Chapter 8. GPU Denoising for Computed Tomography

Andreas Maier[1], Rebecca Fahrig[2]

[1]Pattern Recognition Lab, Friedrich-Alexander-University Erlangen-Nuremberg

[2]Radiological Sciences Lab, Stanford University

8.1 Introduction

The main source of noise in X-ray images is quantum noise. It follows a Poisson process that has a standard deviation of $\sqrt{N}$, where N is the number of photons that arrived at the detector pixel. Thus, resulting signal-to-noise ratio is $\frac{N}{\sqrt{N}} = \sqrt{N}$, i.e. the more photons arrive at the detector, the higher the signal-to-noise ratio. As the number of measured photons is proportional to the number of emitted photons at the source, image noise is inversely proportional to the dose that is applied to the patient. Thus, methods for the reduction of image noise also enable a reduction in patient dose. For this reason, noise reduction techniques have been investigated in literature extensively.

8.2 Noise in X-Ray Images

In order to understand the nature of noise reduction techniques, it is useful to have understanding of the properties of noise in X-ray images.

8.2.1 Noise in Projection Data

The distribution of noise in an X-ray projection image follows Beer-Lambert's Law For the monochromatic case, the measured intensity I_m at the detector is found as

$$I_m = I_0 e^{-\int f(x)\mathrm{d}l}, \tag{8.1}$$

where I_0 is the intensity that was emitted at the source, $f(x)$ the object that was irradiated, and l the line that follows the path of the X-ray. Note that

$$I_0 = N_0 \cdot E_0 \tag{8.2}$$

is composed of the number of photons N_0 and their respective energy E_0. Thus the number of photons is proportional to the measured intensity and the monochromatic case follows exactly the Poisson distribution given by

$$P(n) = \frac{\kappa^n}{n!} \cdot e^{-\kappa} \tag{8.3}$$

where $P(n)$ is the likelihood of observing n photons and $\kappa = N_m = \frac{I_m}{E_0}$ is the expected value of the Poisson process.

For the polychromatic case, the measured intensity I_p is found as an integral along energy ϵ that is present at the source I_ϵ

$$I_p = \int I_\epsilon e^{-\int f(x)\mathrm{d}l} \mathrm{d}\epsilon. \tag{8.4}$$

Figure 8.1 displays an example for an X-ray spectrum at 90 kVp. As a consequence, the noise is no longer a single Poisson process, but its distribution is now an integral over all Poisson processes at each individual energy ϵ. For simplification, we will assume in the following that our source is able to generate monochromatic X-rays.

[insert Fig. 8.1 near hear]

8.2.2 Noise in Reconstructed Image Data

In order to compute noise in reconstruction domain, one has to determine line integrals p from measured intensities I_m:

$$p = \int f(x) = -\ln \frac{I_m}{I_0} \tag{8.5}$$

As I_0 is known from the X-ray tube parameters, the main effect on the noise is the application of the logarithm. For high number of photons, it is valid to approximate the Poisson process by a Gaussian distribution with mean value $\mu = N_m$ and standard deviation $\sigma = \sqrt{N_m}$. By application of a first order Taylor expansion [1, 2], one is able to show that the log transform results in a new mean value of $\kappa_p = -\,ln\frac{N_m \cdot E_0}{I_0}$ and $\sigma_p = \frac{1}{\sqrt{N_m}}$. Figure 8.2 shows a noise-free projection, a noisy projection, and their difference. Note that the noise variance increases with object thickness.

[insert Fig. 8.2 near hear]

The reconstruction process involves filtering and back-projection. Back-projection $b(x)$ is – depending on the imaging geometry – only a weighted sum of the projection values p_i with back-projection weights c_i

$$b(x) = \sum_i c_i p_i. \tag{8.6}$$

Even, if the effect of the filter is not considered, one immediately understands that the noise variance becomes object dependent at this point, as p_i and the noise at this detector pixel σ_{p_i} is dependent on the number of photons that arrived at this detector pixel N_m. Figure 8.3 shows these effects in the reconstructed images. In the water cylinder, the most noise is found in the center of the object. In a non-circular object like the one shown on the right, streak structures appear in the noise. Thus noise removal in the reconstructed image needs to consider the directional nature of the noise generation. The effects of filtering on the noise properties and how to reconstruct the noise variance at every pixel are beyond the scope of this book chapter and we refer to further literature [3, 4, 5].

```
convolution2D (image){
    sumWeight = 0;
    sumFilter = 0;
    step = halfWidth*2+1;
    for (j=0; j < step; j++){
        for (i=0; I < step; i++){
            nx = halfWidth + i;
            ny = halfWidth + j;
            g = geomClose(nx, ny, sigma_spatial)
            sumWeight += g;
            sumFilter += g * image[(x-nx)+((y-ny)*width)];
        }
    }
    return sumFilter/sumWeight;
}
```

Algorithm 8.1: Pseudo code for a 2D convolution for application of a Gaussian kernel of odd neighborhood size, e.g. $\mathcal{N} = 5x5$ using `halfWidth`=2. The function *`geomClose`* refers to the computation of $g(x, y, \sigma_r)$ with *`sigma_spatial`* as σ_r, *`image`* denotes $p(x, y)$ and *`sumWeight`* and *`sumFilter`* are used to compute $k(x, y, \sigma_r)$ and $p'(x, y, \sigma_r)$ incrementally.

[insert Fig. 8.3 near hear]

8.3 Denoising Methods

In the following, we will give a short introduction to the most common denoising methods. We start with the simple Gaussian filter and extend this concept to edge-preserving methods. All reported methods can be employed for 2D and 3D denoising. For sake of simplicity, we will stay in 2D space in the following. Note that we only selected a subset of possible denoising algorithms that we deem as the most important selection, e.g. we omit methods that require an iterative solution such as diffusion filters.

8.3.1 Gaussian Filter

The Gaussian filter is the most common way of suppressing noise in any kind of image. It is applied as convolution in projection space

$$p'(x,y,\sigma_r) = \frac{1}{k(x,y,\sigma_r)} \sum_{x',y' \in \mathcal{N}} p(x',y') \cdot g(x',y',\sigma_r) \tag{8.7}$$

$$g(x,y,\sigma_r) = e^{-\frac{x^2+y^2}{2\sigma_r^2}}$$

$$k(x,y,\sigma_r) = \sum_{x',y' \in \mathcal{N}} g(x',y',\sigma_r)$$

where $p(x,y)$ is the projection image at coordinates (x,y), $g(x,y,\sigma_r)$ is the Gaussian function with standard deviation σ_r, $k(x,y,\sigma_r)$ the mass of the kernel, $\mathcal{N}$ the neighborhood in which the kernel is evaluated, and $p'(x,y,\sigma_r)$ the filtered projection image. This kind of filtering is often included in the filter kernel of filtered back-projection-type reconstruction methods [4]. Thus explicit implementation is often not required. For small kernel sizes, the implementation in spatial domain is efficient. For larger kernels, the convolution in Fourier space is more efficient.

Nonetheless, investigation of the algorithm as graphics card implementation is interesting, as one can see how the computation is parallelized. Algorithm 8.1 lists pseudo code for the spatial implementation on a graphics card. The code omits a loop over the coordinates (x,y) as the kernel code is designed to be executed in a parallel grid structure that matches the image dimensions. The main part of the code are the two for-loops over the double sums that are required to compute $k(x,y,\sigma_r)$ and $p'(x,y,\sigma_r)$. Note that the values of $g(x,y,\sigma_r)$ are the same in all kernel executions as the kernel is shift-invariant. In this example, we do not pre-compute the values of $g(x,y,\sigma_r)$. Depending on the graphics card hardware, this may lead to an increase or decrease in run time. If the hardware has much compute power, the implementation in Algorithm 8.1 will be more efficient, as only a single access to the *global* memory is performed.

In this case, the hardware is *memory-* or *bandwidth-limited*, i.e. the execution is limited by memory transfers. Thus additional operations in the kernel that are computed in *local* or *shared* memory only can be done without increasing the run time as the *global* memory transfers are the dominating factor in the execution time. If the hardware is *compute-limited*, i.e. the computational power is already maxed out. In this case, the online computation of $g(x, y, \sigma_r)$ will add to the run time. It is therefore advisable to pre-compute the values of $g(x, y, \sigma_r)$ as it will result in a reduced run time. Unfortunately, this behavior is problem- and hardware-dependent. Thus, the most efficient implementation depends on the actual use-case. This effect can be exploited for various applications in the field of medical image computing [6, 7].

```
bilateralFilter (image){
     sumWeight = 0;
     sumFilter = 0;
     step = halfWidth*2+1;
     for (j=0; j < step; j++){
          for (i=0; I < step; i++){
               nx = halfWidth + i;
               ny = halfWidth + j;
               g1 = geomClose(nx, ny, sigma_spatial)
               g2 = intensityClose(nx, ny, x, y,
                    sigma_int, image)
               sumWeight += g1*g2;
               sumFilter += g1*g2*image[(x-nx)+((y-ny)*width)];
          }
     }
     return sumFilter/sumWeight;
}
```

Algorithm 8.2: Pseudo code for a bilateral filter. Note the similarity to Algorithm 8.1. The only difference is the introduction of a new function `intensityClose` that computes the closeness of the two intensities of the image.

8.3.2 Bilateral Filter

The bilateral filter [8] is an extension of the Gaussian filter that adds edge-preservation. The concept is very easy to follow, the implementation is straight-forward, and its parameterization can be determined directly from the image. These properties make the bilateral filter to one of the most popular edge-preserving filtering methods. Unfortunately, the bilateral filter has a rather high computational complexity. Parallel execution hardware such as graphics cards have lessened this burden which lead to a broad application of the bilateral filter.

The definition of the bilateral filter is

$$p_b'(x,y,\sigma_r,\sigma_i) = \frac{1}{k(x,y,\sigma_r,\sigma_i)} \sum_{x',y' \in \mathcal{N}} p(x',y') \cdot g(x',y',\sigma_r) \cdot \; g(p(x',y') - p(x\,,y\,),\sigma_i) \quad (8.8)$$

$$k(x,y,\sigma_r,\sigma_i) = \sum_{x',y' \in \mathcal{N}} g(x',y',\sigma_r) \cdot \; g(p(x',y') - p(x\,,y\,),\sigma_i)$$

where we introduce another standard deviation σ_i which describes the intensity similarity between the pixel at $p(x',y')$ and $p(x\,,y\,)$. If they are identical, their difference is zero which leads to a maximum in the Gaussian function. If the kernel is evaluated in a uniform image area, all differences will be zero which leads the normal Gaussian kernel. In the presence of an intensity difference, the pixels that have different intensities will contribute less to the filtering, i.e. the edge is preserved. Doing so, the filtering is not shift-invariant and pre-computation of $g(p(x',y') - p(x\,,y\,),\sigma_i)$ is impossible.

In order to choose σ_i, it is advisable to measure the smallest edge in the image that should still be preserved. For projection images, σ_i should be chosen very small as too high values might introduce streaks in the reconstruction. In this case 10% of the smallest edge value are advisable. In reconstruction space, σ_i can be chosen up to the value of the smallest edge. Note that this will already lead to a slight blurring of the edge [9].

Algorithm 8.2 depicts pseudo kernel code. Again, the parallelization is performed over the image coordinates (x,y). Also in this implementation, we do not pre-computation the geometric closeness as in the previous example which may lead to increased run time depending on the execution hardware and problem size.

8.3.3 Joint Bilateral Filter

The joint bilateral filter is an extension of the bilateral filter that introduces a so-called guidance image [10]. The idea is that one image shows the desired information in very noisy conditions while another one shows the correct edge information. In the original paper, the desired information is a color photograph that was taken without flash and the edge information comes from an image with flash that has suboptimal color information. Thus, the bilateral filter is adjusted in the following manner:

```
jointBilateralFilter (image, guide){
     sumWeight = 0;
     sumFilter = 0;
     step = halfWidth*2+1;
     for (j=0; j < step; j++){
          for (i=0; I < step; i++){
               nx = halfWidth + i;
               ny = halfWidth + j;
               g1 = geomClose(nx, ny, sigma_spatial)
               g2 = intensityClose(nx, ny, x, y,
                    sigma_int, guide)
               sumWeight += g1*g2;
               sumFilter += g1*g2*image[(x-nx)+((y-ny)*width)];
          }
     }
     return sumFilter/sumWeight;
}
```

Algorithm 8.3: Pseudo code for the joint bilateral filter. The only difference is that `intensityClose` now operates on a guidance image `guide`.

$$p_j{}'(x,y,\sigma_r,\sigma_i) = \frac{1}{k(x,y,\sigma_r,\sigma_i)} \sum_{x',y' \in \mathcal{N}} p(x',y') \cdot g(x',y',\sigma_r) \cdot \ g\big(p_g(x',y') - p_g(x\ ,y\ ),\sigma_i\big) \quad (8.9)$$

$$k(x,y,\sigma_r,\sigma_i) = \sum_{x',y' \in \mathcal{N}} g(x',y',\sigma_r) \cdot \ g\big(p_g(x',y') - p_g(x\ ,y\ ),\sigma_i\big)$$

In this formulation, we have now introduced a guidance image $p_g(x\,, y\,)$ that is used to compute the geometric closeness. Everything else is identical to the bilateral filter. Subsequently, also Algorithm 8.3 is almost identical to Algorithm 8.2. The only change is the introduction of the guidance image. If the original image is supplied as guidance image, this implementation will result in the original bilateral filter.

While the implementation is straight-forward, results are astonishing and find applications in many fields from super resolution [11], over multimodal imaging to energy-resolving detectors [9].

8.3.4 Guided Filter

The guided filter [12] is a common alternative to the joint bilateral filter. The main idea is to express the filtered image as a linear transform of the guidance image:

$$p_{gf}{}'(x, y) = a_k p_g(x, y) + b_k \tag{8.10}$$

The local coefficients a_k and b_k are found as the solution to the following optimization problem:

$$a_k, b_k = \operatorname{argmin}_{a,b} \sum_{x',y' \in \mathcal{N}} \left\{ \left[a\, p_g(x', y') + b - p(x', y') \right]^2 + \epsilon a^2 \right\} \tag{8.11}$$

These optimal coefficients are found as

$$a_k = \frac{\mathrm{Cov}_{\mathcal{N}}(p, p_g)}{\mathrm{Var}_{\mathcal{N}}(p_g) + \epsilon} \tag{8.12}$$

$$b_k = \mathrm{Mean}_{\mathcal{N}}(p) - a_k \cdot \mathrm{Mean}_{\mathcal{N}}(p_g)$$

Where $Mean_{\mathcal{N}}(p)$ computes the mean of the image p in the local neighborhood $\mathcal{N}$, $Var_{\mathcal{N}}(p)$ computes the local variance of p and $Cov_{\mathcal{N}}(p, p_g)$ computes the covariance between p and p_g.

If we consider the case where the guidance image is identical to the filtered image and $\epsilon = 0$, we can make the following observations:

- The covariance in the numerator of a_k becomes a variance.

- If the neighborhood is completely homogeneous, this variance will become zero and therewith a_k will become 0. Thus the filtered image will only consist of b_k which is the mean value of the neighborhood.
- If the variance is not 0, a_k will become 1 and thus b_k will be zero. In those regions, the filtered image will consist only of the original image values.

With a value of ϵ that is higher than zero, one can adjust the filter behavior between these two extreme cases. Note that for actual application on an image, the local values of a_k and b_k have to be determined first for every pixel. Then, a low-pass filter has to be applied to all of the a_k and b_k before the final filter is computed. As mean filters can be implemented very efficiently, the guided filter can be applied fast, even if it is applied on large neighborhoods. Algorithm 8.4 describes a summary of the steps that need to be done in order to compute the guided filter. In contrast to previous algorithms, it does not describe kernel code, but subsequent steps that can be executed parallel for every pixel.

```
guidedFilter (image, guide){
    compute $a_k$ and $b_k$ for all pixels
    apply smoothing $a_k$ and $b_k$ for all pixels
    compute final filter for all pixels
}
```

Algorithm 8.4: In contrast to previous algorithms, the guided filter has to be separated into several kernel executions. Thus it cannot be described as a single kernel. The code above gives an abstract description on how the filter is implemented.

8.3.5 Structure Tensor

The idea of the structure tensor-based filtering is to use a structure detector – the structure tensor - to steer the denoising of the image. Before doing so, we decompose the projection stack into directional high frequency components and into a single low frequency component [13]. Next, a structure tensor is computed for every pixel in order to measure the amount of structure and its contributions to the different spatial directions. If a high amount of structure is detected, more weight is given to the respective directional high-pass component. If the tensor reports a low value, more weight is given to the low-pass component. In the end, the filtered image is recovered as a weighted sum of its high and low frequency components.

Again, this filter cannot be described by a single kernel. Algorithm 8.5 reports the individual steps which can be computed in parallel. For the filtering, the use of fast Fourier transforms is

```
structureTensorFilter (image){
     compute directional high-pass components at each pixel
     compute low-pass component at each pixel
     compute structure tensor for each direction at each pixel
     low-pass filter structure tensor
     compute filter output for each pixel
}
```

Algorithm 8.5: In order to filter the image with a structure tensor, the image needs to be decomposed into directional high frequency components and a low pass component. Then the structure tensor is computed for each pixel and low-pass filtered. In the final step the filter output is computed as a weighted sum of low-pass and high-pass components where the weights are determined from the structure tensor.

advisable. Note that if this code is to be executed on a graphics card on an entire projection stack with six directional components in floating point precision up to 24 times the projection stack memory may be required. Thus, it is advisable to process the data in smaller blocks.

8.4 Denoising for CT

The previously described methods are directly applicable to X-ray projection images or to reconstructed CT slice data in 2D or 3D. However, there have been many adaptations of these methods that aim at modeling prior information about X-ray imaging process into the denoising process. In the following, we present a small subset of the literature that is found on this broad topic. We regard these methods to be the most important ones in everyday use.

Projection Domain

In filtered back-projection-type algorithms, streak noise as shown in the beginning of this chapter can be omitted by projection-based noise reduction. All of the previously mentioned methods can be extended to incorporate the average path-length of the photons through the object. Kachelriess et al. developed a method based on triangular filters that performs filtering on the entire projection stack [14]. Zeng at al. included such ray-dependent weighting into the ramp filter of the filtered back-projection algorithm and combined it with an edge preserving filtering in reconstruction domain [15]. Furthermore, also the structure tensor can be applied in projection domain by processing the entire projection stack [13]. Schäfer et al. investigated several implementations of edge-preserving and noise-adaptive filtering and found that filtering minimizing the total variation worked best in their high-contrast examples [16].

8.4.1 Iterative Reconstruction

Although iterative reconstruction is an order of magnitude slower than analytic reconstruction methods, there is a broad variety of iterative reconstruction approaches that aim at reducing the image noise and reducing the X-ray dose. Probably the most well-known one is penalized least squares iterative reconstruction [17]. It aims at weighting each ray with its noise variance. Doing so, the noisy ray get a reduced weight while the more reliable rays with a lower path length get more influence in each iteration. Recently, iterative methods that reduce the total variation in reconstruction space became more and more popular [18]. In these approaches, the assumption is made, that the object of interest is piece-wise constant which allows extreme noise suppression.

If performed in an extreme way, the resulting images are often described as surrealistic or commix-like. Thus such regularization has to be performed with caution. In general, iterative reconstruction methods are able to deliver superior image quality compared to traditional methods. However, their parameterization is difficult and requires a lot of experience and/or grid search of the parameter space which increases their run time even further. Thus, iterative methods are not of great clinical relevance today. If applied in a clinical context, often only a single iteration is performed.

8.4.2. Reconstruction Domain

There is a large body of literature on noise reduction in reconstruction space. Many approaches build on the previously presented denoising methods. A quite different approach that uses wavelets for denoising was presented by Borsdorf et at. [19]. This approach aims at estimating the noise by a dual reconstruction. The set of projection images is split in half and each set is reconstructed individually. Then both reconstructions are correlated against each other in wavelet domain and only correlated structures, i.e. structures that belong to the object are preserved. Bruder et al. have shown that given a high projection number as in the case of CT penalized least squares regularization can be expressed entirely in image domain as a non-linear filter [20]. Given a piece-wise constant object, non-linear filtering can also achieve reconstructions that resemble the outcomes of total variation-based iterative reconstruction methods [21]. In this in line with Manhart et al. who reported a filtered back-projection-type method for perfusion C-Arm CT that delivers image quality that is en par with comparable iterative methods [22]. The run time of the analytic method, however, was more than 23 times faster than the iterative method, although both were implemented on graphics cards.

8.5 Summary

This chapter gave a comprehensive summary of denoising algorithms for GPU. We described the underlying physical processes that cause the noise from photon statistics to the noise in reconstruction domain and gave examples what kind of structure the noise can exhibit. This was followed by an introduction to denoising methods in general with a focus on GPU implementation. We started from shift-invariant Gaussian filtering, introduced the edge-preserving bilateral filter, the guided filter, and structure tensor-based image filters. In the last part of the chapter, we discussed important implementations of noise reduction methods into the reconstruction process which included projection-based, iterative, and reconstruction-based denoising methods. In order to implement fast and reliable noise reduction, a combination of projection-based and reconstruction-based filtering is probably the fastest and easiest way to incorporate noise reduction [9, 15].

All examples in this book chapter have been created with CONRAD an open-source software framework for CT Simulation and Reconstruction [23].

References

[1] Manhart, Michael; Fieselmann, Andreas; Deuerling-Zheng, Yu; Maier, Andreas; Kowarschik, Markus. Dynamic Reconstruction with Statistical Ray Weighting for C-Arm CT Perfusion Imaging. Proc. 12th Fully 3D (12th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine), Lake Tahoe, CA, USA, 18.06.2013, pp. 221-224, 2013

[2] J. Hsieh, “Adaptive filtering approach to the streaking a artifact reduction due to X-ray photon starvation,” Medical Physics, vol. 25, pp. 2139–2147, 1998.

[3] A. Borsdorf. Adaptive Filtering for Noise Reduction in X-Ray Computed Tomography. PhD Thesis. Friedrich-Alexander University Erlangen-Nuremberg. 2010

[4] T. Buzug. Computed Tomography: From Photon Statistics to Modern Cone-Beam CT. Springer Heidelberg, Germany. 2008

[5] J. Baek, N. Pelc. Local and global 3D noise power spectrum in cone-beam CT system with FDK reconstruction. Medical Physics, vol. 38, pp. 2122-2131. 2011.

[6] Zinsser, Timo; Keck, Benjamin. Systematic Performance Optimization of Cone-Beam Back-Projection on the Kepler Architecture. Proceedings of the 12th Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine (Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine (Fully3D)), Lake Tahoe, CA, USA, 16.06-21.06.2013, pp. 225-228, 2013

[7] Maier, Andreas; Hofmann, Hannes; Schwemmer, Chris; Hornegger, Joachim; Keil, Andreas; Fahrig, Rebecca. Fast Simulation of X-ray Projections of Spline-based Surfaces using an Append Buffer. Physics in Medicine and Biology, vol. 57, no. 19, pp. 6193-6210, 2012

[8] V. Aurich and J. Weule, “Non-linear Gaussian filters performing edge-preserving diffusion,”

in Mustererkennung 1995. Informatik aktuell, G. Sagerer, S. Posch, and F. Kummert, Eds. Springer Berlin Heidelberg, 1995, pp. 538–545.

[9] Manhart, Michael; Fahrig, Rebecca; Hornegger, Joachim; Dörfler, Arnd; Maier, Andreas Guided Noise Reduction for Spectral CT with Energy-Selective Photon Counting Detectors Proceedings of the Third CT Meeting (The Third International Conference on Image Formation in X-Ray Computed Tomography), Salt Lake City, UT, USA, 23.06.2014, pp. 91-94, 2014

[10] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, “Digital photography with flash and no-flash image pairs,” ACM Transactions on Graphics (TOG), vol. 23, no. 3, pp. 664–672, 2004

[11] Farsiu, Sina, Michael Elad, and Peyman Milanfar. "Multiframe demosaicing and super-resolution of color images." Image Processing, IEEE Transactions on 15, no. 1 (2006): 141-159.

[12] He, Kaiming, Jian Sun, and Xiaoou Tang. "Guided image filtering." In Computer Vision–ECCV 2010, pp. 1-14. Springer Berlin Heidelberg, 2010.

[13] A. Maier, L. Wigström, H. Hofmann, J. Hornegger, L. Zhu, N. Strobel, and R. Fahrig. Three-dimensional anisotropic adaptive filtering of projection data for noise reduction in cone beam CT. Medical Physics, Vol. 38, No. 11, pp. 5896-5909, 2011.

[14] Kachelriess, Marc, Oliver Watzke, and Willi A. Kalender. "Generalized multi-dimensional adaptive filtering for conventional and spiral single-slice, multi-slice, and cone-beam CT." Medical physics 28, no. 4 (2001): 475-490.

[15] Zeng, Gengsheng L., and Alex Zamyatin. "A filtered backprojection algorithm with ray-by-ray noise weighting." Medical physics 40, no. 3 (2013): 031113.

[16] Dirk Schäfer, Peter van de Haar, Michael Grass. Comparison of Gaussian and non-isotropic adaptive projection filtering for rotational 3D X-ray angiography.Proc. of the The 12th

International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine. 2013. pp 130-133.

[17] Fessler, Jeffrey A. "Penalized weighted least-squares image reconstruction for positron emission tomography." Medical Imaging, IEEE Transactions on 13, no. 2 (1994): 290-300.

[18] X. Pan, E. Sidky, and M. Vannier, “Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction?” Inverse problems, vol. 25, no. 12, pp. 1–50.

[19] Borsdorf, Anja, Rainer Raupach, Thomas Flohr, and Joachim Hornegger. "Wavelet based noise reduction in CT-images using correlation analysis." Medical Imaging, IEEE Transactions on 27, no. 12 (2008): 1685-1703.

[20] Bruder, R. Raupach, J. Sunnegardh, M. Sedlmair, K. Stierstorfer, and T. Flohr, “Adaptive iterative reconstruction,” in SPIE Medical Imaging, vol. 7961, 2011, pp. 79610J–79610J–12. [Online]. Available: http://dx.doi.org/10.1117/12.877953

[21] Christian Riess, Martin Berger, Haibo Wu, Michael Manhart, Rebecca Fahrig and Andreas Maier. TV or not TV? That is the Question. Proc. of the 12th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine. 2013. pp. 341-344.

[22] Manhart, Michael T., André Aichert, Tobias Struffert, Yu Deuerling-Zheng, Markus Kowarschik, Andreas K. Maier, Joachim Hornegger, and Arnd Doerfler. "Denoising and artefact reduction in dynamic flat detector CT perfusion imaging using high speed acquisition: first experimental and clinical results." Physics in medicine and biology 59, no. 16 (2014): 4505.

[23] A. Maier, H. Hofmann, M. Berger , P. Fischer, C. Schwemmer, H. Wu, K. Müller, J. Hornegger, J.-H. Choi, C. Riess, A. Keil, and R. Fahrig. CONRAD - A software framework for

cone-beam imaging in radiology. Medical Physics, Vol. 40, No. 11, 2013.

Figure 8.1: The intensity measured at the detector is found as integral over the source spectrum after attenuation by the object. This spectrum was simulated using CONRAD.

Figure 8.2: Noise-free and noisy projection of a water cylinder. The third image shows the difference between the two. The longer the path length in water, the higher is the noise variance. These images were created using CONRAD.

Figure 8.3: The left side shows the noise in the reconstruction of a water cylinder phantom. The noise is stronger in the center than in the off-center regions. The right side shows an elliptic phantom with two high intensity insets. In the center, streak noise emerges that is caused by the high attenuating structures. For both simulations $N_0 = 90.000$ photons @ 75 keV per pixel were used. The display window/level is [-200,200] HU. These images were created using CONRAD.