3D Motion Blur For PET Using GPGPU

Dóra Varnyú*
Supervised by: László Szirmay-Kalos†

Department of Control Engineering and Information Technology
Budapest University of Technology and Economics
Budapest / Hungary

Abstract

Computing motion blur is important not only in video games and movie rendering, but also in physical simulations where the effect of the movement has to be calculated and compensated. One such application is dynamic Positron Emission Tomography (PET), which is one of today's most important medical imaging techniques. When studying neurological disorders like epilepsy with PET, the involuntary movement of the patient during a seizure can ruin the examination results. Because of this, motion compensation should be incorporated into the reconstruction process. When the movement is rapid, accurate motion blur presents a challenge even if only 2D image space results are needed and hardware capabilities can be exploited. However, in the case of PET, the blurred result is expected in the 3D space. Blurring the motion of a 3D object in the 3D space can be done by dividing its bounding box into homogeneous voxels and tracking the path of each voxel independently. As low execution time is crucial, simplifications must be applied at the expense of accuracy. The solution proposed in this paper approximates the path of each voxel with a sequence of line segments. The segments can then be sampled by properly adapted 3D line drawing algorithms such as the Antialiased Bresenham Algorithm. Implementation is accelerated by parallel execution on GPU using the CUDA platform.

Keywords: 3D, Motion Blur, Line Drawing, Antialiased Bresenham, Positron Emission Tomography, PET, CUDA, GPU, GPGPU

1 Introduction

Dynamic Positron Emission Tomography (dynamic PET) analyzes the dynamic nature of biological processes through observing the accumulation and emptying of drugs in organs [10, 14, 15]. At the beginning of the examination, a small amount of radioactive material called radiotracer is injected into the patient, which then spreads across the body and gets absorbed into the tissues. The emitted radiation is detected by the tomograph and the 3D spatial distribution of the radiotracer is reconstructed. Cells with rapid metabolism, such as cancer cells, absorb more tracer and thus appear as bright spots on the image.

During reconstruction, the measured volume containing the body of the patient is divided into a 3D grid of homogeneous voxels. The task is to determine the amount of the radiotracer in each of the voxels based on the emission detected by the detector ring surrounding the volume.

When studying neurological disorders like epilepsy, the self-motion of the patient is often unavoidable, which can ruin the examination results. The problem can be attacked by measuring real-time the movement using markers [7] and incorporating the gathered information into the reconstruction process. The solution we are proposing uses the information about the movement to apply motion blur on the voxel array during measurements.

Object position at any moment can be described with a geometric transformation consisting of translation and rotation. As motion is a change in position, movement is also described by a geometric transformation in the form of a translation vector and a quaternion for rotation.

The input of the motion blur computation is the voxel array containing the current estimation of the number of radioactive decays that occured in the voxels and the geometric transformation determined using the markers on the patient. The input array is considered to be moving along with the patient. During motion blur, the task is to map the voxels of this moving array into a motionless destination array of the same size. The output is the unmoving array with voxel activities mapped from the moving array.

The classical solution to the problem is to calculate the position of each moving voxel's center point in very small time steps, i.e. determine which voxels of the motionless array does the currently examined moving voxel overlap. This means interpolating the geometric transformations describing object motion and then applying the resulting transformation at every time step. Translation vectors are interpolated linearly and the quaternions using Spherical Linear Interpolation (SLERP, Figure 1).

When accuracy is an important aspect, each voxel path should be sampled at a sufficiently high frequency relative to the motion dynamics. Furthermore, since in most cases the voxels of the moving and the motionless arrays overlap each other only partially (Figure 2), additional points

*varnyu.dora@gmail.com
†szirmay@iit.bme.hu

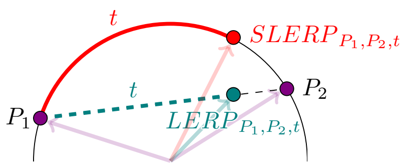


Figure 1: Interpolation of two points using Linear Interpolation (LERP) and Spherical Linear Interpolation (SLERP) with $t = 0.8$

within the volume of a voxel beside its center point should be tracked as well.

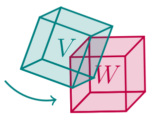


Figure 2: Partially overlapping voxels

The computational complexity of this task can lead to unacceptably long execution times, which necessitates simplifications. One such simplification method is the approximation of the path of each voxel with a polyline, i.e. a sequence of line segments (Figure 3). More precisely, the duration of the movement is divided into several time frames, in each frame assuming both linear and angular velocities to be constant, which results in linear voxel paths. This way, exact voxel positions are calculated only for the beginning and end of the frame and these two points are interpolated during computations. If the time frames are small compared to the dynamics of the object's rotational motion, this simplification can result in significantly less execution times with only a minimal decrease in accuracy.

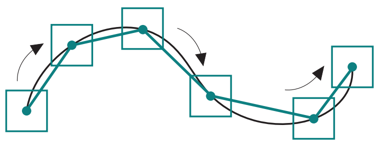


Figure 3: The path of each moving voxel is approximated with a polyline

Assuming that voxels follow a piece-wise linear path, the motion blur algorithm becomes similar to a 3D line rasterization method. Based on our previous study on line drawing algorithms [18], which compared the performance of seven different algorithms known either from computer graphics or tomographic reconstruction, potentially overlapping voxels along a line segment are determined by the *Antialiased Bresenham Algorithm*.

This paper analyzes the solution in terms of speed and accuracy and determine whether the approximation of paths with sequences of line segments proves to be an efficient solution. Our implementation is parallelized with the CUDA platform in order to take advantage of the graphics card's computing power.

2 Related Works

This section covers how motion can be handled during PET reconstruction and analyzes existing motion blur techniques with respect to their applicability in PET.

2.1 Motion Compensation in Tomography

In dynamic tomography, the measurement time is divided into several time frames according to the tracer dynamics. Reconstruction usually iterates two steps. First a maximum-likelihood expectation-maximization (ML-EM) static forward projection and back projection [12] are executed for each frame with frame dependent position and orientation of the patient. Forward projection determines the expected number of detector hits based on our estimate of the radiotracer's spatial distribution, while back projection corrects this estimate based on the ratio of the number of expected and measured hits. Having executed the projections, the parameters of a time function of predefined form are fit in each of the voxels [19]. Since the number of projections is proportional to the number of frames, the frame number has a high impact both on reconstruction time and memory usage. However, rapid movements such as those that occur when the patient has an epileptic seizure require a very high number of frames for an accurate result.

Several solutions were suggested in recent years to tackle this problem, but each one has its limitations.

Gated approaches [5, 17] seek to overcome the problem of motion inside the frame by keeping hits close to discrete time samples, but this results in losing significant information, thus they are less reliable for low statistics or low dose experiments.

A different solution aims to find the list that would have been measured if the subject had remained motionless by re-binning the list of detector hit events [3, 4, 8]. Afterwards, a conventional reconstruction is executed for the modified sinogram. Although such sinogram filtering techniques are simple to implement and efficient, they cannot be applied for effects happening in the detectors. Because of this, they either add noise to the image or result in the measurement data being filtered so that it is not of Poisson distribution anymore and suffer from the missing or lost data problem. This missing data problem comes from the fact that events can be transformed out of the field of view of the tomograph, which would be compensated by events transformed in. However, events outside the field of view are not measured, thereby they cannot be transformed in.

The method we are proposing incorporates continuous motion during the frames by executing motion blur on the voxel array before both forward and back projection. Apart from these two additional computations, all other components of the simulation are the same as in the dynamic reconstruction system developed without motion compensation.

2.2 Motion Blur Techniques

Motion blur is a phenomenon that manifests as a visible trail along the trajectory of a moving object. It is the result of the physical limitations of recording devices as they require a finite exposure time to integrate light. The effect is widely used by photographers to create the perception of motion on still images and also commonly applied in movie and video game rendering where the goal is to make movement on the scene look more natural. While motion blur is automatically incorporated in images recorded using real cameras, for computer-generated images it needs to be explicitly simulated. Several solutions exist with different assumptions, limitations and associated visual artefacts. An overview of the state-of-the-art in motion blur rendering techniques was reported by Navarro *et al.* [9]. They divided existing methods into seven categories: analytical, geometrical substitution, texture clamping, Monte Carlo, post-production, hybrid and physically inspired techniques.

Analytic methods use closed form expressions that can be exactly evaluated to get pixel intensities. Because of the non-linearity of lighting equations, this family of methods often rely on heavy assumptions to construct an analytical description and are as a result limited in their application.

Geometric substitution methods replace moving objects with new geometry built from the original primitives and their evolution over time. A simple example is depicting moving particles of a particle system with antialiased line segments outlining their trajectories [11].

Monte Carlo methods use point sampling and are therefore able to handle phenomena where no analytical or geometrical description can be given. However, the result tends to contain random noise artefacts due to low levels of sampling and the image is not deterministic as sample points are chosen stochastically.

Post-processing methods blur pre-rendered image snapshots using motion information extracted from the objects' animation data or the images themselves. As all operations are executed in image space, the approach fully decouples motion blur from rendering algorithms. This results in a considerable performance gain at the cost of lower quality.

Hybrid methods target specific aspects of the general motion blur problem by combining different models. They are widely accepted due to their efficiency in a broad set of scenarios and the good quality of their results. However, most of these techniques are restricted to objects with polygonal geometry.

Models inspired by the geometrical and optical characteristics of cameras are also becoming increasingly relevant. A number of different approaches aim to simulate the optics of recording devices and the shutter geometry to achieve motion blur [1].

It can be concluded that simulating motion blur is challenging even if only 2D image space results are needed. Many different solutions were proposed to reduce the complexity of the problem by relying on approximations or predefined information about the scene, its objects and lighting conditions. In the case of PET, where the result is expected in the 3D space, the 2D rendering pipeline cannot be utilized and therefore the efficiency of the motion blur algorithm becomes even more relevant.

A strong limitation imposed by the characteristics of tomographic reconstruction is that the geometry of the moving object is not known. This makes it impossible to use a number of motion blur techniques that depend on geometric information about the object or can only operate a specific type of geometrical representations such as polygons, spheres or particles. This opts out analytic, geometric substitution and hybrid methods.

After careful evaluation of the discussed techniques, we concluded that a stochastic point sampling method would be the most appropriate to apply in tomographic reconstruction as it is independent of geometry, can be implemented in a technically simple framework and can be flexibly combined with motion information extracted from the markers. The downside of this solution is the noise artefacts caused by insufficient sampling that can be attacked by increasing the number of sample points if the resources and time limits of the examination enable it.

3 Motion Blur Implementation

Our solution operates on two voxel arrays. The object-space array uses a coordinate system local to the target object, while the tomograph-space array is centered around the unmoving tomograph. Our task is to blur the activities, i.e. the number of radioactive decays occuring in the object-space voxels into the tomograph-space voxels.

Let $x_{V,F}$ denote the activity of voxel V in frame F, and $o_{V,W,F}$ the average overlap factor of object-space voxel V and tomograph-space voxel W ($o_{V,W,F} \in [0,1]$, $\sum_W o_{V,W,F} = 1$). The activity of each tomograph-space voxel W in frame F can then be calculated as

$$x_{W,F} = \sum_V o_{V,W,F} \cdot x_{V,F}. \qquad (1)$$

We use a gather type, i.e. tomograph-space centric approach to determine overlapping voxels in order to avoid the need for atomic operations on the GPU. Each tomograph-space voxel W is mapped to object space via the inverse of the geometric transform describing object motion. Its path is tracked and in each visited object-space voxel, the activity of W is increased by the activity of the visited voxel proportionally to the degree of their overlap.

W is assumed to follow a linear path during the frame. Therefore, efficient 3D line drawing algorithms can be utilized to determine the object-space voxels that W overlaps. Based on previous research, the *Antialiased Bresenham Algorithm* was chosen for the task. The weights calculated by the algorithm equal to the overlap factors $o_{V,W,F}$.

3.1 Antialiased Bresenham Algorithm

The Antialiased Bresenham Algorithm is the antialiased version of Bresenham's line drawing algorithm developed originally for digital plotters [2].

In 3D, the Bresenham Algorithm iterates in unit steps through the coordinate axis direction in which the line changes the most rapidly. At each iteration, it takes the 2D slice of the voxel array orthogonal to the given axis and selects one single voxel from it, the one that is the closest to the geometric line. When the algorithm steps onto the next slice, it chooses whether to increment the two coordinates identifying voxels in the slice. This is decided with the help of two error variables, one in both coordinate axis directions. The variables keep track of the distance between the geometric line and the center of the selected voxel along their designated direction.

A drawback of the Bresenham Algorithm is that it obtains just a single voxel from each slice. When continuous motion is examined, even if the moving voxel was a single point instead of a cuboid, it could cross multiple voxels of the motionless array in each slice (Figure 4). Since the voxels of the two arrays are generally the same size, ignoring partial overlap results in large errors.

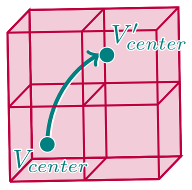


Figure 4: Moving voxel crosses multiple voxels of the motionless array

The Antialiased Bresenham Algorithm complements the Bresenham Algorithm with on-the-fly box filtering. Let's examine first only the special case when the geometric line is on an axial plane. In 2D, box filtering a one-voxel-wide line segment requires the calculation of the area of the line segment's intersection with the voxels concerned.

A one-voxel-wide line segment with a slant between 0° and 45° can overlap up to three voxels in every column (Figure 5). Let the vertical distance of the three overlapped voxels' centers from the geometric line be R, S and T, respectively. Let's assume that $S < T \leq R$. Due to geometric considerations, $S + T = 1$.

The areas of intersection, A_S, A_T and A_R depend not only on the vertical distances, but on the slant of the line segment as well. However, this dependency can be eliminated using the following approximations [16]:

$$A_S \approx 1 - S = T \tag{2}$$

$$A_T \approx 1 - T = S \tag{3}$$

$$A_R \approx 0 \tag{4}$$

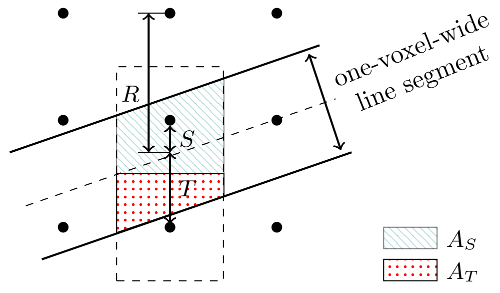


Figure 5: The line segment can overlap up to three voxels in every column if its slant is between 0° and 45°.

These formulae can be evaluated incrementally together with the Bresenham Algorithm's incremental coordinate calculation. In every column, the two selected voxels are weighted by the area of their intersection with the line segment, i.e. A_S and A_T. The farthest voxel is disregarded.

Like the original Bresenham Algorithm, the 3D generalization of the Antialiased Bresenham Algorithm iterates through the direction in which the line changes the most rapidly. In each slice orthogonal to the given axis, a total of four voxels are selected. Namely, the voxel that is the closest to the geometric line, one of its two horizontal neighbors depending on the slope, one of its two vertical neighbors, and finally the diagonal neighbor between them (Figure 6). That is, in 3D, the method executes simultaneously the 2D special case of the algorithm both horizontally and vertically relative to the orientation of the slices, with the addition of a fourth voxel that is adjacent to the closest voxel only through its corner. Each of the four visited voxels gets two independent weights, one horizontal weight ($A_{X,S}$ or $A_{X,T}$) and one vertical weight ($A_{Y,S}$ or $A_{Y,T}$), which are then multiplied to form the final weight of the voxel. This weight expresses the degree of the currently tracked voxel W that overlaps the voxel V to which the weight is assigned, i.e. $o_{V,W,F}$.

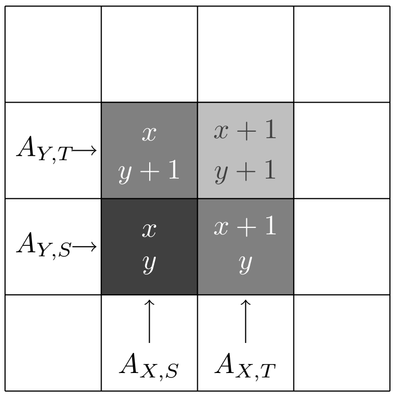


Figure 6: Coordinates (x,y) mark the voxel closest to the geometric line in the slice. Beside that, a horizontal, vertical and a diagonal neighbor get selected in the slice. Each voxel gets an independent horizontal and vertical weight.

3.2 GPU Acceleration

CUDA is a parallel computing platform and programming model that enables general purpose computing on GPU (GPGPU). In our implementation, each parallel thread is assigned to track one tomograph-space voxel during the frame. A new set of threads are deployed for each frame.

On GPU, memory operations are often a bottleneck of calculations. Since fast access to the voxels is a key factor for high performance, the object-space voxel array is stored in the fast but read-only texture memory. Texture memory has built-in trilinear interpolator units that can be used for reducing the number of memory accesses in the Antialiased Bresenham Algorithm. By utilizing the built-in interpolation, the four selected voxels in the array slice can be sampled with only one memory fetch [13], which also includes proper weighting. The memory address is calculated as the interpolation of the four voxel centers with the weights calculated by the Antialiased Bresenham Algorithm (Figure 7). Horizontal and vertical coordinates are interpolated separately.

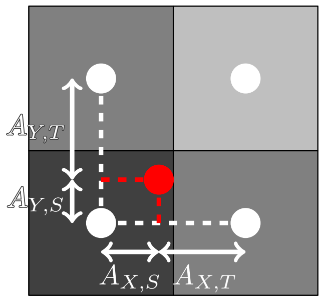


Figure 7: The horizontal and the vertical weights calculated by the Antialiased Bresenham Algorithm are used in the interpolation to reduce the number of memory reads from 4 to 1 in each voxel array slice.

Another advantage of the texture memory is its cache specialized for memory access patterns following spatial locality. This feature can be fully utilized by our implementation as threads read adjacent memory addresses.

4 Examination setup

Measurements were focused around two questions. First, the impact of approximating paths with polylines was examined. At this stage, the Antialiased Bresenham Algorithm was not yet utilized, voxel paths were simply sampled at predefined time steps.

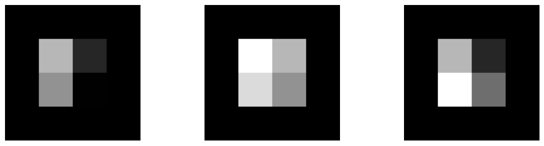
The classical motion blur technique interpolates the geometric transformations describing object motion at the beginning and end of the frame. Assuming voxel paths to be linear during the frame simplifies this computation to interpolate the voxel's starting and ending position so that no transformations are needed at internal steps. This latter technique is similar to ray marching, since it samples along a line segment or ray at predefined distances.

The second question to be examined was whether the Antialiased Bresenham Algorithm proves to be more efficient than the discussed ray marching technique.

As accurate computations are usually not possible due to tight time limits, comparisons were executed using low sampling rate. The number of time steps for both the classical motion blur and the ray marching was set to equal to the number of voxel array slices used by the Antialiased Bresenham Algorithm, i.e. the largest coordinate change of the line segment's direction.

We used three phantoms in different motion scenarios.

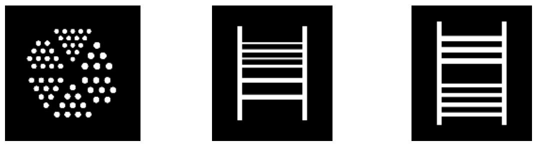
- The **Homogeneity Phantom** (Figure 8) is built of 8 constant activity cubes and consists of $256 \times 256 \times 256$ voxels.



(a) Sagittal ($x = 150$) (b) Coronal ($y = 150$) (c) Transverse ($z = 150$)

Figure 8: Slices of the Homogeneity Phantom

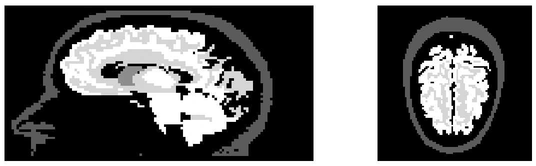
- The **Derenzo Phantom** [6] (Figure 9) is built of rods with varying diameters between two plates and consists of $256 \times 256 \times 256$ voxels.



(a) Sagittal ($x = 100$) (b) Coronal ($y = 103$) (c) Transverse ($z = 137$)

Figure 9: Slices of the Derenzo Phantom

- The **Zubal Brain Phantom** [20] (Figure 10) models a human's brain inside its skull and consists of $128 \times 128 \times 64$ voxels.



(a) Sagittal ($x = 66$) (b) Transverse ($z = 69$)

Figure 10: Slices of the Zubal Brain Phantom

Reference was calculated in each scenario with the classical motion blur technique using a high sampling rate and multiple sample points within the voxels' volume as well.

In each voxel, 200 random points were selected for tracking. It should be noted that the classical technique used in the comparisons tracked only one point in each voxel due to performance considerations.

Methods were compared in terms of accuracy and speed. Accuracy comparison used error rates relative to the reference activity:

$$Error\,\% = \frac{\sum_V (x_V - \hat{x}_V)^2}{\sum_V x_V^2} \cdot 100, \tag{5}$$

where x_V is the reference activity and $\hat{x}_V$ is the estimated activity of voxel V.

Execution times were measured with the CUDA event API offered by NVIDIA for performance metrics. All computations were executed on a NVIDIA GeForce GTX 960M graphics card.

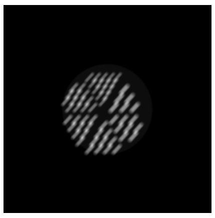
5 Results

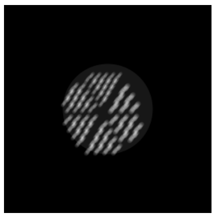
Table 1, Table 2 and Table 3 summarize the results while Figures 11–14 display slices of the blurred arrays to demonstrate the differences between the algorithms.

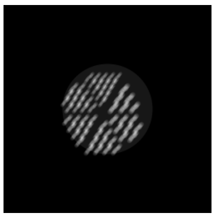
Results show that when the rotational motion of the object is not too significant, approximating paths with polylines successfully competes with the classical motion blur technique. In these scenarios, ray marching achieves the exact same or only a little worse error rates as the classical approach in half or even less time. The greater the motion, the more advantage the ray marching gains over the classical motion blur in speed. This implies that when high execution times cannot be afforded, the suggested approximation proves to be a reasonable solution. However, if the rotational motion of the object is large (scenarios #6 and #8), applying such simplification can lead to unacceptably high errors.

Focusing on the comparison of the ray marching and the Antialiased Bresenham Algorithm, it can be concluded that the Antialiased Bresenham Algorithm performs at least as good as the ray marching in the examined scenarios. When the movement is small, there is no considerable difference between the two methods neither in accuracy nor in execution time. As the motion gets greater, the Antialiased Bresenham Algorithm provides greater accuracy, although at a slightly slower speed. In scenario #3, which measures large linear movement with no rotation, the Antialiased Bresenham Algorithm can achieve up to two orders of magnitude lower error than ray marching depending on the blurred object.

Although the Antialiased Bresenham Algorithm is generally a little slower than ray marching, the difference is not significant compared to the execution overhead of applying transformations at each step as in the case of the classical approach. If the object motion tends to be large, which happens often when neurological disorders are examined, the Antialiased Bresenham Algorithm may prove to be the most favorable in terms of overall performance.


(a) Reference


(b) Classical)


(c) Ray marching

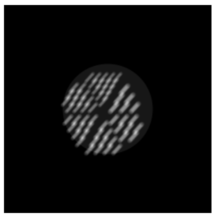
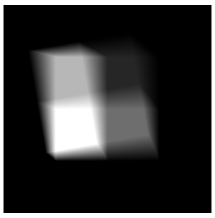
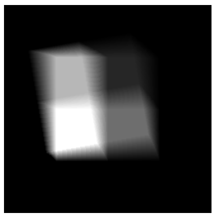
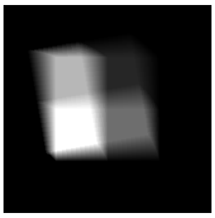

(d) Antialiased Bresenham

Figure 11: Blurred images of the Derenzo Phantom, Scenario 2 ($(12, 10, -11)$ voxel translation).


(a) Reference


(b) Classical)


(c) Ray marching

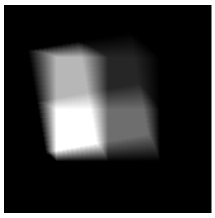

(d) Antialiased Bresenham

Figure 12: Blurred images of the Homogeneity Phantom, Scenario 5 (10° rotation).

6 Summary

This paper addressed the problem of efficient motion blur in 3D space for compensating involuntary movement in dynamic Positron Emission Tomography examinations. To accelerate calculations, voxel paths were approximated with polylines and blurring was executed with the Antialiased Bresenham Line Drawing Algorithm. The suggested method was compared to the classical motion blur technique and ray marching in terms of speed and accuracy. Results show that if the measured object has no significant rotational motion to distort the accuracy of the linear approximation, the Antialiased Bresenham achieves the greatest accuracy at a very good speed.

Scenario	Object motion		Error [%]			Execution time [ms]		
#	**translation**	**rotation**	**CL**	**RM**	**AB**	**CL**	**RM**	**AB**
1	(2, 2, 2)	0	0.0611	0.0611	0.0611	68.5384	33.6528	34.9513
2	(12, 10, -11)	0	0.0355	0.0355	0.0316	168.5775	52.9623	53.7336
3	(-20, 30, 15)	0	0.0211	0.1337	0.0087	4620.0122	1035.0973	984.2599
4	(-20, 30, 0)	0	0.0229	0.0238	0.0110	335.5022	84.4063	108.6518
5	(0, 0, 0)	10	0.0278	0.0474	0.0473	454.1396	100.6399	113.5833
6	(0, 0, 0)	60	0.0093	17.6623	17.6603	2472.8540	278.6437	377.6001
7	(2.5, -3.2, -0.7)	-5	0.0587	0.0734	0.0787	229.5226	59.7012	61.3481
8	(42, 21, 12)	20	0.0146	0.5404	0.5393	1059.9015	182.7321	218.4056

Table 1: Results on the Homogeneity Phantom of the comparison of the classical motion blur (CL), the ray marching (RM) and the Antialiased Bresenham Algorithm (AB). Translation is given in voxels and rotation in degrees.

Scenario	Object motion		Error [%]			Execution time [ms]		
#	**translation**	**rotation**	**CL**	**RM**	**AB**	**CL**	**RM**	**AB**
1	(2, 2, 2)	0	0.7696	0.7696	0.7696	68.5586	34.8277	34.9368
2	(12, 10, -11)	0	0.8664	0.8664	0.5819	167.6756	52.9707	53.7349
3	(-20, 30, 15)	0	0.4557	0.4655	0.1534	327.9224	84.4359	108.6342
4	(-20, 30, 0)	0	0.1105	0.1105	0.0963	338.2086	86.5922	112.1805
5	(0, 0, 0)	10	0.4108	0.5632	0.4818	458.1706	100.6931	113.6382
6	(0, 0, 0)	60	0.1754	50.4148	50.2363	2478.4565	278.5847	376.0340
7	(2.5, -3.2, -0.7)	-5	1.2496	1.4567	1.1130	222.5278	59.4808	61.4398
8	(42, 21, 12)	20	0.2196	10.1651	9.7513	1062.7415	182.6560	217.8306

Table 2: Results on the Derenzo Phantom of the comparison of the classical motion blur (CL), the ray marching (RM) and the Antialiased Bresenham Algorithm (AB). Translation is given in voxels and rotation in degrees.

Scenario	Object motion		Error [%]			Execution time [ms]		
#	**translation**	**rotation**	**CL**	**RM**	**AB**	**CL**	**RM**	**AB**
1	(2, 2, 2)	0	2.2777	2.2777	2.2777	7.8501	4.0458	4.0384
2	(12, 10, -11)	0	1.1337	1.1337	0.3329	19.2592	7.3780	7.2972
3	(-20, 30, 15)	0	0.5874	0.5876	0.0898	39.7277	11.0532	12.7923
4	(-20, 30, 0)	0	0.1854	0.1888	0.0599	43.5365	11.6059	13.9375
5	(0, 0, 0)	10	0.6587	1.0414	0.7998	32.1842	7.9588	9.9130
6	(0, 0, 0)	60	0.2541	27.8996	27.5332	165.0488	19.4201	27.1628
7	(2.5, -3.2, -0.7)	-5	2.5813	2.7822	1.9601	14.9824	6.0723	6.1744
8	(42, 21, 12)	20	0.3330	1.4790	1.0891	99.6831	17.2297	20.1701

Table 3: Results on the Zubal Brain Phantom of the comparison of the classical motion blur (CL), the ray marching (RM) and the Antialiased Bresenham Algorithm (AB). Translation is given in voxels and rotation in degrees.

References

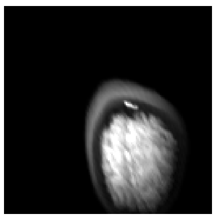
[1] Brian Barsky, Daniel Horn, Stanley Klein, Jeffrey A. Pang, and Meng Yu. Camera models and optical systems used in computer graphics: Part ii, image-based techniques. volume 2669, pages 256–265, 05 2003.

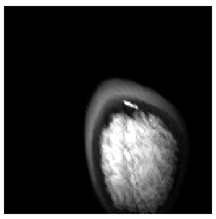
[2] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

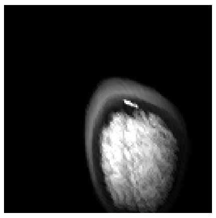
[3] P. Buhler, U. Just, E. Will, J. Kotzerke, and J. van den Hoff. An accurate method for correction of head movement in pet. *IEEE Transactions on Medical Imaging*, 23(9):1176–1185, Sept 2004.

[4] C. Chan, X. Jin, E. K. Fung, M. Naganawa, T. Mulnix, R. E. Carson, and C. Liu. Event-by-event respiratory motion correction for pet with 3-dimensional internal-external motion correlation. In *2012 IEEE Nuclear Science Symposium and Medical Imaging Conference Record (NSS/MIC)*, pages 2117–2122, Oct 2012.

[5] J. Cui, J. Yang, E. Graves, and C. S. Levin. Gpu-enabled pet motion compensation using sparse and low-rank decomposition. In *2012 IEEE Nuclear Sci-*


(a) Reference


(b) Classical)


(c) Ray marching

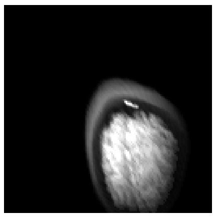
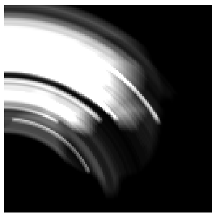
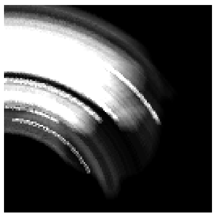
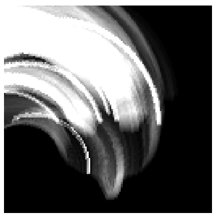

(d) Antialiased Bresenham

Figure 13: Blurred images of the Zubal Brain Phantom, Scenario 3 ($(-20, 30, 15)$ voxel translation, 0° rotation).


(a) Reference


(b) Classical)


(c) Ray marching

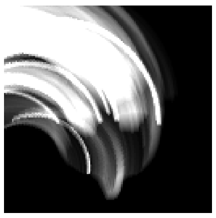

(d) Antialiased Bresenham

Figure 14: Blurred images of the Zubal Brain Phantom, Scenario 6 (60° rotation).

ence Symposium and Medical Imaging Conference Record (NSS/MIC), pages 3367–3370, Oct 2012.

[6] S. E. Derenzo. Mathematical removal of positron range blurring in high resolution tomography. *IEEE Trans. Nucl. Sci.*, 33:546–549, 1986.

[7] J. Jiao, A. Bousse, K. Thielemans, N. Burgos, P. S. J. Weston, J. M. Schott, D. Atkinson, S. R. Arridge, B. F. Hutton, P. Markiewicz, and S. Ourselin. Direct parametric reconstruction with joint motion estimation/correction for dynamic brain pet data. *IEEE Transactions on Medical Imaging*, 36(1):203–213, Jan 2017.

[8] M. Menke, M. S. Atkins, and K. R. Buckley. Compensation methods for head motion detected during pet imaging. *IEEE Transactions on Nuclear Science*, 43(1):310–317, Feb 1996.

[9] Fernando Navarro, Francisco Serón, and Diego Gutierrez. Motion blur rendering: State of the art. *Comput. Graph. Forum*, 30:3–26, 03 2011.

[10] Andrew J Reader and Jeroen Verhaeghe. 4d image reconstruction for emission tomography. *Physics in Medicine and Biology*, 59(22):R371, 2014.

[11] W.T. Reeves. Particle system-a technique modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 17:359–376, 01 1983.

[12] L. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Trans. Med. Imaging*, 1:113–122, 1982.

[13] C. Sigg and M. Hadwiger. Fast third-order texture filtering. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, pages 313–329. Matt Pharr(ed.), Addison-Wesley, 2005.

[14] László Szirmay-Kalos and Ágota Kacsó. Regularizing direct parametric reconstruction for dynamic pet with the method of sieves. In *Molecular Imaging Conference*, MIC '16, pages M16D–1, 2016.

[15] László Szirmay-Kalos, Ágota Kacsó, Milán Magdics, and Balázs Tóth. Dynamic pet reconstruction on the gpu. *Periodica Polytechnica Electrical Engineering and Computer Science*, 62(4):134–143, 2018.

[16] L. Szirmay-Kalos (editor). *Theory of Three Dimensional Computer Graphics*. Akadémia Kiadó, Budapest, 1995. http://www.iit.bme.hu/˜szirmay.

[17] M. Toussaint, J. P. Dussault, and R. Lecomte. Revisiting motion compensation models in pet image reconstruction. In *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pages 90–94, April 2016.

[18] Dóra Varnyú and László Szirmay-Kalos. Thick line integration with filtered sampling. In *KEPAF 2019, Debrecen*, 2019.

[19] Guobao Wang and Jinyi Qi. An optimization transfer algorithm for nonlinear parametric image reconstruction from dynamic pet data. *IEEE Trans Med Imaging*, 31(10):1977–1988, 2012.

[20] I. George Zubal, Charles R. Harrell, Eileen O. Smith, Zachary Rattner, Gene Gindi, and Paul B. Hoffer. Computerized three-dimensional segmented human anatomy. *Medical Physics*, 21(2):299–302, 1994.