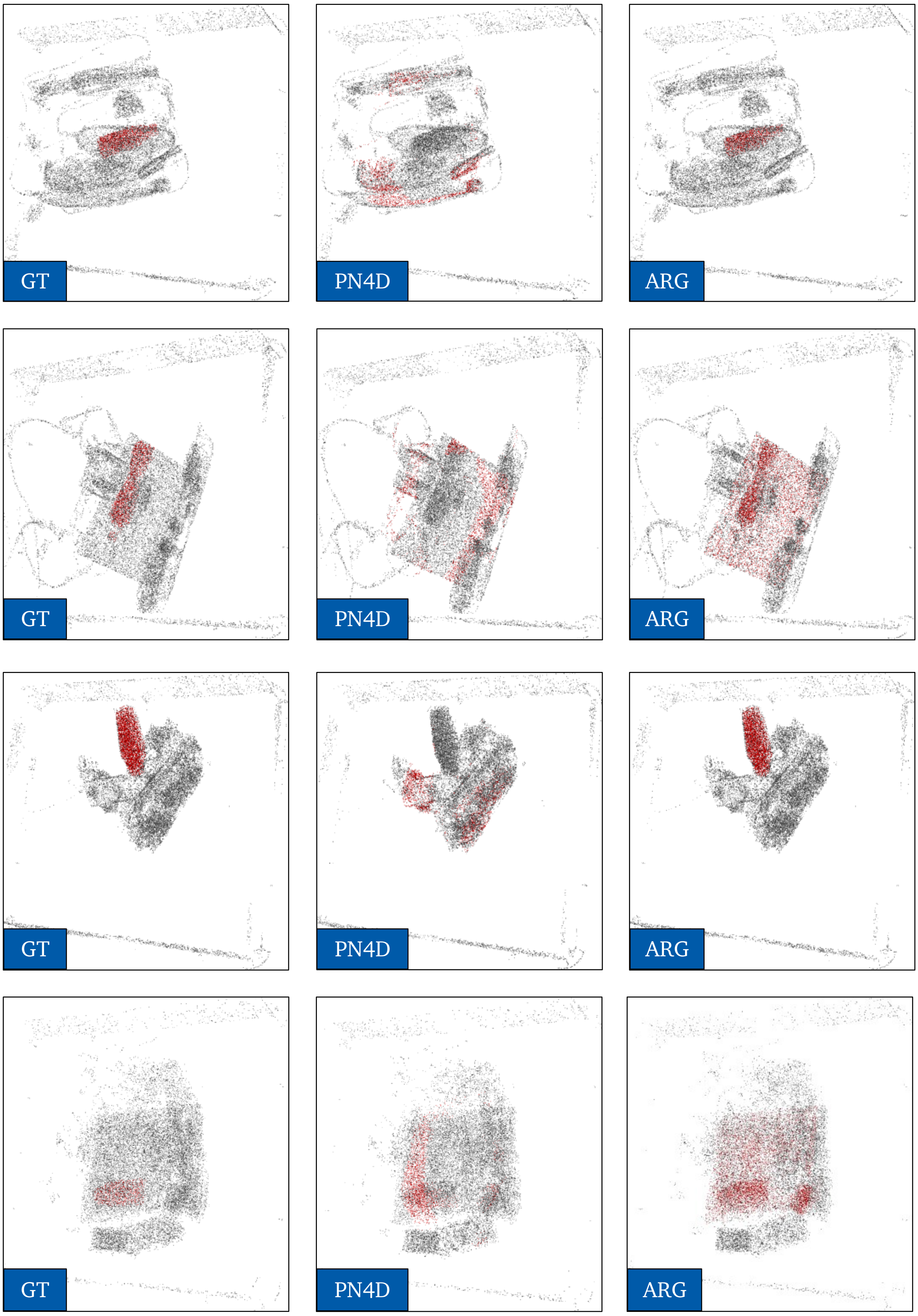


Figure 15: Ground truth (left) *vs. PointNet 4D* (middle) *vs. Automatic Region Growing* (right)