# GPU-Acceleration of Attenuation and Scattering Compensation in Emission Computed Tomography

Fang Xu and Klaus Mueller

***Abstract*—In functional imaging, the modeling of emission ray attenuation and scattering during reconstruction can greatly improve image quality. Proper modeling of these effects in the forward projection step of iterative algorithms, such as OS-EM, can yield a significantly more accurate estimate of the required grid correction in the subsequent back projection step. This, in turn, leads to faster and more accurate reconstruction of the emission image. Graphics hardware boards (GPUs) have recently emerged as a powerful and affordable means to accelerate many 3D tomographic reconstruction algorithms, both analytical and iterative. This paper presents an efficient and GPU-amenable method that incorporates attenuation and scattering modeling into these.**

***Index Terms*—Computed Tomography, Emission Tomography, Iterative Algorithms**

## I. Introduction

WHILE in transmission tomography the objective is to reconstruct a 3D image (volume) of the object's X-ray attenuation properties, using an external radiation source, in emission tomography one seeks to reconstruct the (internal) radiation sources themselves. This gives rise to a very different (projection) image generation scenario, somewhat reminiscent to volume rendering with self-emitting particles. The projection scenario is different since, depending on their distance from the detector, local emissions can have varied impacts on the projection image, since they potentially lose different amounts of energy on their paths across the tissue to the detector. These attenuation variations can be significant and must be incorporated into the weight matrix [3].

Another source of error are the scattering events occurring along the photon paths [2]. These are mostly due to photon interactions with the traversed tissue, leading to stray photons that are eventually counted in detector bins neighboring the intended one. Here, the amount of detector blurring a particular emission site causes is influenced by the local scattering properties of the traversed tissue as well as the distance of this site to the detector. The most appropriate technique to model scattering is a Monte-Carlo simulation, using the current reconstruction instance as the source. However, such a simulation is (currently) computationally infeasible to conduct within an iterative procedure, at least when it comes to clinical routine. A good approximation can be obtained by the slice-by-slice blurring method that has been proposed for emission CT [1] as well as for volume rendering [4]. In both approaches, the blurring is guided by the local scattering properties of the tissue. The scattering as well as the attenuation modeling rely on the existence of a prior CT scan to provide an estimate of the attenuation and scattering properties of the tissue.

Commodity graphics hardware boards (GPUs) have achieved remarkable speedups in various disciplines of Computed Tomography (CT). Their programmability allows even complex schemes to be fully accelerated in their pipelined multi-processor array. Peak performances of 500 GFlops ($10^9$ floating point operations/s) and more are now possible on a single PC equipped with a board that is available in every computer outlet for less than $500. Although the trend is currently towards generalizing these architectures to general purpose computing within the SIMD (Same Instruction Multiple Data) data processing paradigm, adhering to the native graphics (hardwired) pipeline facilities and strategies can be advantageous in CT reconstruction. For example, we have recently shown [5] that a $512^3$ full floating point precision volume can be reconstructed (using Feldkamp's filtered back-projection algorithm) from 360 X-ray projections oriented in general position in less than 9s. In less recent work [6] we have also described a general framework that expresses a popular set of analytical and iterative reconstruction algorithms in a form that resonates well with the underlying architecture and programming model of GPUs. However, that paper did not address the specific GPU-implementation aspects of emission reconstruction with proper modeling of attenuation and scattering effects. These are now described in this paper.

The outline of our paper is as follows. In Section 2, we describe our method from a more theoretical angle, and in Section 3 we provide implementation details. Section 4 present our results, and Section 5 ends with conclusions.

Manuscript received February 15, 2007. This work was supported in part by the NIH under Grant R21 EB004099-01 and the Keck Foundation.

Fang Xu and Klaus Mueller are with the Center of Visual Computing, Computer Science Department, Stony Brook University, Stony Brook, NY, 11794 (phone: 631-632-1524; fax: 631-632-8445; e-mail: {fxu, mueller}@cs.sunysb.edu).

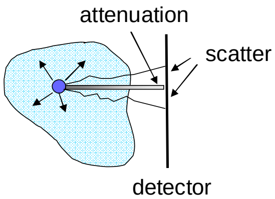
## II. Theoretical Considerations

The figure below illustrates the effects we attempt to model: attenuation and scattering. For the former we use the standard volume rendering integral. Here, *s* parameterizes a linear ray terminating in a given detector bin *p*. The ray integrates emissions *c* along its path, subsequently attenuated by the encountered attenuations $\mu$..

$$p = \int_{s=0}^{l} c(s)\, e^{-\int_{t=0}^{s} \mu(t)dt}\, ds$$

$$\approx \sum_{i=0}^{L/\Delta s} c(i\Delta s)\, e^{-\sum_{j=0}^{i-1} \mu(j\Delta s)} = \sum_{i=0}^{L/\Delta s} c(i\Delta s) \prod_{j=0}^{i-1} e^{-\mu(j\Delta s)}$$

The second part of this equation approximates this integral as a sequence of equidistant interpolations, $\Delta s$ apart.



To describe the (approximate, non-Monte Carlo) technique used to model the scattering, let us first consider a scattering event in a differential volume patch *dV*. Here, depending on the patch's scatter properties, a ray of photons suffers some amount of diffusion, which can be modeled by a suitable diffusion function kernel, such as a box, tent, or Gaussian, where the extent of this kernel is determined by the amount of local scattering potential, estimated from a map indexed by the underlying CT data. This local model can be extended to a global one by recursion, where a detector-aligned slice buffer is advanced step-by-step from the rear of the volume to the detector, and the scatter-diffusion process is modeled, at each such step, by convolving the slice image with a variable-extent blurring function. Here, the size of the filter is dependent on the local scattering properties (higher scattering coefficients widen the filter). Also at each step, an emission volume slice is interpolated and added to the advancing slice buffer. In fact, this scheme can be combined with the attenuation modeling, as we shall show in Section III.

Modeling these effects in the forward projection provides a better estimate of the actual image generation process, given the current state of the emission volume under reconstruction. This in turn provides a better estimate of the required grid correction for back-projection (which favors reconstruction speed and quality). This estimate can then be back-projected via regular means (without modeling attenuation and scattering effects) or by including these effects. The former leads to the concept of unmatched projector/back-projector pair, which has been frequently used. We have also found the reconstructions to be of higher quality with the unmatched scheme, at least for our (relatively simple) phantom, but we will provide implementation details and results for the matched projector/back-projector pair as well.

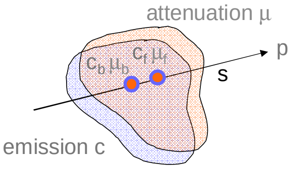
## III. Implementation Details

We first reformulate the above attenuation equation into a framework more convenient for implementation on graphics hardware. Using a Taylor expansion we get:

$$p \approx \sum_{i=0}^{L/\Delta s} c(i\Delta s) \prod_{j=0}^{i-1} (1-\mu(j\Delta s))$$

Now normalizing all CT density $\mu$ values into a range [0.0, 1.0] we obtain the following recursive expression:

$$\mathrm{c}_b = c_b(1-\mu_f) + c_f = c_b t_f + c_f$$

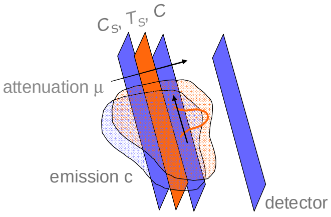


In volume rendering, this is called *back-to-front compositing*. The new (back) emission $c_b$ is calculated by adding the previous back emission $c_b$ to the newly interpolated front emission $c_f$. But before it is added it must be attenuated by the interpolated transparency $t_f$ at this point (note, the higher the attenuation, the lower the transparency). An alternative form is *front-to-back compositing*:

$$c_f = c_f + c_b t_f \qquad t_f = t_f(1-\mu_b) = t_f t_b$$

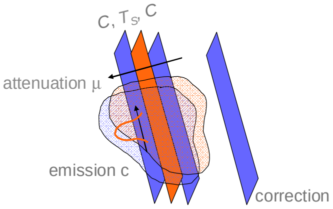
For this we need to keep two advancing buffers., one for the advancing transparency, $t_f$, and one for the advancing emission front, $c_f$ (now $c_b$ is the currently interpolated emission). First the emission buffer is updated, then the transparency buffer. Since the effect of each transparency layer is multiplicative, a new (front) $t_f$ is formed by multiplying the present $t_f$ by the currently interpolated $\mu_b$ (in fact, *$(1-\mu_b)$*).

Now combining this with the scattering yields the following implementation strategy, illustrated next, along with the respective algorithm. We first show the approach for projection, which uses the back-to-front compositing scheme:



```
Set emission buffer $C$=0
Step back to front, at each step:
  interpolate emission slice $C_S$
  interpolate transp. slice $T_S$
  blur $C$ using $T_S$
  composite $C = C \cdot T_S + C_S$
```

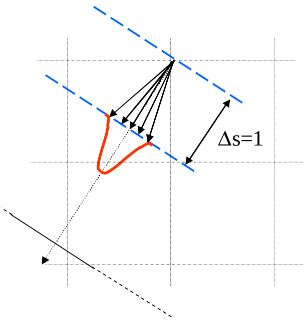
The back-projection, using front-to-back compositing is:



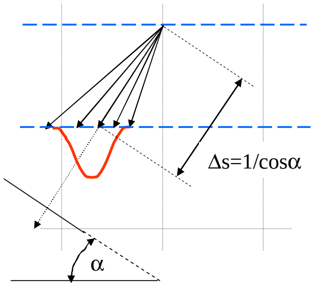
```
Initialize correction buffer $C$
Step front to back, at each step:
  spread (and add) $C$ into the
emission volume
  interpolate transp. slice $TS$
  blur $C$ using $TS$
  update $C = C \cdot TS$
```

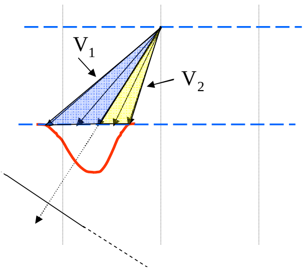
A possible hardware implementation of this scheme would be to store the *T* (CT) volume and the *C* (emission) volume in 3D textures and then use the detector-aligned 3D slicing for interpolation in the forward projection, and volume-slice aligned updates (volume writes) in the back-projection step. There are, however, some practical limitations to this scheme. These originate in the current lack of hardware support for efficient writes to a 3D texture. While Framebuffer objects (FBOs) provide this ability now for the latest GPU generation, this capability is still not optimized and not well supported by the hardware drivers. Even though one could perform the forward projection with a 3D texture and then switch to a stack of 2D textures for back-projection, the repeated data reformatting from 3D to 2D textures and back would be very expensive. We therefore propose a scheme that uses 2D texture stacks for both projection modes. In that respect, what needs to be resolved is the question if the 2D texture slice-based forward projection can be made sufficiently accurate, compared to the true detector-aligned slicing shown in the figures above. We have designed a few schemes to that effect, which are described next.

The figure to the right shows the blurring kernel in the ideal detector-aligned buffer configuration. Here the recursive blurring (the red kernel) can occur at any arbitrary buffer (blue dashed lines) distance Δs (Δs=1 is reasonable). This is the configuration used when a 3D texture is interpolated.



Next, the figure on the right shows the texture-slice (or axis) aligned situation (assuming the texture slice distance is 1). In this case the scattering will occur at a larger distance and therefore the width of the blurring kernel should be scaled up accordingly, that is, by a factor 1/cosα, where α is the rotation angle of the detector.





Due to the perspective (cone-beam) distortion, the scattering on one side of the principal direction (here the right half-cone as seen from the scattering source) will occur in a larger distance before entering the slice buffer than the other (the left half-cone), see the figure on the right. We can correct for this as well, by additionally scaling the two kernel side lobes according to their subtended half-cone volumes, $V_1$ and $V_2$.

## IV. Results

To validate the accuracy and performance of our GPU reconstruction framework, we used a PC equipped with an Athlon 2.2GHz CPU and a Geforce 8800 GTX graphics card. We employed a 3D phantom composed of a set of ellipsoids of varying emission values.

Fig. 1 compares the slice-by-slice blurring results obtained with the detector-aligned vs. the axis-aligned projector for three orientations α: 0˚, 20˚, 40˚. Both simulations use the same scattering model and volume. Next to the projections we show the difference images and the intensity profile along the line indicated in the first image. A good fit is observed. We also computed the overall RMS error and found it to be between 1-2% of the maximum value in a projection. We conclude from this study that the axis-aligned projector is well suited for the recursive scatter simulation we use in our GPU-accelerated iterative emission reconstruction framework.

Fig. 2 shows a set of representative projections obtained with our simulator, with emissions only, (E) emission with attenuation (E+A), emission with scattering (E+S), and emission with scattering and attenuation (E+S+A). Scattering creates substantially more blur, while attenuation weakens the projections of emissions traversing highly attenuating material, both with and without scattering.

Fig 3 shows a representative slice from a 3D reconstruction of our phantom (10 SART iterations), for various modeling scenarios arranged into rows. The first column shows the reconstructed slice when the (row) effects are not modeled, while the second column shows the slice when modeling took place. In the final column we show the intensity profiles for the line indicated in the first image. We compare the original phantom profile (solid grey), the profile obtained when the effect has not been modeled (dotted) and the profile when the effect has been modeled (solid black). We see that in all cases the contrast is greatly improved, the features are sharper, and the profiles match the original better when the effect is modeled. We also observe that without attenuation/scattering (A+S) modeling, the small ellipsoid between the two large ones in the upper third of the phantom can not be detected.

Finally, Table 1 compares our timing results. We observe that adding attenuation and scattering only to the projector (in an unmatched projector/back-projector reconstruction framework) has a relatively small impact on performance (less than a factor of 2). This verifies the observations of [1]. The results shown in Fig. 3 were all obtained with configuration. On the other hand, modeling these effects for the back-projection operator is about 10 times more expensive. Overall, it takes 5 times longer if attenuation and scattering are modeled in both he projection and back-projection stages (in the matched projector/back-projector).

**Table I**

RECONSTRUCTION TIMINGS: $128^3$ VOLUME, 160 PROJECTIONS

| | **Projection** | **Backproj.** | **1 iter.** | **10 iter.** |
|---|---|---|---|---|
| Transmission CT | 0.7s | 0.7s | 1.7s | 18s |
| Emission CT (matched) | 1.2s | 7.3s | 8.7s | 90s |
| Emission CT (unmatched) | 1.2s | 0.7s | 2.2s | 23s |

## V. CONCLUSIONS

We have demonstrated that attenuation and scattering effects as well as their compensation in iterative emission CT can be efficiently performed with GPUs. Unmatched operators are significantly more efficient and already seem effective to improve reconstruction quality in the presence of these effects. While the axis-aligned projector appeared sufficient, future work will aim to include the 3D texture approach once it is better supported by the hardware.

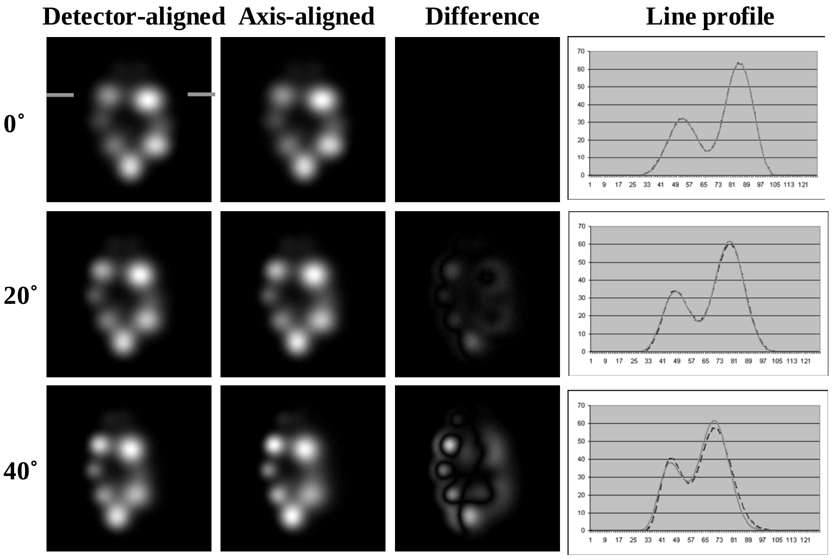


**Figure 1:** Projection evaluation of detector and axis-aligned scattering models. Profiles: detector-aligned (solid), axis aligned (dashed) projections

## REFERENCES

[1] C. Bai, G. Zeng. and G. Gullberg, “A slice-by-slice blurring model and kernel evaluation using the Klein-Nishina formula for 3D scatter compensation in parallel and converging beam SPECT,“ *Physics in Medicine and Biology*, vol. 45, pp. 1275-1307, 2000.

[2] Z. Liang, T.G. Turkington, D.R. Gilland, R.J. Jaszczak, and R.E. Coleman, “Simultaneous compensation for attenuation, scatter and detector response for SPECT reconstruction in three dimensions,” *Phys. Med. Biol.*, vol. 37, pp. 587- 603, 1992.

[3] M.A. King, B.M.W. Tsui, and T.S. Pan, ”Attenuation compensation for cardiac single-photon emission computed tomographic imaging: Part 1, Impact of attenuation and methods of estimating attenuation maps,” *J. Nucl. Cardiol.*,vol.2, pp. 513-524, 1995.

[4] J. Kniss, S. Premoze, C. Hansen, P. Shirley, and A. McPherson, "A model for volume lighting and modeling," *IEEE Trans. on Visualization and Computer. Graphics,* vol. 9, no. 2, pp. 150-162, 2003.

[5] K. Mueller, F. Xu, and N. Neophytou, "Why do GPUs work so well for acceleration of CT?," *SPIE Electronic Imaging '07* San Jose, Jan. 2007.

[6] F. Xu and K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," *IEEE Trans. on Nuclear Science*, vol. 52, no. 3, pp. 654-663, 2005.

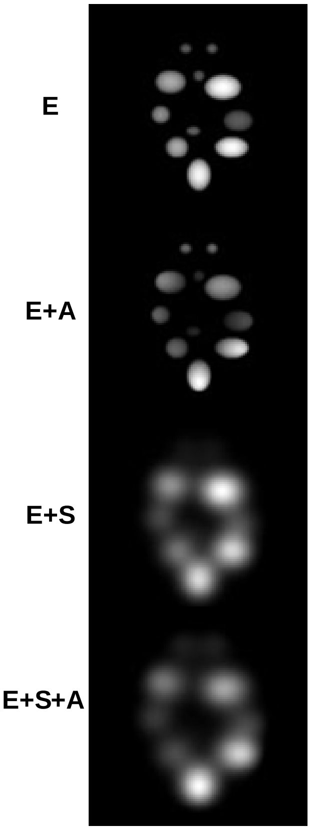


**Figure 2.** Projections obtained with different effects. **E**: emission; **A**: attenuation; **S**: scattering.

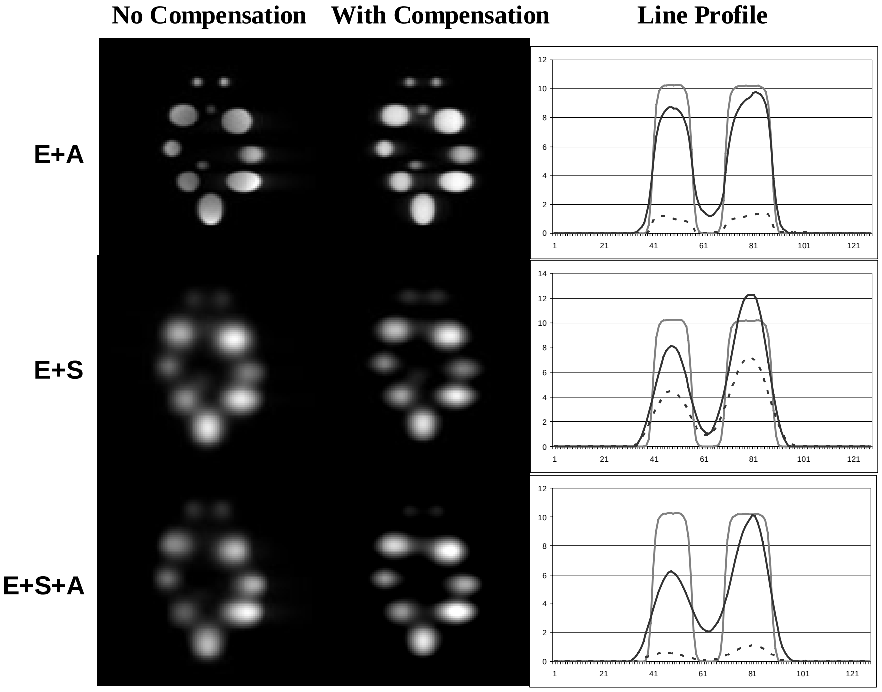


**Figure 3.** Reconstructions results obtained with different effects. **E**: emission; **A**: attenuation; **S**: scattering