Published in final edited form as:

Phys Med Biol. 2011 March 7; 56(5): 1431–1441. doi:10.1088/0031-9155/56/5/014.

GPU-based fast gamma index calculation

Xuejun Gu, Xun Jia, and Steve B. Jiang

Center for Advanced Radiotherapy Technologies and Department of Radiation Oncology,
University of California San Diego, La Jolla, CA 92037-0843

Steve B. Jiang: sbjiang@ucsd.edu

Abstract

The γ -index dose comparison tool has been widely used to compare dose distributions in cancer radiotherapy. The accurate calculation of γ -index requires an exhaustive search of the closest Euclidean distance in the high-resolution dose-distance space. This is a computational intensive task when dealing with 3D dose distributions. In this work, we combine a geometric method (Ju et al. *Med Phys* **35** 879-87, 2008) with a radial pre-sorting technique (Wendling et al. *Med Phys* **34** 1647-54, 2007), and implement them on computer graphics processing units (GPUs). The developed GPU-based γ -index computational tool is evaluated on eight pairs of IMRT dose distributions. The γ -index calculations can be finished within a few seconds for all 3D testing cases on one single NVIDIA Tesla C1060 card, achieving 45-75 $\times$ speedup compared to CPU computations conducted on an Intel Xeon 2.27 GHz processor. We further investigated the effect of various factors on both CPU and GPU computation time. The strategy of pre-sorting voxels based on their dose difference values speeds up the GPU calculation by about 2.7-5.5 times. For n -dimensional dose distributions, γ -index calculation time on CPU is proportional to the summation of γ^n over all voxels, while that on GPU is affected by γ^n distributions and is approximately proportional to the γ^n summation over all voxels. We found increasing the resolution of dose distributions leads to quadratic increase of computation time on CPU, while less-than-quadratic increase on GPU. The values of dose difference (DD) and distance-to-agreement (DTA) criteria also have their impacts on γ -index calculation time.

1. Introduction

The γ -index concept introduced by Low *et al.* (Low *et al.*, 1998) has been widely used to compare two dose distributions in cancer radiotherapy. The original γ -index calculation algorithm of Low *et al.* (Low *et al.*, 1998) has been improved for better accuracy and/or efficiency (Depuydt *et al.*, 2002; Bakai *et al.*, 2003; Stock *et al.*, 2005; Jiang *et al.*, 2006; Spezi and Lewis, 2006). However, since these modified algorithms still involve computational intensive tasks such as interpolation of dose grid and exhaustive search, it is very time-consuming (*e.g.*, many minutes) to compare two 3D dose distributions of clinically relevant sizes. On the other hand, the comparison of 3D dose distributions becomes necessary in patient specific QA for recently developed and more sophisticated treatment modalities such as volumetric modulated arc therapy (VMAT) (Teke *et al.*, 2010). Therefore, there is a clinical need to significantly speed up 3D γ -index computations.

In more recent years, much effort has been invested to develop fast and/or accurate γ -index calculation algorithms. Wendling *et al.* (Wendling *et al.*, 2007) speeded up the exhaustive search by pre-sorting involved evaluation dose points with respect to their spatial distances to a reference dose point and performing interpolation on-fly in a fixed searching radius region. This fixed search region induces an overestimation of γ -index values at certain cases when dose difference values are very large inside the search region and have sharp drop just beyond the search region boundary. This algorithm also relies on a fine dose interpolation to

secure accuracy. The geometric interpretation of γ -index evaluation technique proposed by Ju *et al.* (Ju *et al.*, 2008) implies a linear interpolation by calculating the distance from a reference point to a subdivided simplex formed by evaluation dose points in search regions. Thus, high accuracy and efficiency can be achieved without interpolating the dose grid to a fine resolution. However, searching the closest distance over all subdivided simplexes is still time-consuming. Later, Chen *et al.* (Chen *et al.*, 2009) reports a method based on using fast Euclidean distance transform (EDT) of quantized n -dimensional dose distributions. Fast γ -index evaluation can be achieved with the complexity of $O(N^n M)$, where N is the size of dose distribution in each dimension, n is the number of dimensions, and M is the number of quantized values for dose distribution. This method brings in discretization errors when quantizing dose distributions. It also requires M times' more memory space of original searching based algorithm. Thus, a full 3D application of EDT method is limited by its memory requirement. The searching based algorithm's complexity is $O(N^n N_s)$, where N_s represents exhaustive search steps. For cases where γ is not very large, where N_s is much smaller than M , this EDT method loses its advantage of efficiency. Recently, Yuan *et al.* (Yuan and Chen, 2010) proposes a technique using a k -d tree technique for nearest neighbor searching. The searching time for N^n voxels dose distribution can be reduced to $(N^n)^{1/k}$, where $2 < k < 3$ for 2D and 3D dose distributions. However, this method requires interpolating dose grid to secure accuracy. Moreover, in certain cases, the overhead of k -d tree construction time is longer than γ -index calculation time.

One approach to achieve fast γ -index evaluation is to implement an accurate algorithm on graphics processing unit (GPU) platform. The GPU is originally designed for graphics rendering. It has recently been introduced into the radiotherapy community to accelerate computational tasks such as DTS reconstruction, CBCT reconstruction, rigid and deformable image registration, dose calculation, and treatment plan optimization (Li *et al.*, 2005; Sharp *et al.*, 2007; Yan *et al.*, 2007; Li *et al.*, 2008; Samant *et al.*, 2008; Jacques *et al.*, 2008; Hissoiny *et al.*, 2009; Gu *et al.*, 2009; Men *et al.*, 2009; Jia *et al.*, 2010b; Gu *et al.*, 2010; Jia *et al.*, 2010a; Men *et al.*, 2010a; Men *et al.*, 2010b). The GPU is especially well-suited for problems that can be expressed as data-parallel computations (NVIDIA, 2010). γ -index calculation belongs to this category, because the evaluation of each reference point is totally independent. Instead of implementing the memory demanding EDT method or the large overhead k -d tree method, we decide to combine the accuracy of geometric interpretation technique (Ju *et al.*, 2008) and the efficiency of the pre-sorting technique (Wendling *et al.*, 2007). We will also revise this modified algorithm to make it GPU-friendly and then implement it on GPU to achieve both accuracy and high efficiency.

The remainder of this paper is organized as follows. In Section 2, we will discuss the modified γ -index algorithm and its implementation on GPU. Section 3 will present the evaluation of our GPU-based algorithm using eight 3D IMRT dose distributions pairs. We will first study the speedup factor achieved by GPU implementation. Then, we will study the effects of dose difference sorting and the r -index values on the computation time. We will also investigate how the dose distribution resolution and dose difference (DD) and distance-to-agreement (DTA) criteria impact on computation time. Conclusion will be given in Section 4.

2. Methods and Materials

2.1 A modified γ -index algorithm

The γ -index is the minimum Euclidean distance in normalized dose-distance space (Low *et al.*, 1998):

$$\gamma(\mathbf{r}_r) = \min \{ \Gamma(\mathbf{r}_r, \mathbf{r}_e) \},$$

with

$$\begin{aligned} \Gamma(\mathbf{r}_r, \mathbf{r}_e) &= |\tilde{\mathbf{r}}_r - \tilde{\mathbf{r}}_e|, \forall \{ \mathbf{r}_e \}, \\ \tilde{\mathbf{r}}_r &= \left(\frac{r_r}{\Delta d}, \frac{D_r(\mathbf{r}_r)}{\Delta D} \right), \\ \tilde{\mathbf{r}}_e &= \left(\frac{r_e}{\Delta d}, \frac{D_e(\mathbf{r}_e)}{\Delta D} \right). \end{aligned} \quad (1)$$

Here, $D_r(\mathbf{r}_r)$ is the reference dose distribution at position $\mathbf{r}_r$ and $D_e(\mathbf{r}_e)$ is the evaluated dose distribution at position $\mathbf{r}_e$. ΔD and Δd refer to dose DD criterion and DTA criterion, respectively. Using the geometric method (Ju *et al.*, 2008), the accurate Γ can be obtained by calculating the distance from the reference point $\tilde{\mathbf{r}}_r$ to the continuous evaluation surface formed by discrete evaluation points $\tilde{\mathbf{r}}_e$. And the minimum Γ value is achieved by accelerated exhaustive search with pre-sorting algorithm (Wendling *et al.*, 2007). The algorithm A1 illustrates the CPU implementation of combined presorting and geometric γ -index algorithm.

Algorithm A1: A modified γ -index calculation algorithm implemented on CPU

1. Calculate the maximum DD: $\max(DD(\mathbf{r}_r)) = \max(D_r(\mathbf{r}_r) - D_e(\mathbf{r}_r)), \forall \{ \mathbf{r}_r \}$;
2. Calculate the geometric distance set L_n , which defines the maximum search range for each reference point;

$$L_n = \sqrt{(i\Delta x)^2 + (j\Delta y)^2 + (k\Delta z)^2} / \Delta d;$$

with $|i(\text{or } j, k)| \leq \frac{\max(DD(\mathbf{r}_r))}{\Delta D} \frac{\Delta d}{\Delta x(\text{or } \Delta y, \Delta z)} \& |L_n| < \frac{\max(DD(\mathbf{r}_r))}{\Delta D}$, where i, j, k are discretized coordinates in x, y, z directions and $\Delta x, \Delta y, \Delta z$ refer to resolutions in x, y, z directions;

3. Sort the geometric distance set $\{n, L_n\}$ in ascending order of L_n , with n as the total number of voxel inside the search range
4. For each reference dose point:
 - a. Set $\gamma(\mathbf{r}_r) = DD(\mathbf{r}_r) / \Delta D$;
 - b. For $n = 1: N$ (N is the length of $\{L_n\}$)

For $j = 1:nS$ (nS is the number of simplexes in one voxel)

- i. Calculate Euclidean distance $\Gamma(\tilde{\mathbf{r}}_r, S_j)$ from reference point $\tilde{\mathbf{r}}_r$ to a k -simplex S_j :

$$\Gamma(\tilde{\mathbf{r}}_r, S_j) = \begin{cases} \min \left| \tilde{\mathbf{r}}_r - \sum_{i=1}^{k+1} \omega_i \mathbf{v}_i \right|, & \text{if all } \omega_i > 0 \\ \min_{S_i \in \partial S_j} \Gamma(\tilde{\mathbf{r}}_r, S_i), & \text{others} \end{cases};$$

- ii. If $\Gamma(\tilde{\mathbf{r}}_r, S_j) < \gamma(\mathbf{r}_r) : \gamma(\mathbf{r}_r) = \Gamma(\tilde{\mathbf{r}}_r, S_j)$;

End For

If $\gamma(\mathbf{r}_r) < L_n$ break;
End For

End For

Similar to Wendling *et al.* (Wendling *et al.*, 2007), at the Steps 2 and 3 of Algorithm A1, we establish a sorted table of normalized geometric distance $\{L_n\}$ of all the voxels in the maximum search range. However, instead of using a manually selected search range as in (Wendling *et al.* 2007), we choose search radius as $\max(DD(\mathbf{r}_r)) \Delta d / \Delta D$, which can avoid overestimating γ -index values.

The $\Gamma(\tilde{\mathbf{r}}_r, S_j)$ in Algorithm A1 Step 4 is obtained when $\{\omega_1, \dots, \omega_k\} = (V^T V)^{-1} V^T P$,

$\omega_{k+1} = 1 - \sum_{i=1}^k \omega_i$, where P and V are $K \times 1$ and $K \times k$ matrices with a form

$P = \begin{Bmatrix} c_1(\tilde{\mathbf{r}}_r) - c_1(v_{k+1}) \\ \vdots \\ c_n(\tilde{\mathbf{r}}_r) - c_n(v_{k+1}) \end{Bmatrix}$, $V = \begin{Bmatrix} c_1(v_1) - c_1(v_{k+1}) & \cdots & c_1(v_k) - c_1(v_{k+1}) \\ \vdots & \vdots & \vdots \\ c_n(v_1) - c_n(v_{k+1}) & \cdots & c_n(v_k) - c_n(v_{k+1}) \end{Bmatrix}$. Here, for a k -dimensional dose distribution $K = k + 1$, $c_j(q)$ is the j th coordinate of point of q and $\{v_1, \dots, v_{k+1}\}$ are the vertices of a k -th simplex. Regarding $\Gamma(\tilde{\mathbf{r}}_r, S_j)$ calculation, we follow the computational acceleration techniques presented by Ju *et al.* (Ju *et al.*, 2008), where the computation is conducted recursively in the simplexes set and the recursive computation is only limited to the subset of simplexes where corresponding weights ω_i are negative. Detailed information regarding $\Gamma(\tilde{\mathbf{r}}_r, S_j)$ calculation can be found in the reference (Ju *et al.*, 2008).

2.2 GPU implementation

In this work, we implement the γ -index algorithm (Algorithm A1) on GPU using Compute Unified Device Architecture (CUDA) programming environment. In the Algorithm A1 Step 4, for each reference point the minimum Γ value is searched around the reference point in a search range of a radius $(DD(\mathbf{r}_r) / \Delta D) \Delta d$. On CPU, Step 4 is repeated for all reference points in a sequential manner. On GPU, this step can be parallelized for a large number of reference points and executed simultaneously using multiple threads. A key point of the GPU implementation of this algorithm is to ensure all threads in the same batch (strictly speaking *warp* in CUDA terminology) to have similar numbers of arithmetic operations. This is because, if some threads in a warp require much longer execution time, the other threads in this warp will finish first and then wait in idle until the longer execution time threads finish, implying a waste of computational power. Therefore, directly mapping the CPU version of γ -index algorithm (Algorithm A1) onto GPU cannot guarantee that all threads in a warp have similar computation burdens, and consequentially cannot achieve maximum speed up. As we know, the upper boundary of the search range for each reference point is $(DD(\mathbf{r}_r) / \Delta D) \Delta d$. The computation task for each reference point is then approximately proportional to the dose difference $DD(\mathbf{r}_r)$. The larger the $DD(\mathbf{r}_r)$, the more evaluation dose points will be involved, leading to longer computation time. We therefore pre-sort the voxels according to $DD(\mathbf{r}_r)$ (for convenience we call it DD sorting) and perform γ -index calculation on GPU according to the pre-sorted voxel order. This DD-sorting procedure, along with the pre-sorting geometric distance set $\{n, L_n\}$, can be parallelized using recently developed Thrust library functions (Hoberock *et al.*, 2010), which can sort a (or multiple) millions-element array(s) within subseconds. The completed GPU-based γ -index algorithm is illustrated as following:

Algorithm A2: A modified γ -index calculation algorithm implemented on GPU

1. Transfer dose distributions data from CPU to GPU;

2. CUDA Kernel 1: calculate in parallel the dose difference $DD(\mathbf{r}_r) = D_r(\mathbf{r}_r) - D_e(\mathbf{r}_r)$, $\forall \{\mathbf{r}_r\}$;
3. Sort in parallel $\{\text{Voxel Index}, DD(\mathbf{r}_r)\}$ array pair in ascending order of $DD(\mathbf{r}_r)$ using Thrust parallel sorting function and obtain $\max(DD(\mathbf{r}_r))$;
4. CUDA Kernel 2: calculate in parallel the geometric distance set $\{L_n\}$;
5. Sort in parallel the geometric distance set $\{n, L_n\}$ in ascending order of L_n using Thrust parallel sorting function;
6. CUDA Kernel 3: calculate in parallel the γ -index values using the algorithm illustrated in Step 4 of Algorithm A1;
7. Sort $\{\text{Voxel Index}, \gamma\}$ back to the original voxel index order;
8. Transfer the γ -index data from GPU to CPU.

We would like to point out that the Step 4-b-i of Algorithm A1 utilizes a recursive algorithm for computing $\Gamma(\mathbf{r}_r, S_j)$ only in the subset of simplexes which involves many IF conditions and thus creates a branching issue in GPU implementation. To avoid this problem, in Step 6 (Kernel 3) of Algorithm A2, we calculate $\Gamma(\mathbf{r}_r, S_j)$ in all simplexes.

2.3 Experimental data sets

We tested our GPU implementation on eight IMRT dose-distribution pairs (4 lung cases (L1-L4) and 4 head-neck cases (H1-H4)), which were generated using a Monte Carlo dose engine called MCSIM (Ma *et al.*, 2002) as well as an in-house developed pencil beam algorithm (Gu *et al.*, 2009). Monte Carlo dose calculation results were treated as the reference dose distributions while results obtained from the pencil beam algorithm were used as the evaluation dose distributions. All the doses were originally calculated with the voxel size of $4.0\text{mm} \times 4.0\text{mm} \times 2.5\text{mm}$ and normalized to the prescription dose and interpolated to various resolution levels for comparison studies. CPU computation was conducted on a 4-core Intel Xeon 2.27 GHz processor. GPU computation was performed on one single NVIDIA Tesla C1060 card, which has 240 processors cores (1.3 GHz) and 4 GB device memory. We would like to point out that our GPU implementation did not affect the calculation accuracy; in all scenarios, the γ -index values calculated on GPU agree with those calculated on CPU within $\sim 10^{-6}$. In the following sections, we present results under various conditions. For the CPU implementation based on Algorithm A1, we divide the total computation time T^C into two parts, *i.e.*, $T^C = T_p^C + T_\gamma^C$, where T_p^C is the data processing time (Steps 1, 2, and 3 of Algorithm A1) and T_γ^C is the γ -index calculation time (Step 4 of Algorithm A1). For the GPU implementation based on Algorithm A2, we split the total computation time T^G into three parts, *i.e.*, $T^G = T_t^G + T_p^G + T_\gamma^G$, where T_t^G is the data transferring time between CPU and GPU (Steps 1 and 8 of Algorithm A2), T_p^G is the data processing time (Steps 2-5 and Step 7 of Algorithm A2), and T_γ^G is the γ -index calculation time (Step 6 of Algorithm A2).

3. Experimental Results and Discussion

3.1 Speedup of GPU vs. CPU

For this study, we set the resolution of dose distributions to be $256 \times 256 \times 144$ (or 160, 206) and use 3% for DD criterion and 3 mm for DTA criterion. Table 1 lists computation time for the CPU implementation (Algorithm A1) and the GPU implementation (Algorithm A2). We present two speedup factors in Table 1, with and without CPU-GPU data

transferring time, *i.e.*, T^C/T^G and $T^C/(T^G - T_t^G)$. These two speedup factors are quite similar (within 3% for all cases), indicating that the data transferring time in GPU calculation is not significant compared to γ -index computation time T_γ^G . We can also see that the data processing time in both CPU and GPU implementations is relatively insignificant compared to the γ -index calculation time (Step 4 of Algorithm A1 and Step 6 of Algorithm A2). Overall, the GPU implementation can achieve about 45~75 $\times$ speedup compared to its CPU implementation.

3.2 The effect of DD sorting on computation time T_γ^G

As mentioned in Section 2.2, introducing of DD sorting (Step 3 in Algorithm A2) can better synchronize the computational tasks on CUDA threads and consequently reduce

computation time. We illustrate the effect of DD sorting on T_γ^G in Table 2. The speedup achieved by DD sorting is around 2.7- 5.5 times.

3.3 The effect of γ -index values on computation time T_γ^C and T_γ^G

From Table 1, we see that, both T_γ^C and T_γ^G change significantly from case to case. Taking cases L3 and L4 as an example, they have the same number of voxels, but their γ -index calculation time differs by around 3 times. We know that computation time t for each reference point is proportional to the number of searched voxels N_s , *i.e.*, $t \propto N_s$. The relationship of N_s with the search length L can be expressed as $N_s \propto L^n$, where n is the dimension of dose distributions. Here, for all the testing cases in this paper, $n = 3$. On the other hand, from Algorithm A1 Step 4-b-ii, we can deduce the search length L is proportional to the γ value at each reference point, *i.e.*, $L \propto \gamma$. Thus, we can state that the computation time t for each reference point is proportional to γ^n : $t \propto \gamma^n$. In Figure 1(a), we plot T_γ^C and T_γ^G versus the summation of γ^3 over all voxels ($\Sigma \gamma^3$) for each of 8 testing cases, respectively. We see that both T_γ^C and T_γ^G are monotonically increasing with the value of $\Sigma \gamma^3$. To further illustrate our point, we choose one set of patient data (H2) and shift the evaluation dose distribution (normalized to the prescription dose) by -10%, -9%, ..., up to 10%, at a step size of 1%, inside the region of 10% iso-dose line. Figure 1(b) illustrates the γ -index calculation time T_γ^C and T_γ^G with respect to $\Sigma \gamma^3$. We can see that T_γ^C versus $\Sigma \gamma^3$ can be fitted with a straight line (dashed line in Figure 1(b)), indicating that T_γ^C strictly follows the rule $T_\gamma^C \propto \Sigma \gamma^n$. However, the data points for T_γ^G are much more scattered. This is because the GPU computation time is not only the function of $\Sigma \gamma^3$, but also the function of γ^n distribution that determines the variation of threads computation time in a warp.

3.4 The effect of dose distribution resolution on the computation time T_γ^C and T_γ^G

We choose the case H2 to test the effect of the dose distribution resolution on computation time T_γ^C and T_γ^G . We interpolate the dose distributions to various resolution levels, including $128 \times 128 \times 72$, $128 \times 128 \times 144$, $256 \times 256 \times 72$, $256 \times 256 \times 144$, and $512 \times 512 \times 72$.

We illustrate T_γ^C and T_γ^G change with respect to the resolution changes in Figure 2. As indicated by the power trend lines (dashed lines in Figure 2), T_γ^C increases approximately as $N^{1.90}$ while T_γ^G increases approximately as $N^{1.75}$, when the resolution of dose distribution increases N times. As illustrated in Algorithm A1, the CPU based γ -index calculation is completed with two loops. The outer loop is over all the reference dose points and the inner loop is an exhaustive search in a limited region around each reference dose point. The

computation time of the outer loop is increased linearly with respect to the increase of resolution of dose distributions. The inner loop computation time is proportional to the number of voxels involved. For the geometric method, the number of involved voxels in a fixed region is increased linearly as the resolution increases. Overall, it leads to a quadratic increase of computational time ($\propto N^{1.90}$) for a linear change of resolution. For the GPU algorithm A2, the computation time increases as $(N)^{1.75}$ in this testing case. This slight difference might be due to the fact that the memory accessing time can be hidden by large arithmetic operations in GPU computation.

3.5 The effect of DD and DTA criteria on computation time T_{γ}^C and T_{γ}^G

In this study, we choose case H2 and fix the resolution to $256 \times 256 \times 144$, then vary DD and DTA criteria. Table 3 lists the computation time obtained from varying criteria. There are three interesting phenomena: 1) when we increase the DD criterion value and fix the DTA criterion value, the computation time decreases; 2) when we fix the DD criterion value and increase the DTA criterion value, the computation time increases; 3) when we increase both DD criterion value and DTA criterion value proportionally, for example, from 1%, 1mm to 2%, 2mm, or 3%, 3mm, the computation time do not change. As we mentioned in Section 3.4, the γ -index calculation time for each reference dose point $t \propto \gamma^n$. For phenomenon 1), when we increase the DD criterion value and fix the DTA criterion value, the γ -index value decreases. Consequently, the required searching steps decreases, and computation time decreases. For phenomenon 2), when we fix the DD criterion value, but increase the DTA criterion value by k times, the γ -index values will decrease by k' times, with $k' < k$. This decreases computation time by $(k')^n$ times. However, when the DTA criterion value increases by k times, the resolution in the normalized dose-distance space will also increase by k^n times, which consequently increases the computation time by k^n times. The net change of the computation time should be $(k/k')^n$. Since $k' < k$, the overall computation time will then increase. For phenomenon 3), when we increase both DTA and DD criteria values simultaneously by k times, the γ -index values decrease by $k' = k$ times. The increase rate of computation time will be $(k/k)^n = 1$. The computation time under this situation will not change. From Table 3, we can see that the change of DD and DTA criteria values does not affect the speedup factor achieved with GPU implementation.

4. Conclusions

In this paper, we implemented a modified γ -index algorithm on GPU. We evaluated our GPU implementation on eight pairs of IMRT dose distributions. Overall, when using one single Tesla C1060 GPU card for GPU computation and one core Intel Xeon 2.27GHz processor, our GPU implementation has achieved about 45~75 $\times$ speedup compared to the CPU implementation and can finish the γ -index calculation within a few seconds. We also studied the effects of various factors on the calculation time on both CPU and GPU for our implementation of the modified γ -index algorithm. We found that the pre-sorting procedure based on the dose difference speeds up the GPU calculation by about 2.7~5.5 times. The CPU computation time is proportional to the summation of γ^n over all voxels, where n is the dimensions of dose distributions. The GPU computation time is approximately proportional to the summation of γ^n over all voxels, but affected by the variation of γ^n among different voxels. We also found that increasing the resolution of dose distribution leads to a quadratic increase of computation time on CPU, while less-than-quadratic increase on GPU. We observed that both CPU and GPU computation time decrease when increasing the DD criterion value and fixing the DTA criterion value, increase when increasing the DTA criterion value and fixing the DD criterion value, and don't vary when both DD and DTA criterion values change proportionally. Both CPU and GPU codes developed in this work for γ -index dose evaluation are in public domain and available upon request.

Acknowledgments

This work is supported in part by the University of California Lab Fees Research Program, by Varian Medical Systems, Inc., and by an NIH/NCI grant 1F32 CA154045-01. We would like to thank NVIDIA for providing GPU cards for this project.

References

- Bakai A, Alber M, Nusslin F. A revision of the gamma-evaluation concept for the comparison of dose distributions. *Phys Med Biol*. 2003; 48:3543–53. [PubMed: 14653561]
- Chen ML, Lu WG, Chen Q, Ruchala K, Olivera G. Efficient gamma index calculation using fast Euclidean distance transform. *Phys Med Biol*. 2009; 54:2037–47. [PubMed: 19287084]
- Depuydt T, Van Esch A, Huyskens DP. A quantitative evaluation of IMRT dose distributions: refinement and clinical assessment of the gamma evaluation. *Radiotherapy and Oncology*. 2002; 62:309–19. [PubMed: 12175562]
- Gu XJ, Choi DJ, Men CH, Pan H, Majumdar A, Jiang SB. GPU-based ultra-fast dose calculation using a finite size pencil beam model. *Phys Med Biol*. 2009; 54:6287–97. [PubMed: 19794244]
- Gu XJ, Pan H, Liang Y, Castillo R, Yang DS, Choi DJ, Castillo E, Majumdar A, Guerrero T, Jiang SB. Implementation and evaluation of various demons deformable image registration algorithms on a GPU. *Phys Med Biol*. 2010; 55:207–19. [PubMed: 20009197]
- Hisson S, Ozell B, Despres P. Fast convolution-superposition dose calculation on graphics hardware. *Medical Physics*. 2009; 36:1998–2005. [PubMed: 19610288]
- Hoferock, J.; Bell, N. Thrust: A Parallel Template Library. 2010.
- Jacques R, Taylor R, Wong J, McNutt T. Towards Real-Time Radiation Therapy: GPU Accelerated Superposition/Convolution. *High-Performance MICCAI Workshop*. 2008
- Jia X, Gu XJ, Sempau J, Choi D, Majumdar A, Jiang SB. Development of a GPU-based Monte Carlo dose calculation code for coupled electron-photon transport. *Phys Med Biol*. 2010a; 55:3077–86. [PubMed: 20463376]
- Jia X, Lou YF, Li RJ, Song WY, Jiang SB. GPU-based fast cone beam CT reconstruction from undersampled and noisy projection data via total variation. *Medical Physics*. 2010b; 37:1757–60. [PubMed: 20443497]
- Jiang SB, Sharp GC, Neicu T, Berbeco RI, Flampouri S, Bortfeld T. On dose distribution comparison. *Phys Med Biol*. 2006; 51:759–76. [PubMed: 16467577]
- Ju T, Simpson T, Deasy JO, Low DA. Geometric interpretation of the gamma dose distribution comparison technique: Interpolation-free calculation. *Medical Physics*. 2008; 35:879–87. [PubMed: 18404924]
- Li G, Xie HC, Ning H, Capala J, Arora BC, Coleman CN, Camphausen K, Miller RW. A novel 3D volumetric voxel registration technique for volume-view-guided image registration of multiple imaging modalities. *International Journal of Radiation Oncology Biology Physics*. 2005; 63:261–73.
- Li G, Xie HC, Ning H, Citrin D, Capala J, Maass-Moreno R, Guion P, Arora B, Coleman N, Camphausen K, Miller RW. Accuracy of 3D volumetric image registration based on CT, MR and PET/CT phantom experiments. *Journal of Applied Clinical Medical Physics*. 2008; 9:17–36.
- Low DA, Harms WB, Mutic S, Purdy JA. A technique for the quantitative evaluation of dose distributions. *Medical Physics*. 1998; 25:656–61. [PubMed: 9608475]
- Ma CM, Li JS, Pawlicki T, Jiang SB, Deng J, Lee MC, Koumrian T, Luxton M, Brain S. A Monte Carlo dose calculation tool for radiotherapy treatment planning. *Phys Med Biol*. 2002; 47:1671. [PubMed: 12069086]
- Men CH, Gu XJ, Choi DJ, Majumdar A, Zheng ZY, Mueller K, Jiang SB. GPU-based ultrafast IMRT plan optimization. *Phys Med Biol*. 2009; 54:6565–73. [PubMed: 19826201]
- Men CH, Jia X, Jiang SB. GPU-based ultra-fast direct aperture optimization for online adaptive radiation therapy. *Phys Med Biol*. 2010a; 55:4309–19. [PubMed: 20647601]
- Men CH, Jia X, Jiang SB, Romeijn HE. Ultrafast treatment plan optimization for volumetric modulated arc therapy (VMAT). *Medical Physics*. 2010b; 37:5787–91. [PubMed: 21158290]

- NVIDIA. NVIDIA CUDA Compute Unified Device Architecture, Programming Guide version 3.2. ed NVIDIA. 2010.
- Samant SS, Xia JY, Muyan-Ozcelik P, Owens JD. High performance computing for deformable image registration: Towards a new paradigm in adaptive radiotherapy. *Medical Physics*. 2008; 35:3546–53. [PubMed: 18777915]
- Sharp GC, Kandasamy N, Singh H, Folkert M. GPU-based streaming architectures for fast cone-beam CT image reconstruction and demons deformable registration. *Phys Med Biol*. 2007; 52:5771–83. [PubMed: 17881799]
- Spezi E, Lewis DG. Gamma histograms for radiotherapy plan evaluation. *Radiotherapy and Oncology*. 2006; 79:224–30. [PubMed: 16697065]
- Stock M, Kroupa B, Georg D. Interpretation and evaluation of the gamma index and the gamma index angle for the verification of IMRT hybrid plans. *Phys Med Biol*. 2005; 50:399–411. [PubMed: 15773719]
- Teke T, Bergman AM, Kwa W, Gill B, Duzenli C, Popescu IA. Monte Carlo based, patient-specific RapidArc QA using Linac log files. *Medical Physics*. 2010; 37:116–23. [PubMed: 20175472]
- Wendling M, Zijp LJ, McDermott LN, Smit EJ, Sonke JJ, Mijnheer BJ, Van Herk M. A fast algorithm for gamma evaluation in 3D. *Medical Physics*. 2007; 34:1647–54. [PubMed: 17555246]
- Yan H, Ren L, Godfrey DJ, Yin FF. Accelerating reconstruction of reference digital tomosynthesis using graphics hardware. *Medical Physics*. 2007; 34:3768–76. [PubMed: 17985622]
- Yuan JK, Chen WM. A gamma dose distribution evaluation technique using the k-d tree for nearest neighbor searching. *Medical Physics*. 2010; 37:4868–73. [PubMed: 20964204]

Figure 1.

(a) GPU and CPU computation time for eight testing cases vs. the summation of γ^3 values over all evaluated points. (b) GPU and CPU computation time for case H2 with various dose shifts on the evaluation dose distribution vs. the summation of γ^3 values over all evaluated points. For convenient purpose, we scaled down CPU computation time by a factor of 60.0 to illustrate them in the same vertical axis of GPU computation time.

Figure 2.

CPU and GPU computation time as functions of dose distribution resolution. Again, For the convenient purpose, we scaled down CPU computation time by a factor of 60.0 to illustrate CPU computation time in the same axis of GPU computation time.

Table 1
Calculation time of λ -index for CPU and GPU implementations for 8 IMRT dose distributions pairs.

Case	Voxel number	CPU (sec)			GPU (sec)			Speedup factor	
		T_p^C	T_γ^C	T^C	T_i^G	T_p^G	T_γ^G	T^G	$T^C/(T^G - T_i^G)$
L1	256×256×206	0.33	64.93	65.26	0.07	0.18	1.18	1.43	47.99
L2	256×256×160	0.24	65.64	65.89	0.06	0.15	1.13	1.34	51.47
L3	256×256×160	0.28	101.46	101.74	0.06	0.14	1.68	1.88	55.90
L4	256×256×160	0.25	30.10	30.35	0.06	0.14	0.43	0.63	53.25
H1	256×256×144	0.49	47.73	47.95	0.05	0.11	0.91	1.07	47.27
H2	256×256×144	0.45	242.23	242.68	0.05	0.12	3.13	3.30	74.67
H3	256×256×144	0.24	116.14	116.38	0.05	0.12	2.10	2.27	52.42
H4	256×256×144	0.22	107.61	107.86	0.05	0.12	1.75	1.92	57.66

Table 2

Speedup achieved in GPU computation by sorting voxels based on the dose difference values.

Case	T_{γ}^G (Non-DD sorting) (sec)	T_{γ}^G (DD sorting) (sec)	Speedup factor achieved by DD sorting
L1	4.37	1.18	3.70
L2	4.62	1.13	4.09
L3	6.71	1.68	3.99
L4	1.18	0.43	2.74
H1	4.99	0.91	5.48
H2	14.31	3.13	4.57
H3	9.01	2.10	4.29
H4	12.54	1.75	7.17

Table 3

CPU and GPU computation time varies with DD and DTA criteria values.

DD criteria	DTA criteria (mm)	Computational time (sec)		Speedup factor $T_{\gamma}^C / T_{\gamma}^G$
		T_{γ}^G	T_{γ}^C	
1%	1	3.13	240.37	76.80
1%	2	11.33	863.33	76.20
1%	3	23.91	1545.09	64.62
2%	1	0.97	74.27	76.57
2%	2	3.13	265.82	84.93
2%	3	6.53	486.68	74.53
3%	1	0.55	40.09	72.89
3%	2	1.55	119.32	76.98
3%	3	3.13	245.34	78.38