2
Real-time segmentation for tomographic imaging

2.1 Introduction

Tomographic imaging is a widely applicable technique for studying the internal structure of objects using some form of penetrating radiation such as X-rays or an electron beam. Projection images are obtained from a range of angles and a tomographic reconstruction algorithm subsequently computes a 3D image of the internal structure of the object. Currently, reconstruction and analysis are often performed after image acquisition has completed. If processing, reconstruction, and analysis of tomographic data can be run in real time during the experiment, internal dynamic processes of the imaged object can be visualized and analyzed as they occur. Real-time feedback enables online optimization and steering of the imaging setup and experimental conditions which increases the efficiency of experiments and avoids costly repetition.

Despite advances in computationally efficient reconstruction algorithms [13, 112] and in specialized hardware such as Graphic Processing Units (GPUs) [174] and supercomputers [16], full 3D tomographic reconstructions at the rate of data acquisition remain out of reach for most applications. Recently it was shown that real-time reconstruction can be achieved for a small set of arbitrarily oriented 2D slices [27]. These slices can be adjusted on the fly, thereby giving access to a virtual full 3D volume at a fraction of the computational cost. This methodology is called quasi-3D reconstruction, and is implemented in the RECAST3D software package.

To enable adaptive imaging, where the imaging process is adjusted based on the observations, just having access to a reconstructed volume is not sufficient, as the image analysis step should also be included in the real-time processing pipeline.

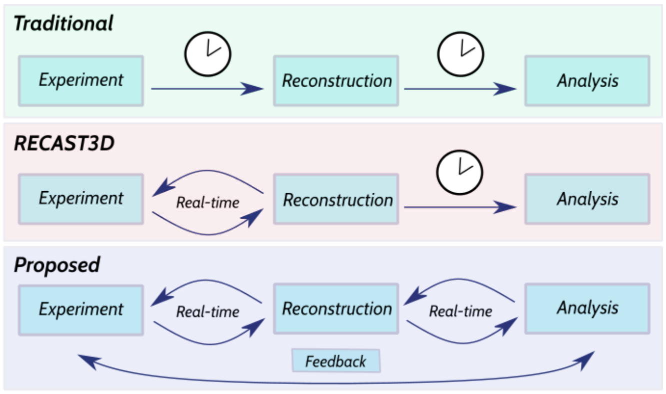


Figure 2.1: Traditional experiments (top) involving tomography require significant time for both the reconstruction phase and the offline analysis phase. With RECAST3D (middle) the reconstruction phase is performed in real time. Our method (bottom) additionally includes a real-time segmentation step.

An important step in many image analysis pipelines is segmentation, which is the problem of assigning to each pixel the appropriate class label from a finite set of classes, for example segmenting bone for calcaneal fractures in CT images [194]. In this article we introduce a real-time imaging pipeline to reconstruct, segment, and visualize quasi-3D volumes implemented as an extension of the existing RECAST3D software package. Our method adds real-time segmentation to the existing real-time reconstruction capabilities of the RECAST3D framework, as outlined in Figure 2.1.

As quasi-3D reconstruction employs direct reconstruction methods such as filtered backprojection (FBP) [106] and Feldkamp-David-Kress (FDK) [59] without additional image regularization, limited-data artefacts are typically present in the reconstructions. These artifacts limit the applicability of computationally efficient unsupervised segmentation algorithms, such as Otsu's method [170], since they are often unable to separate artifacts and noise from important features. Furthermore, because image analysis algorithms may be sensitive to noise in the segmentation [32, 150, 199], analysis based on such traditional segmentation methods may not be accurate. In addition, many unsupervised segmentation methods operate exclusively on the basis of the pixel values [124, 163, 170], limiting their applicability to general segmentation problems as they are unable to segment features that are not based on pixel values.

To overcome these issues, we propose to use a convolutional neural network (CNN) to segment the quasi-3D reconstructions in real time. To apply CNNs in a quasi-3D setting, we introduce an adapted training strategy that takes the arbitrary orientations of the slices into account. We show that a CNN is capable of achieving similar accuracy to segmentations based on computationally more expensive total variation minimization (TV-MIN) reconstructions [14] which are too slow to compute for real-time applications. In addition, we show that a CNN can be implemented efficiently as a plugin within the existing RECAST3D framework without significantly increasing the processing time.

This article is structured as follows. In Section 2.2 we introduce the tomographic reconstruction problem and define the FDK and TV-MIN reconstruction algorithms. We introduce quasi-3D reconstructions, the segmentation problem, and provide more details on the segmentation plugin. Lastly, we outline our adapted training strategy for randomly oriented slices. In Section 2.3 we present the experimental results, and analyze the training strategies. We perform a real-world experiment on a dynamic X-ray CT dataset and two simulated experiments. Finally, in Section 2.4 we state our final conclusions.

2.2 Method

2.2.1 Prerequisites

Tomographic Reconstruction

The tomographic reconstruction problem is to recover a volume from a series of its projections. In this article we consider circular cone-beam tomography, where the object is placed in between a point source and flat-panel detector which are situated on opposite sides of a circle. The object is rotated and X-ray projections are taken at a selection of equidistant angles. The approach generalizes to other acquisition geometries (e.g. parallel beam) in a straightforward manner.

The tomographic reconstruction problem can be modelled as an inverse problem:

$$K\mathbf{u} = \mathbf{f}. \tag{2.1}$$

Here K is the forward projection operator, $\mathbf{u} \in \mathbb{R}^{N_x \times N_y \times N_z}$ represents the object, and $\mathbf{f} \in \mathbb{R}^{N_\theta \times N_a \times N_b}$ is the measured projection data, with N_θ is the number of projection angles, and N_a, N_b are the number of detector rows and columns respectively. In this article we use the FDK reconstruction algorithm, given by

$$\mathbf{u}_{\text{FDK}} = K^T(\mathbf{h} * \tilde{\mathbf{f}}). \tag{2.2}$$

Here $\tilde{\mathbf{f}}$ denotes weighted projection data, which compensates for diminishing intensity at distance from detector center, and $\mathbf{h} \in \mathbb{R}^{N_b}$ is a 1D filter. We used the Ram–Lak filter for this work.

Instead of using FDK, equation 2.1 can be solved by iteratively minimizing $\|K\mathbf{u} - \mathbf{f}\|$. In addition, we can add prior information about the gradient of the image being sparse by adding a total variation term [14] to improve reconstruction accuracy when projection data is limited or noisy:

$$\frac{1}{2}\|K\mathbf{u} - \mathbf{f}\|_2^2 + \lambda\|\nabla\mathbf{u}\|_1.$$

This function can be minimized by a range of convex optimization algorithms.

Quasi-3D Reconstruction

Quasi-3D reconstruction has recently been proposed as a method to make real-time tomographic reconstruction feasible [27]. Instead of computing a full 3D volume, only a small collection of arbitrarily oriented 2D slices is reconstructed and visualized in real-time. When these slices are translated and/or rotated by the user, they are reconstructed on the fly, so that it appears as though a full 3D reconstruction is available. This on-demand 2D reconstruction significantly reduces the total computational cost compared to full 3D reconstruction. This approach is implemented in the open source RECAST3D software package and more implementation details can be found in [27].

In RECAST3D, the filtering and weighting steps of the FDK algorithm are performed in parallel. The computation of $\mathbf{h} * \tilde{\mathbf{f}}$ is performed in real time from the incoming data. When a slice is requested, the application of K^T (called backprojection) is performed using GPU-based high-performance routines from the ASTRA toolbox [1]. In addition, a low-resolution 3D FDK reconstruction is created so that the user can preview the object. Our quasi-3D pipeline for segmentation is implemented by extending the RECAST3D software package with a computationally efficient segmentation plugin.

Segmentation

Mathematically, segmenting an image can be described as finding a function $g : \mathbb{R}^{m\times n} \to \mathbb{Z}_k^{m\times n}$, where m, n are the rows and columns of the image and k is the number of object classes to be assigned.

Classical segmentation methods (for example local and global thresholding [124, 170], watershed methods [163]) typically operate on the image greyvalues to separate classes and have the high computational efficiency that is need for real-time segmentation. As an example, Otsu's method performs a segmentation of an image by selecting a threshold that minimizes intra-class variance. In addition to the greyscale distribution, segmentation can be performed on other properties by for example clustering pixels [48] or defining edge boundaries in the image [157]. Recently, CNNs have proven successful for image segmentation [9, 202].

CNNs for segmentation

In this work we use CNNs to segment the tomographic reconstructions. In a segmentation network, the final output layer will assign one of k classes to each pixel. The CNN is defined by its architecture with weights Θ which can be altered to change the output. For a given Θ, a CNN corresponds to a function $F_\Theta : \mathbb{R}^{m\times n} \to \mathbb{R}^{k\times m\times n}$ which aims to approximate g by computing a probability vector with predictions for each class for each pixel. The highest probability class can be chosen from the network predictions to obtain a final segmentation.

The weights Θ are found in a training phase, where input samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are processed by the network and compared to known labelled output samples $\mathbf{y}_1, \ldots, \mathbf{y}_N$. A loss function $\mathcal{J} : \mathbb{R}^{k\times m\times n} \times \mathbb{Z}_k^{m\times n} \to \mathbb{R}$, such as cross-entropy loss,

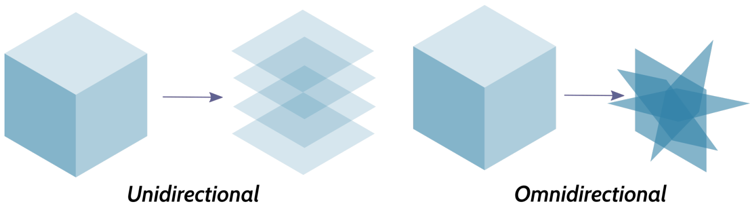


Figure 2.2: Diagram outlining unidirectional training on slices (left) and omnidirectional training (right).

measures the error of the network on the training samples. The aim of the training phase is to find a Θ that minimizes the loss on the training dataset

$$\Theta^* = \arg\min_{\Theta} \left\{ \sum_{i=1}^{N} \mathcal{J}(F_\Theta(\mathbf{x}_i), \mathbf{y}_i) \right\}.$$

For a CNN, we can compute the partial derivatives of $\mathcal{J}$ with respect to the weights using backpropagation. The weights can be updated using gradient-based optimization algorithms [113].

2.2.2 Quasi-3D training strategies

In 3D image segmentation, neural networks are often trained on 2D slices from the volumes since full 3D networks are typically computationally too expensive [134, 183]. The 2D input slices are obtained by extracting slices in a single direction. In contrast, slices in a quasi-3D reconstruction can have an arbitrary orientation. A network trained only on unidirectional slices may not recognize object classes from a different view. Therefore, the standard training procedure has to be adapted to enable application of CNNs to arbitrary oriented slices. Here, we introduce a training strategy where arbitrarily oriented 2D slices of the tomographic volume are supplied as input for the neural network.

Let $\mathbf{X} \in \mathbb{R}^{n\times n\times n}$ be an input volume and $\mathbf{Y} \in \mathbb{Z}_k^{n\times n\times n}$ the aligned target volume. Define $E_{\alpha,\beta,\gamma} : \mathbb{Z} \times \mathbb{R}^{n\times n\times n} \to \mathbb{R}^{n\times n}$ to be a rotated extraction operation. $E_{\alpha,\beta,\gamma}(i, \mathbf{X})$ extracts the i-the slice rotated by angles α, β, γ with respect to the sagittal slice from the volume $\mathbf{X}$. For *omnidirectional training* we create a dataset of slices with pairs $(E_{\alpha,\beta,\gamma}(i, \mathbf{X}), E_{\alpha,\beta,\gamma}(i, \mathbf{Y}))$ where the angles are randomly generated. For *unidirectional training* we create pairs of sagittal slices $(E_{0,0,0}(i, \mathbf{X}), E_{0,0,0}(i, \mathbf{Y}))$ (see Figure 2.2).

2.2.3 Segmentation Plugin

To construct the pipeline for real-time segmentation we developed a plugin for RECAST3D which segments quasi-3D reconstructions. The segmented slices are then visualized in RECAST3D. The plugin is GPU-based and can be disabled, altered and reenabled while the projection data is processed simultaneously. A command line interface allows for online tuning of parameters with immediate visual feedback, and the user can select which class is visualized. The plugin operates independently from the quasi-3D reconstruction pipeline, and can be run on a separate node.

The plugin is implemented in Python, and includes three CNNs implemented in PyTorch: the MS-D network [89, 183], U-Net [202], and ResNet [84]. In addition, the plugin includes several traditional segmentation methods, including Otsu's method [170], cross entropy thresholding [124], contour evolution [158], region based random walk segmentation [69] and the watershed algorithm [163].

2.3 Results and Discussion

2.3.1 Setup

To assess the accuracy and computational efficiency of our CNN-based segmentation approach, we compare it with traditional unsupervised methods. The neural network used in this work is the MS-D network [183], chosen because of its low number of trainable parameters.

Since the MS-D network can flexibly adapt to different problems, we used the same network architecture for each experiment. We used an MS-D network, implemented in PyTorch [89], of 100 layers with a width of 1. The dilations in layer i were set to $1 + (i \mod 10)$. The networks were trained using the ADAM algorithm [113], using a batch size of 20, and the cross-entropy loss function. For each experiment, the data was split in training, validation, and test sets. The network is trained on the training set for 100 epochs. The network with the lowest validation error was selected to be evaluated on the test set. All experiments were run on a workstation with an AMD Ryzen 3800X processor and NVIDIA GeForce RTX 2070 Super GPU. To quantify our comparisons we use the F1-score (Dice coefficient) which is the harmonic mean of precision and recall, and the accuracy. For multi-class problems we report the macro F1-score (average of per-class F1-score), and the global accuracy.

2.3.2 Simulated data

We created two sets of simulated tomographic data to investigate the difference between omni- and unidirectional training, and to quantitatively compare Otsu's method to the MS-D network. For the former we created twenty 512^3 volumes filled with fibre strands and spheres. Each volume contained 20 spheres and 20 fibres which were generated with a random shape and location. Greyscale intensities were the same for both classes. 3D renderings of the noiseless volumes are shown in Figure 2.3.

For the second simulation experiment we created twenty 512^3 volumes filled with 40 fibre strands generated with random shapes surrounded by a cylinder

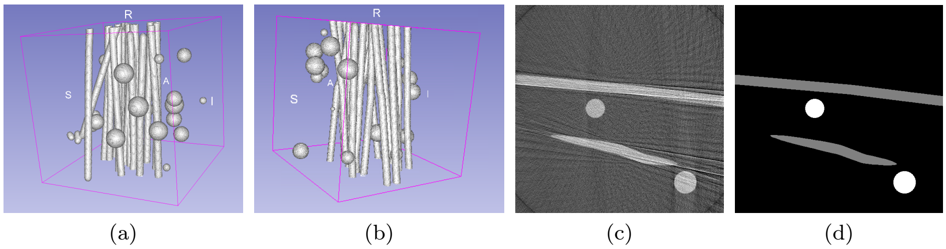


Figure 2.3: (a), (b) Example volumes of the fibre-sphere data, (c) slice of the noisy FDK reconstruction, and (d) slice of the ground truth (spheres and fibres have different labels).

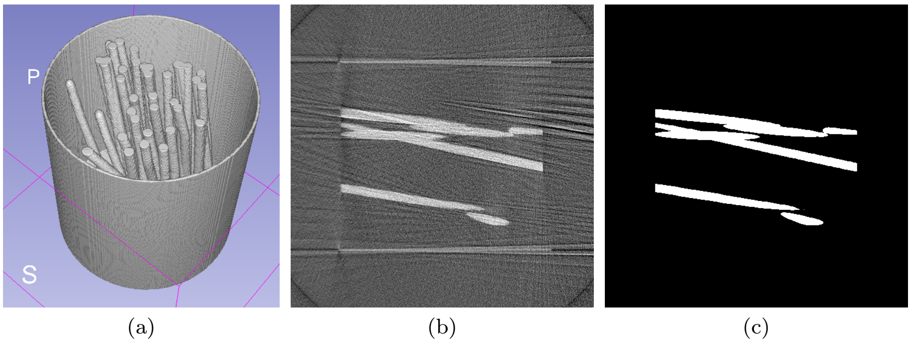


Figure 2.4: (a) Example volume of the fibre-container data with container, (b) slice of the noisy FDK reconstruction, and (c) slice of the ground truth (the container is not to be segmented).

forming a container (see Figure 2.4). Greyscale intensities were the same for the container and the fibres to represent a segmentation problem where the fibres are to be labeled but the container is not.

Using the ASTRA toolbox [1] we simulated a cone-beam projection dataset of 60 projections for each volume. Furthermore, Poisson noise was applied to the projection dataset where the sample absorbed roughly 70% of the incoming photons. Out of the 20 phantoms, 14 were randomly selected for training, 4 for validation and 2 for testing. For each scan we obtained 500 randomly oriented, and 500 unidirectional 512×512 slices for both the ground truth labeled volume, and the noisy FDK reconstruction (see Figures 2.3 and 2.4).

Comparison of uni- and omnidirectional training slices

To investigate the importance of the slicing direction in the training strategy for CNNs, we compare unidirectional and omnidirectional training strategies. We

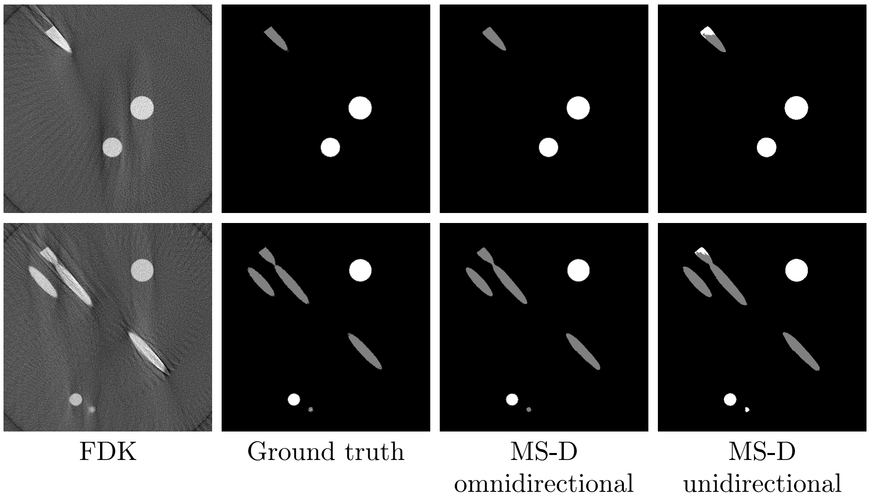


Figure 2.5: Slices of the simulated fibre-sphere test dataset and MS-D network predictions. From left to right we have the FDK reconstructions, the labeled ground truths, the MS-D network output trained on randomly oriented slices, and the MS-D network output trained on unidirectional slices.

	Random orientation		Sagittal orientation	
Training	**F1**	**Accuracy**	**F1**	**Accuracy**
Omnidirectional	**0.9989**	**0.9979**	0.9951	0.9950
Unidirectional	0.6857	0.9792	**0.9995**	**0.9995**

Table 2.1: Macro F1 and accuracy on the fibre-sphere (left) randomly oriented test set, and (right) sagittal test set, for MS-D networks trained with the omnidirectional, and unidirectional strategies.

trained one network on the slices in a single direction and the other on randomly oriented slices, using the fibre-spheres dataset (Figure 2.3). Otsu's method, and any other unsupervised method that works solely on greyvalues, are not applicable to this type of multi-class segmentation problem as they cannot categorize objects with the same greyvalue. This indicates an advantage of deep-learning approaches.

Example results from the test set are shown in Figure 2.5. Note that both networks were able to remove the Poisson noise and the tomographic artifacts. Some notable differences between both networks can be seen where the unidirectional network identifies parts of the fibre strands as spheres. In Table 2.1 we report the macro F1-score, and accuracy for both networks on both the test set with randomly oriented slices and the test set with slices in a single direction.

The MS-D omnidirectional network outperforms the unidirectional network on both quantitative measures for the randomly oriented test set, which is in

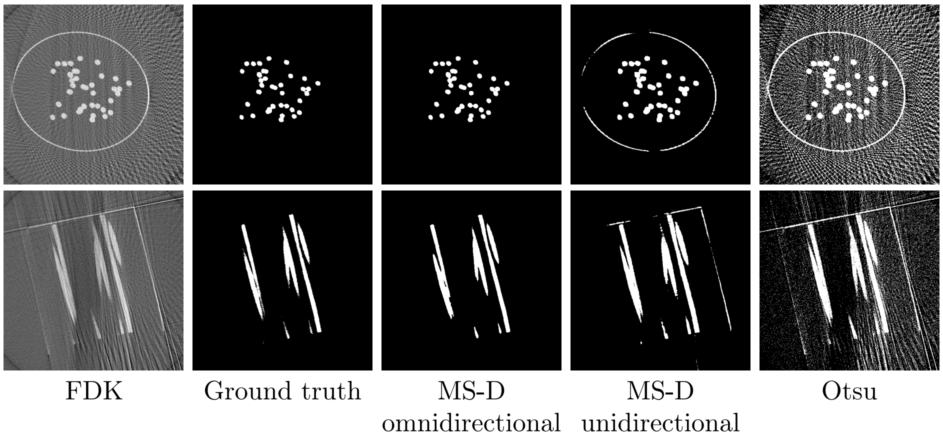


Figure 2.6: Slices of the simulated fibre-container test dataset, with (from left to right): the FDK reconstructions, the labeled ground truths, the omnidirectional MS-D network output, the unidirectional MS-D network output, and Otsu's method.

line with the qualitative comparison shown in Figure 2.5. When tested on the sagittal direction, although the unidirectional networks performs better than the omnidirectional network, the difference in performance is significantly smaller. This can be explained by the fact that the omnidirectional network has also encountered images sliced in the sagittal direction.

Comparison of MS-D segmentation and Otsu's method

Here, we compare Otsu's method to the uni- and omnidirectional MS-D networks. Both the MS-D networks and Otsu's method were tested on randomly oriented slices from the fibre data test phantoms because the user can select arbitrary slices in RECAST3D. To more accurately determine the performance of Otsu's method inside the container, we created a version of Otsu's method where a ROI-mask was applied on the segmented slices with the proper rotation. We used a cylindrical volume around the simulated container to remove misclassified background. Some example results from the test set can be seen in Figure 2.6 and in Table 2.2 we report the F1-score and accuracy.

The results show that the omnidirectional MS-D network was able to accurately segment the fibres and remove the applied Poisson noise. We see that the unidirectional MS-D network misclassified the container in the randomly oriented slices, and that Otsu's method occasionally does not remove the FDK artifacts and noise. In addition, Otsu's method classifies the container as a fibre since it is unable to distinguish it from the fibres on the basis of intensity. Even if we manually mask the region-of-interest, the interior of the container is significantly more noisy than the MS-D network segmentation.

	F1-score	**Accuracy**
MS-D omnidirectional	**0.9544**	**0.9989**
MS-D unidirectional	0.6777	0.9885
Otsu	0.0585	0.6086
ROI-Otsu	0.2568	0.9300

Table 2.2: F1 and accuracy on the fibre-container randomly oriented test set for MS-D omnidirectional, MS-D unidirectional, Otsu, and ROI Otsu.

The MS-D network outperforms Otsu and ROI-Otsu on both metrics. Otsu's method performs significantly worse on F1-score, which can be explained by the greater amount of false positives segmented by Otsu as opposed to the MS-D network.

2.3.3 Experimental data

To show the feasibility of our method in real-world applications, we applied the real-time segmentation pipeline to a real-world dynamic X-ray CT dataset of a dissolving tablet suspended in gel [38, 39]. A container with a dissolving tablet was filled with gel to create moving air bubbles which we segmented. The container was rotated at 100 deg/s and 60 projections were acquired every 180 degrees with an exposure time of 30 ms for each projection. In total 9960 projections of size 647×768 were taken and the experiment lasted 5 minutes.

In RECAST3D, the full processing step of a batch of projections takes approximately 140 ms on our workstation. The computation time to compute the backprojection for a slice is about 2 milliseconds. The segmentation with Otsu's method is about 3 milliseconds and with the MS-D network about 30 milliseconds. This means that the pipeline would be able to dynamically visualize the projection data stream every 170 milliseconds for a batch of 60 projections. In this experiment, data acquisition was at a rate of 1.8 seconds per batch of 60 projections (180 degrees), well within the computational limits of the pipeline. Figure 2.8 shows an example real-time quasi-3D reconstruction and segmentation of the data in RECAST3D.

To create training data for the neural networks we created TV-MIN reconstructions for every 60 projections with a regularization parameter $\lambda = 0.001$ for 2000 iterations. We used the Douglas–Rachford primal-dual splitting algorithm [19] to iteratively minimize the functional. Each TV-MIN reconstruction took roughly 20 hours on our workstation and is therefore infeasible to compute in real time. Next, we created 25 labeled ground truth volumes by applying Otsu's method to the TV-MIN reconstructions and masking the region outside the container. As a final processing step we removed small objects with a mass smaller than 4 pixels with the scikit-image `remove_small_objects` function from the morphology package [242]. The scans were randomly separated into 18 training scans (9216 slices), 4 validation scans (2048 slices) and 3 test scans (1536 slices) for the unidirectional

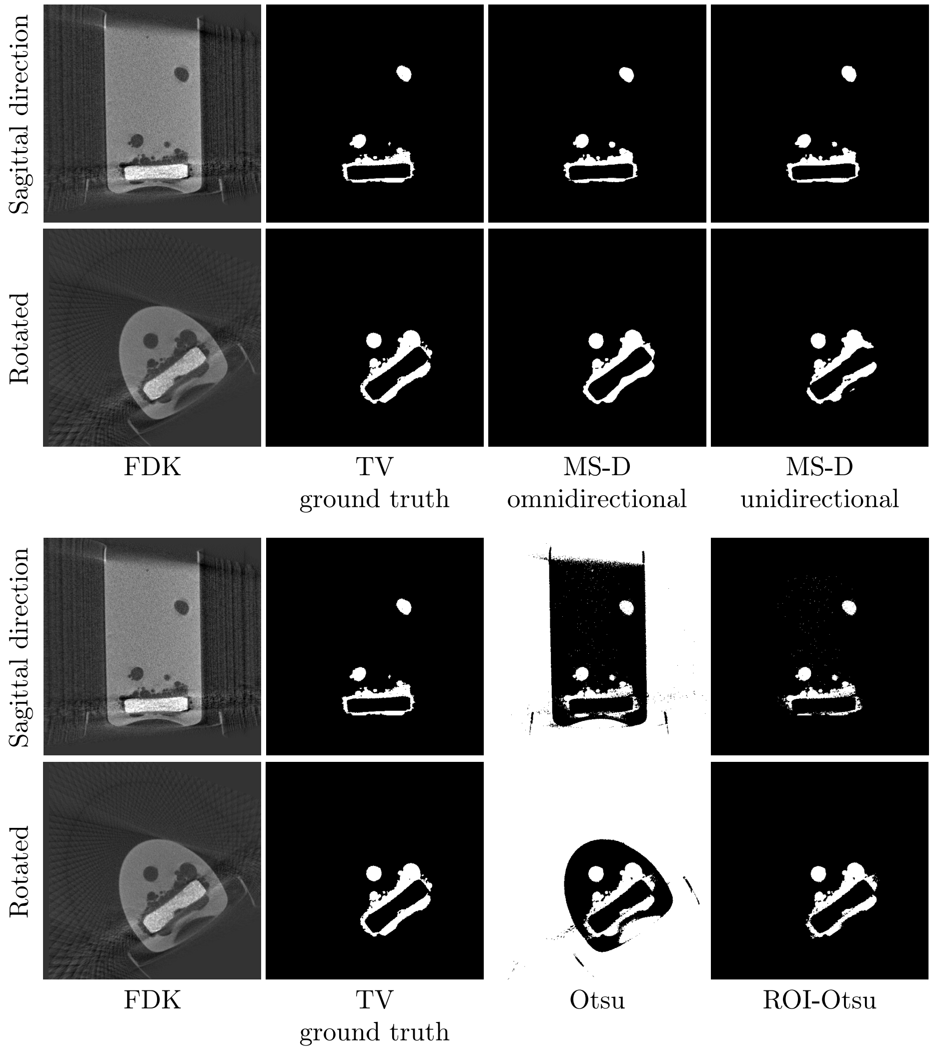
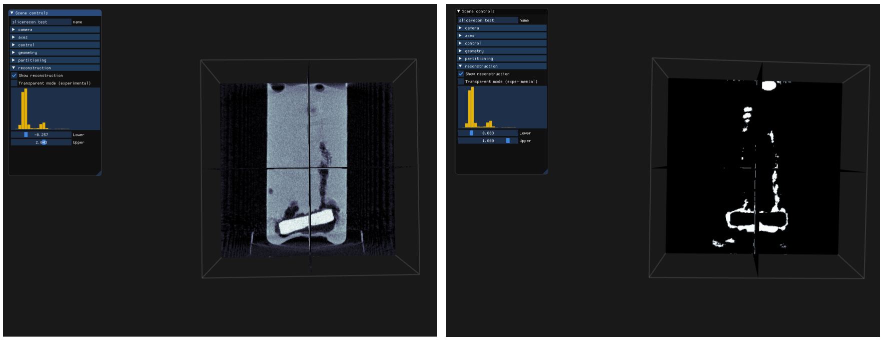


Figure 2.7: Slices of the TabletInFluid test dataset and experimental predictions. The first row is from the unidirectional dataset, the second from the randomly rotated dataset. From left to right we have the FDK reconstructions, the labeled TV-MIN ground truths, the MS-D network output trained on randomly oriented slices, the MS-D network output trained on slices in one direction, Otsu's segmentation, and Otsu's segmentation using a ROI cylindrical mask.



slicerecon test name
[✓] Show reconstruction
-8.257 Lower
2.043 Upper

slicerecon test name
[✓] Show reconstruction
0.083 Lower
1.000 Upper

Figure 2.8: (Left) example screenshot of dissolving tablet data reconstructed dynamically in RECAST3D, (right) example of segmented network output in RECAST3D.

	F1-score	**Accuracy**
MS-D omnidirectional	**0.8816**	**0.9983**
MS-D unidirectional	0.7595	0.9968
Otsu	0.0142	0.2538
ROI-Otsu	0.8229	0.9977

Table 2.3: F1 and accuracy on the real-world TabletInFluid randomly oriented test set for MS-D omnidirectional, MS-D unidirectional, Otsu, and ROI Otsu.

network. For the arbitrarily oriented slices, we chose the same amount of slices at random 3D orientations for each scan.

To compare our method to an existing computationally efficient method, we segmented each FDK slice with Otsu's method and manually applied a cylindrical ROI-mask to remove misclassified background. The results can be seen in Figure 2.7. In Table 2.3 we report the F1-score and accuracy for the randomly oriented slices.

The MS-D network trained on randomly rotated slices is able to create an accurate segmentation of the bubbles in real time and it outperformed the other three methods on all metrics. It is able to create real-time segmentations with similar quality to the computationally expensive segmented TV-MIN reconstructions. This shows that our method can be used to perform quasi-3D reconstruction and segmentation in real time by training on randomly oriented slices of segmented TV-MIN reconstructions. Note that, in practice, acquiring training data and training the networks has to be performed offline. The results show that our method outperforms Otsu's method with masking on all metrics. Notably, the unidirectional network regularly misclassifies sections of the randomly oriented slices. The importance of the training method is highlighted when comparing the F1-scores for both MS-D networks.

2.4 Conclusions

In this paper, we introduced a real-time pipeline to process, reconstruct, and segment quasi-3D tomographic images, representing an important step for online and real-time analysis of tomographic experiments. We showed the importance of including arbitrarily oriented slices in the training dataset to achieve accurate results. We demonstrated that a deep-learning based approach can perform better than Otsu's method in terms of accuracy on both simulated data and real-world dynamic tomographic data. In addition, our deep-learning based approach is more generalizable to multi-class segmentation problems than traditional intensity-based unsupervised segmentation methods. Using our method, one can perform real-time and online segmentation of quasi-3D volumes, enabling immediate feedback and analysis during experiments.