2010 Fourth Pacific-Rim Symposium on Image and Video Technology

Real-Time GPU-Aided Lung Tumor Tracking

Yin Yang, Zichun Zhong
Guodong Rong, Xiaohu Guo
University of Texas at Dallas
Richardson, Texas, USA
Email: {yxy061100|zxz082020|gxr071100|xguo}
@utdallas.edu

Jing Wang, Timothy Solberg
Weihua Mao
UT Southwestern Medical Center
Dallas, Texas, USA
Email: {Jing.Wang|Timothy.Solberg|Weihua.Mao}
@UTSouthwestern.edu

***Abstract*—A real time solution of tracking daily lung tumor motion is proposed in this paper in order to achieve an accurate dose delivery for radiation therapy as recently developed cone-beam computed tomography (CBCT) technique is not able to catch tumor motions due to the patient's respiration. We develop a novel GPU-based fast digitally reconstructed radiograph (DRR) generation algorithm which enables an instant DRR computation and rendering from patients' radiation therapy treatment planning CTs. In the meantime, classic image correlation algorithm is extended as the main method to locate tumors in X-ray 2D projections, the raw data of CBCT scans. With the GPU-aided implementation, this algorithm is capable of capturing movement of lung tumors as fast as the CBCT image acquisition in real time, which greatly facilitates the radioactive treatment.**

***Keywords*-GPU; image correlation; DRR; radiation; therapy**

I. Introduction

Radiation therapy is an essential modality in the treatments of cancer, and used in over 50% of the patients with cancer, either alone or in combination with other forms of treatments. Current radiation therapy delivery techniques and dose calculations are based on a treatment planning *computed tomography* (CT) scan acquired before a course of therapy begins. Modern radiation therapy techniques, such as *3D conformal radiation therapy* (3DCRT), *intensity-modulated radiation therapy* (IMRT), and *stereotactic body radiation therapy* (SBRT) can conform radiation doses closely to a tumor volume while sparing surrounding sensitive structures, but high conformality is achieved only if relevant anatomy (tumor and dose-limiting structures) can be perfectly replicated relative to the treatment beam on a daily basis. This is particularly challenging for treating tumors moving due to patients' respiration. A tight planning margin could be applied on 3D *planning target volume* (PTV) based on the respiratory pattern so that normal tissues are spared from toxic high dose radiation. An on-board *cone-beam CT* (CBCT) imaging technique has been commercialized in the past decade. It is mounted on the treatment linear accelerator and capable of providing high quality volumetric imaging prior to radiation therapy. A typical on-board CBCT scan takes from 1 to 2 minutes and acquires about 660 2D x-ray projections over a gantry rotation of 360 degrees. CBCT technique provides high contrast imaging for soft tissue however, it is difficult to catch respiratory motions. It is essential to evaluate respiratory motion on a daily basis to setup patient accurately for treatment and confirm breathing patterns used for treatment planning. Motivated by the challenges in terms of both accuracy and efficiency in such clinical requirement, we extend the well-known image correlation algorithm to eliminate the search miss while powerful parallel computation ability of GPU is fully utilized boosting the computation performance. The combination of these result an accurate, real-time solution for tracking daily lung tumor motion that could be widely adopted in clinics, hospitals and medical centers with low cost.

II. Related Work

As *digital reconstructed/rendererd radiographs* (DRRs) generation is actually a subset of volume rendering [1] which enables rendering of 2D projection images out of 3D discretely sampled data, most volume rendering techniques can be directly transplanted to DRR generation such as raycasting, splatting and shear warp. The raycasting [2], [3] is the most intuitive as each X-ray is simulated with line integral while splatting [4], [5] normally sets a threshold such that the voexls under this threshold are considered as none-contribution and removed, thus effectively reduces rendering load. However, due to the absence of thresholded voxels, aliasing problem arises. This requires additional processing suppressing such aliasing. The shear warping [6] is a little bit similar to the method used in this paper where a set of parallel slices are used to represent the volume data. In many DRR-generation-involved real applications (including our work), the generation speed is sensitive and the fast DRR generation rate is always preferred. The GPU-based DRR generation algorithms [3], [7], [8] have been proposed considering this procedure is highly parallelable. Similar to [6], [9], [10], texture slices (quads set) sampled from a 3D texture [11] are used in this work, which makes the rendering much more efficient than voxel-based rendering. The CT scanning is simulated with OpenGL built-in perspective projection while additional amendment is performed after the projection with Cg [12] fragment program. The resulting DRR is very close to DRR generated using raycasting (figure 8) and instant DRR generation rate ($0.016\ sec$ per DRR) is made possible.

Certain similarity measurement is needed based on the generated DRR. A nice survey can be found in [13] where several classic registration algorithms are compared within the context of medical imaging. The *normalized cross correlation* (NCC) [14] becomes our prime choice. Other methods may not be applicable in this work: the tumor signals suffer a very strong interference from other featured structures' signals. Different image modalities also make this mission more challenging. The original correlation is further extended by double correlation with different resolutions (similar to [15]).

The time-consuming cross correlation search becomes the performance bottle neck in this work. Several hardware accelerating techniques have been proposed using both CPU [16]and GPU [17]. GPU shows a greater potential under such purposes. Though CPU based method also claims [16] a remarkable acceleration, it essentially still uses the idea of parallelization requiring a dedicated multiprocessor hardware architecture. GPU with its indigenous

978-0-7695-4285-0/10 $26.00 © 2010 IEEE
DOI 10.1109/PSIVT.2010.89

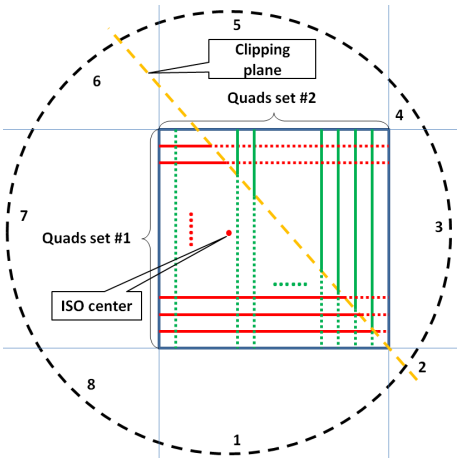


Figure 1. The setup of the two directional DRR generation.

multi-core is, from the parallelization point of view more suitable for this task: the NVIDIA GeForce 8800GTX video card (the graphic device used in this work) is equipped with 128 stream processors clocked at 1.35 GHz. Cg [12] instead of CUDA [17] or other GPGPU [18] languages is chosen in this step again. We implement three acceleration strategies. The one with OpenGL built-in 2D convolution beats other two methods. Such results also suggest that more pipeline-oriented method/language, when handling the pipeline-oriented tasks is likely to benefit more from the graphical parallelization.

III. DRR Generation

Essentially, this work is to perform a rigid registration between 3D (PTV) and 2D (CBCT) medical images. This, normally calls for the fast dimension reduction of higher dimensional data (PTV in this case). The resulting 2D images (DRRs) are then subjected to certain similarity measurement in order to estimate tumor position in the CBCT image.

The production of DRR is a highly computation-involved and time-consuming procedure. Theoretically, the calculation of line integral along each simulated X-ray beam from the source to a certain position on the expecting DRR should be performed [19]. Unfortunately, the direct implementation of this method is not feasible under most circumstances as 3D/2D registration applications normally require repeated DRR productions for the registration adjustment. In our case, though only one DRR is necessary because the simulated X-ray projection follows the same geometry configuration as the on-board CBCT scan does, yet a big number of DRR generations is still needed to match the actual CBCT series (about 660 images) with different gantry angles.

A. Two Directional DRR Generation

During CT scanning, each pixel on the resulting image is assigned a numerical value (CT number), which is the average of all the attenuation values/density contained within the corresponding voxels. This number is compared to the attenuation value of water and displayed on a scale of arbitrary units named *Hounsfield units* (HU). We take use of graphical perspective projection plus blending to simulate the X-ray cone-beam volume and the summation of the attenuation values respectively. The PTV used as DRR source has around the size of $512 \times 512 \times 200$ $voxels$. The distance between X-ray source and the resulting image plane is $1,536$ mm and the radius of the rotation centered at the ISO center which is manually set close to labeled lung tumor margin is $1,000$ mm. The rotation of X-ray source is performed within a plane parallel to the top and bottom of the PTV. The given CT volume is passed to GPU video memory as a 3D texture. Quads are used as the elementary rendering primitives which could be much less rendering intensive than voxel-oriented methods [5]. Each quad corresponds to one slice of voxels with the texture coordinates computed based on its spatial position. The blending function, afterwards sums up all the voxel slices and generates the preliminary DRR image. This configuration could be problematic when the camera which simulates the X-ray source travels on the orbit (the dash circle in figure 1): when the camera rotates to the position where the cone-beam volume is somehow parallel to the quad-set, we are not able to get the valid DRR images any longer.

In order to solve this problem, we can define the quad-set dynamically based on the camera position making the quad-set always perpendicular to the CBCT volume as most volume rendering techniques do [9], [10]. However because the voxels are intrinsically arranged in a regular fashion. When the quad-set fits the original arrangement of the voxels, we do not miss or try to guess any information; when the quad-set diverges from the original voxel arrangement, we have to do certain interpolation to estimate volume intensity at some positions. Such interpolation though could be accomplished by OpenGL built-in routines, still effects the accuracy of the resulting DRR. In addition, the distance between every successive two quads must be handled very carefully so that voxels in the PTV are cut by one and only one quad which is not possible in some extreme angles. Alternatively, we propose a new method named as *two directional rendering* to solve this problem. The idea is simple, where two sets of quads are to be drawn instead of merely one set (figure 1). As shown in the figure, the original PTV defines 8 segments of the camera's traveling path. In region 1, 3, 5 and 7 where the camera is mostly orthogonal to one dimension of PTV while aligned to the other dimension (the third dimension is eliminated as the orbit plane is perpendicular to one dimension of the PTV), only one set of quads is to be drawn. More specifically, in regions 1 and 5 the quad-set $\#1$ is to be drawn while in regions 3 and 7, the quad-set $\#2$ is to be drawn. In other regions, both quad-sets are to be drawn. An additional clipping plane is applied to remove unnecessary voxel information (the yellow dash line in the figure 1) which can be defined by one of the four edges (depending on which region camera is sitting in) perpendicular to the orbit plane plus the camera position. As shown in figure 1 assuming camera is in region 2, the cone-beam volume is partitioned into two parts. For the part on the left hand side of the plane, only quad-set $\#1$ is to be rendered (solid red lines). On the contrary, for the part on the other side of the plane, we only draw quad-set $\#2$ (solid green lines). Intuitively, the portions that are most orthogonal to the cone-beam volume of each set are rendered and the rest (dotted lines) are clipped. In this method, all the quad-sets follow the original voxel arrangement and the intervals between quads that are freed of any texture interpolations are just the real size of the voxel. It is also noteworthy that the extension of this method to *three directional rendering* is straightforward which enables the arbitrary camera positioning in 3D space.

B. Integral Adjustment

The OpenGL blending function sums up all the voxel intensities along the simulated X-ray beams, however the lengths of the beams have not yet been incorporated. Thus an amendment is required for the preliminary generated DRR images. We assume that the beam

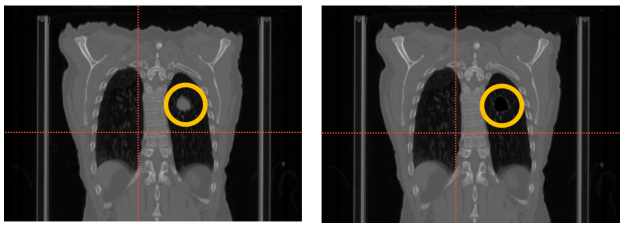


Figure 2. One slice of the original PTV (left) and tumor-removed PTV (right).

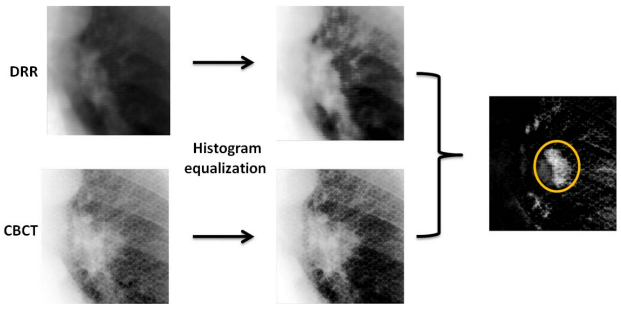


Figure 3. The subtraction-based tumor intensification.

always travels within one voxel between two successive quads[1]. The beams are actually subdivided evenly by each quad based on the assumption, and the corresponding integrals can be simplified as:

$$\sum l \times i(t) = l \times \sum i(t), \tag{1}$$

where, $0 \leq t \leq 1$ is the 1D coordinate of the parameterized beam; l is the length of the line segment between two successive quads and $i(t)$ is the volume intensity at t. This implies only a constant is needed for each beam. Fortunately, the constant of each beam can be pre-computed which is just the length of line segment from the camera to the pixel on the final DRR image. The DRR without amendment is rendered to a *frame buffer object* (FBO) and A Cg fragment program is then used for this pixel-wise operation carried out simultaneously at each fragment/pixel.

IV. Tumor Tracking

After the simulated CT for certain gantry angle is produced, the rigid registration is then performed between the pair of 2D images: one DRR image and one CBCT image. The classic NCC is extended and used in the similarity measurement.

A. Normalized Cross Correlation Search

The NCC is one of the most widely used methods for the similarity measurement. In short, it calculates a scalar named *correlation coefficient* (CC) such that:

$$CC = \frac{\sum_s \sum_t [f(s,t) - \bar{f}][w(x+s, y+t) - \bar{w}]}{\sqrt{\sum_s \sum_t [f(s,t) - \bar{f}]^2 \sum_s \sum_t [w(x+s, y+t) - \bar{w}]^2}}, \tag{2}$$

where, f and w are the two images under correlation. f is the image pattern under search and w is the image where this pattern is to be located. Obviously, f should have smaller size than w does. We, here name f and w as template image and target image respectively. For each valid candidate position (x, y) in w, a correlation window with the same size of f is defined. The summation indices s, t traverse all the pixels in f and the correlation window in order to compute the corresponding CC at this position. When CC hits the maximum value, 1, it indicates the best similarity based on the pixel *grey level* (GL) has been reached. And the corresponding correlation window is considered as the located pattern in the target image. The procedure that loops over all the candidate positions in target image and keeps tracking the maximum CC value is called "cross correlation search" or "cross search" in short. Prior to the planning stage, experienced and skilled doctors have labeled a tight margin of lung tumor based on the regular CT scanning of the patient as mentioned. Such information is preserved when PTV is reconstructed: additional bit is added to the conventional voxel data indicating whether this voxel is within the margin or not. So in DRR, a 2D margin of the tumor is also generated which, ideally should serve as template image during cross search. Unfortunately, the direct cross search based on labeled lung tumor and CBCT image is not applicable due to the low signal intensity of the tumor portion comparing with other internal structures (i.e. bony structures) and the cross search gets constantly missed. Consequently, some other processes are necessary in order to emphasize the tumor signals.

[1]This could not be the real case. However our experiments show that such approximation does not effect the accuracy of upcoming cross search

B. First Cross Search

The intensities of all the labeled tumor voxels in PTV are cleared and the DRRs are generated based on the tumor-free CT volume (figure 2). On the tumor-free DRRs, a region is selected based on the image contents such that the following two conditions must be satisfied: 1) the bounding box of tumor margin is fully contained and 2) extra strong GL pattern such as bony structures are desired. This region is called *matching window* which serves as the template image for the first cross search. The size of matching window is normally set as two to four times of the tumor bounding box. However, user can change this parameter for the various tumor positions in order to achieve the best search result. The first condition effectively reduces the number of candidate positions on the target image and the second condition guarantees a reasonable result from the cross search and in our experiments, the first cross search can always return an accurate result due to the included strong features. After the search is finished, we have a subregion in the target image (CBCT) with the same size of matching window. Most importantly, it now can be safely assumed that the tumor must be located within this subregion.

C. Subtraction-Based Tumor Intensification

With the newly-defined target image which is the subregion on CBCT that corresponds to the matching window, we can roughly have some ideas of the tumor position. However, due to the condition 2, the new target image is still suffering none-tumor signals that is in orders-of-magnitude stronger than original tumor signals. Based on the assumption that the new target image and the matching window in DRR contain almost the same materials except that the tumor portion is removed in matching window, an image substraction is operated which removes most strong GL features within the target image. Before the subtraction, a histogram equalization is performed at both images to minimize the modality difference between images. Figure 3 shows a detailed step-by-step result of this procedure.

D. Second Cross Search

Naturally, the resulting image with tumor intensified after subtraction is to serve as the new target image. Another DRR is created from the volume where only tumor-labeled voxels are kept and all other voxels are cleared which actually is the DRR of the tumor margin. A rectangle tumor bounding box can be extracted out of this DRR which is the template image in the second cross search. Because most interfering information has been removed after the subtraction, this step operates smoothly and fast (only searching within the matching window rather than the entire CBCT image). The second cross search is a low resolution search as compared with the first search which runs on the whole CBCT image.

E. Constrained Cross Search

It is well-known that NCC, though is easy to implement and in most cases gives reasonable results is very computation-intensive. In our implementation using Matlab, the time needed for searching on one complete CBCT series which contains about 660 2D images of 512×512 $pixels$ is about 20 hours. However, running the cross search over the entire target image region (in the first cross search) could be unnecessary based on the observation that the CBCT images are generated continuously. The time latency between two CBCT images is about 0.2 sec and within such limited time, the tumor is not likely to move far away from its previous position. Thus it is greatly wasteful to compute CC for all candidate positions in target image. Following this consideration, the search region is constrained to a small area close to its previous search result. With this constraining strategy, we can eliminate over 95% unnecessary computation. The time, in consequence is reduced to about 35 min executed with Matlab implementation.

V. GPU-Based Search Acceleration

In order to make this work a real-time application for the practical clinical usage, some efforts are still needed in terms of the time performance. Recall that in equation 2, the computation of CC at different candidate positions are totally independent of each other indicating that a parallelization-based acceleration is promising. GPU naturally becomes our prime choice. Both DRR generation and cross search are of huge potential in terms of parallelization and the whole procedure can get fitted into the conventional graphics pipeline seamlessly. We use Cg [12] as the main GPU language in our implementation instead of other GPGPU language (CUDA for example) because generally speaking, a pipeline-oriented application gets more accelerated using more pipeline-oriented GPU language. The Cg fragment program is executed on the fragments concurrently and each fragment corresponds to a pixel in the image. We re-arrange the NCC equation as:

$$c = \frac{\sum\sum(fw - \bar{f}w - f\bar{w} + \bar{f}\bar{w})}{A\sqrt{\sum\sum(w^2 - 2w\bar{w} + \bar{w}^2)}}, \quad (3)$$

where, $A = \sqrt{\sum\sum(f - \bar{f})^2}$ can be considered as a pre-computed constant as template image f is fixed. The summation is over s and t which are the two dimensions of template image as in equation 2. In order to compute CC, we have to do massive video memory accesses which include double $s \times t$ accesses for term fw and another several $s \times t$ accesses for $\bar{w}$ related terms for a single candidate position. A typical tumor bounding box with the size of 60×60 $pixels$ results in $(512 - 60) \times (512 - 60) \times 60 \times 60$ memory accesses. This is a very large number impeding the real-time processing rate even with GPU acceleration. Some pre-computation is necessary to reduce the number of memory access.

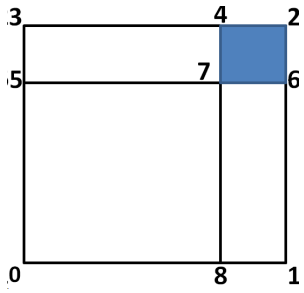


Figure 4. The summed area table.

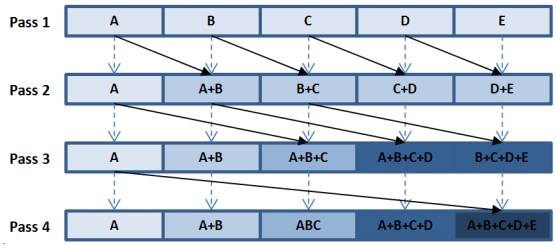


Figure 5. The horizontal phase of SAT generation.

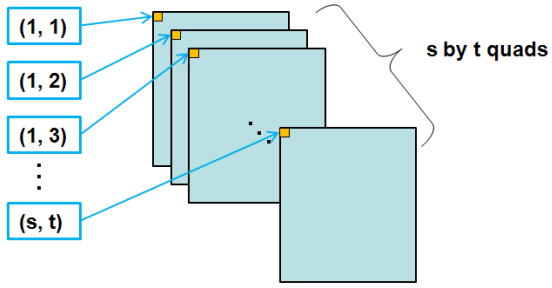


Figure 6. The blending-based implementation: $s \times t$ quads with shifts are drawn at different depth. Blending is used to compute the summation.

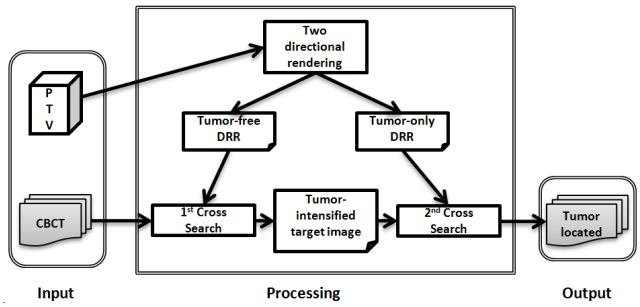


Figure 7. The general data flow and computational procedures.

(a) (b)

Figure 8. Generated DRR images using (a) our method and (b) raycasting.

Method	Time performance
Linear raycasting on CPU	16.1 sec/DRR
Two directional rendering	0.016 sec/DRR
Cross search on CPU (no constraint)	20.4 $hour$/660 CBCT
Cross search on CPU (with constraint)	35.8 min/660 CBCT
GPU-aided cross search (with blending)	4.6 min/660 CBCT
GPU-aided cross search (no blending)	1.6 min/660 CBCT
GPU-added cross search (with 2D convolution)	1.2 min/660 CBCT

Table I
TIME PERFORMANCE.

The auxiliary data structure called *summed area table* (SAT) is built. This data structure was first introduced for fast computation of graphical texture mapping [20]. In SAT, each pixel stores the sum of a rectangle area top-right-cornered by this pixel. As a result, a constant number of readings are sufficient to get the sum of any sub-rectangle area of the image. As shown in figure 4, in order to get the summed area of the shadowed rectangle A_{7624}, only the value at four corners are read such that $SAT(2) = A_{0123}, SAT(4) = A_{0843}, SAT(7) = A_{0875}$ and $SAT(6) = A_{0165}$. $A_{7624} = A_{0123} + A_{0875} - A_{0843} - A_{0165} = SAT(2) + SAT(7) - SAT(4) - SAT(6)$. We follow the famous *recursive doubling* [21] to generate the SAT for the target image w and w^2. The generation consists of one horizontal phase and one vertical phase that must be carried out sequentially. Figure 5 illustrates the horizontal phase for one pixel strip. Total number of $\log N$ (N denotes the strip width) passes are to be executed. As shown in the figure, in pass k, every element is added by the element that is 2^k ahead. The out-bounding access returns 0. Vertical phase begins after the horizontal phase with the similar fashion. The fragment programs are running simultaneously at all the horizontal/vertical pixel strips. After SAT of w and w^2 are computed. $\bar{w}$, $\sum\sum w$ and $\sum\sum w^2$ become instant-accessible quantities in equation 3 which can be further written as:

$$c = \frac{\sum\sum fw + B}{AC}, \tag{4}$$

where, A, B and C are all terms that can be evaluated in constant time. Thus, the most time-consuming term is $\sum\sum fw$ which are to be parallelized using GPU. Three parallelization strategies are used in our implementation. In the first implementation, we take use of OpenGL blending function to do the summation over two dimensions of template image f. More specifically, $s \times t$ quads are drawn on the screen with different depth. Each quad is of the same size of target image w while takes a index shift corresponding s and t (as shown in figure 6). The second approach is running a fragment program on each candidate pixel in target image and compute CC using equation 4 and the third one is to use OpenGL built-in 2D convolution routine to compute fw. The use of OpenGL built-in routine gives the best computation speed while the blending based method is the slowest one. This result also coincide with the-more-pipeline-oriented-the-faster assumption. All GPU-aided methods greatly accelerate the computation of CC as compared to the CPU-implemented linear computation.

VI. EXPERIMENT AND RESULTS

All the tests are run on a Windows 7 PC with Intel Q6600 CPU, 4GB DDR2 RAM, and NVIDIA 8800GTX video card (768M GDDR3 video memory, 128 stream processors). Cg is used as major GPU shader language. The whole framework has two main components: DRR production and cross search. Both of them are run mainly on GPU. Figure 7 shows a general data flow and step-by-step computational procedure. The CT volume with labeled tumor margin and 660 CBCT images are the general inputs. Two types of DRRs are generated: the ones without tumor along as the CBCT images are the inputs of the first cross search; the ones with only tumor plus the tumor-intensified target images after subtraction are the inputs of the second cross search. Finally, the tumor position is tracked. The accompanied demo video is available at:http://www.utdallas.edu/~yxy061100/demo. For the purpose of comparison, the common raycasting method is also implemented using C++. Our two directional rendering method gives very similar results of DRR images (as shown in figure 8). However, the time needed for one DRR image generation using our method is about $0.01632\ sec$ which is almost $1,000$ times faster comparing with linear raycasting executed on CPU ($16.121\ sec$). 4 anonymous patients' on-board CBCT images with lung tumor are analyzed using our method. Each patient has 6-8 sets of CBCT series. Each series contains about 660 images of size $512 \times 512\ pixels$. We can locate tumor accurately on over 98% CBCT images (figure 9) which is a very high rate in oncology. The original cross search without any constraints takes over 20 hours to locate one CBCT series with Matlab implementation. This number decreases to $1.2\ min$ with GPU based implementation. The detailed benchmark can be found in table I. Because it takes about $2\ min$ to have one series of CBCT ready during treatment, we are able to locate tumor position as soon as the CBCT scanning finishes which makes this method run at a real-time processing rate. It can greatly facilitate the radioactive treatment.

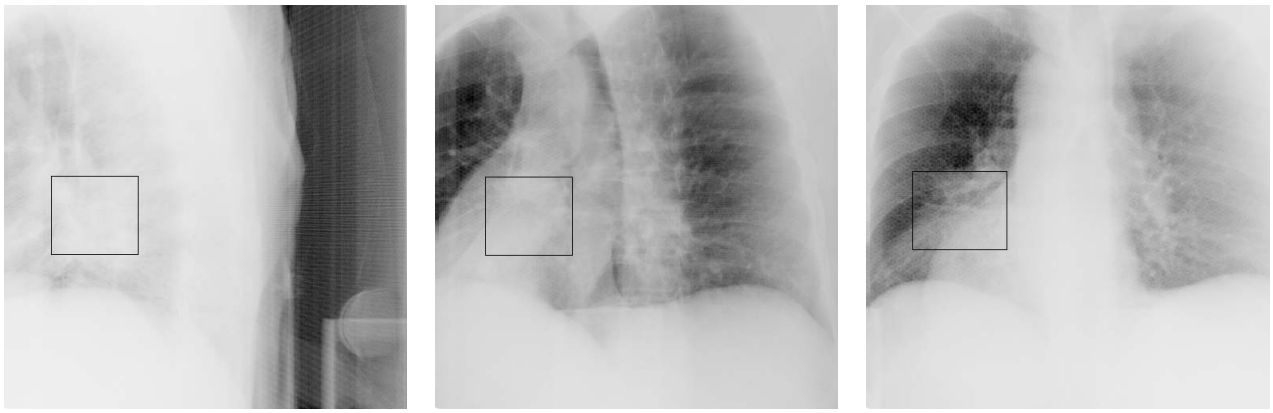
VII. CONCLUSION

We present a GPU-based real-time solution to track daily lung tumor movements. The classic NCC algorithm is extended with ampliation of tumor signal which makes the cross search much more accurate. In DRR generation, a new rendering strategy is used with two orthogonal texture slices avoiding texture interpolation. The Cg fragment program is used to adjust final CT numbers. In NCC computation, the summed area table is employed which saves at least half memory readings. We compare three GPU-based implementation of cross search and the OpenGL 2D convolution is the one with the fastest computation rate. The constrained searching with parallelized GPU-aided implementation gains enormous performance increase. This work makes the real-time lung tumor tracking possible, thus is of substantial potential for real clinical applications.

REFERENCES

[1] R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume rendering," in *SIGGRAPH '88*, 1988, pp. 65–74.

[2] P. M. Joseph, "An improved algorithm for reprojecting rays through pixel images." *IEEE Trans. Med Imaging*, vol. 1, no. 3, pp. 192–6, 1982.



(a) (b) (c)

Figure 9. Cross search results for different gantry angles with tumor highlighted.

[3] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, "The volumepro real-time ray-casting system," in *SIGGRAPH '99*, 1999, pp. 251–260.

[4] L. Westover, "Footprint evaluation for volume rendering," in *SIGGRAPH '90*, 1990, pp. 367–376.

[5] W. Birkfellner, R. Seemann, M. Figl, J. Hummel, C. Ede, P. Homolka, X. Yang, P. Niederer, and H. Bergmann, "Wobbled splatting - a fast perspective volume rendering method for simulation of x-ray images from ct," *Phys. in Med. and Bio.*, vol. 50, no. 9, p. N73, 2005.

[6] P. Lacroute and M. Levoy, "Fast volume rendering using a shear-warp factorization of the viewing transformation," in *SIGGRAPH '94*, 1994, pp. 451–458.

[7] J. Spoerk, H. Bergmann, W. Birkfellner, F. Wanschitz, and S. Dong, "Fast DRR splat rendering using common consumer graphics hardware," *Med. Phys.*, vol. 34, pp. 4302–4308, 2007.

[8] D. Ruijters, B. M. ter Haar-Romeny, and P. Suetens, "GPU-accelerated digitally reconstructed radiographs," in *BioMED '08: Proceedings of the Sixth IASTED International Conference on Biomedical Engineering*, 2008, pp. 431–435.

[9] O. Wilson, A. VanGelder, and J. Wilhelms, "Direct volume rendering via 3D textures," Tech. Rep., 1994.

[10] J. Kruger and R. Westermann, "Acceleration techniques for GPU-based volume rendering," in *Proceedings of the 14th IEEE Vis 2003 (VIS'03)*, 2003, p. 38.

[11] K. Akeley, "Reality engine graphics," in *SIGGRAPH '93*, 1993, pp. 109–116.

[12] W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard, "Cg: a system for programming graphics hardware in a C-like language," in *SIGGRAPH '03*, 2003, pp. 896–907.

[13] G. Penney, J. Weese, J. Little, P. Desmedt, D. Hill, and D. Hawkes, "A comparison of similarity measures for use in 2-d-3-d medical image registration," *IEEE Trans. Med. Imaging*, vol. 17, no. 4, pp. 586–595, 1998.

[14] C. Wilson and J. Theriot, "A correlation-based approach to calculate rotation and translation of moving cells," *IEEE Trans. Image Processing*, vol. 15, no. 7, pp. 1939 –1951, july 2006.

[15] S.-D. Wei and S.-H. Lai, "Fast template matching based on normalized cross correlation with adaptive multilevel winner update," *IEEE Trans. Image Processing*, vol. 17, no. 11, pp. 2227–2235, 2008.

[16] M. Cavadini, M. Wosnitza, and G. Tröster, "Multiprocessor system for high-resolution image correlation in real time," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 9, no. 3, pp. 439–449, 2001.

[17] P. J. Lu, H. Oki, C. A. Frey, G. E. Chamitoff, L. Chiao, E. M. Fincke, C. M. Foale, S. H. Magnus, W. S. M. Jr., D. M. Tani, P. A. Whitson, J. N. Williams, W. V. Meyer, R. J. Sicker, B. J. Au, M. Christiansen, A. B. Schofield, and D. A. Weitz, "Orders-of-magnitude performance increases in GPU-accelerated correlation of images from the international space station," *Journal of Real-Time Image Processing*, 2009.

[18] W. Liu, B. Schmidt, and W. Müller-Wittig, "Performance analysis of general-purpose computation on commodity graphics hardware: a case study using bioinformatics," *J. VLSI Signal Process. Syst.*, vol. 48, no. 3, pp. 209–221, 2007.

[19] M. Levoy, "Efficient ray tracing of volume data," *ACM Trans. Graph.*, vol. 9, no. 3, pp. 245–261, 1990.

[20] F. C. Crow, "Summed-area tables for texture mapping," in *SIGGRAPH '84*, 1984, pp. 207–212.

[21] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra, "Fast summed-area table generation and its applications," *Computer Graphics Forum*, vol. 24, pp. 547–555, 2005.