Accelerated C-Arm Reconstruction by Out-of-Projection Prediction

Hannes G. Hofmann, Benjamin Keck, Joachim Hornegger

Pattern Recognition Lab, University Erlangen-Nuremberg
hannes.hofmann@informatik.uni-erlangen.de

Abstract. 3D image reconstruction from C-arm projections is a computationally demanding task. However, for interventional procedures a fast reconstruction is desirable. We present a method to reduce the number of actually back-projected voxels and ultimately the processing time by 20–30 % without affecting image quality. The proposed method detects and skips subvolumes that are not visible in the current view. It works with projection matrices and is thus capable of handling arbitrary geometries. It can easily be incorporated into existing algorithms and is also suitable for the back-projection step of iterative algorithms.

1 Introduction

Clinical applications – in particular interventional procedures – require fast 3D reconstruction of tomographic data. Therefore, means to reduce the reconstruction time are a strong research topic. Currently, the most wide-spread algorithms in clinical X-ray CT reconstruction belong to the class of filtered back-projection (FBP), e.g. the FDK method [1] for cone-beam data. Another class of reconstruction algorithms is based on iterative techniques which require multiple alternating back- and forward-projections. The latter can incorporate various corrections and provide superior image quality in certain cases like sparse or irregular data [2]. The fact that their complexity is a multiple of the FDK's is a reason why this class of reconstruction algorithms is less commonly used in clinical systems. However, both classes of reconstruction algorithms benefit from an acceleration of the back-projection step.

Several groups investigate acceleration using specialized hardware (e.g. FPGAs [3, 4, 5]), or general purpose high-performance hardware (Cell processor [6, 7], GPUs [8, 9, 10], Larrabee [11]).

Yu et al. [12] proposed to reconstruct only cylinder- or sphere-shaped region of interest – approximating the Field-of-View (FOV) of the scan – to reduce the number of back-projected voxels. Their method, however, requires a priori knowledge about the trajectory and offline computation of the FOV before reconstruction.

Yu et al. further suggested data partitioning which is similar to the subvolumes approach used e.g. by Scherl et al. [6] and Kachelriess et al. [7]. They used subvolumes to fit the problem into the 256 KB-sized local stores of the Cell

Algorithmus 3 Original back-projection step, called once for each projection image I_n.

```
for all voxels $\underline{x}$ do
    Project voxel onto detector plane
    Check if point lies on detector
    Fetch projection values
    Bilinear interpolation
    Update $f_{\mathrm{FDK}}(\underline{x})$
end for
```

processor. They also proposed that subvolumes improve caching on CPUs resulting in better performance. However, they did not actually skip subvolumes as proposed in this paper.

Our proposed method adapts dynamically to the FOV of each projection image during back-projection. It uses projection matrices from the scanner and requires no prior knowledge about the acquisition trajectory. It is easy to integrate into existing reconstruction algorithms and does not affect image quality.

2 Materials and Methods

Alg. 3 shows pseudo-code of the back-projection step of the FDK method. It is called once for each projection image $\boldsymbol{I}_n$ and updates every voxel of the reconstructed volume.

In cone-beam CT, the source position and the corners of a projection image define a pyramid, the current FOV. Areas of the reconstructed volume that are outside of the FOV are not visible on the detector. Hence, the current view cannot contribute information about their content.

Alg. 4 shows the modified back-projection step. Before a subvolume is back-projected a check is performed to decide whether it is inside the current FOV. Therefore, its corner voxels are projected to figure out its shadow. If the intersection of the shadow with the detector has a positive area it is back-projected as usual. Otherwise, the whole subvolume is skipped and processing continues with the next one. The computation of a subvolume's shadow involves 3D/2D-projections of its 8 corners, computation of the minimum and maximum of the resulting 2D coordinates, clamping of the coordinates and computation of the shadow's area. On the other hand, for every voxel contained in a skipped subvolume a 3D/2D-projection, the coordinates check, 4× pixel access, bilinear interpolation and the voxel update are saved.

The proposed method skips only subvolumes that did not contribute to the current projection. Hence, there should be no difference in the result compared to the reference method. The reconstruction benchmark RabbitCT [13] was used to test our implementation for correctness and measure its performance.

Obviously, the performance of our new method depends on the acquisition geometry. To rule out the possibility of "cooperative data" the method was evaluated on two other clinical datasets, additional to the public RabbitCT

Algorithmus 4 Modified back-projection step, called once for each projection image I_n.

```
Partition volume $f_{\text{FDK}}$ into subvolumes $f_{\text{FDK},i}$
for all subvolumes $f_{\text{FDK},i}$ do
  // Test if $f_{\text{FDK},i}$ is visible
  Compute shadow of $f_{\text{FDK},i}$ by projecting corners
  if $\emptyset == \textit{shadow} \cap I_n$ then
    Skip $f_{\text{FDK},i}$
  else
    for all voxels $\underline{x} \in f_{\text{FDK},i}$ do
      Project voxel onto detector plane
      Check if point lies on detector
      Fetch projection values
      Bilinear interpolation
      Update $f_{\text{FDK}}(\underline{x})$
    end for
  end if
end for
```

dataset. All datasets were acquired with real C-arm systems using clinical protocols. Their sizes are shown in Tab. 1 where dataset A is the public RabbitCT dataset.

To show that the new method also improves the performance of highly optimized code it was incorporated into our vectorized and multi-threaded FDK implementation for CPUs (described in more detail in [14]).

The new method was tested on two multi-core systems. The first system featured two Core i7 (“Nehalem”) quad-core processors at 2.66 GHz (85.12 GFlops total) and was equipped with 12×1 GB of DDR3-1066 RAM. This system had a very high memory bandwidth. The other one had four hexa-core Xeons (X7460) at 2.66 GHz (255.35 GFlops total) and 32 GB of DDR2-1066 RAM. All tests were performed using 64-bit Linux and the Intel compilers in version 11.1.056. The “X7460” and “Nehalem” systems used in this report are preproduction systems. We expect production hardware to deliver similar performance levels.

3 Results

The reconstructed volume had a size of 512^3 voxels. Tab. 1 shows the results for both computer systems. The subvolume size was empirically chosen to be 128×64×4 as this performed overall best. In the worst case, this subvolume size was about 5 % slower than the best one. The large size in x-direction is justified by the fact that neighboring voxels in this direction are stored successively in memory and therefore supportive for the hardware prefetching mechanism.

Results for three different implementations are shown. The baseline is defined by our vectorized and multi-threaded implementation [14] (“base” in Tab. 1). Using subvolumes allows to optimize the cache usage of modern processors by

Table 1. Description of the three datasets. Further, reconstruction times (seconds). Results are shown for two systems ("Nehalem" and "X7460") and three implementations ("base", "cache" and "skip"). The volume size was 512^3, the subvolume size 128×64×4.

	Number of	Projection	"Nehalem"			X7460		
Dataset	projections	size	base	cache	skip	base	cache	skip
A	496	1240 × 960	84.71	80.28	61.38	60.53	42.89	31.96
B	543	1240 × 960	92.89	87.56	67.39	66.04	48.77	35.08
C	414	1024 × 1024	70.54	68.23	54.54	50.59	36.05	29.21

re-using voxel data before storing it back to main memory. Consequently, the memory bandwidth restriction is relaxed ("cache"). Finally, the results of the method proposed in this work ("skip") are presented.

The optimized cache usage gives a speedup of at least 1.35× on the bandwidth-starving "X7460" system. On the "Nehalem" system, however, it shows only little effect (up to 1.06×). This result was expected due to the higher memory bandwidth of the Nehalem architecture.

With the proposed method of skipping subvolumes the reconstruction time is further reduced to about 72–80% compared to the cache-optimized version on both systems (1.23–1.39×). The effectiveness of our approach can be seen by looking on the numbers of a single dataset, e.g. A. Given the chosen size of 128×64×4 voxels about 24 % of the subvolumes are outside of the FOV. The reconstruction time is 75 and 76 % of the "cache" implementation's on the "X7460" and the "Nehalem" system, respectively.

4 Discussion

We have tried numerous different subvolume sizes and shapes and found out that the optimal solution depends on several factors. Large subvolumes – e.g. 512×64×8 – provide better data locality and use less memory bandwidth if caches are large enough. However, this advantage is absorbed by the fact that they are less likely to be skipped since smaller subvolumes resemble the real FOV more accurately. While small subvolumes can have skip-ratios of up to 32 % (for 16×16×4 and dataset A) they come with the overhead of more shadow computations and with reduced savings per skipped subvolume.

On the other hand, subvolume shape has an impact on other performance factors. From a theoretical point, square-shaped subvolumes should be optimal as their mean shadow size over all viewing angles is minimal. But given current CPUs' caching and prefetching algorithms it is a good idea to use subvolumes which are longer in the direction of contiguous memory access.

The savings of the proposed method depend on the acquisition geometry, namely the ratio between volume and FOV sizes. To show its relevance the method was tested using three different clinical datasets. The proposed method delivers a considerable speedup even to highly optimized implementations. Although all datasets used here were circular trajectories the method is suitable for arbitrary trajectories. It requires no prior knowledge other than the projection matrices. Since it works on-the-fly it can easily be incorporated into existing algorithms.

References

1. Feldkamp LA, Davis LC, Kress JW. Practical cone-beam algorithm. J Opt Soc Am. 1984;A1(6):612–9.
2. Kunze H, Härer W, Stierstorfer K. Iterative extended field of view reconstruction. Proc SPIE. 2007;6510:5X–1–8.
3. Xue X, Cheryauka A, Tubbs D. Acceleration of fluoro-CT reconstruction for a mobile C-Arm on GPU and FPGA hardware: A simulation study. Proc SPIE. 2006;6142:1494–501.
4. Churchill M. Hardware-accelerated cone-beam reconstruction on a mobile C-arm. Proc SPIE. 2007;6510:5S–1–8.
5. Heigl B, Kowarschik M. High-speed reconstruction for C-arm computed tomography. Radiol Nucl Med. 2007; p. 25–8.
6. Scherl H, Koerner M, Hofmann H, et al. Implementation of the FDK algorithm for cone-beam CT on the cell broadband engine architecture. Proc SPIE. 2007;6510:58–1–10.
7. Kachelrieß M, Knaup M, Bockenbach O. Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware. Med Phys. 2007;34(4):1474–86.
8. Mueller K, Yagel R. Rapid 3D cone-beam reconstruction with the algebraic reconstruction technique (ART) by utilizing texture mapping graphics hardware. IEEE Nucl Sci Symp Conf Rec. 1998;3:1552–9.
9. Xu F, Mueller K. Real-time 3D computed tomographic reconstruction using commodity graphics hardware. Phys Med Biol. 2007;52(12):3405–19.
10. Scherl H, Keck B, Kowarschik M, et al. Fast GPU-Based CT reconstruction using the common unified device architecture (CUDA). In: IEEE Nucl Sci Symp Conf Rec. vol. 6; 2007. p. 4464–6.
11. Hofmann HG, Keck B, Rohkohl C, et al. Towards C-arm CT reconstruction on Larrabee. In: Proc Fully 3D Meeting and HPIR Workshop; 2009. p. 1–4.
12. Yu R, Ning R, Chen B. High speed cone beam reconstruction on PC. Proc SPIE. 2001;4322:964–73.
13. Rohkohl C, Keck B, Hofmann HG, et al. RabbitCT: an open platform for benchmarking 3D cone-beam reconstruction algorithms. Med Phys. 2009;36(9):3940–4.
14. Hofmann HG, Keck B, Rohkohl C, et al. Putting 'p' in rabbitCT: fast CT reconstruction using a standardized benchmark. In: PARS-Mitteilungen. GI, Gesellschaft für Informatik e.V.; 2009. p. in press.