Technical Note: RabbitCT—an open platform for benchmarking 3D cone-beam reconstruction algorithms[a)]

C. Rohkohl,[b)] B. Keck, H. G. Hofmann, and J. Hornegger
Department of Computer Science, Chair of Pattern Recognition, Friedrich-Alexander University Erlangen-Nuremberg, Martensstrasse 3, 91058 Erlangen, Germany

(Received 16 February 2009; revised 26 June 2009; accepted for publication 26 June 2009; published 12 August 2009)

Purpose: Fast 3D cone beam reconstruction is mandatory for many clinical workflows. For that reason, researchers and industry work hard on hardware-optimized 3D reconstruction. Backprojection is a major component of many reconstruction algorithms that require a projection of each voxel onto the projection data, including data interpolation, before updating the voxel value. This step is the bottleneck of most reconstruction algorithms and the focus of optimization in recent publications. A crucial limitation, however, of these publications is that the presented results are not comparable to each other. This is mainly due to variations in data acquisitions, preprocessing, and chosen geometries and the lack of a common publicly available test dataset. The authors provide such a standardized dataset that allows for substantial comparison of hardware accelerated backprojection methods.
Methods: They developed an open platform RabbitCT (www.rabbitCT.com) for worldwide comparison in backprojection performance and ranking on different architectures using a specific high resolution C-arm CT dataset of a rabbit. This includes a sophisticated benchmark interface, a prototype implementation in C++, and image quality measures.
Results: At the time of writing, six backprojection implementations are already listed on the website. Optimizations include multithreading using Intel threading building blocks and OpenMP, vectorization using SSE, and computation on the GPU using CUDA 2.0.
Conclusions: There is a need for objectively comparing backprojection implementations for reconstruction algorithms. RabbitCT aims to provide a solution to this problem by offering an open platform with fair chances for all participants. The authors are looking forward to a growing community and await feedback regarding future evaluations of novel software- and hardware-based acceleration schemes.
[DOI: 10.1118/1.3180956]

Key words: benchmark, 3D reconstruction, backprojection, high performance computing

I. INTRODUCTION

The typical clinical environment requires fast 3D reconstruction. Therefore, research focuses on implementations which reduce the runtime of cone beam reconstruction. A large class of reconstruction algorithms is backprojection based[1–5] requiring a projection of each voxel onto each projection image, interpolating the projection data, and finally updating the voxel value. This step is highly time consuming and, thus, is the focus of optimization in recent publications.[6–13]

Despite the numerous publications and acceleration techniques, there is still a lack of comparability of the achieved results for several reasons: First, datasets used in publications are not open to public and thus experiments are not reproducible. Further, the objective comparison of implementations is not possible due to differing assumptions and, lastly, there exist hundreds of different acquisition protocols and parameters used for the reconstruction algorithms.

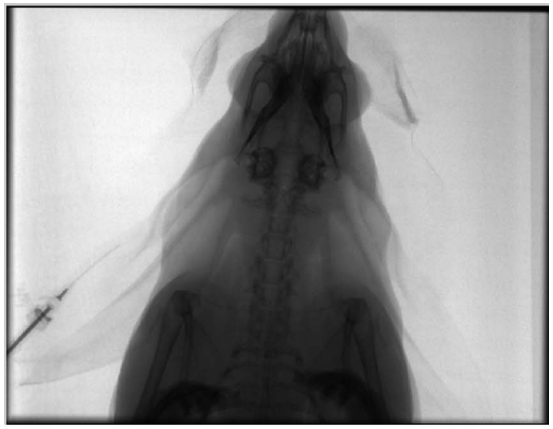
We aim to close this gap by providing an open platform, called RabbitCT, for the worldwide comparison in backprojection performance on different architectures using a specific high resolution C-arm CT dataset of a rabbit.

II. METHODS AND MATERIALS

The development of an open and uniform benchmark environment for backprojection performance requires several steps to be considered. Each step is devoted one of the subsequent sections. A publicly accessible dataset along with geometry information has to be provided (Sec. II A). A reconstruction task with different levels of difficulty needs to be specified (Sec. II B). Finally, a reconstruction algorithm (Sec. II C) and the evaluations metrics (Sec. II D) complete the benchmark environment.

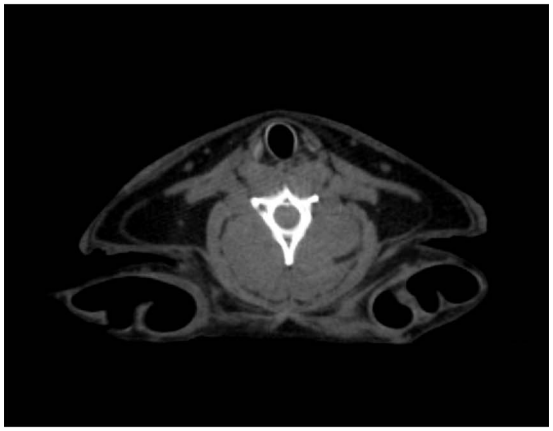
II.A. Dataset

A preprocessed dataset of a rabbit suitable for cone beam 3D reconstruction was acquired at the Department of Neuroradiology, Friedrich-Alexander-University Erlangen-Nuremberg, Germany, and is available for download at the competition website.[14] It consists of $N=496$ projection images $\boldsymbol{I}_n \in \mathbb{R}^{S_x \times S_y}$, $n=1,\ldots,N$ from a C-arm system (Siemens AG, Artis Zee) acquired on a 200° circular short-scan trajec-



(a)

(b)



(c)

FIG. 1. RabbitCT projection image, volume rendering, and axial slice of the corresponding 3D reconstruction.

tory. The size of a projection image is $S_x=1248$ pixels in width and $S_y=960$ pixels in height at an isotropic resolution of 0.32 mm/pixel.

For each projection image (see Fig. 1) a precalibrated projection matrix $\boldsymbol{A}_n \in \mathbb{R}^{3\times 4}$ is available encoding the perspective projection, in homogeneous coordinates, of a 3D object point onto the 2D projection image.[15,16]

II.B. Reconstruction task

The aim of RabbitCT is the reconstruction of an isocentric cubic volume of 256^3 mm^3. For that it has been decided for four different samples of different computational costs. The side lengths of the cubic reconstruction are $L \in \{128, 256, 512, 1024\}$ voxels, respectively, at an isotropic voxel size of $R_L=(256/L)$ mm.

The origin of the world coordinate system (in mm) is at the isocenter of the C-arm system. The volume to be reconstructed is denoted $f(x,y,z)$ where $x,y,z \in [-128, 128]$. The discrete volume is denoted $f_L(i,j,k)$, where $i,j,k \in [0,\ldots,L-1]$ and is related to the world coordinates as follows:

$$f_L(i,j,k) = f(O_L + iR_L, O_L + jR_L, O_L + kR_L), \tag{1}$$

with $O_L=-\frac{1}{2}R_L(L-1)$.

II.C. Reconstruction algorithm

The FDK algorithm[1] is utilized with Parker weighting[17] for the reconstruction of short-scan C-arm projection data. It requires the projection data to be preprocessed, e.g., physical corrections, cosine weighting, and ramp filtering.[18] All these steps have already been applied to the available projection image $\boldsymbol{I}_n$. The discrete version of the FDK algorithm thus breaks down to

$$f(x,y,z) = \sum_{n=1}^{N} \frac{1}{w_n(x,y,z)^2} \cdot \hat{p}_n(u_n(x,y,z), v_n(x,y,z)), \tag{2}$$

where

$$u_n(x,y,z) = (a_0 x + a_3 y + a_6 z + a_9) \cdot w_n(x,y,z)^{-1},$$

$$v_n(x,y,z) = (a_1 x + a_4 y + a_7 z + a_{10}) \cdot w_n(x,y,z)^{-1},$$

$$w_n(x,y,z) = a_2 x + a_5 y + a_8 z + a_{11},$$

and

$$\boldsymbol{A}_n = \begin{pmatrix} a_0 & a_3 & a_6 & a_9 \\ a_1 & a_4 & a_7 & a_{10} \\ a_2 & a_5 & a_8 & a_{11} \end{pmatrix}.$$

This particular mathematical notation has been chosen for its closeness to the implementation. The function $\hat{p}_n: \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ performs a bilinear interpolation with a zero boundary condition in the projection image $\boldsymbol{I}_n$. It is given by

$$\begin{aligned}\hat{p}_n(x,y) = {} & (1-\alpha)(1-\beta)p_n(i,j) + \alpha(1-\beta)p_n(i+1,j) \\ & + (1-\alpha)\beta p_n(i,j+1) + \alpha\beta p_n(i+1,j+1),\end{aligned}$$

with $i=\lfloor x \rfloor$, $j=\lfloor y \rfloor$, $\alpha=x-\lfloor x \rfloor$, and $\beta=y-\lfloor y \rfloor$. An integral value of the image matrix $\boldsymbol{I}_n$ is accessed by the function $p_n: \mathbb{Z} \times \mathbb{Z} \mapsto \mathbb{R}$, which is given by

$$p_n(i,j) = \begin{cases} \boldsymbol{I}_{n,i,j} & \text{if } i \in \{0,\ldots,S_x-1\} \wedge j \in \{0,\ldots,S_y-1\} \\ 0 & \text{otherwise.} \end{cases}$$

The output of the reconstruction algorithm is scaled to be in the 12-bit range of {0,…,4095} Hounsfield units.

II.D. Evaluation metrics

For comparing different implementations of Eq. (2) in terms of speed and quality, we utilize four different evaluation metrics. The first one, t_{avg}, measures the efficiency of the backprojection implementation by computing the average time that is required to process all voxels for one projection image. This includes the time required for data access, interpolation, and computation. This parameter is the most important one in a clinical environment as it tells how long the physician has to wait for the result of the reconstruction, depending on the number of acquired projections. Other performance measures, e.g., GUPS,[19] can be derived from it. Comparing the GUPS between different problem sizes, one can assess the scalability of a specific implementation.

The three other metrics measure the quality of the reconstruction. For that, a reference reconstruction f_L^{ref} is provided for each problem size, which is reconstructed using the reference implementation LolaBunny provided on our website. LolaBunny features a straightforward implementation of the reconstruction task. This reference is then used to estimate the numerical accuracy of a new reconstruction f_L. The mean squared error $q_{\text{mse}}(f_L)$ of the reconstruction in Hounsfield units (HU^2) in comparison with the reference reconstruction is given by

$$q_{\text{mse}}(f_L) = \frac{1}{L^3}\sum_{i,j,k}[f_L(i,j,k) - f_L^{\text{ref}}(i,j,k)]^2. \tag{3}$$

While the peak signal to noise ratio q_{psnr}, measured in decibels (dB), is calculated according to

$$q_{\text{psnr}}(f_L) = 10\log_{10}\left(\frac{4095^2}{q_{\text{mse}}(f_L)}\right). \tag{4}$$

Further, a histogram of the absolute errors is computed and presented in the ranking on the website.

III. BENCHMARK DESIGN

III.A. Benchmark execution

A RabbitCT benchmark reconstruction consists of three steps:

(1) *Benchmark program and dataset*. Visit the download section of our website[14] and download RABBITCTRUNNER and the dataset described in Sec. II A. RABBITCTRUNNER is the program that performs the benchmark and evaluation. It is available for various operating systems, e.g., Linux, Windows, and Mac OS X in 32 and 64 bits.
(2) *Backprojection module*. Implement your own backprojection module (see Sec. III B) or download a precompiled backprojection module, e.g., our basic C++ example LolaBunny.
(3) *Benchmark participation*. Execute RABBITCTRUNNER from the command line. It will produce a volume file along with an encoded result file which can be uploaded to our servers. More details are provided in Sec. IV.

Algorithm 1. RABBITCTRUNNER execution scheme.

input: $L, O_L, R_L, \boldsymbol{I}, \boldsymbol{A}$
output: Reconstruction f_L, metrics t_{avg}, q_{mse}, q_{psnr}
call RCTLoadAlgorithm;
for $n=1$ to N do
  $t=currentTime(\,)$;
  call RCTAlgorithmBackprojection $(\boldsymbol{I}_n, \boldsymbol{A}_n, L, O_L, R_L)$;
  $t_{total}=t_{total}+(currentTime(\,)-t)$;
end
call RCTFinishAlgorithm; // set f_L
$t_{avg}=t_{total}/N$;
$q_{mse}=q_{mse}(f_L)$;
$q_{psnr}=q_{psnr}(f_L)$;
call RCTUnloadAlgorithm;

Algorithm 2. RCTAlgorithm backprojection for the nth projection image.

input: $\boldsymbol{I}_n, \boldsymbol{A}_n, L, O_L, R_L, f_L$
output: updated reconstruction f_L
// Visit each voxel in the sampled grid
for $i=0$ to $L-1$ do
  for $j=0$ to $L-1$ do
    for $k=0$ to $L-1$ do
      // Calculate coordinates in world coordinate system
      $x=O_L+iR_L$;
      $y=O_L+jR_L$;
      $z=O_L+kR_L$;
      // Update the volume
      $f_L(i,j,k)=f_L(i,j,k)+\frac{1}{w_n(x,y,z)^2}\cdot\hat{p}_n(u_n(x,y,z),v_n(x,y,z))$;
    end
  end
end

III.B. Backprojection modules

The backprojection module is the core concept of our platform. It has to be implemented as a dynamically loadable shared object (e.g., a DLL in Windows) and requires four functions in the symbol table: initialization, backprojection, finish, and cleanup. All functions are of the form bool name (RabbitCtGlobalData*) where name is replaced by the actual function name. They return the success of the requested operation and are passed a parameter structure RabbitCtGlobalData containing all relevant data, e.g., $\boldsymbol{I}_n$, $\boldsymbol{A}_n$, L, O_L, and R_L. The pseudocode of the RABBITCTRUNNER execution model is provided in Algorithm 1. More details about compilation and an example C++ implementation called LolaBunny are provided on the website.[14] In the following, a brief overview of each function is given.

III.B.1. Initialization

A function RCTLoadAlgorithm needs to be defined. Generally this function can be used to initialize required data structures, e.g., allocate memory for the volume on the graphicscard. It returns the success of the operation.

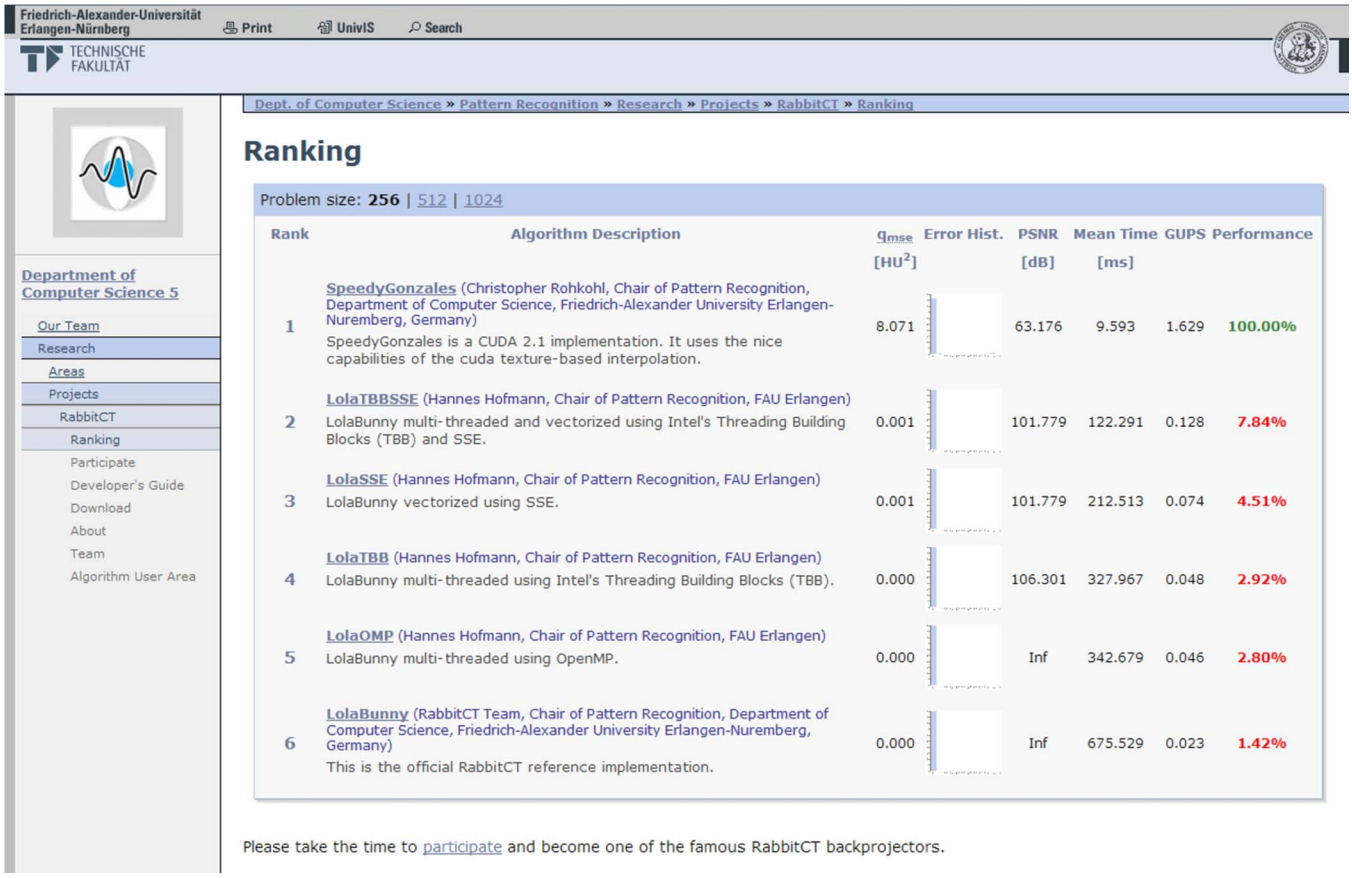


FIG. 2. Screenshot of the ranking table on the RabbitCT website.

III.B.2. Backprojection

The function RCTAlgorithmBackprojection implements Eq. (2) and is called once for each projection image along with the required projection data. The required execution time for this function is averaged to obtain the efficiency measure t_{avg}. The pseudocode for a simple backprojection routine is provided in Algorithm 2.

III.B.3. Finish

The function RCTFinishAlgorithm is called after the last projection image has been processed. Here, the reconstructed volume needs to be stored in a memory buffer and passed back via the global structure. The returned volume is then used for evaluating the image quality.

III.B.4. Cleanup

The function RCTUnloadAlgorithm is called after the reconstruction quality was assessed. A common task of this routine is, for example, freeing the volume memory buffer.

IV. PARTICIPATION AND RANKING

After executing a benchmarked reconstruction using RABBITCTRUNNER, a result file is created. It can be submitted on our website[14] and will be listed after approval. The submission is a two-step procedure.

First, a registration of the algorithm is required. The participant is asked to describe his algorithm and to provide some reference, e.g., technical report, paper, or website. After approval, the algorithm's author gets a username and password and can add as many benchmark results as desired in a user area. For each benchmark submission the output of RABBITCTRUNNER is required along with the information about the system the benchmark was run on. Once approved, a result is published on our website along with all the relevant information provided by its author (cf. Fig. 2).

The ranking contains the results for the problem sizes $L \geqslant 256$ only, as smaller problem sizes do not allow a fair comparison but are useful for debugging and testing.

V. CONCLUSION AND OUTLOOK

There is a need for objectively comparing backprojection implementations for reconstruction algorithms. RabbitCT aims to provide a solution to this problem by offering an open platform with fair chances for all participants. At the time of writing, six backprojection implementations are already listed on the website. Optimizations include multithreading using Intel threading building blocks and OpenMP, vectorization using SSE and computation on the GPU using CUDA 2.0. The straightforward CPU implementation is provided open source as reference. We are looking forward to a growing community and await feedback regarding future

evaluations of novel software- and hardware-based acceleration schemes.

ACKNOWLEDGMENTS

The authors thank Dr. Tobias Stuffert and Professor Dr. Arnd Dörfler from the Department of Neuroradiology, Friedrich-Alexander-University Erlangen-Nuremberg, Germany for providing the RabbitCT dataset.

[a]URL: http://www.rabbitct.com
[b]Electronic mail: christopher.rohkohl@informatik.uni-erlangen.de

[1]L. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone-beam algorithm," J. Opt. Soc. Am. A **1**, 612–619 (1984).
[2]F. Dennerlein, A. Katsevich, G. Lauritsch, and J. Hornegger, "Exact and efficient cone-beam reconstruction algorithm for a short-scan circle combined with various lines," Proc. SPIE **5747**, 388–399 (2005).
[3]S. Hoppe, F. Noo, F. Dennerlein, G. Lauritsch, and J. Hornegger, "Cone-beam tomography from short-scan circle-plus-arc data measured on a C-arm system," *2006 IEEE Nuclear Science Symposium Conference Record*, San Diego (IEEE, 2006), pp. 2873–2877.
[4]A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging* (IEEE, New York, 1988).
[5]H. Turbell, "Cone-beam reconstruction using filtered backprojection," Ph.D. thesis, Linköping University, 2001.
[6]I. Goddard and M. Trepanier, "High-speed cone-beam reconstruction: An embedded systems approach," Proc. SPIE **4681**, 483–491 (2002).
[7]K. Mueller and F. Xu, "Practical considerations for GPU-accelerated CT," *IEEE International Symposium on Biomedical Imaging* (IEEE, 2006), Paper No. SU-AM-OS3.3.
[8]K. Mueller, F. Xu, and N. Neophytou, "Why do commodity graphics hardware boards (GPUs) work so well for acceleration of computed tomography?," Proc. SPIE **6498**, 64980N (2007).
[9]M. Kachelrieß, M. Knaup, and O. Bockenbach, "Hyperfast perspective cone-beam backprojection," *IEEE Nuclear Science Symposium and Medical Imaging Conference*, San Diego (IEEE, 2006), p. M01-7.
[10]H. Scherl, M. Koerner, H. Hofmann, W. Eckert, M. Kowarschik, and J. Hornegger, "Implementation of the FDK algorithm for cone-beam CT on the cell broadband engine architecture," Proc. SPIE **6510**, 651058 (2007).
[11]M. Churchill, "Hardware-accelerated cone-beam reconstruction on a mobile c-arm," Proc. SPIE **6510**, 1605–7422 (2007).
[12]F. Xu and K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," IEEE Trans. Nucl. Sci. **52**(3), 654–663 (2005).
[13]R. Yu, R. Ning, and B. Chen, "High-speed cone-beam reconstruction on PC," Proc. SPIE **4322**, 964–973 (2001).
[14]Friedrich-Alexander University Chair of Pattern Recognition. RabbitCT, Open Platform for worldwide comparison in backprojection performance, (http://www.rabbitct.com) 2009.
[15]O. Faugeras, *Three-Dimensional Computer Vision (Artificial Intelligence)*, (MIT Press, Cambridge, 1993).
[16]K. Wiesent, K. Barth, N. Navab, P. Durlak, T. Brunner, O. Schuetz, and W. Seissler, "Enhanced 3-D-reconstruction algorithm for C-arm systems suitable for interventional procedures," IEEE Trans. Med. Imaging **19**(5), 391–403 (2000).
[17]J. R. Parker, *Algorithms for Image Processing and Computer Vision* (Wiley, New York, 1996).
[18]M. Zellerhoff, B. Scholz, E.-P. Ruehrnschopf, and T. Brunner, "Low contrast 3D reconstruction from C-arm data," Proc. SPIE **5745**, 646–655 (2005).
[19]I. Goddard, A. Berman, O. Bockenbach, F. Lauginiger, S. Schuberth, and S. Thieret, "Evolution of computer technology for fast cone beam backprojection," *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, February 2007, Vol. 6498 (unpublished).