# Fast trapping force calculation by utilizing graphic processing unit

Sheng-Yang Tseng*[a], Ai-Tang Chang[a], Chih-Wei Luo[a], Long Hsu[a], Takayoshi Kobayashi[a,b,c,d]
[a]Department of Electrophysics, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan 300, ROC; [b]ICORP, JST, 4-1-8 Honcho, Kawaguchi, Saitama, 332-0012, Japan; [c]University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, 182-8585, Japan; [d]Institute of Laser Engineering, Osaka University, 2-6 Yamada-oka, Suita, Osaka, 565-0971, Japan

## ABSTRACT

Fast calculation of trapping force provides a more direct way for optimizing designs of optical systems which generate optical traps. In this study, a graphic processing unit (GPU), NVIDIA GTX 275, is used to boost the speed of trapping efficiency calculation under ray optics approximation. The codes of trapping efficiency calculation are implemented in C++. The computing power of GPU is utilized through compute unified architecture device (CUDA) toolkit 4.0. The computing speed is compared with that of central processing unit (CPU), Intel Core 2 Quad Q9550. Over 100x speedup is achieved when single-precision floating-point numbers were used in the calculation.
**Keywords:** Graphic processing unit, optical trapping force, Ray optics

## 1. INTRODUCTION

The most important thing in generating optical traps is focusing a laser beam tightly enough to exert noticeable gradient forces on particles. In optical tweezers[1], this work is often done by a high-numerical-aperture objective lens. The objective lens is corrected to be free of aberrations and can bring a laser beam into a diffraction-limited spot. Hence, optical tweezers are often built around a commercial microscope. In recently years, some researchers fabricated optical components for trapping, such as micro-mirror array[2, 3], micro-lens array[4, 5], or cylindrical mirror[6]. In these cases, researchers have to design and optimize their own optical components for generating effective traps. However, instead of maximizing trapping efficiency, most of optimizations are done by minimizing aberrations of the systems, which may lead to that the obtained result is not the best one. This is due to calculating the trapping efficiency is computationally intense.

In the calculation of trapping force under ray optics (RO) approximation, the incident laser beam is often decomposed into many rays. The force exerted on a particle by each ray is calculated. The trapping force is then obtained by adding the forces contributed by all the rays. When the force calculation is implemented in a central processing unit (CPU) code, the force of each ray is calculated sequentially. When the number of rays is large, for example hundreds of thousands, or a two-dimensional force scan is required, the calculation takes more time. This makes using trapping efficiency for optimizing optical designs impractical. In contrast to CPU, modern programmable graphic processing unit (GPU) can lunch many thousands of threads at the same time[7]. Each thread can calculate the force of each ray simultaneously so the calculation speed can be greatly increased.

In this study, we use a GPU to increase the calculation speed of trapping force. The method we used to calculate trapping force is first introduced. The calculation result and speed of CPU and GPU are then compared.

## 2. METHOD

### 2.1 Ray optics approximation of trapping force

In RO approximation of trapping force calculation, a laser beam is modeled by rays[8]. Every ray carries momentum and energy. When these rays enter a particle, they change propagating directions and therefore their momentum due to the refractive index change. The momentum change implies that the particle exerts a force on each ray. Every ray thus exerts a reaction force, which has the same magnitude but opposite direction, on the particle simultaneously. The total sum of these reaction forces is the trapping force on the particle.

*twsone@gmail.com

Optical Trapping and Optical Micromanipulation IX, edited by Kishan Dholakia, Gabriel C. Spalding, Proc. of SPIE Vol. 8458, 84582Y · © 2012 SPIE · CCC code: 0277-786/12/$18 · doi: 10.1117/12.929238

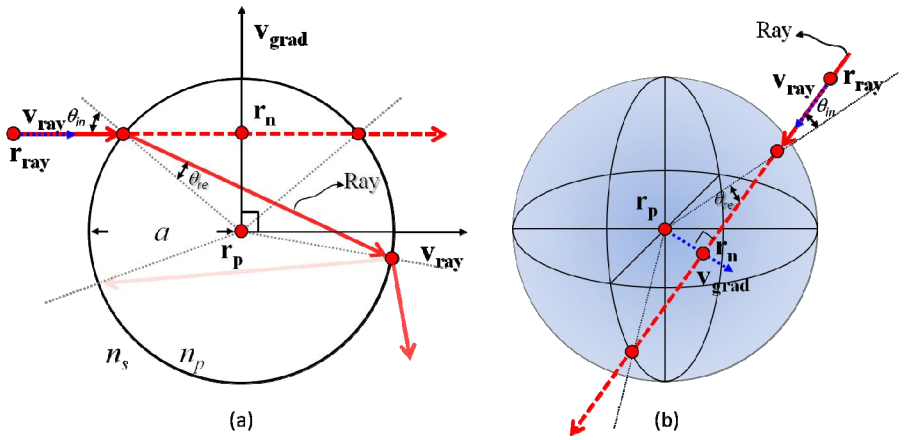


Figure 1. A path of a ray entering a particle. (a) is ray path on the incident plane and (b) is a three-dimensional illustration of the ray path.

Figure 1 shows a path of ray entering a particle. The particle is located at $\mathbf{r}_p$ and has radius $a$ and refractive index $n_p$. When a circular polarized ray of power $P$ which originates from position $\mathbf{r}_{ray}$ and is along a unit direction vector $\mathbf{v}_{ray}$ is incident on the particle, the force $\mathbf{F}_{ray}$ contributed by the ray is given by[8]

$$\mathbf{F}_{ray} = \frac{n_s P}{c}\left\{R\sin(2\theta_{in}) - \frac{T^2[\sin(2\theta_{in}-2\theta_{re}) + R\sin(2\theta_{in})]}{1+R^2+2R\cos(2\theta_{re})}\right\}\mathbf{v}_{grad}$$

$$+\frac{n_s P}{c}\left\{1+R\cos(2\theta_{in}) - \frac{T^2[\cos(2\theta_{in}-2\theta_{re}) + R\cos(2\theta_{in})]}{1+R^2+2R\cos(2\theta_{re})}\right\}\mathbf{v}_{ray} \quad (1)$$

where $c$ is the speed of light in vacuum, $n_s$ is the refractive index of the surrounds, $R$ and $T$ are the Fresnel reflection and transmission coefficients[9] of the surface at incident angle $\theta_{in}$, $\theta_{re}$ is the refractive angle, and $\mathbf{v}_{grad}$ is a unit direction vector which are perpendicular to $\mathbf{v}_{ray}$ and directs from the particle center $\mathbf{r}_p$ to the closest position of the ray from the particle $\mathbf{r}_n$. The first term of equation (1) is the gradient force, which attracts the particle to the brightest region. The second term is the scattering force, which pushes the particle along the ray. The trapping force $\mathbf{F}_{total}$ is obtained by summing the contributions of all the incident rays, that is

$$\mathbf{F}_{total} = \sum_{i=1}^{N}(\mathbf{F}_{grad,i} + \mathbf{F}_{scat,i})$$

$$= \frac{n_s P_{total}}{cN}\sum_{i=1}^{N}(\mathbf{Q}_{grad,i} + \mathbf{Q}_{scat,i}), \quad (2)$$

where the index $i$ indicates $i$-th ray, $P_{total}$ is the total incident power, $N$ is the total number of incident rays and $\mathbf{Q}_{grad,i}$ and $\mathbf{Q}_{scat,i}$ are dimensionless vectors which represent the efficiencies of the gradient and scattering force of the $i$-th ray respectively. $\mathbf{Q}_{grad,i}$ and $\mathbf{Q}_{scat,i}$ are given by

$$\mathbf{Q}_{grad,i} = \left\{R_i\sin(2\theta_{in,i}) - \frac{T_i^2[\sin(2\theta_{in,i}-2\theta_{re,i}) + R_i\sin(2\theta_{in,i})]}{1+R_i^2+2R_i\cos(2\theta_{re,i})}\right\}\mathbf{v}_{grad,i} \quad (3)$$

and

$$\mathbf{Q}_{\mathbf{scat},i} = \left\{1 + R_i \cos(2\theta_{in,i}) - \frac{T_i^{\,2}[\cos(2\theta_{in,i} - 2\theta_{re,i}) + R_i \cos(2\theta_{in,i})]}{1 + R_i^{\,2} + 2R_i \cos(2\theta_{re,i})}\right\}\mathbf{v}_{\mathbf{ray},i}\,. \tag{4}$$

$\mathbf{v}_{\mathbf{grad},i}$ in equation (3) is defined by

$$\mathbf{v}_{\mathbf{grad},i} = (\mathbf{r}_{\mathbf{n},i} - \mathbf{r}_{\mathbf{p}}) / |\,\mathbf{r}_{\mathbf{n},i} - \mathbf{r}_{\mathbf{p}}\,|, \tag{5}$$

where $\mathbf{r}_{\mathbf{n},i}$ is the closest position of the $i$-th ray from the particle and

$$\mathbf{r}_{\mathbf{n},i} = \mathbf{r}_{\mathbf{ray},i} + [(\mathbf{r}_{\mathbf{p}} - \mathbf{r}_{\mathbf{ray},i}) \cdot \mathbf{v}_{\mathbf{ray},i}]\mathbf{v}_{\mathbf{ray},i}\,. \tag{6}$$

In addition, $T_i$ and $R_i$ of circular polarized light in equations (3) and (4) are

$$R_i = \frac{1}{2}\left[\left(\frac{n_p \cos(\theta_{in,i}) - n_s \cos(\theta_{re,i})}{n_p \cos(\theta_{in,i}) + n_s \cos(\theta_{re,i})}\right)^2 + \left(\frac{n_s \cos(\theta_{in,i}) - n_p \cos(\theta_{re,i})}{n_s \cos(\theta_{in,i}) + n_p \cos(\theta_{re,i})}\right)^2\right], \tag{7}$$

and

$$T_i = \frac{1}{2}\left[\left(\frac{2n_s \cos(\theta_{in,i})}{n_p \cos(\theta_{in,i}) + n_s \cos(\theta_{re,i})}\right)^2 + \left(\frac{2n_s \cos(\theta_{in,i})}{n_s \cos(\theta_{in,i}) + n_p \cos(\theta_{re,i})}\right)^2\right], \tag{8}$$

where $\theta_{in}$ and $\theta_{re}$ can be expressed as

$$\theta_{in,i} = \sin^{-1}\left(\frac{|\mathbf{r}_{\mathbf{n},i} - \mathbf{r}_{\mathbf{p},i}|}{a}\right), \tag{9}$$

and

$$\theta_{re,i} = \sin^{-1}\left(\frac{n_s}{n_p}\frac{|\mathbf{r}_{\mathbf{n},i} - \mathbf{r}_{\mathbf{p},i}|}{a}\right). \tag{10}$$

Therefore, with $\mathbf{r}_{\mathbf{ray},i}$, $\mathbf{v}_{\mathbf{ray},i}$ of the rays and the position of the particle center $\mathbf{r}_{\mathbf{p}}$, all the incident angles of the rays and the directions of scattering and gradient forces are determined. The magnitude of the forces can be determined further when $n_s$, $n_p$ and $P_{total}$ are known.

A vector **Q** used to indicate force efficiency of a trap is defined as

$$\mathbf{Q} = \mathbf{Q}_{\mathbf{grad}} + \mathbf{Q}_{\mathbf{scat}}, \tag{11}$$

where $\mathbf{Q}_{\mathbf{grad}}$ and $\mathbf{Q}_{\mathbf{scat}}$ are the efficiencies of gradient and scattering force of a trap respectively and are given by

$$\mathbf{Q}_{\mathbf{grad}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{Q}_{\mathbf{grad},i} \tag{12}$$

and

$$\mathbf{Q}_{\mathbf{scat}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{Q}_{\mathbf{scat},i}\,. \tag{13}$$

In the following sections, **Q** is computed in all the numerical calculations.

## 3. CPU AND GPU COMPUTING

In this section, the computing results and speeds of CPU and GPU are compared. The trapping efficiency **Q** of a trap was calculated by using a CPU (Core 2 Quad Q9550, Intel) and a GPU (GTX 275, NVIDIA) separately. Both of the CPU and GPU codes of Q calculation are implemented in C++. The computing power of GPU is utilized through compute unified architecture device (CUDA) toolkit 4.0.

The calculation of the trapping efficiency **Q** of a trap formed by an optical system is started by generating an array of rays, which are specified by $\mathbf{r}_{ray,i}$ and $\mathbf{v}_{ray,i}$. The final values $\mathbf{r'}_{ray,i}$ and $\mathbf{v'}_{ray,i}$ of the rays are then calculated after the rays propagate through the optical system, such as an objective lens. The trapping efficiency contributed by a ray $\mathbf{Q}_i$ is obtained by substituting the final values of $\mathbf{r}_{ray,i}$ and $\mathbf{v}_{ray,i}$ into equation (3) through equation (10). After summing up all of the obtained $\mathbf{Q}_i$ and normalized by the total number of incident rays, the **Q** of the trap is obtained.

When writing a CPU code to compute **Q** of a trap, a loop is often used to scan the entire ray array and compute all the $\mathbf{Q}_i$. The obtained result is added together in each iteration sequentially. After all the $\mathbf{Q}_i$ have been computed and added, **Q** of the trap is obtained. However, in order to have an accurate result, the number of rays is usually very large, which is around hundreds of thousands or more. The sequential calculation takes lots of time especially when a two-dimensional or three-dimensional distribution of **Q** is required.

In contrast to CPU, a modern programmable GPU can lunch a large number of threads at the same time. Each thread can propagate a ray through the optical system and then calculate the $\mathbf{Q}_i$. The obtained $\mathbf{Q}_i$ in each thread is added together by using parallel reduction algorithm[10]. Because of parallel computing, the calculation speed can thus be greatly increased.

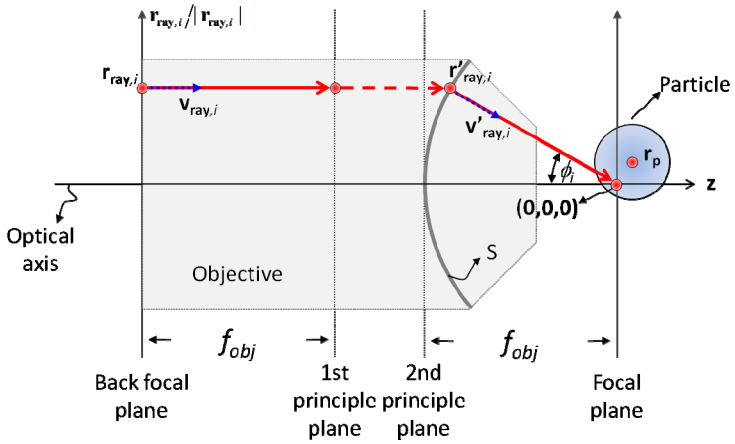


Figure 2. a ray path in a trap which is formed by focusing a normal-incident, collimated laser beam with an infinite-corrected objective lens.

To compare the computing speeds and results of CPU and GPU, **Q** of a trap which is formed by focusing a normal-incident, collimated laser beam with an infinite-corrected, water-immersion objective lens is calculated. We assume that the particle has refractive index of 1.55 and is placed in water, which has refractive index of 1.33. In addition, the objective lens has numerical aperture (NA) of 1.1.

Figure 2 is a schematic illustration of a ray path of the trap. The objective, which obeys Abbe sine condition[11], is modeled by focal planes, principle plane, and a spherical surface S. The optical axis is along the *z*-axis. The distance between a focal plane and a nearby principle plane is $f_{obj}$. The surface S is centered at the focal point and is tangent to the 2nd principle. The ray starts from position $\mathbf{r}_{ray,i}$ and has a direction vector $\mathbf{v}_{ray,i} = (0,0,1)$. After the ray passes through the objective lens, it leave at position $\mathbf{r'}_{ray,i}$ on surface S, whose transverse component is the same as that of $\mathbf{r}_{ray,i}$, and converges to the focal point with a direction vector $\mathbf{v'}_{ray,i}$, which is given by

$$\mathbf{v'}_{ray,i} = -\mathbf{r'}_{ray,i} / |\mathbf{r'}_{ray,i}| \tag{14}$$

and

$$\mathbf{r'}_{\mathbf{ray},i} = \mathbf{h}_{\mathbf{ray},i} - \sqrt{f_{obj}{}^2 - |\mathbf{h}_{\mathbf{ray},i}|^2}\,\mathbf{z}, \tag{15}$$

where $\mathbf{h}_{\mathbf{ray},i}$ is the transverse component of $\mathbf{r}_{\mathbf{ray},i}$. Once an array of $\mathbf{r}_{\mathbf{ray},i}$ and $\mathbf{v}_{\mathbf{ray},i}$ are given, the $\mathbf{r'}_{\mathbf{ray},i}$ and $\mathbf{v'}_{\mathbf{ray},i}$ can be obtained according to equations (14) and (15). **Q** of the trap can then be obtained. Figure 3(a) and 3(b) show GPU results of $\mathbf{Q}_{\mathbf{grad}}$ and $\mathbf{Q}_{\mathbf{scat}}$ of the trap in the *x* and *z* directions when the particle moves along the *x* and *z* axis respectively. The spatial coordinates is normalized by the radius of the particle. The total ray number in the calculation is 768 x 768. **Q** was calculated at 512 positions which are equally distributed along an axis from -3 to 3. Figure 3(c) and 3(d) show the difference distribution of **Q** between the results of CPU and GPU. The ΔQ is defined by subtracting the Q calculated by using CPU from the Q calculated by using GPU. The difference comes from that only single-precision floating-point numbers were used in the GPU calculation for a faster computing speed. The maximum value of ΔQ is less than $5\times10^{-5}$. The larger differences distribute only in the regions where the focus of the laser beam is outside the particle. In that case, the particle will not stay in the trap and thus the difference of Q could be neglected. Some recent GPUs have already supported double-precision floating-point numbers and the computing speed has also increased. The accuracy of the GPU computing can be improved by using double-precision floating-point numbers while the computing speed is still the same as or even faster than the speed achieved in this study.

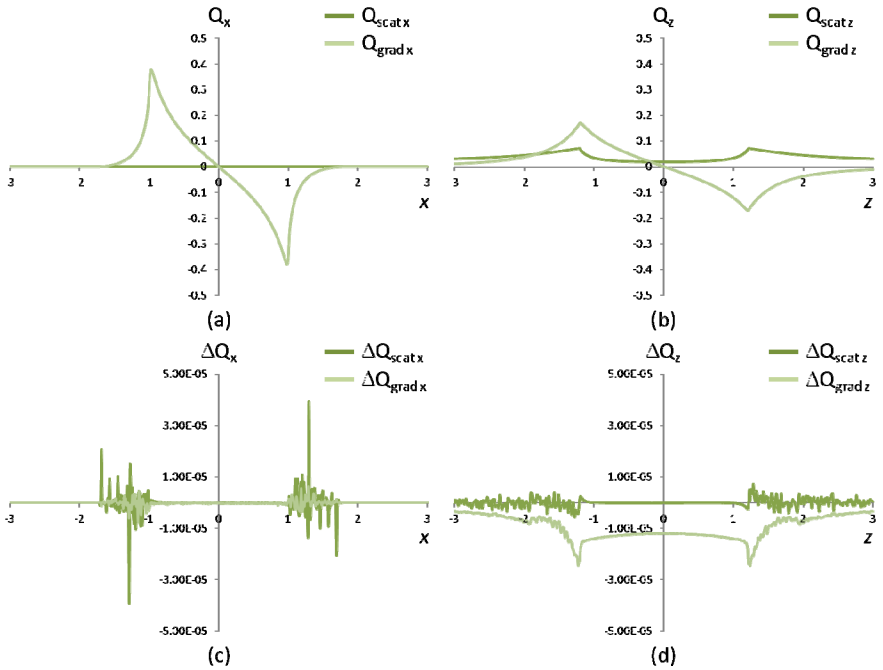


Figure 3. GPU results of Q distribution along the *x* and *z* axis. (a) is the *x* component of $\mathbf{Q}_{\mathbf{scat}}$ and $\mathbf{Q}_{\mathbf{grad}}$ along the *x* axis. (b) is the *z* component of $\mathbf{Q}_{\mathbf{scat}}$ and $\mathbf{Q}_{\mathbf{grad}}$ along the *z* axis. (c) is the difference between the CPU and GPU results along the *x* axis. (d) is the difference between the CPU and GPU results along the *z* axis.

Figure 4 shows speedup of GPU computing at different total ray number. The speedup is obtained by dividing the computation time of CPU by that of GPU. When the total ray number is small, the speedup is small because only a small number of threads were used in the calculation. As the total ray number increases, the number of threads used in the calculation also increases. The speedup grows rapidly at first and reaches a maximum value of speedup, which is about 108. The speedup is limited by the number of threads which can be lunched at the same time in a GPU.

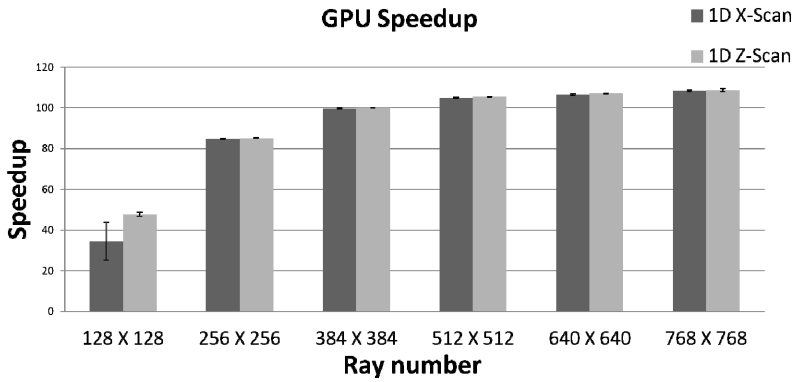


Figure 4. GPU speedup at different total ray number.

## 4. CONCLUSION

In this study, we used a GPU to calculate the trapping efficiency of a trap formed by focusing a collimated normal-incident laser beam with an infinite-corrected objective lens. A modern programmable GPU can lunch hundreds of thousands threads simultaneously. Each thread calculates the contribution of an incident ray. In this way, GPU computing can provide more than 100 times the speed of CPU computing when single-precision floating-point numbers were used in the calculation. In addition, the GPU results were compared with the double-precision floating-point results of CPU. The maximum difference of **Q** between the results of CPU and GPU is less than $5\times10^{-5}$. Larger differences of **Q** only occur at the positions where the laser focus is outside the particle. In that situation, the particle does not stay in the trap. The value of Q cannot be measured accurately by an experiment. The difference could be neglected. Therefore, calculating trapping efficiency for optimizing optical designs of trapping systems becomes practical.

## REFERENCES

[1] A. Ashkin, J. M. Dziedzic, J. E. Bjorkholm et al., "Observation of a single-beam gradient force optical trap for dielectric particles," Optics Letters, 11(5), 288-290 (1986).
[2] F. Merenda, J. Rohner, J.-M. Fournier et al., "Miniaturized high-NA focusing-mirror multiple optical tweezers," Opt. Express, 15(10), 6075-6086 (2007).
[3] Y. S. Ow, M. B. H. Breese, and S. Azimi, "Fabrication of concave silicon micro-mirrors," Opt. Express, 18(14), 14511-14518 (2010).
[4] X. Zhao, Y. Sun, J. Bu et al., "Microlens-array-enabled on-chip optical trapping and sorting," Appl. Opt., 50(3), 318-322 (2011).
[5] F. Chen, H. Liu, Q. Yang et al., "Maskless fabrication of concave microlens arrays on silica glasses by a femtosecond-laser-enhanced local wet etching method," Opt. Express, 18(19), 20334-20343 (2010).
[6] A. T. Chang, S. Y. Tseng, and L. Hsu, "Optical guiding with cylindrical mirror system," Proc. SPIE 7762, 77622T.
[7] D. B. Kirk, and W.-m. W. Hwu, [Programming Massively Parallel Processors: A Hands-on Approach ] Morgan Kaufmann, (2010).
[8] A. Ashkin, "Forces of a single-beam gradient laser trap on a dielectric sphere in the ray optics regime," Biophysical Journal, 61(2), 569-582 (1992).
[9] M. Born, and E. Wolf, [Principles of optics : electromagnetic theory of light] Cambridge University Pr, Cambridge ; New York.
[10] M. Harris, "Optimizing Parallel Reduction in CUDA," NVIDIA GPU Computing SDK 4.0.
[11] M. Mansuripur, [Classical optics and its applications] Cambridge University Press, (2002).