A versatile tomographic forward- and backprojection approach on Multi-GPUs

Andreas Fehringer[a], Tobias Lasser[b,c], Irene Zanette[a], Peter B. Noël[d] and Franz Pfeiffer[a]

[a]Chair of Biomedical Physics (E17), Technische Universität München,
James-Franck-Str. 1, Garching, 85748, Germany
[b]Chair for Computer Aided Medical Procedures (I-16), Technische Universität München,
Boltzmannstr. 3, Garching, 85748, Germany
[c]Institute for Biomathematics and Biometry, Helmholtz Zentrum München,
Building 58a, Ingolstädter Landstr. 1, Neuherberg, 85764, Germany
[d]Department of Radiology, Technische Universität München,
Ismaninger Straße 22, München, 81675, Germany

ABSTRACT

Iterative tomographic reconstruction gets more and more into the focus of interest for x-ray computed tomography as parallel high-performance computing finds its way into compact and affordable computing systems in form of GPU devices. However, when it comes to the point of high-resolution x-ray computed tomography, e. g. measured at synchrotron facilities, the limited memory and bandwidth of such devices are soon stretched to their limits. Especially keeping the core part of tomographic reconstruction, the projectors, both versatile and fast for large datasets is challenging. Therefore, we demonstrate a multi-GPU accelerated forward- and backprojector based on projection matrices and taking advantage of two concepts to distribute large datasets into smaller units. The first concept involves splitting up the volume into chunks of slices perpendicular to the axis of rotation. The result is many perfectly independent tasks which then can be solved by distinct GPU devices. A novel ultrafast precalculation kernel prevents unnecessary data transfers for cone-beam geometries. Datasets with a great number of projections can additionally take advantage of the second concept, a split-up into angular wedges. We demonstrate the portability of our projectors to multiple devices and the associated speedup on a high-resolution liver sample measured at the synchrotron. With our splitting approaches, we gained factors of 3.5 – 3.9 on a system with four and 7.5 – 8.0 with eight GPUs. The computing time for our test example decreased from 23.5 s to 2.94 s in the latter case.

Keywords: tomography, forwardprojector, backprojector, multiple GPUs, versatile, large data

1. DESCRIPTION OF PURPOSE

Tomographic imaging modalities cover a broad spectrum of length scales and competing acquisition approaches. One example is phase-contrast CT at the synchrotron, a technique that provides high-resolution in combination with significantly improved soft-tissue contrast compared to absorption-based techniques. Each approach, however, has special demands concerning geometry, data size and reconstruction algorithm.

The motivation for our investigations was to develop a versatile scheme for processing datasets from very different geometries and sizes efficiently. After a brief summary of how to use projection matrices for describing tomographic cone-beam geometries very generally, we explain the course of action of the GPU kernels for back- and forwardprojection as well as some memory considerations. The focus of our work, however, lies on the two concepts we use to make the projector capable of handling large datasets and using multiple GPUs. A comparison of the benchmarks for the forward- and backprojection of a high-resolution phase-contrast synchrotron dataset on different single- and multi-GPU hardware setups will show the capabilities of our approach.

Medical Imaging 2014: Image Processing, edited by Sebastien Ourselin,
Martin A. Styner, Proc. of SPIE Vol. 9034, 90344F · © 2014 SPIE
CCC code: 1605-7422/14/$18 · doi: 10.1117/12.2043860

2. METHOD

2.1 The General concept of the implementation

The basic implementation of forward- and backprojector was originally based on the work of R. Bippus et al.,[1] but further developed and adapted to work with projection matrices as suggested by Galigekere et al.[2] We decided to use bilinear hardware interpolation and OpenCL instead of the more common CUDA™ framework because it provides support for more devices. All computations are carried out in single floating point precision.

2.1.1 Projection matrices for tomographic projections

Projection matrices are used to perform central projections of points in a 3 D setup onto a 2 D plane. The following section shows how to take advantage of them for tomographic reconstruction.

Each single projection of a scan can be described with one corresponding 3×4 projection matrix P that is a direct transformation from voxel indices in the image volume to pixel indices on the detector. If $\boldsymbol{X} = (X, Y, Z, 1)^{\mathrm{T}}$ is a homogeneous voxel coordinate, the corresponding detector pixel coordinate $\boldsymbol{x} = (x, y, 1)^{\mathrm{T}}$ is obtained by

$$\boldsymbol{x}\, w = P\, \boldsymbol{X}$$

where P is the projection matrix for the corresponding projection angle and w is the homogenization factor.

Vice versa, tracing a ray

$$\boldsymbol{X}(\lambda) = \boldsymbol{X}_0 + \lambda\, \boldsymbol{U}_0$$

through the image is possible by varying the parameter λ where $\boldsymbol{X}_0\, W = P^{+}\, \boldsymbol{x}$ (W is the homogenization factor) and $\boldsymbol{U}_0 = \boldsymbol{X}_0 - \tilde{\boldsymbol{S}}$ is the stepping vector along the ray. The position of the rotated source $\tilde{\boldsymbol{S}}$ can be obtained by solving $P\, \tilde{\boldsymbol{S}} = 0$. For the implementation of a ray caster, the main direction of propagation of the ray through the image volume can be determined by $d_{\mathrm{main}} = \arg\max_i \left| \left\{ \boldsymbol{m}^3 \right\}_i \right|$ where $\boldsymbol{m}^3$ is the principle axis vector which is built up from the first three elements of the third row of the projection matrix P. If the stepping vector $\boldsymbol{U}_0$ is normalized to 1 along the main direction of propagation and the image voxel indices in this direction are in the range of $\left[0, \{\boldsymbol{I}\}_{d_{\mathrm{main}}}\right)$, then $\lambda \in \left[-\{\boldsymbol{X}_0\}_{d_{\mathrm{main}}}, \{\boldsymbol{I}\}_{d_{\mathrm{main}}} - \{\boldsymbol{X}_0\}_{d_{\mathrm{main}}}\right)$. Further details about projection matrices can be found in the appendix and in the book of Hartley et. al.[3]

2.1.2 The backprojector

For both, the back- and the forwardprojector, the input arrays are kept in the partly cached 3 D texture memory to take advantage of its short access times and hardware interpolation feature. The output arrays are held in global memory.

Each thread of the backprojector computes the intensity in one voxel. Starting from its corresponding position given in image voxel coordinates $\boldsymbol{X}$, it computes the projection points $\boldsymbol{x}$ on the detector for each projection matrix P as described in the previous section. In the end, the thread pushes or adds the locally summed up intensities to global memory.

For the last step, the dimensions of the array holding the image volume are made a multiple of the OpenCL workgroup size of $8\,\mathrm{vx} \times 8\,\mathrm{vx} \times 8\,\mathrm{vx}$ so that no boundary checks are necessary. The fastest dimension is further forced to become a multiple of 16 (that is 64 Bytes) in order to guarantee coalesced access to global memory for all threads. The cubic workgroup dimensions are designed to keep the covered area on the detector texture as compact and symmetric as possible because all points within the spacial neighborhood are automatically cached. All projection matrices and parameters are held in constant memory because it is fully cached and optimized for concurrent access of all threads.

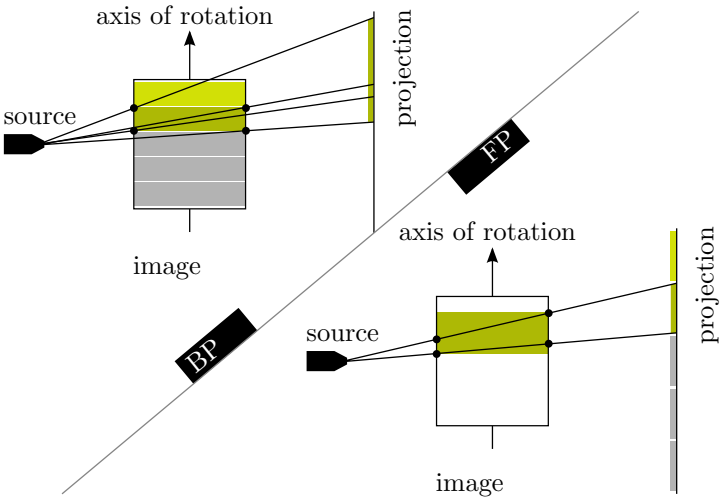

Figure 1. **An illustration for the concept of slice chunks.**
In a preliminary step, all corners of the output volume corresponding to one chunk are projected to the input volume to figure out the minimum and maximum detector row or image slice required.
The upper schematic illustrates the computation for the **BP**. The eight corners of the image chunk (labeled with black dots) are projected to the detector to determine the required slices. For the **FP** the four corners of the detector chunk are back projected into the image volume. The deciding entry points are marked each with a black dot.

2.1.3 The forwardprojector

The forwardprojector operates on the pixels of the detector under each angle. The dimensions of the array holding the projections are enlarged in the same way as those of the image volume above. Each thread traces the ray $\boldsymbol{X}(\lambda)$ through the source computed from the corresponding detector coordinate $\boldsymbol{x}$ as shown in 2.1.1. The fastest choice for the workgroup size turned out to be 32 angles $\times$ 8 px $\times$ 2 px.

To speed up the computation of the stepping vector $\boldsymbol{U}_0$ and the starting point $\boldsymbol{X}(\lambda_{\min})$ in the beginning, precomputed arrays in constant memory hold the source positions $\tilde{\boldsymbol{S}}$, the main directions d_{main} and the pseudo-inverse projection matrices P^+ for each angle. The projection matrices P itself are not required in the forwardprojection kernel. Analog to the backprojector, there is only one explicit access to global memory at the very end in order to store the sum of the locally collected intensities along the ray.

2.2 Concepts to handle large datasets

Non-clinical datasets, e. g. taken at a synchrotron, can have large detector arrays or numbers of projections that would cause an overflow of constant or global memory if processed all at once. We spent much effort in developing and implementing the following two concepts in order to split up large datasets into several chunks without loosing performance. Both concepts work for the forward- as well as for the backprojector. A clever division into chunks is also essential to distribute work over multiple GPUs or concurrently transfer data to the device while computing on other data.

2.2.1 The concept of slice chunks

The first, newly developed concept involves splitting up a whole stack of image slices for a backprojection and detector rows for a forwardprojection, respectively, into chunks of a predefined height. Therefore, we introduce an additional GPU kernel running before the projection itself. It projects the positions of the corners defining the chuck subvolumes to the output coordinate system, i. e. the image space for the backprojector and the projection space for the forwardprojector, respectively (Figure 1). Each thread cares for the projection of one corner under one angle. Afterwards, a reduction algorithm makes it possible to use the GPU also for finding the corresponding minimum and maximum required slice or row of the output volume for each chunk. This way, only the parts of the volumes have to be transferred to the device that are really necessary for the projection of one chunk saving transfer time and memory usage. The concept of slice chunks also ensures that the maximum possible measure of the sample along the axis of rotation is not limited by memory, even not for cone-beam reconstructions.

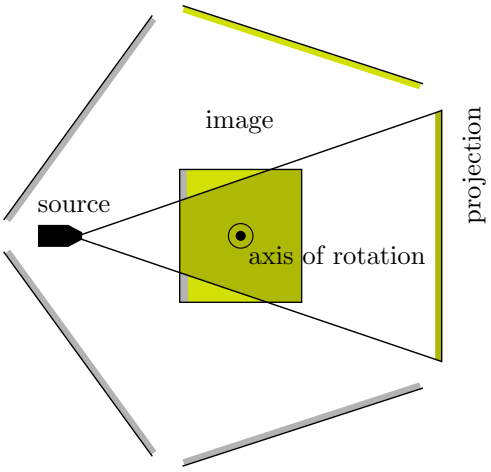

Figure 2. **A schematic of the concept of angular wedges.**
If there are many projections, only a part of them are processed at once. The maximum wedge size can automatically be computed from the size of the available texture and constant memory. In the BP all wedges are preferably computed on the same GPU device. This way the image volume only needs to be transferred once.

2.2.2 The concept of angular wedges

The second concept ensures that all angular parameters fit into the very size-limited constant memory, as described in 2.1.2 and 2.1.3, and that the projection data does not overflow the GPU memory. The maximum limit for a constant memory of 64 kByte is 1365 projection matrices for a backprojection (12 floats for each P) and 1024 for a forwardprojection (15 floats and 1 integer for each set of P^+, $\tilde{\boldsymbol{S}}$, and d_{main}). The maximum limit for the projection data is given by the maximum allocation size in global memory depending on the GPU device. If a dataset exceeds one of the two limits, the angular splitting feature performs a whole projection for a subset of angles first and then continues processing the next subset as shown in Figure 2. It is convenient to perform the subsequent projections of angular subsets on the same GPU device so the image volume does not have to be transferred back and forth.

3. RESULTS

Figure 3 shows the benchmarks for the suggested forward- and backprojector for two different hardware setups, one providing four and the other one eight GPU devices, and each for two different data sizes. Each benchmark was measured once with all available GPUs and once on a single GPU in order to obtain the speedup and overhead produced by the suggested data splitting.

Independent from the hardware setup and data size, the speedup was close to the number of GPU devices meaning that the overhead from our split-up approaches is very small. Some of the little overhead that can be seen despite, can be explained by the fact that the uneven number of slices was not completely divisible by the workchunk size of 16 px for the backprojector and 24 px for the forwardprojector.

4. NEW OR BREAKTHROUGH WORK TO BE PRESENTED

We presented an implementation of a forward- and a backprojector including sophisticated considerations of how to split up data efficiently. The great advantage of the concept of slice chunks shown is that all chunks are completely independent from each other and can perfectly be distributed over multiple GPUs. Our profiling showed that the required preliminary kernel only takes less than a millisecond of additional computing time for the profitable precomputations.

Both, the splitting into slice chunks and the splitting into angular wedges allow concurrently fetching the results from a previous work unit and pushing data for the next one while computing.

5. CONCLUSION

We introduced an implementation of a multi-GPU tomographic forward- and backprojector which is designed to handle each setup geometry that can be represented by projection matrices.

With the help of an example from synchrotron X-ray tomography, we also showed that utilizing multiple GPUs is a great asset for tomographic projectors when distributing the data in a clever way. Therefore, we introduced the concept of slice chunks which can be calculated without loss of computation time and are able to subdivide both forward- and backprojection into really distinct work units. For datasets with many projections, the concept of angular wedges can additionally be applied in order to stay within the memory limits of one device.

Further improvements that might be investigated is using half floats instead of floats in order to increase the number of cached texels and minimize the transfers. Furthermore, a comparison of the OpenCL to a CUDA version of the implementation would be valuable.

With projector designs such as ours it will be possible to develop in the near future far more advanced iterative reconstruction algorithms especially for large amounts of data.

REFERENCES

1. R. Bippus, T. Koehler, F. Bergner, B. Brendel, E. Hansis, and R. Proksa, "Projector and backprojector for iterative CT reconstruction with blobs using CUDA," *Proceedings of Fully 3 D* **3**, pp. 68–71, 2011.
2. R. Galigekere, K. Wiesent, and D. Holdsworth, "Cone-beam reprojection using projection-matrices," *Medical Imaging, IEEE Transactions on* **22**(10), pp. 1202–1214, 2003.
3. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2011.
4. P. B. Noël, J. Herzen, A. A. Fingerle, M. Willner, M. K. Stockmar, D. Hahn, M. Settles, E. Drecoll, I. Zanette, T. Weitkamp, E. J. Rummeny, and F. Pfeiffer, "Evaluation of the potential of phase-contrast computed tomography for improved visualization of cancerous human liver tissue," *Zeitschrift für Medizinische Physik* **23**(3), pp. 204–211, 2013.

	Hardware	Data size	Single GPU	Multi GPU	Speedup
FP	4× Nvidia® Titan	full	15.07 s	4.28 s	3.5×
		reduced	7.50 s	2.00 s	3.7×
	4× Nvidia® Tesla K10 (2 GPUs each)	full	62.53 s	7.87 s	7.9×
		reduced	23.55 s	2.94 s	8.0×
BP	4× Nvidia® Titan	full	32.65 s	8.40 s	3.9×
		reduced	16.30 s	4.30 s	3.8×
	4× Nvidia® Tesla K10 (2 GPUs each)	full	71.36 s	9.47 s	7.5×
		reduced	36.05 s	4.82 s	7.5×

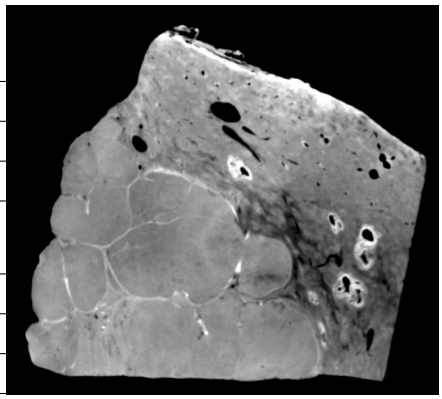

Figure 3. **A comparison of the computation times between a single GPU and multiple GPUs taking advantage of the suggested slice-wise and angle-wise data splitting.**
The measurements were carried out on a system containing four of the very recent Nvidia® Titan graphics cards as well as on a system holding four Nvidia® Tesla K10 devices which have two GPUs each. The time values given include all splitting precalculations, transfers and kernel executions. The two data sizes are the full image volume size of $\left(373 \times 1376^2\right)$ vx^3 and a reduced size of $\left(373 \times 976^2\right)$ vx^3 which is more adapted to the actual size of the liver sample. Each computation considered 1200 projection angles.
The image on the right is a slice of the reconstructed liver sample. The phase-contrast dataset was kindly provided for testing by P. B. Noël et al.[4]

APPENDIX A. COMPUTING THE PROJECTION MATRICES FROM THE MEASURES IN A CT SETUP

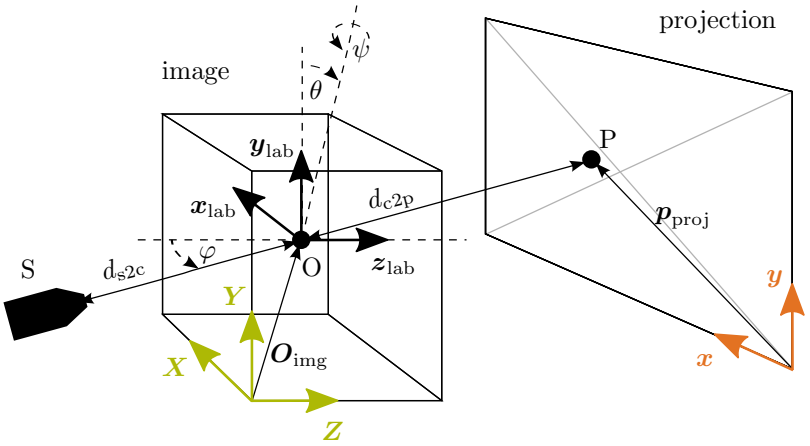

Figure 4. **A schematic of the three coordinate systems, the geometric parameters and their relations.** The three Euler angles (φ, θ, ψ) successively describe the rotation of the detector plane and the opposing source S around the axes $\boldsymbol{y}_{\text{lab}}$, $\boldsymbol{z}'_{\text{lab}}$, and $\boldsymbol{y}''_{\text{lab}}$ of the world coordinate system where each dash indicates a previous rotation of the axis itself. P, the principal point, is defined as the intersection of the orthogonal projection ray and the detector plane, i. e. the foot of a perpendicular of S on the detector. The distance between S and O is called d_{s2c} and d_{c2p} is the subsequent distance to P.
The coordinate system $(\boldsymbol{x}_{\text{proj}}, \boldsymbol{y}_{\text{proj}})$ on the detector plane has the offset $\boldsymbol{p}_{\text{proj}}$ from P and its coordinates are expected to be in units of detector pixels so that they can directly be used as array indices. Same, for the coordinate system $(\boldsymbol{x}_{\text{img}}, \boldsymbol{y}_{\text{img}}, \boldsymbol{z}_{\text{img}})$ in the image volume which has the same orientation as the world system, but measures of image voxels and its origin is displaced by the vector $\boldsymbol{O}_{\text{img}}$ from O.
The definition of the Euler angles, as well as the order of the axes in each system are only meant to serve as a concrete example for the computation of projection matrices and have no influence on the generality of the latter or the implementation below.

In order to use projection matrices in the context of tomography, each single projection of a scan has to be described with one corresponding matrix P consisting of the following components. The relevant coordinate systems and names of important geometry parameters of a tomographic setup are introduced in Figure 4. This appendix follows the book of Hartley et. al.[3]

A.1 The intrinsic parameters

are represented by the 3×3 Camera Calibration Matrix K. It describes a camera geometry, or the geometry of a tomographic setup in our case. Mapping the point $\boldsymbol{X} = (X, Y, Z, 1)^{\text{T}}$ in homogeneous coordinates to the corresponding point $\boldsymbol{x} = (x, y, 1)^{\text{T}}$ on the detector can be written as

$$\begin{pmatrix} x\,w \\ y\,w \\ w \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha_{\text{x}} & & p_{\text{x}} \\ & \alpha_{\text{y}} & p_{\text{y}} \\ & & 1 \end{pmatrix}}_{K} \left[\text{I}_3 \mid 0\right] \cdot \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

where I_i is the identity in the i-th dimension and w is a measure for the depth of the point that will not play any role in the further investigations.

The parameters α_{x} and α_{y} contain firstly the focal length $f = d_{\text{s2c}} + d_{\text{c2p}}$, the distance from the source to the principal point, in arbitrary length units, e. g. SI units. Secondly, further rescaling is caused by the pixel densities $\boldsymbol{m} = (m_{\text{x}}, m_{\text{y}})$ of the detector in both directions, given in the same system of units. Altogether, $\boldsymbol{\alpha} = (\alpha_{\text{x}}, \alpha_{\text{y}})$ can be written as $\boldsymbol{\alpha} = f \cdot \boldsymbol{m}$.

The other parameter in K is the offset $\boldsymbol{p}_{\mathrm{proj}} = (p_{\mathrm{x}}, p_{\mathrm{y}})$ of the origin of the detector coordinate system from the principal point in detector pixel units.

A.2 The extrinsic parameters

are firstly the camera translation vector $\boldsymbol{s}$ describing, in our case, the position of the source in the world coordinate system before any rotation.

Secondly, there is the camera rotation matrix R. Following the definition of the Euler angles above, it reads

$$\begin{aligned} R &= R_{\boldsymbol{y}''_{\mathrm{lab}}} \cdot R_{\boldsymbol{z}'_{\mathrm{lab}}} \cdot R_{\boldsymbol{y}_{\mathrm{lab}}} = \\ &= \begin{pmatrix} \mathrm{c}_\psi & & \mathrm{s}_\psi \\ & 1 & \\ -\mathrm{s}_\psi & & \mathrm{c}_\psi \end{pmatrix} \begin{pmatrix} \mathrm{c}_\theta & -\mathrm{s}_\theta & \\ \mathrm{s}_\theta & \mathrm{c}_\theta & \\ & & 1 \end{pmatrix} \begin{pmatrix} \mathrm{c}_\varphi & & \mathrm{s}_\varphi \\ & 1 & \\ -\mathrm{s}_\varphi & & \mathrm{c}_\varphi \end{pmatrix} = \\ &= \begin{pmatrix} -\mathrm{s}_\psi\mathrm{s}_\varphi + \mathrm{c}_\theta\mathrm{c}_\psi\mathrm{c}_\varphi & -\mathrm{s}_\theta\mathrm{c}_\psi & \mathrm{s}_\psi\mathrm{c}_\varphi + \mathrm{c}_\theta\mathrm{c}_\psi\mathrm{s}_\varphi \\ \mathrm{s}_\theta\mathrm{c}_\varphi & \mathrm{c}_\theta & \mathrm{s}_\theta\mathrm{s}_\varphi \\ -\mathrm{c}_\psi\mathrm{s}_\varphi - \mathrm{c}_\theta\mathrm{s}_\psi\mathrm{c}_\varphi & \mathrm{s}_\theta\mathrm{s}_\psi & \mathrm{c}_\psi\mathrm{c}_\varphi - \mathrm{c}_\theta\mathrm{s}_\psi\mathrm{s}_\varphi \end{pmatrix} \end{aligned}$$

where $\mathrm{s}_x = \sin x$ and $\mathrm{c}_x = \cos x$.

Together with the intrinsic parameters, the whole projection reads

$$\boldsymbol{x}\, w = \underbrace{K \Big[R \,|\, \boldsymbol{s} \Big]}_{\text{extrinsic parameters}} \cdot \boldsymbol{X}.$$

The projection matrix is then defined as

$$P_{\mathrm{lab}} = K \Big[R \,|\, \boldsymbol{s} \Big].$$

A.3 The transformation to image coordinates

could in principle be replaced by an appropriate choice of the preliminary parameters, but we decided to introduce it as an extra step in order to have a convenient separation of arbitrary units for the geometry parameters and voxel units for direct access to the indices in the image array. The transformation can be written as

$$T = \begin{pmatrix} \Delta_{\mathrm{img,x}} & & & o_{\mathrm{img},x} \\ & \Delta_{\mathrm{img,y}} & & o_{\mathrm{img},y} \\ & & \Delta_{\mathrm{img,z}} & o_{\mathrm{img},z} \\ & & & 1 \end{pmatrix}$$

where $\boldsymbol{\Delta}_{\mathrm{img}}$ is the image voxel size and $o_{\mathrm{img},i} = \Delta_{\mathrm{img},i} \cdot O_{\mathrm{img},i}$ the position of the center of rotation within the image volume in voxel units multiplied with the voxel size.

The resulting final projection matrix reads

$$P = T\, P_{\mathrm{lab}} = T\, K \Big[R \,|\, \boldsymbol{s} \Big].$$