New GPU implementation of Separable Footprint (SF) Projector and Backprojector : first results

Camille Chapdelaine, Nicolas Gac, Ali-Mohammad Djafari, Estelle Parra-Denis

New GPU implementation of Separable Footprint (SF) Projector and Backprojector : first results

Camille Chapdelaine, Nicolas Gac, Ali Mohammad-Djafari and Estelle Parra

***Abstract*—Model-based iterative reconstruction methods enable to improve the quality of reconstruction in 3D X-ray Computed Tomography (CT). The main computational burden of these methods lies in successive projection and backprojection operations. Among existing pairs of projector and backprojector, Separable Footprint (SF) pair combines computational efficiency and accurate modelling of X-rays passing through the volume to image. In order to accelerate these operators, implementations on Graphical Processor Units (GPUs) for parallel-computing have been proposed for SF pair. Due to a CPU-loop, these implementations involve many memory transfers between CPU and GPU which are known to be the main bottleneck for GPU computing. In this paper, we investigate a new GPU implementation of SF projector and backprojector in order to minimize these memory transfers. Our proposed GPU SF projector and backprojector have no CPU-loop, and use two ray-driven kernels for the projection and one voxel-driven kernel for the backprojection. After having described their implementations, we study these operators as single modules and validate it in a MBIR method. Perspectives for this work are GPU optimizations and comparisons with the other existing implementations of SF pair.**

***Index Terms*—Computed Tomography, Separable Footprint, Graphical Processor Unit, iterative reconstruction methods**

I. Introduction

Compared with conventional filtered backprojection (FBP) methods, model-based iterative reconstruction (MBIR) methods have shown their advantages in terms of robustness and reconstruction quality [1]. Their major drawback is that they are highly computationally-costly due to successive projection and backprojection operations. The projection operator models the linear process of X-rays passing through the volume. The backprojector is defined as the mathematical adjoint of the projector.

Particularly in 3D applications, unmatched pairs of projector and backprojector (P/BP pairs) [2] have been widely used in order to alleviate the overall computational cost of iterative reconstruction methods. Although it has been found sufficient conditions for not having troubles when using an unmatched P/BP pair [2], these conditions have been derived for very simple iterative reconstruction methods which minimize the Euclidean distance between theoretical and measured projections by a gradient descent. For regularized least-squares using more sophisticated optimization algorithms, it has been highlighted in [3] that an unmatched P/BP pair can lead to suboptimal solutions or to divergence of the reconstruction algorithm. In order to avoid this mathematical approximation, computationally-efficient matched pairs have been recently proposed. The separable footprint (SF) pair [1] approximates the footprint of a voxel on the detector as a separable function which is trapezoidal in transaxial direction and rectangular in axial direction. The distance-driven (DD) pair [4] does the same kind of approximation, but models the footprint as rectangular both in transaxial and axial directions, which makes it less accurate than the SF pair [1].

To the best of our knowledge, two GPU implementations of SF P/BP pair have been proposed in [5], [6], the one of [6] having been shown faster than the one of [5]. Nevertheless, both these implementations have a CPU-loop on projection angles. This implies many memory transfers between CPU and GPU which are known to be the main bottleneck for accelerating computations on GPU. In order to minimize these memory transfers, in this paper, we investigate a new GPU implementation of SF P/BP pair and present first results. Our GPU implementation of SF projector is ray-driven and runs two independent kernels. Each of these kernels projects rays if the source is closer to x- or y-axis respectively. Our GPU SF backprojector is voxel-driven and runs only one kernel. These implementations have no CPU-loop.

In the following, we first present our SF projector and backprojector on GPU. Next, we analyze these operators as single modules, and validate it in simulation in a full MBIR method we detailed in [7]. Compared with an unmatched pair, we show the reconstruction converges in less global iterations with better convergence properties.

II. SF pair on GPU

A. SF projection

We consider a volume $\boldsymbol{f} = \{f(x_e, y_e, z_e)\}$ discretized in cubic voxels of side δ. Centers of voxels have normalized coordinates (x_e, y_e, z_e). On the detector, each cell's center has (u, v)-coordinates $(u_e\delta u, v_e\delta v)$. The volume is put between the detector and a source modeled as a point from which X-rays are sent in a conic beam. To acquire several perspectives, the source and the detector are rotated by an angle ϕ around z-axis. The rotation center has normalized (x, y, z)-coordinates $(x_{0_e}, y_{0_e}, z_{0_e})$. After X-rays have crossed the volume and

C. Chapdelaine is with Laboratoire des Signaux et des Systèmes (L2S) (CNRS–CentraleSupélec–Université Paris-Saclay) and SAFRAN SA, Safran Tech, Pôle Technologie du Signal et de l'Information (e-mail : camille.chapdelaine@l2s.centralesupelec.fr).

A. Mohammad-Djafari and N. Gac are with Laboratoire des Signaux et des Systèmes (L2S) (CNRS–CentraleSupélec–Université Paris-Saclay).

E. Parra is with SAFRAN SA, Safran Tech, Pôle Technologie du Signal et de l'Information.

reached the detector, the SF projection on a cell at angle ϕ reads

$$g(u_e, v_e, \phi) = l_{\theta_c}(u_e, v_e) \sum_{x_e, y_e} l_{\psi_v}(\phi; x_e, y_e) F_t(u_e, \phi; x_e, y_e) \times \sum_{z_e} F_a(v_e, \phi; x_e, y_e, z_e) f(x_e, y_e, z_e) \quad (1)$$

where $F_t(u_e, \phi; x_e, y_e)$ is the transaxial footprint and $F_a(v_e, \phi; x_e, y_e, z_e)$ the axial footprint of voxel (x_e, y_e, z_e) on cell (u_e, v_e) with projection angle ϕ [1]. Amplitude functions $l_{\theta_c}(u_e, v_e)$ and $l_{\psi_v}(\phi; x_e, y_e)$ are given by the A2 method in [1].

In order to avoid writing conflicts between threads, our GPU implementation of SF projector is ray-driven, i.e. one thread computes the SF projection of one ray defined by (u_e, v_e, ϕ). We follow each ray according to its primary direction, which is x-axis if the source is closer to x-axis, and y-axis otherwise [6]. Next, along this primary direction, we compute the voxels for which transaxial footprint and axial footprint are both non-zero. To ensure local memory accesses, the volume to project is copied on texture memory. Furthermore, variables related to the geometry of the acquisition are copied in constant memory.

Considering a ray with primary direction x or y leads to different calculations we need to do separately. That is why our SF projector runs two kernels. First kernel $proj\_x\_ker$ projects rays with primary direction x, while second kernel $proj\_y\_ker$ handles rays with primary direction y. These two kernels are independent since they compute disjoint sets of projections. In this paper, they are executed successively on one GPU but their parallel applications on two different GPUs can be considered.

For a ray with primary direction x, the corresponding thread performs a loop on x_e, $1 \leq x_e \leq N_x$. For each x_e, it computes the intersecting location $(x_e, y_e(x_e))$ with the ray, similarly to Joseph's method [8] :

$$y_e(x_e) = \frac{1}{\delta}\left(y_s(\phi) + \frac{y(u_e, \phi) - y_s(\phi)}{x(u_e, \phi) - x_s(\phi)}(x_e\delta - x_s(\phi))\right) \quad (2)$$

where $(x_s(\phi), y_s(\phi))$ is the (x, y)-location of the source and $(x(u_e, \phi), y(u_e, \phi))$ is the (x, y)-location of the center of cell (u_e, v_e) at projection angle ϕ. Then, for current x_e, the thread looks the voxels of which projections onto the median plane are between $(x_e, y_e(x_e) - 1)$ and $(x_e, y_e(x_e) + 1)$, i.e. pixels (x_e, y_e) of which left side is before $(y_e(x_e) + 1)$ and right side after $(y_e(x_e) - 1)$. Hence, the thread looks each y_e, such that

$$y_{e_{min}} \leq y_e \leq y_{e_{max}}, \begin{cases} y_{e_{min}} = \lfloor y_e(x_e) - 1.5 \rfloor \\ y_{e_{max}} = \lceil y_e(x_e) + 1.5 \rceil \end{cases} . \quad (3)$$

This makes a loop on y_e for each thread of kernel $proj\_x\_ker$ which is very small (typically size 4 or 5). For rays with primary direction y, the calculations are the same by reverting roles of x and y : the main loop is on y_e and we have a second loop on x_e, $x_{e_{min}} \leq x_e \leq x_{e_{max}}$. Here, we see the interest of dealing with rays with different primary directions in different kernels, in order to avoid divergence between threads.

In the double loop over (x_e, y_e) (with y_e or x_e varying in a very little set depending on the executed kernel), for each considered (x_e, y_e), we calculate the scaled transaxial footprint

$$F'_t(u_e, \phi; x_e, y_e) = l_{\psi_v}(\phi; x_e, y_e) F_t(u_e, \phi; x_e, y_e) \quad (4)$$

as described in [1]. Next, we find indices z_e for which $F_a(v_e, \phi; x_e, y_e, z_e) \neq 0$. Thanks to the chosen rectangular shape of the axial footprint, these indices are very simple to compute :

$$z_{e_{min}} \leq z_e \leq z_{e_{max}} \quad (5)$$

where

$$\begin{cases} z_{e_{min}} = \lfloor z_{0_e} - 0.5 + \frac{x_{\phi_e}\delta v}{D}(v_e - v_{c_e} - 0.5) \rfloor \\ z_{e_{max}} = \lceil z_{0_e} + 0.5 + \frac{x_{\phi_e}\delta v}{D}(v_e - v_{c_e} + 0.5) \rceil \end{cases} , \quad (6)$$

where D is the source-to-detector distance, and

$$x_{\phi_e} = \frac{R}{\delta} + (x_e - x_{0_e})\cos\phi + (y_e - y_{0_e})\sin\phi \quad (7)$$

where R is the source-to-rotation center distance. Knowing the bounds for z_e, threads run a loop on z_e to compute

$$F'_a(v_e, \phi; x_e, y_e) = \sum_{z_e = z_{e_{min}}}^{z_{e_{max}}} F_a(v_e, \phi; x_e, y_e, z_e) f(x_e, y_e, z_e). \quad (8)$$

This loop is very small and is typically size 3. Iteratively, through the double loop over x_e and y_e, threads calculate the sum

$$g'(u_e, v_e, \phi) = \sum_{x_e} \sum_{y_e = y_{e_{min}}}^{y_{e_{max}}} F'_t(u_e, \phi; x_e, y_e) F'_a(v_e, \phi; x_e, y_e) \quad (9)$$

in kernel $proj\_x\_ker$, and

$$g'(u_e, v_e, \phi) = \sum_{y_e} \sum_{x_e = x_{e_{min}}}^{x_{e_{max}}} F'_t(u_e, \phi; x_e, y_e) F'_a(v_e, \phi; x_e, y_e) \quad (10)$$

in kernel $proj\_y\_ker$. Finally, the thread handling ray (u_e, v_e, ϕ) computes the final value for the projection

$$g(u_e, v_e, \phi) = l_{\theta_c}(u_e, v_e) g'(u_e, v_e, \phi), \quad (11)$$

which is stored from GPU to CPU.

B. SF backprojection

Because SF projector and backprojector are matched, for a voxel (x_e, y_e, z_e), the SF backprojection is

$$b(x_e, y_e, z_e) = \sum_{\phi} \sum_{u_e} F_t(u_e, \phi; x_e, y_e) l_{\psi_v}(\phi; x_e, y_e) \times \sum_{v_e} F_a(v_e, \phi; x_e, y_e, z_e) l_{\theta_c}(u_e, v_e) g(u_e, v_e, \phi). \quad (12)$$

To prevent from writing conflicts between threads, we compute volume $\boldsymbol{b}$ by running a kernel $back\_ker$ which is voxel-driven : one thread calculates the backprojection of one voxel (x_e, y_e, z_e). Kernel $back\_ker$ has a main loop on projection angles ϕ. For each projection angle, a thread finds cells (u_e, v_e) overlapped by transaxial and axial footprints of voxel (x_e, y_e, z_e). u_e-coordinates of these cells are given by ordering

the projections of the four corners of pixel (x_e, y_e) in the middle plane, $\tau_0 \leq \tau_1 \leq \tau_2 \leq \tau_3$ [1] :

$$u_{e_{min}} \leq u_e \leq u_{e_{max}}, \begin{cases} u_{e_{min}} = \lfloor u_{c_e} - 0.5 + \frac{\tau_0}{\delta u} \rfloor \\ u_{e_{max}} = \lceil u_{c_e} + 0.5 + \frac{\tau_3}{\delta u} \rceil \end{cases} . \quad (13)$$

The set of v_e for which axial footprint $F_a(v_e, \phi; x_e, y_e, z_e) \neq 0$ is

$$v_{e_{min}} \leq v_e \leq v_{e_{max}}, \begin{cases} v_{e_{min}} = \lfloor v_{c_e} - 0.5 + \frac{\chi_0}{\delta v} \rfloor \\ v_{e_{max}} = \lceil v_{c_e} + 0.5 + \frac{\chi_1}{\delta v} \rceil \end{cases} \quad (14)$$

where $\chi_0 \leq \chi_1$ are the projections of voxels $(x_e, y_e, z_e - 0.5)$ and $(x_e, y_e, z_e + 0.5)$ respectively [1].

Next, double-loop on u_e and v_e is performed to compute the backprojection of voxel (x_e, y_e, z_e). The size of this double-loop is approximately the same for each voxel [5]. Projections $\boldsymbol{g}$ are copied on texture memory, so local memory accesses are ensured. Because of the separation of the voxel's footprint between transaxial and axial directions, the double-loop can be done on u_e then v_e or on v_e then u_e indifferently. Here, we chose the loop on u_e as main loop. For each u_e between $u_{e_{min}}$ and $u_{e_{max}}$, we run a loop on v_e, $v_{e_{min}} \leq v_e \leq v_{e_{max}}$. Like for z_e-loop in SF projector, this loop is typically size 3 and calculates each axial footprint to return

$$b_a(u_e, \phi; x_e, y_e, z_e) = \sum_{v_e=v_{e_{min}}}^{v_{e_{max}}} F_a(v_e, \phi; x_e, y_e, z_e) l_{\theta_c}(u_e, v_e) g(u_e, v_e, \phi). \quad (15)$$

Hence, the double-loop performs the summation

$$b_\phi(\phi; x_e, y_e, z_e) = \sum_{u_e=u_{e_{min}}}^{u_{e_{max}}} F_t(u_e, \phi; x_e, y_e) \times l_{\psi_v}(\phi; x_e, y_e) b_a(u_e, \phi; x_e, y_e, z_e) \quad (16)$$

Then, the backprojection can be updated

$$b(x_e, y_e, z_e) += b_\phi(\phi; x_e, y_e, z_e) \quad (17)$$

until all the projection angles have been considered.

III. First results on simulated data

In this section, we present first results of our GPU implementation of SF pair on a volume with 256^3 voxels. We use 64 projections of this volume uniformly distributed over $[0, 2\pi]$. Each projection has 256^2 pixels. For our tests, we use only one GPU, which is a NVIDIA's GeForce GTX TITAN X. We analyze our GPU SF P/BP as single modules and in a full MBIR method.

A. GPU SF projector and backprojector as single modules

Like [6], we show in table I the normalized root mean square error (NRMSE) with respect to our CPU version, which has been implemented following [1]. The NRMSE is

$$NRMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i^{(GPU)} - y_i^{(CPU)}}{y_i^{(CPU)}} \right)^2}, \quad (18)$$

where y_i denotes projection or backprojection for ray or voxel i respectively. We see NRMSE is very low for both projector and backprojector, so our GPU implementation implies no deviation with respect to our CPU version. Furthermore, the coupling degree, introduced in [3], of our GPU SF pair is 1.0005, i.e. very close to 1, which means that our GPU SF P/BP pair is very well matched [3].

In table I, we also measure the number of used registers per thread. Because each of our kernels do many calculations, this number is quite high. Because the number of registers per block is limited, dimensioning thread blocks of the GPU must be done very carefully. The size of each block is $16 \times 16 \times 1$ for our SF projector, as for the backprojector. The size of the grid is $16 \times 16 \times 64$ for the projector and $16 \times 16 \times 256$ for the backprojector.

Lastly, we give in table I the computation times for our GPU SF projector and backprojector. We may underline that the presented version is the first one. Further optimizations, such as using multiple GPUs or finding a way to merge kernels $proj\_x\_ker$ and $proj\_y\_ker$ for the projection, are still needed and are undergoing works.

Operator	Computation time		NRMSE (%)	Registers/thread
	CPU	GPU		
Projector	143.9 s	8.3 s	1.2×10^{-4}	84
Backprojector	98.7 s	4.4 s	3.2×10^{-5}	63

TABLE I: Proposed GPU SF projector and backprojector as single modules

B. GPU SF pair in full MBIR algorithm

In order to fully validate our GPU implementation, we now test our GPU SF projector and backprojector in a full MBIR algorithm we presented in [7] with an unmatched pair. This algorithm performs alternate reconstruction and segmentation until convergence, and maximizes the joint posterior distribution of volume $\boldsymbol{f}$ and hyperparameters $\boldsymbol{\theta}$ of the prior models defined for the volume and the uncertainties on the projections. These prior models are detailed in [7]. Hyperparameters $\boldsymbol{\theta}$ are estimated jointly with the volume. According to Bayes's rule and by removing constant terms from the log-joint posterior distribution of $\boldsymbol{f}$ and $\boldsymbol{\theta}$, the stop criterion to maximize is

$$\mathcal{L}(\boldsymbol{f}, \boldsymbol{\theta}) = \ln(p(\boldsymbol{g}|\boldsymbol{f}, \boldsymbol{\theta})) + \ln(p(\boldsymbol{f}|\boldsymbol{\theta})) + \ln(p(\boldsymbol{\theta})). \quad (19)$$

We run the algorithm with our matched SF pair on GPU and compare the obtained results with the unmatched pair we used in [7]. This unmatched pair is described in [7], [9]. In order to compare the two considered P/BP pairs, for both experiments with matched and unmatched pair, we use same dataset, initialization and parameters, given in [7]. The algorithm has a maximum number of global iterations fixed to 20 and can be stopped before if the criterion (19) does not change by more than 10^{-6} % between two global iterations.

The results are shown in figure 1. We see the quality of reconstruction is good for both P/BP pairs : this validates our GPU implementation of SF P/BP pair. The obtained reconstructions with matched and unmatched pairs look visually the same : the SSIM [10] computed by MATLAB is approximately 1. Furthermore, the $\mathcal{L}_2$-relative error with respect to the original phantom, in table II, is approximately the same for both pairs. Nevertheless, as shown in table II,

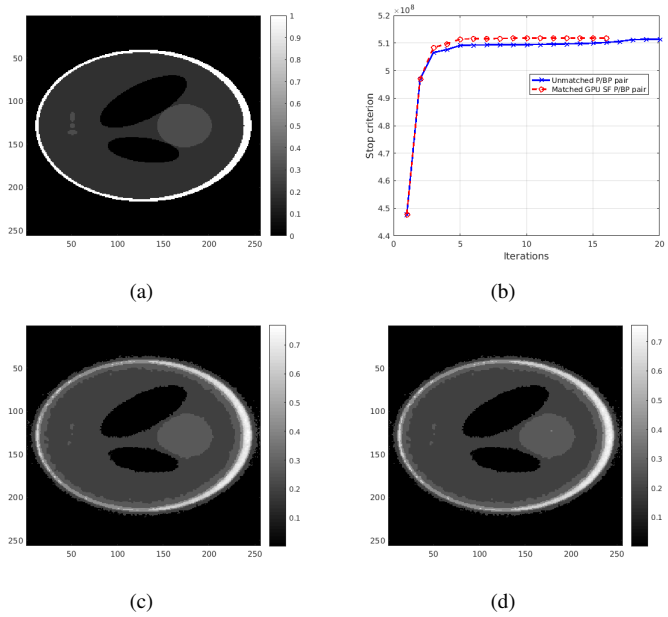

Fig. 1: Reconstruction of Shepp-Logan phantom (a) by the same MBIR method [7], with unmatched P/BP pair (c) and proposed matched GPU SF pair (d). Convergences are shown in (b).

the computation time is much longer when using matched SF pair, since it requires more calculations.

To deepen the comparison, we also analyze the convergence of the MBIR method. In table II and figure 1, we see the algorithm with matched SF pair converges in 16 global iterations, while with unmatched P/BP pair, it reaches the maximum number of 20. Moreover, in figure 1, the convergence of the algorithm is shown to be much smoother when using matched SF pair. In table II, the comparison of the final value of the stop criterion (19) shows that it is greater when using matched SF pair : that means the reconstruction obtained with an unmatched pair is suboptimal, as it has been already noticed in [3].

To conclude this section, we may emphasize that our GPU implementation of SF pair has been fully validated by its use in a MBIR method we developed with another P/BP pair [7]. The comparison between these two pairs has shown that using matched SF pair is more computationally intensive than using an unmatched pair, but ensures better convergence properties.

Used P/BP pair	Computation Time	$\mathcal{L}_2$-relative error	Number of global iterations	Final value of the criterion ($\times 10^8$)
Unmatched	**629.3 s**	**18.5 %**	20	5.1136
GPU SF	6517.0 s	18.7 %	**16**	**5.1183**

TABLE II: Comparison of the results for a MBIR method [7] with unmatched P/BP pair and proposed GPU SF pair

IV. Conclusion and perspectives

In this work, we have investigated a new GPU implementation of Separable Footprint (SF) projector and backprojector (P/BP) which minimizes memory transfers between CPU and GPU. We have presented a ray-driven GPU SF projector with two independent kernels which handle rays depending on their primary direction. Concerning SF backprojector, our GPU implementation is voxel-driven and uses only one kernel which takes advantage of the separability of the footprint in SF pair. Both our GPU SF projector and backprojector have no CPU-loop, so memory transfers are minimized.

As first results, we have fully validated the proposed GPU implementation. By computing the coupling degree [3], we have shown that our GPU SF implementation is well-matched. Our proposed implementation of SF pair has been shown to obtain very good results in a Model-Based Iterative Reconstruction (MBIR) algorithm. Compared with an unmatched P/BP pair, we have emphasized that the use of matched SF pair provides better convergence properties and accelerates the convergence in terms of global iterations.

Nevertheless, we have also seen the computational cost when using a matched pair is much higher, and results in a much longer computation time, as highlighted in table II. To reduce this computation time, further GPU optimizations are still necessary, such as, for instance, executing the two kernels of our GPU SF projector on two different GPUs, which is possible since these two kernels are independent. Moreover, our experiments have been done on relatively small volumes. Consequently, scaling our implementation for much larger volumes, as the aforementioned optimizations, is a perspective for this work. After having proceeded with these optimizations, comparisons with previous GPU implementations of SF P/BP pair [5], [6] will remain to be done.

References

[1] Y. Long, J. A. Fessler, and J. M. Balter, "3D forward and back-projection for X-ray CT using separable footprints," *IEEE transactions on medical imaging*, vol. 29, no. 11, pp. 1839–1850, 2010.

[2] G. L. Zeng and G. T. Gullberg, "Unmatched projector/backprojector pairs in an iterative reconstruction algorithm," *IEEE transactions on medical imaging*, vol. 19, no. 5, pp. 548–555, 2000.

[3] F. Arcadu, M. Stampanoni, and F. Marone, "On the crucial impact of the coupling projector-backprojector in iterative tomographic reconstruction," *arXiv preprint arXiv:1612.05515*, 2016.

[4] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," *Physics in medicine and biology*, vol. 49, no. 11, p. 2463, 2004.

[5] M. Wu and J. A. Fessler, "GPU acceleration of 3D forward and backward projection using separable footprints for X-ray CT image reconstruction," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, vol. 6. Citeseer, 2011, p. 021911.

[6] X. Xie, M. G. McGaffin, Y. Long, J. A. Fessler, M. Wen, and J. Lin, "Accelerating separable footprint (SF) forward and back projection on GPU," in *SPIE Medical Imaging*. International Society for Optics and Photonics, 2017, pp. 101 322S–101 322S.

[7] C. Chapdelaine, A. Mohammad-Djafari, N. Gac, and E. Parra, "A 3D Bayesian Computed Tomography Reconstruction Algorithm with Gauss-Markov-Potts Prior Model and its Application to Real Data," *Fundamenta Informaticae*, vol. 155, no. 4, pp. 373–405, 2017.

[8] P. M. Joseph, "An improved algorithm for reprojecting rays through pixel images," *IEEE transactions on medical imaging*, vol. 1, no. 3, pp. 192–196, 1982.

[9] N. Gac, A. Vabre, A. Mohammad-Djafari, A. Rabanal, and F. Buyens, "GPU implementation of a 3D Bayesian CT algorithm and its application on real foam reconstruction," in *The First International Conference on Image Formation in X-Ray Computed Tomography*, 2010, pp. 151–155.

[10] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.