Learned Cone-Beam CT Reconstruction Using Neural Ordinary Differential Equations

Mareike Thies[a], Fabian Wagner[a], Mingxuan Gu[a], Lukas Folle[a], Lina Felsner[a], and Andreas Maier[a]

[a]Pattern Recognition Lab, Friedrich-Alexander Universität Erlangen-Nürnberg, Erlangen, Germany

ABSTRACT

Learned iterative reconstruction algorithms for inverse problems offer the flexibility to combine analytical knowledge about the problem with modules learned from data. This way, they achieve high reconstruction performance while ensuring consistency with the measured data. In computed tomography, extending such approaches from 2D fan-beam to 3D cone-beam data is challenging due to the prohibitively high GPU memory that would be needed to train such models. This paper proposes to use neural ordinary differential equations to solve the reconstruction problem in a residual formulation via numerical integration. For training, there is no need to backpropagate through several unrolled network blocks nor through the internals of the solver. Instead, the gradients are obtained very memory-efficiently in the neural ODE setting allowing for training on a single consumer graphics card. The method is able to reduce the root mean squared error by over 30% compared to the best performing classical iterative reconstruction algorithm and produces high quality cone-beam reconstructions even in a sparse view scenario.

Keywords: Inverse problems, computed tomography, iterative reconstruction, known operators

1. INTRODUCTION

Extending analytical or iterative reconstruction algorithms for computed tomography (CT) by deep learning modules has shown to improve the quality of the reconstructed images, especially in challenging cases like high noise levels or insufficient projection data.[1,2] For CT reconstruction, the input and output data of the problem are connected in a non-trivial geometrical manner. This is the reason why most deep-learning-based reconstruction approaches, instead of learning direct mapping from sinogram to image domain, incorporate knowledge about the physical operator connecting both domains and replace single components in the reconstruction pipeline by their learned counterpart, mostly operating in one domain only.[3]

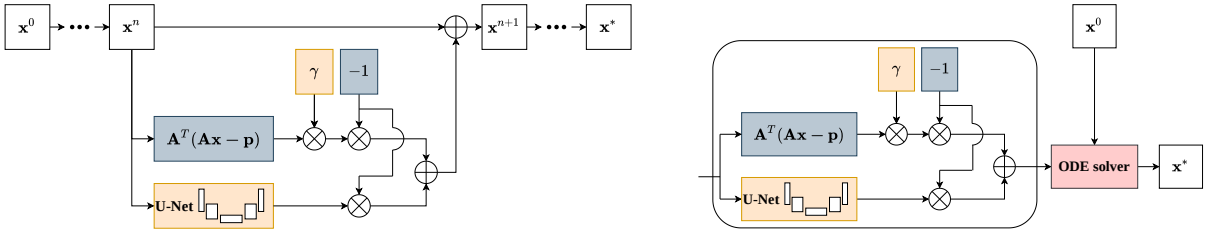
Unrolled iterative approaches seek to solve the reconstruction problem by loosely mimicking known iterative algorithms for inverse problems. This is done by unrolling a fixed number of iterations in depth as a deep learning architecture and incorporating learnable components in each step which are trained end-to-end. The exact architecture and the role of the trainable modules can vary giving rise to a number of approaches for MRI[4,5] and CT.[1,6,7]

While these unrolled iterative algorithms have achieved superior performance for the reconstruction of 2D images, their extension to the 3D case is challenging. Training requires gradient backpropagation through the entire unrolled sequence of trainable and known operators, thereby consuming a large amount of memory on the graphics card (GPU). In the 3D case, the amount of memory occupied by intermediate representations needed during backpropagation exceeds the memory of modern GPUs making the direct application of iterative approaches infeasible.

Previous work addressed the prohibitively high memory consumption of unrolled 3D models. Greedy training of each iteration independently reduces memory consumption and allows training on patches but does not result in an optimal joint weight configuration.[8] When increasing the volume resolution with network depth,

Further author information: Send correspondence to Mareike Thies)
Mareike Thies: E-mail: mareike.thies@fau.de

7th International Conference on Image Formation in X-Ray Computed Tomography, edited by Joseph Webster Stayman, Proc. of SPIE Vol. 12304, 1230409
 · 0277-786X · doi: 10.1117/12.2646442



(a) (b)

Figure 1: Comparison of an unrolled network (a) and a neural ODE-based version (b). While the unrolled version explicitly repeats the same network block for a fixed number of steps, the ODE solver receives only one network block parameterizing the temporal derivative of the volume. The solver computes the numerical integration internally. All trainable parts of our architecture are highlighted in yellow, the fixed parts are gray.

memory consumption is dominated by the single final iteration on full scale, but image quality is coupled to the expressiveness of the last iteration.[9] Further, the use of invertible networks avoids storing intermediate representations which makes the memory requirement constant in depth but requires the network architecture to meet certain criteria for invertibility.[10,11]

In this work, we propose to interpret the series of unrolled iterations as a continuous residual process and formulate the problem in terms of an ordinary differential equation (ODE). This allows us to map the reconstruction problem onto an initial value problem which can be solved and trained memory-efficiently using recently proposed neural ODEs.[12] The key idea is that the memory requirement does not depend on the number of iterations, i.e., network depth. Instead, the forward pass is replaced by a call to an ODE solver and gradients are obtained by solving another adjoint ODE without storing the intermediate representations of the forward pass. Whereas a similar idea has been applied to MRI reconstruction,[13] to the best of our knowledge we are the first ones to apply neural ODEs to CT reconstruction. We show that using this method we obtain 3D cone-beam CT reconstructions from few angles with superior image quality compared to classical analytic and algebraic algorithms.

2. METHODS

2.1 Learned Iterative CT Reconstruction

The forward model of a CT acquisition can be written as

$$\mathbf{p} = \mathbf{A}\mathbf{x} + \epsilon \ , \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^M$ is the volume, $\mathbf{p} \in \mathbb{R}^N$ is the projection data, $\mathbf{A} \in \mathbb{R}^{N\times M}$ is the forward operator defined by the imaging geometry (cone-beam in this case), and $\epsilon \in \mathbb{R}^N$ is additive noise. Typically, recovering the volume $\mathbf{x}$ from the measured data $\mathbf{p}$ is an ill-posed inverse problem meaning that $\mathbf{A}$ is not square and there are multiple solutions for $\mathbf{x}$ which are consistent with the measured data $\mathbf{p}$. Hence, the reconstruction is formulated as a regularized optimization problem

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\{D(\mathbf{x},\mathbf{p}) + \mu R(\mathbf{x})\} \ . \tag{2}$$

Here, $D : \mathbb{R}^M \times \mathbb{R}^N \to \mathbb{R}$ is a function which measures the consistency of volume $\mathbf{x}$ and measured data $\mathbf{p}$. We choose the data consistency term in a least squares sense given as $D(\mathbf{x},\mathbf{p}) = \frac{1}{2}\|\mathbf{Ax}-\mathbf{p}\|_2^2$. $R : \mathbb{R}^M \to \mathbb{R}$ is a regularizer which helps finding a favorable solution $\mathbf{x}^*$ and is weighted against the data consistency term by a scalar $\mu \in \mathbb{R}$. Using a simple gradient descent optimization scheme, the resulting update formula is

$$\begin{aligned} \mathbf{x}^{n+1} &= \mathbf{x}^n - \lambda\nabla\{D(\mathbf{x},\mathbf{p}) + \mu R(\mathbf{x})\} \\ &= \mathbf{x}^n - \lambda(\mathbf{A}^T(\mathbf{Ax}-\mathbf{p}) + \mu\nabla R(\mathbf{x})) \ , \end{aligned} \tag{3}$$

where $\mathbf{A}^T \in \mathbb{R}^{M\times N}$ is the adjoint operator of $\mathbf{A}$, $\lambda \in \mathbb{R}$ is a sufficiently small step size, and n is the iteration index. To learn a flexible regularizer from data, we replace its gradient by a network $N_\theta : \mathbb{R}^M \to \mathbb{R}^M$ with free

parameters θ which allows to fit the regularizing component directly from data. The final update formula is given as

$$\mathbf{x}^{n+1} = \mathbf{x}^n - \lambda(\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{p}) + \mu N_\theta(\mathbf{x})) \ . \tag{4}$$

If $\mathbf{x}$ represents a 2D image, this equation could inspire an unrolled network architecture (Figure 1a) and the parameters θ can be trained from pairs of input projection data and ground truth reconstruction for a fixed number of unrolled iterations.[1] This is infeasible for 3D cone-beam data due to extremely high GPU memory requirements.

2.2 Neural Ordinary Differential Equations

Equation 4 has a residual form of the type

$$\mathbf{x}^{n+1} = \mathbf{x}^n + f_\theta(\mathbf{x}, \mathbf{p}) \ , \tag{5}$$

with $f_\theta(\mathbf{x}, \mathbf{p}) = -\lambda(\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{p}) + \mu N_\theta(\mathbf{x}))$. The function f_θ describes how the volume $\mathbf{x}^n$ changes incrementally. Chen et al.[12] proposed to regard such residual neural network architectures as the numerical integration of some underlying continuous ordinary differential equation. Here, the continuous differential equation would be $\frac{d\mathbf{x}}{dt} = f_\theta(\mathbf{x}(\mathbf{t}), \mathbf{p})$. Following that idea, a full unrolled iterative reconstruction is similar to the solution of an initial value problem of the given ODE starting from an initial condition $\mathbf{x}^0$ integrated until some end time T

$$\mathbf{x}^* = \mathbf{x}^0 + \int_0^T f_\theta(\mathbf{x}(\mathbf{t}), \mathbf{p}) \mathrm{dt} \ . \tag{6}$$

There exist a number of different numerical solvers to approximate solutions of such initial value problems. In this work, we use a fixed step-size Runge-Kutta solver of order 4 which integrates Eq. 6 by dividing the interval $[0, .., T]$ into a fixed number of steps S to solve the integral numerically. This highlights the analogy to residual networks of depth S.

To be able to combine this ODE-based problem formulation with a trainable network architecture, we need to compute a loss that is based on the output of the ODE solver and use its gradient to update the weights θ contained in f_θ. As demonstrated by Chen et al.,[12] this gradient can be obtained without backpropagating through the internals of the solver. Instead, the solver is regarded as a black box and the gradient with respect to θ is computed by solving another ODE backward in time (adjoint sensitivity method). This allows to obtain gradients with a memory cost that is independent of the number of steps S taken by the solver. Coming back to the analogy with residual networks, we can unroll the reconstruction problem in many steps using neural ODEs without further increasing the memory cost.

2.3 Network Architecture

In the neural ODE setting, a neural network defines the dynamics of the system, i.e., its temporal derivative. Following the classical problem formulation in Eq. 4, we design this network using two branches: (1) A data consistency branch with no trainable parameters incorporating the system's forward and backward model as known operator and (2) a regularization branch which is trained from data. Figure 1b illustrates the proposed network architecture. The data consistency branch implements the term $\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{p})$. Operators $\mathbf{A}$ and $\mathbf{A}^T$ are the CT forward and backprojection under the correct cone-beam geometry, respectively. We use the differentiable version of these operators described by Syben et al.[14] which computes analytical gradients and allows for a direct embedding of these operators in neural networks. For the regularization branch, we use a standard 3D U-Net[15] with depth 4 and 8 feature maps on the first level which are doubled in each stage, ReLU activation function, and instance normalization. In total, this leads to a network with 255 000 free parameters. We further introduce an additional single trainable parameter γ (referred to as data consistency weight) which is multiplied to the output of the data consistency branch before adding the output of both branches together in order to enable a data-optimal weighting between the data consistency and regularization component. This network together with the initialization $\mathbf{x}^0$ is passed to the ODE solver.

3. EXPERIMENTS

3.1 Data

The data set consists of 42 walnuts scanned under cone-beam CT geometry (cone angle: 40°).[16] It contains the raw projection data and a corresponding ground truth reconstruction. The projection images are acquired on three circular trajectories on different heights along the walnuts' long axes with a full rotation of 360° divided into 1200 angular steps each. The ground truth reconstruction is computed iteratively from the full set of acquired projections. As input to our algorithm, we use projection data from only the central one of the three trajectories and downsample the projections by a factor of 10 in angular direction and 2 in the spatial directions. This results in 120 projection images of size 384×486 pixels with an angular increment of 3° covering a full rotation of 360°. Hence, the algorithm has much less projection data available than has been used for computing the ground truth reconstruction which serves as learning target and is downsampled by a factor of 2 resulting in volumes of size 251^3. We use walnut number 1 for validation and walnut number 2 for testing. The rest is used for training. The used data, its preprocessing and the train-test-splits are identical to Rudzusika et al.[10]

3.2 Training Details

We use the neural ODE solver provided by Chen et al.[12] Integration of Eq. 6 is performed from 0 to $T = 1$ with a fixed step size of 0.05. This results in 20 steps and 80 evaluations of the network per forward and backward pass as the fourth order Runge-Kutta solver takes four evaluations per step. The initial volume $\mathbf{x}^0$ is a Feldmann-Davis-Kress (FDK) reconstruction of the projection data. The last layer of the U-Net is initialized with zeros such that the regularizer has no influence upon initialization and the data consistency weight is initialized with $\gamma = 0.01$. The parameters are optimized by an Adam optimizer with learning rate 1×10^{-4} for the U-Net and 1×10^{-2} for the data consistency weight. Training is performed with batch size 1 for 80 epochs and an L1-Loss evaluated only inside the cylindrical scan field of view (FOV) captured in each projection. The model weights corresponding to the lowest validation loss during training are selected for further evaluation.

3.3 Reference Methods

The reconstruction of the proposed method is compared to (1) an FDK reconstruction, (2) a SIRT reconstruction with non-negativity constraint (500 iterations)[17] and (3) a total variation (TV) regularized reconstruction using gradient updates (300 iterations).[14]

4. RESULTS

The proposed model requires a maximum of 13 GB GPU memory during training. Reconstruction results of the test walnut are shown in Figure 2. The FDK reconstruction exhibits strong streaking artifacts in the xy-plane as well as cone-beam artifacts at the top and bottom of the walnut visible in the xz- and yz-planes. The iterative SIRT algorithm only partly removes these artifacts. The TV regularized algorithm produces a very smooth image with homogeneous gray values for different structures inside the walnut. Nevertheless, cone-beam artifacts in the xz- and yz-plane are still visible. In contrast, our method achieves images with no noticeable cone-beam artifacts. Additionally, it performs well in removing the streaks in the xy-plane and produces images which are visually closest to the ground truth. The proposed method also performs best regarding all quantitative metrics (Tab. 1). Compared to the SIRT which is the second-best performing method, the RMSE is reduced by 31.7 % and PSNR and SSIM are increased by 11.1 % and 11.3 %, respectively. All metrics have only been evaluated inside the cylindrical scan FOV captured in each projection. Concerning reconstruction time, our method lies between the two investigated iterative methods with a run time of 43.4 s. The data consistency weight converges to a value of $\gamma = 0.036$.

5. DISCUSSION

Our proposed method is able to train a network inspired from classical iterative reconstruction for 3D cone-beam data incorporating a trainable regularizer with a GPU memory consumption which is independent of the number of incremental update steps on the volume. We can reconstruct volumes of practically relevant size (251^3) while using only 13 GB of GPU memory during training. This is feasible with a single recent consumer graphics card.

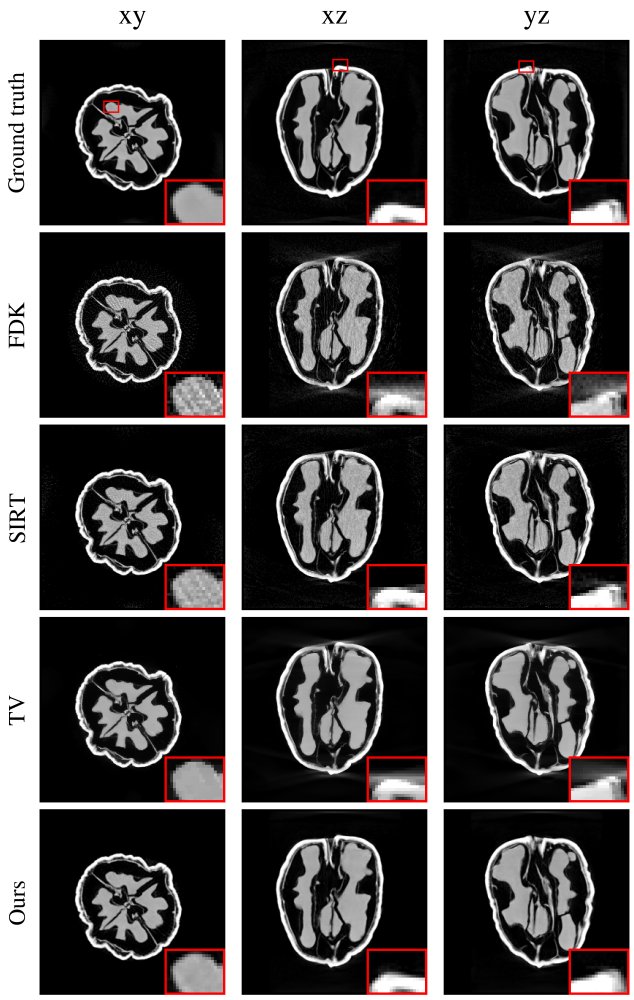


Figure 2: Reconstructed center slices of the test walnut along each dimension using an analytical FDK algorithm, two classical iterative algorithms (SIRT, TV) and the proposed method. The gray value window is 0 to $0.06\,\mathrm{mm}^{-1}$. Zoomed regions are indicated in red.

Table 1: Quantitative results in terms of root mean squared error (RMSE), peak signal-to-noise ratio (PSNR), and structural similarity index measure (SSIM).

	FDK	SIRT	TV	Ours
RMSE $[\cdot 10^{-3}] \downarrow$	3.735	2.321	2.688	1.586
PSNR $\uparrow$	25.682	29.813	28.539	33.121
SSIM $\uparrow$	0.562	0.813	0.777	0.904

The considered reconstruction problem is severely ill-posed due to the strong undersampling of projection data in angular direction. Hence, the analytical FDK reconstruction leads to strong artifacts in the reconstructed images. The trained algorithm removes the noise and streak artifacts successfully. While the classical TV-regularized iterative reconstruction also performs well in this regard, the proposed algorithm is the only one which is able to remove the cone-beam artifacts. We hypothesize that one main advantage of the learned regularizer parameterized by the U-Net over hand-crafted ones such as TV is its larger receptive field. It can suppress artifacts with non-local extent such as the cone-beam artifacts while TV depends only on the local gradient information in the image. A detailed comparison of our method to other learning-based approaches, such as,[10] will be performed in future work.

Once trained, the reconstruction time of the presented method is comparable to that of classical iterative algorithms. Training time is rather high due to the high number of network evaluations. Potentially, using an adaptive ODE solver instead of the fixed step size solver can shorten training and inference times by adaptively adjusting the step size and hence the number of network evaluations to a given tolerance.

6. CONCLUSION

This paper presents a method which uses neural ODEs to train a cone-beam reconstruction algorithm inspired by iterative reconstruction schemes. It ensures consistency with the measured data by incorporating a data consistency branch which exploits analytical knowledge about the physical operator connecting sinogram and image domain along with a trained regularizer. The proposed method outperforms well-known FDK and iterative reconstruction algorithms on the used walnut data set and is able to remove artifacts with non-local extent such as the cone-beam artifacts while being tractable concerning GPU memory.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC Grant No. 810316).

REFERENCES

[1] Chen, H., Zhang, Y., Chen, Y., Zhang, J., Zhang, W., Sun, H., Lv, Y., Liao, P., Zhou, J., and Wang, G., "LEARN: Learned experts' assessment-based reconstruction network for sparse-data CT," *IEEE Trans. Med. Imag.* **37**(6), 1333–1347 (2018).

[2] Jin, K. H., McCann, M. T., Froustey, E., and Unser, M., "Deep Convolutional Neural Network for Inverse Problems in Imaging," *IEEE Trans. Image Process.* **26**(9), 4509–4522 (2017).

[3] Adler, J., "Learned Iterative Reconstruction," *Handbook of Mathematical Models and Algorithms in Computer Vision and Imaging: Mathematical Imaging and Vision* , 1–22 (2021).

[4] Hammernik, K., Klatzer, T., Kobler, E., Recht, M. P., Sodickson, D. K., Pock, T., and Knoll, F., "Learning a variational network for reconstruction of accelerated MRI data," *Magn. Reson. Med.* **79**(6), 3055–3071 (2018).

[5] Gadjimuradov, F., Benkert, T., Nickel, M. D., and Maier, A., "Robust partial Fourier reconstruction for diffusion-weighted imaging using a recurrent convolutional neural network," *Magn. Reson. Med.* (2021).

[6] Vishnevskiy, V., Rau, R., and Goksel, O., "Deep Variational Networks with Exponential Weighting for Learning Computed Tomography," in [*Proc. MICCAI*], 310–318, Springer International Publishing (2019).

[7] Adler, J. and Öktem, O., "Learned Primal-Dual Reconstruction," *IEEE Trans. Med. Imag.* **37**(6), 1322–1332 (2018).

[8] Wu, D., Kim, K., and Li, Q., "Computationally efficient deep neural network for computed tomography image reconstruction," *Med. Phys.* **46**(11), 4763–4776 (2019).

[9] Hauptmann, A., Adler, J., Arridge, S., and Öktem, O., "Multi-scale learned iterative reconstruction," *IEEE Trans. Comput. Imag.* **6**, 843–856 (2020).

[10] Rudzusika, J., Bajic, B., Öktem, O., Schönlieb, C.-B., and Etmann, C., "Invertible Learned Primal-Dual," in [*Proc. NeurIPS*], (2021).

[11] Kellman, M., Zhang, K., Markley, E., Tamir, J., Bostan, E., Lustig, M., and Waller, L., "Memory-efficient learning for large-scale computational imaging," *IEEE Trans. Comput. Imag.* **6**, 1403–1414 (2020).

[12] Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D., "Neural ordinary differential equations," in [*Proc. NeurIPS*], 6572–6583 (2018).

[13] Chen, E. Z., Chen, T., and Sun, S., "MRI Image Reconstruction via Learning Optimization Using Neural ODEs," in [*Proc. MICCAI*], 83–93, Springer (2020).

[14] Syben, C., Michen, M., Stimpel, B., Seitz, S., Ploner, S., and Maier, A. K., "PYRO-NN: Python reconstruction operators in neural networks," *Med. Phys.* **46**(11), 5110–5115 (2019).

[15] Wolny, A., Cerrone, L., Vijayan, A., et al., "Accurate and versatile 3D segmentation of plant tissues at cellular resolution," *eLife* **9**, e57613 (2020).

[16] Der Sarkissian, H., Lucka, F., van Eijnatten, M., Colacicco, G., Coban, S. B., and Batenburg, K. J., "A cone-beam X-ray computed tomography data collection designed for machine learning," *Sci. Data* **6**(1), 1–8 (2019).

[17] van Aarle, W., Palenstijn, W. J., Cant, J., Janssens, E., Bleichrodt, F., Dabravolski, A., Beenhouwer, J. D., Batenburg, K. J., and Sijbers, J., "Fast and flexible X-ray tomography using the ASTRA toolbox," *Opt. Express* **24**, 25129–25147 (Oct 2016).