Generating Simplified Trapping Probability Models from Simulation of Optical Tweezers System

Ashis Gopal Banerjee[a], *Arvind Balijepalli*[a,b], *Satyandra K. Gupta*[a,1], *and Thomas W. LeBrun*[b]

[a]Department of Mechanical Engineering and the Institute for Systems Research
University of Maryland
College Park, MD 20742, USA
[b]Precision Engineering Division, Manufacturing Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899, USA

Abstract

This paper presents a radial basis function based approach to generate simplified models to estimate the trapping probability in optical trapping experiments using offline simulations. The difference form of Langevin's equation is used to perform physically accurate simulations of a particle under the influence of a trapping potential and is used to estimate trapping probabilities at discrete points in the parameter space. Gaussian radial basis functions combined with kd-tree based partitioning of the parameter space are then used to generate simplified models of trapping probability. We show that the proposed approach is computationally efficient in estimating the trapping probability and that the estimated probability using the simplified models is sufficiently close to the probability estimates from offline simulation data.

1. Introduction

Simulations have emerged as a powerful tool in the traditional manufacturing and assembly areas to facilitate a wide variety of decision making. We expect simulations to play a major role in microscale assembly as well. However, simulations at the microscale pose new challenges due to the presence of a higher level of uncertainty due to random forces arising from Brownian motion of the components in the surrounding fluid. Furthermore, accurate simulations at the micro-scale can require time-steps smaller than a microsecond, which results in very long simulation runs. Hence, real-time simulations can be challenging to use for assembly planning. On the other hand, offline simulations can be used to generate data at discrete points in the parameter space, which can then be parameterized by computationally efficient simplified models. The work described in this paper investigates this approach using the optical tweezers as a case study.

Light beams exert optical forces on objects. Using this property of light, optical tweezers (OT) have been developed to successfully trap and move nanoscale and microscale components of many different sizes and shapes [1-3]. Components can be simply released from optical traps by switching off the laser beam, thereby overcoming the challenges associated with contact based grippers at the microscale. Optical tweezers also provide a broad range of positioning and orienting capabilities to place components at desired locations in a fluid workspace (also referred to as the assembly cell in this paper). By utilizing the principle of synchronized time sharing,

[1] E-mail address of corresponding author: skgupta@umd.edu

multiple laser beams in an OT system can perform several operations in parallel. These characteristics make optical tweezers a very promising technology for nano and microscale assemblies. The optical tweezers setup, currently being used in the Manufacturing Engineering Laboratory at the National Institute of Standards and Technology (NIST) [4] can be utilized for manipulating microspheres, nanowires, and cells. Other types of OT setup are described in [5]. Locations of components in the workspace can be determined using real-time imaging [6].

Autonomous OT operation requires real-time motion planning, which involves trapping the desired components and avoiding collisions with other components in the workspace. Information that is very useful in this context is the probability with which a component will be trapped in a spatial region close to a moving laser beam center, referred to as the *trapping probability*. Since many decisions need to be taken within a few milliseconds during real-time planning, the trapping probability estimation method must be extremely fast. As already discussed, trapping physics based simulations are computationally intensive. Currently, it takes few seconds to complete a single simulation run at any particular point in space. Moreover, several simulation runs are required at each point to average the noise, which results from the stochastic nature of the simulations. The raw data set generated is also expected to be very large in size, since it includes results from varying several parameters in a combinatorial manner. This paper describes a computational framework to perform offline microscale particle motion simulations and proposes a model simplification technique to represent the raw data in a compact form to estimate probabilities at any arbitrary point in the parameter space at run-time.

2. Framework for Simulating Sphere Behavior in Single Beam OT system

We have chosen glass microspheres to illustrate our approach for the following reasons. Firstly, the physics of trapping glass spheres is well understood using a ray optics model and considerable literature exists, which discusses this force model and provides experimental validation [1,7,8]. Secondly, glass spheres can be reliably trapped and are often used as probes in many biological applications.

A computational framework has been developed using the C++ programming language to simulate the physical behavior of a sphere in a single beam optical tweezers system. Laser beam be kept either stationary or moved with a constant speed in the horizontal plane or along the vertical axis. As shown in Fig. 1, our coordinate system is oriented such that the positive Z-axis points vertically downwards and coincides with the beam propagation direction.

2.1. Simulation of Sphere Motion

A particle moving with a velocity *V, which is a function of time,* in fluid experiences a rapidly fluctuating force (due to a large number of collisions with the surrounding liquid molecules), as well as a hydrodynamic drag force. These forces are intimately related with each other and are modeled using Langevin's equation given by Equation (1) [9].

$$\frac{dV(t)}{dt} = -\frac{\gamma}{m}V(t) + \frac{\xi}{m}\Gamma(t) \qquad (1)$$

This equation assumes a fluid with viscosity η as a function of temperature T (η=1.002 x 10^{-3} Pa-

s at T=293 K for water). The drag coefficient γ for a spherical particle is given by Stokes' law as $6\pi\eta R_a$, where R_a is the radius of the spherical particle. The scaling constant $\xi = \sqrt{2\gamma K_B T}$, where K_B is Boltzmann's constant, is obtained by imposing the requirements of the fluctuation-dissipation theorem [9]. The presence of the stochastic force term $\Gamma(t)$ prevents direct analytical solution of this equation. Therefore, we first need to express Langevin's equation in finite difference form before proceeding to integrate it numerically. The difference form of this equation is given by Equation (2), where the stochastic term $\Gamma(t)$ is replaced with a standard normal distribution and the scaling constant includes the time-step δt. The external force term, F_{Ext}, allows us to include optical trapping, gravitational, and buoyancy forces. The latter two forces are constant for a sphere of a given size.

$$\frac{V(t+\delta t)-V(t)}{\delta t} = A(t+\delta t) = -\frac{\gamma}{m}V(t) + \frac{1}{m}\sqrt{\frac{2\gamma K_B T}{\delta t}}N(0,1) + \frac{F_{Ext}}{m} \qquad (2)$$

The optical trapping forces are calculated by numerically integrating the force equations given in [1]. We choose the power of the trapping laser, P to be 100 mW, with wavelength of the incident light, λ = 500 nm. The speed of light in free space is, $c = 3\text{x}10^8$ m/s, the refractive index of water n_1=1.33, the refractive index of glass n_2=1.515 and the numerical aperture (NA) of the objective is 1.2. The density of glass is ρ_s = 2600 kg/m^3, that of water is ρ_f = 1000 kg/m^3, and the acceleration due to gravity g = 9.81 m/s^2. Fig. 2 and 3 show the trapping forces along the axial and transverse directions for a 5.0 μm radius sphere that is trapped offset from the trap focus. Fig. 2 shows that for axial displacement the transverse forces are negligible and the axial forces have opposite signs above and below the trap focus. Moreover, the axial forces acting on a sphere are much stronger when the center of the sphere is above the trap focus rather than below it. Fig. 3 reveals that the trapping forces are much stronger for transverse displacements than for axial displacements and the transverse forces are anti-symmetric about the trap focus while the axial forces are symmetric. We find that the two Figures agree qualitatively and quantitatively with the data published in the literature [1,8]. In order to minimize the computation time, we generate a lookup table of the trapping forces with a uniform grid spacing of 0.5 μm for a given sphere size. Bilinear interpolation is then used to recover the optical trapping force at run time.

Once we calculate the optical trapping forces, we include them in Equation (2) to calculate the acceleration, $A(t + \delta t)$ at the end of the time-step, δt. We then use the velocity form of the second order Verlet integrator [10,11] to generate a list of the particle's position and velocity at the end of each time step. The integration time-step is chosen so that it is much smaller than the characteristic time-scale of the physical process. From Equation (1), we see that the characteristic time-scale for this model is given by the relaxation time $\frac{m}{\gamma}$. Therefore, we choose the time-step such that $\delta t << \frac{m}{\gamma}$. In our simulations, we pick δt as the closest multiple of 10 μs that is smaller than the relaxation time. A particular run is terminated when one of the following conditions are satisfied: (a) the sphere is trapped, (b) the sphere goes completely outside the geometric boundary of the laser beam, or (c) the sphere reaches the bottom of the cell under the influence of gravity.

Termination criterion (a) depends on various factors. A stably trapped sphere is usually located a small distance (a few nm to several hundred nm depending on the size of the sphere) below the beam focus, where the axial component of optical trapping force exactly balances the combined effect of gravity and buoyancy. In the case of a stationary trap, the displacement from the trap center along the X and Y-axes are comparatively small, whereas for a moving beam, these displacements can be larger and depend on the direction and magnitude of the trap velocity. In order to determine whether a particle was stably trapped in all cases, we first conducted a pilot test to ascertain the approximate position a sphere trap of a given size relative to the trap, moving along a particular direction with a constant speed. This information is then utilized to quickly determine whether a sphere has been trapped. We consider the sphere to be trapped if moves in the direction of the beam, maintaining the same displacement relative to the beam focus.

Termination criterion (b) is suppressed in two cases to incorporate certain important physical effects. First, it is not used when a sphere goes outside the laser cone boundary sufficiently close to the beam focal region. This is done because such a sphere still experiences the effect of strong gradient force due to its proximity to the trap and may be eventually trapped. Second, it is not applied when a beam is moving in the horizontal (X-Y) plane. This is required due to the fact that a sphere, outside the influence of the trap cone at any point of time, can diffuse (along the Z-axis) and come under the influence of strong optical trapping forces at a later time and be trapped.

During our simulation runs, the sphere is initially placed close to the laser. These close locations ensure that the beam motion will result in the sphere boundary intersecting the laser beam boundary within one path update interval. Henceforth, we refer to this subset of the workspace as the *local workspace*. The planning time span depends upon the system setup. Based on our system setup, we have chosen this time span to be equal to 50 ms in our simulations (assuming that the beam position will be updated at a frequency of 20 Hz). The trapping probability of any sphere that lies outside the local workspace can be estimated and utilized for motion planning purposes at subsequent time steps.

Fig. 4 graphs the spatial trajectory of a 7.5 μm radius sphere that is initially placed below the focal plane inside the local workspace. Although simulation is performed at intervals of 0.02 ms (computed value of δt for 7.5 μm sphere), data generated after every 1 ms is used for plotting purposes. The laser beam focus (shown by an 'x' mark) is initially placed at the origin. It can be seen from Fig. 4 (c) that the sphere settles at a certain distance below the focus. Fig. 4(b) shows that the total displacement along the X-axis is approximately an order of magnitude smaller than the displacements along Y and Z axes. When the sphere center is not initially displaced along the X-axis, displacement occurs purely due to random Brownian motion coupled with viscous drag. However, as soon as the sphere drifts away from the origin of X-axis, strong optical trapping forces pull it back to its equilibrium position.

2.2. Generating Trapping Probability Estimates using Dynamics Simulation

Simulation is performed $N = 100$ times at every discrete point in the 4-dimensional space (3 spatial coordinates and 1 velocity component at any given time) to estimate the trapping

probability based on the number of times the sphere gets trapped by the laser beam. The reason for using only 1 velocity component has been explained later in this section. Since we wanted to restrict the error bound in probability estimate within ±0.1 with 95% confidence level, we chose *N* to be equal to 100. In any particular trial, the outcome can be either 0 (non-trapped) or 1 (trapped).

Overall, four separate sets of experiments were performed. We used two sphere sizes having radii of 5.0 μm and 7.5 μm respectively. For each sphere size, we conducted two experiments. In the first experiment, the in-plane velocity component (v_{xy}) was kept zero. In the second experiment, the out-of-plane component (v_z) was kept zero. In order to reduce the computational burden, the coordinate system is rotated so that the new *Y*-axis always coincides with the direction of the in-plane velocity vector. Thus, in any particular simulation, four parameters are present, namely the relative *x*, *y*, and *z* coordinate of the sphere with respect to the beam focus and one velocity component.

For the sake of convenience in setting up experimental conditions, the spatial 3D volume is represented in polar coordinates. Discretization is carried out at an interval of every 0.5 μm for 7.5 μm sphere and 1.0 μm for 5.0 μm sphere along the radial (r_{xy}) and axial directions (r_z) and 5° along the azimuthal direction (θ_{xy}). Since time interval δt is much smaller for 5.0 μm radius sphere, it takes much longer to complete the set of simulation experiments. Therefore, a larger spatial discretization interval is chosen for a smaller sized sphere to reduce the time required for gathering all the data. Since the diffusion length is larger for a smaller sized sphere, trapping probability estimates change over larger distances and hence, no significant disadvantage to performing the simulation at larger spatial intervals.

The expressions for maximum permissible laser speeds in the transverse and axial directions such that the optical trapping force is sufficient to overcome viscous drag are given in Equation (3).

$$v_{xy,\max} = \frac{Q_{tr,\max} n_1 P}{6c\pi\eta R_a} \text{ and } v_{z,\max} = \frac{Q_{ax,\max} n_1 P}{6c\pi\eta R_a} \tag{3}$$

The maximum transverse and axial trapping efficiencies, $Q_{tr,\max}$ and $Q_{ax,\max}$ [1] have been chosen to be 0.3 and 0.15, respectively. These values are selected based on the peak values of the axial and transverse force components in Fig. 3. Although axial forces have higher peaks as seen in Fig. 2, it is reasonable to select the trapping efficiency conservatively so that Equation (3) is applicable in all scenarios. Simulation experiments reveal that near the maximum speed, the sphere often escapes from the trap. Hence, we have further restricted the maximum velocities to 75% of the value given by Equation (3) and sampled this domain uniformly with 5 points (including zero) in order to estimate the trapping probabilities. For the out-of-plane component, both upward and downward laser movement has been studied. Combinatorial experiments have then been carried out by varying one parameter at a time and keeping all the others constant. However, this means that a large number of grid points are constructed for estimating trapping probabilities inside the local workspace. In order to obtain results in a reasonably short period of time, computationally redundant simulations are avoided based on our insight into the physical behavior of the system.

- Azimuthal variation is not considered in the experiments where the laser beam is stationary or moves along the axial direction since the *x* and *y*-components of the OT force are radially symmetric about the vertical axis.
- Data points are placed only at the top half-plane at every horizontal cross-section of the 3D volume where azimuthal variations need to be taken into account. This is because all points in the bottom half-plane will exhibit symmetry about the transverse (*Y*) axis since the laser velocity has no component along the *X*-axis.
- Simulation is not performed at all the data points lying in the top half-plane at any particular cross-section even where azimuthal variations have to be taken into account. Experiments are carried out at all the points lying along the direction of motion of the trap (*Y*-axis) to identify the domain of interest, which refers to the region along *Y*-axis within which the estimated trapping probability changes from 1 to 0 or vice versa. At the next azimuthal direction, this domain of interest is elongated by 1.5 μm at the boundary where trapping probability transitions from a fractional value to 0 or vice versa and by 3 μm where it transitions from 1 to a fractional value or vice versa. This process is repeated by obtaining the domain of interest iteratively at any θ_{xy}.

Approximately 6 weeks were required to collect data using three PCs; the first two contain Pentium 4 processors, 1 GB RAM with a clock speed of 3.6 GHz, whereas the third one has Pentium D processor, 1 GB RAM and 3.2 GHz clock speed. All the PCs run Microsoft Windows XP Professional operating system and Microsoft Visual.Studio.Net 2003 was used as the compiler in all the cases[*].

Simulation data is used to generate intuitive visualizations of the trapping probability estimates. Figures 5-8 show the probability contour plots for 7.5 μm radius sphere in different planes under the influence of a stationary or moving trap. The simulation data includes noise as a result of stochastic, thermal forces on the spheres, which introduces short-term variability in the trapping probabilities. Thus, the trapping probability at any two neighboring grid points may show a local trend that is different from the overall trend. The contour plots from the simulation have therefore been smoothed using an in-built MATLAB filter function.

Fig. 5 shows a baseline contour plot, which represents the trapping behavior of a sphere under the influence of a stationary laser beam and follow circular patterns in *XY*-plane. The plot is shown only in half-*YZ* plane, as the contours in the other half are symmetric about the *Z*-axis and similar trends can be observed in the *XZ* plane. In Fig. 5, contours meet the beam axis below the focal plane. However, this phenomenon is not seen above the focal plane. Instead, higher probability contours tend to come towards the beam axis, whereas, lower probability contours run roughly along the beam axis.

Fig. 6 and 7 illustrate the case when the laser moves along +*Y*-axis with a speed of 0.352 μm/ms, where significantly different behavior from the stationary case is observed. Fig. 6 reveals that the contours form nearly parallel bands before converging along the negative *Y*-axis of the Y-Z

[*] Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

plane. Higher probability contours occur closer to the origin where the optical trapping forces are stronger. The contours also tend to shift upwards with increasing positive values in *Y* due to the combined effect of laser motion and gravity. As the laser moves along positive *Y*-axis, spheres that are originally positioned far above the focal plane and quite far away from origin, have a higher probability of being trapped as they come under the influence of strong trapping forces. But spheres that are positioned near the focal plane are less likely to be trapped as they may fall outside the trapping region due to gravity by the time the XY location of the trap approaches their XY locations. This differential behavior at two horizontal cross-sections is clearly shown in Fig. 7. At both the cross-sections, the contours do not follow circular patterns unlike in the stationary beam case; rather they take elliptical shapes and are distorted along the direction of laser motion.

Fig. 8 shows the contour plot when the laser moves along the negative *Z*-axis with a constant speed of 0.176 μm/ms. The plot is similar to that in Fig. 8 for the stationary beam case. The only significant difference lies in the span or coverage of the contours.

3. Generating Simplified Models from Simulation Data

3.1. Selecting Model Simplification Technique

Although offline simulation has resulted in a large amount of data, it is difficult to use it directly to estimate the trapping probability in real-time. As already discussed in Section 1, a simplified model is necessary to represent this data compactly and then use it efficiently to compute the estimated trapping probability at any point in 4-dimensional parametric space. Surface fitting and interpolation are the two major classes of such models. A detailed comparison of different surface fitting techniques has been performed by Chivate et al. [13]. Similarly, a comparative review of various scattered data interpolation methods has been carried out by Amidror [14]. Many of the limitations associated with the surface fitting and interpolation techniques have been overcome using radial basis functions (RBFs). RBFs are circularly symmetric functions centered about a single point known as the mean or the center. Their advantages include their compact representation, sensitivity to local features, ability to suppress noise in data, and ability to incorporate varying level of smoothness [15]. All these characteristics make RBFs an ideal choice in our case. Not only do they allow us to preserve local features that may be present in our high-dimensional trapping probability surface, they are also robust with respect to outliers which typically occur in stochastic experiments. We have selected Gaussian RBFs for developing our simplified models [16].

Each of the four data sets obtained from simulation is separately represented and modeled using Gaussian RBFs of the following form:

$$p\left(\vec{x}\right) = w_0 + \sum_{i=1}^{m'} w_i \phi_i\left(\vec{x}\right) \tag{4}$$

where $\vec{x}$ is the position vector of a point in 4-dimensional parameter space,

$p\left(\vec{x}\right)$ is the computed trapping probability value at that point,

m' is the number of RBFs,

ϕ_i is the ith RBF,

w_0 is a constant and

w_i is the weight of the ith RBF

For a Gaussian RBFs, $\phi_i\left(\vec{x}\right) = \exp\left(-\frac{r^2}{2}\right)$, where the RBF radius, $r = \left\| \frac{\vec{x} - \vec{\mu}_i}{\vec{\sigma}_i} \right\|$. Here, $\vec{\mu}_i$ is the position vector of the i^{th} RBF center and $\vec{\sigma}_i$ represents the width of the i^{th} RBF. We may select the components of this width vector anisotropically so that it assumes different values along the different axes. However, we restrict ourselves to axis-aligned RBFs only, thereby sacrificing some generality for the sake of a more compact representation. By axis-aligned, we mean that the axes of the hyper-ellipsoidal Gaussian RBF surface are aligned along the axes of the 4-dimensional parameter space.

3.2. Fitting Gaussian RBFs to Simulation Data Sets

We have adopted and extended the method reported in [15] for this application. Specifically, Juba and Varshney have developed a method for fitting 3D volumetric data using RBFs. Our parametric space is four dimensional and involves both distance and velocity parameters. Hence, we had to develop a different spatial partitioning scheme.

We start by fitting a single RBF to a 2^4 resolution downsampled version of the given data set. This down sampling is done by obtaining all possible combinations of the minimum and maximum values of each of the four parameters (i.e., 2^4 points). This is taken as the root of our RBF hierarchy, which is represented in the form of a k-dimensional tree (kd-tree). This fitted RBF is then evaluated at all the data points in the entire parameter space. If the root mean square (RMS) error is above 2% and the maximum error is above 4%, then the parameter space is partitioned into two halves by a hyperplane that is perpendicular to the *X*-axis such that the two half-spaces have approximately same number of data points. These threshold errors are just representative values that have been chosen to evaluate the performance of this fitting procedure. Henceforth, each sub-space will be termed as a *region* that actually represents a node in the kd-tree. Thus, we can see that kd-tree is a binary tree which will be progressively constructed during the fitting operation.

We now obtain two other 2^4 resolution downsampled versions of the parameter space in the newly constructed regions by selecting points using uniform, random distribution. We then fit a new child RBF to the residual error in each of the two regions using the downsampled versions and evaluate the error using both the child as well as the parent RBF at all the data points. If both error measures lie below their respective thresholds in any of the two regions, then it is not sub-divided any further. Otherwise, that region is partitioned by a hyperplane perpendicular to the *Y*-axis, the kd-tree is updated and this process continues. If region partitioning is required at the 3rd and 4th levels of the kd-tree, then hyperplanes perpendicular to *Z* and *V*-axis are used respectively. This cycle of choice of partitioning hyperplane is then repeated for subsequent levels in the tree. Moreover, if we are working with a region where some or all of the four parameters assume boundary values that are different from the minimum or maximum values,

then one additional data point is considered from the corresponding neighboring region. This avoids noticeable discontinuities at the region boundaries.

Although down sampling could have been done again by considering minimum and maximum parameter values, we found that random sampling leads to a slightly lower number of RBFs. However, it is better to follow the former approach while obtaining the root RBF as it enables us to fit a function that can broadly span the entire parameter space. The process terminates when all the regions have been adequately fitted using varying number of RBFs such that RMS and maximum errors lie below their respective thresholds in each and every one of them. It may be argued that the overall process can be sped up by using a hex tree (analogous to octree in 3D) decomposition method as this directly splits up the space into 16 sub-spaces in one run. However, a kd-tree is a better choice in this case because of the differences in dimensions of the parameters along the 4 axes (3 are distances and 4th is velocity). It also provides a convenient way of discretizing the overall space into sub-spaces of roughly equal cardinality by forcing the partitioning plane to pass through the median value of the corresponding parameter. Moreover, any dimensionality of the parametric space can be easily handled by the kd-tree. This only entails increasing the number of partitioning planes (equal to *k*) to a higher value. Thus, kd-tree based partitioning offers lot of flexibility in incorporating additional parameters in future applications.

Therefore, obtaining this compact model reduces to a careful selection of the constant term w_0, RBF weights, RBF centers, and RBF widths. We have selected w_0 as zero for all results reported in this work, therefore, this term is neglected from now on. Selecting a non-zero value of w_0 increases computational overhead significantly without significantly improving the accuracy. The initial guess value of the center of the newly created RBF is taken as the position vector of the data point that has the maximum residual approximation error. The RBF's weight is set as the RMS error value of the region under consideration. This can be rationalized by taking note of the fact that this causes the center data point to have zero error since Gaussian RBFs have a value of 1 at their centers.

The anisotropic widths are selected using maximum likelihood estimation (MLE) scheme which assumes that the data to be fitted to a particular RBF is a histogram of samples from a Gaussian (normal) probability density function with the previously selected mean or center. Then the width or spread correlates to its standard deviation and is computed such that this set of samples will have the highest probability of being generated. Since our data values (trapping probabilities) are estimated based upon a set of 100 simulation runs, it is reasonable to adopt this approach in the current work. The *x*-component of the *i*th RBF's width vector (σ_{xi}^2) is calculated using Equation (5) below. Other components can be calculated similarly.

$$\sigma_{xi}^2 = \frac{\sum_{j=1}^{n} p_j}{2\sum_{j=1}^{n}\left(x_j - \frac{\sum_{j=1}^{n} x_j p_j}{\sum_{j=1}^{n} p_j}\right) p_j} \tag{5}$$

where n is the number of data points that are being approximated by the ith RBF and p_j is the estimated trapping probability of the jth data point.

Once the RBF center and all the widths have been obtained, improved values of the weights are computed by minimizing the sum of squared errors

$$\psi = \sum_{j=1}^{n} \left[\sum_{i=1}^{m'} w_i \phi_i - p_j \right]^2 \tag{6}$$

This is equivalent to solving the following system of linear equations in the least-squares sense, as usually $m' < n$.

$$\begin{bmatrix} \phi_1\left(\vec{x}_1\right) & \cdots & \phi_{m'}\left(\vec{x}_1\right) \\ \vdots & \ddots & \vdots \\ \phi_1\left(\vec{x}_n\right) & \cdots & \phi_{m'}\left(\vec{x}_n\right) \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_{m'} \end{bmatrix} = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} \tag{7}$$

The entire method has been implemented in MATLAB. All the steps are formally described in the algorithm FIT-GAUSSIAN-RBFs and the performance of this algorithm on all the four data sets has been shown in Table 1. The first two data sets correspond to 7.5 μm radius sphere, while the other two correspond to 5.0 μm radius sphere. Odd numbered data sets deal with the cases where horizontal component of the laser velocity is varying, while even numbered data sets capture the effect of out-of-plane laser velocity. Even if simulation data is gathered only in the top-half cross-sectional plane or at few locations using adaptive sampling, probability estimates at other data points are stored by copying values suitably using a pre-processor program.

Algorithm: FIT-GAUSSIAN-RBFs

Input: A set of M data points in 4-dimensional parameter space $D = \{d_1, \ldots, d_M\}$, with an estimated trapping probability p_i associated with every point d_i, $i = 1, \ldots, M$. The parameters associated every data point include x, y, z-coordinates of the initial location of sphere center under consideration and either in-plane or out of plane component of laser velocity. This can be represented by a 4-tuple (x_i, y_i, z_i, v_i).

Output: An incrementally constructed partial kd-tree K that represents the input data concisely using M' number of Gaussian RBFs (same as the total number of tree nodes), such that every node corresponding to a region in 4D parametric space is fitted by a single RBF.

Steps:

1. Obtain a 2^4 resolution downsampled version from D by obtaining all the points whose 4-tuple is formed by some combination of minimum and/or maximum values of the parameters. These points represent the boundary points of the local workspace.
2. Fit a Gaussian RBF to this downsampled data set using the function FIT-SINGLE-RBF and store it as the root node of previously empty K.

3. Estimate the RMS error of approximating the entire parameter space consisting of M data points by the single Gaussian RBF $e_{RMS} = \sqrt{\frac{\sum_{i=1}^{M}\left(p_i - p_i^{/}\right)^2}{M}}$, where $p_i^{/}$ is the reconstructed trapping probability value using the fitted RBF at the i^{th} data point.
4. Compute the maximum error in approximating the parameter space by the same Gaussian RBF $e_{Max} = \max\left(\left|p_i - p_i^{/}\right|\right), \forall i = 1,...,M$.
5. **If** $e_{RMS} > 2\%$ **or** $e_{Max} > 4\%$, then proceed to step 6; **else** output K and terminate.
6. Initialize current level (l) to be equal to 1, current parent node (n_c) to be equal to *NULL* and current region (r_c) as the entire parameter space.
7. **While** $e_{RMS} > 2\%$ **or** $e_{Max} > 4\%$ in r_c **and** r_c contains at least 2 data points, **do**
    i. Partition r_c into two halves by a plane perpendicular to one of the axes such that each half-plane roughly contains the same number of points. Choice of this axis depends on the level l. If l belongs to the sequence {1, 5, …} select X-axis; if it is a part of the sequence {2, 6, …} select Y-axis; if it belongs to {3, 7, …} choose Z-axis and if it is a multiple of 4, then choose V-axis. Store the two half-planes as children nodes of the root of K. By pre-sorting the original data set D along each of the 4 dimensions, median coordinate along the corresponding dimension is used to split the current data set.
    ii. Increment l by 1 and update n_c as the node corresponding to the current region and r_c as the left child region.
    iii. Obtain a 2^4 resolution downsampled version from the points stored in r_c by random selection.
    iv. Search whether any of the 4 parameters have a boundary value that is different from its minimum and maximum values. **If** yes, then add another randomly selected point from the neighboring region that shares a common partitioning hyperplane perpendicular to that parameter axis to the downsampled version.
    v. Fit a new Gaussian RBF to r_c by calling the function FIT-SINGLE-RBF.
    vi. Estimate e_{RMS} in approximating p_i values in *all* the data points present in the left child region using the newly fitted RBF as well as all of its parent or ancestral RBFs spanning up to the root node from current level l. Similarly estimate e_{Max} as well.
8. **If** $e_{RMS} < 2\%$ **and** $e_{Max} < 4\%$ in all the K regions, then output K and terminate. **Else** go to next step.
9. **If** node corresponding to r_c is the left child of n_c, then update r_c as the right child region and go to step 7 iii and continue recursively inside the while loop. However, **if** the corresponding node is the right child of n_c, then traverse backwards in K until we reach a node, such that the previously visited node is its left child. Store this node value as n_c, update r_c as the right child region of n_c and set l as the tree level value of right child. Then go to step 7 iii.

Function: FIT-SINGLE-RBF

Input: A downsampled set of data points $D^{/}$ present in a particular K node region, such that $D^{/} \subseteq D$, and corresponding $p_i^{/}$ values.

Output: A newly constructed Gaussian RBF that approximates p_i values for all the points present in D' and updated p_i' values.

Steps:

1. Perform a linear search to obtain the data point belonging to set D' that has maximum *residual* error using the expression $e_{Max} = \max\left(\left|p_i - p_i'\right|\right), \forall i = 1,...,n'$, where n' is the cardinality of D'. Select this data point as the RBF center. **If** we are constructing the root RBF, then the geometric center corresponding to the mean value of each of the 4 parameters is selected.
2. Choose anisotropic widths for the Gaussian RBF using Equation (5).
3. Represent the RBF using Equation (4), with the constant term always set to zero and neglect the weight temporarily to obtain updated e_{RMS} only for this downsampled data set D'. Updated e_{RMS} is computed by modifying p_i' to incorporate the effect of the new RBF. Set the weight to be equal to e_{RMS}.
4. Obtain the final value of weight by performing a least squares error fitting over the data set D' using Equations (6) and (7).

We found that maximum errors typically occur repetitively in two local zones – a) while fitting an RBF to a kd-tree region that has been formed by partitioning parent region based on velocity parameter and b) while fitting to a region created by partitioning based on *z*-coordinate value above the focal plane only when in-plane velocity is taken into account. The former scenario arises in all the 4 data sets due to the presence of only 5 discretization levels in the velocity parameters. However, the latter scenario occurs exclusively in odd numbered data sets due to the differential trapping behavior at various cross-sections explained using Fig. 7. The number of fitted RBFs (and hence the fitting time) depends on the size of data set, the size of the sphere as well as which velocity component is considered. In general, it is lower when the data set size is smaller and out-of-plane velocity is considered (lesser features observable in the contour plots as compared to the in-plane velocity cases). However, proportionately it is slightly higher for the smaller sized sphere as data points are spread apart.

Table 1: Results from fitting Gaussian RBFs to 4 different simulation data sets

Data set number	Number of data points (millions)	Fitting time (sec)	Number of fitted RBFs (thousands)	RMS error (%)	Maximum error (%)
1	3.441	263	19.41	1.75	3.97
2	0.036	1.5	0.08	1.97	3.23
3	1.058	112	8.54	1.76	3.98
4	0.007	0.4	0.02	1.98	3.25

3.3. Querying kd-Tree to Compute Estimated Trapping Probability Values

Once all the data sets have been modeled, it is straightforward to compute the estimated trapping probability value in real-time at any point in the parameter space. The point of interest need not coincide with any of the grid or data points because we may be interested in computing the probability value at any arbitrary location in the parameter space. Summing the values of all RBFs that overlap at the point of interest results in the probability estimation and the steps to perform this operation are outlined below in the algorithm OBTAIN-PROBABILITY-VALUE.

Algorithm: OBTAIN-PROBABILITY-VALUE

Input:

- A point in local workspace with fully specified 4-tuple $(x_{\text{int}}, y_{\text{int}}, z_{\text{int}}, v_{\text{int}})$ at which we are interested in computing the trapping probability.
- kd-tree K obtained as output of the algorithm FIT-GAUSSIAN-RBFs

Output:

- Estimated trapping probability value p_{est}.

Steps:

1. Initialize node (n_e) and region (r_e) that are being currently explored as the root of K and the entire parameter space respectively.
2. Initialize a list of explored kd-tree nodes L as an empty set.
3. **While** p_{est} is not computed **or** both children of n_e have not been explored, **do**
   i. Insert node n_e in list L.
   ii. Perform a linear search of all the data points stored in r_e to ascertain whether the given point lies inside r_e based on all the values specified in the 4-tuple.
   iii. **If** yes, then go to left child of n_e (if one exists) and update n_e and r_e accordingly. However, **if** n_e is a leaf node, then carry out a backward traversal of K from n_e to root node to identify all the parent RBFs and compute p_{est} using Equation (4).
   iv. **Else** backtrack to nearest unexplored node in K (present in the set of nodes of K but not in L) as is done in any *depth-first* tree traversal, update n_e and r_e and continue inside the loop recursively.
4. Output p_{est}.

One thousand points are randomly chosen from each of the four parametric spaces such that none of the points coincides with any of the data points that were used to fit the Gaussian RBFs. Simulation experiment is then carried out 100 times at each such point and the trapping probability is estimated. The query algorithm is then used to compute trapping probabilities at all those selected points. The performance of the algorithm (and hence that of our overall fitting technique) in terms of both computational speed as well as accuracy is shown in Table 2. Timing data clearly reveals the significance of developing a simplified model and the impracticality of using offline simulation for real-time motion planning. As expected, the error measures are slightly higher than the corresponding numbers in Table 1 because those values are recorded based on grid points, whereas these points never coincide with any point that has been used to fit RBFs. However, none of the error values is markedly high which shows the efficacy of our fitting technique.

Table 2: Performance of query function to estimate trapping probability at 1000 randomly selected points in 4 parametric spaces

Data set number	Overall simulation time (hours)	Overall query time (ms)	RMS error (%)	Maximum error (%)
1	5.86	38	1.81	4.09
2	5.41	2.5	2.04	3.34
3	12.27	23	1.83	4.22
4	11.53	1.6	2.05	3.40

In order to transport one or more spheres from a given initial location to target location, we need to develop intelligent motion planning strategies that can perform this operation without any manual intervention in the least possible time. This requires avoiding collisions with other spheres present in the workspace, which often results in losing the trapped particle. In the event of a collision with another particle in the workspace, the laser must be moved to a location where the particle is trapped again, so it can be moved to its final assembly location.

Run-time knowledge of the trapping probability in a region around the beam focus at any permissible laser speed will help us quickly predict possible collisions. Once these spheres are identified, multiple traps can be switched on so that they can trap those spheres and either hold them stationary or move them away from the spheres that are being transported. Alternatively, the controller can command the currently operational laser beams to go along different paths at a faster or slower rate to circumvent the obstructing spheres. The trapping probability information allows us greater flexibility in positioning the trap and greatly improves the chances of successfully trapping a sphere anywhere in the workspace for any combination of position and velocity parameters. This in turn will help us develop more robust and efficient algorithms.

4. Conclusions

Since it is very challenging to use real-time simulations in automated planning applications, due to extremely long computation times, we have developed a systematic approach to generate simplified trapping probability models based upon offline simulation data. The main contributions of this work are summarized as follows:

- The trapping behavior of a dielectric sphere in a spatial region close to the focus of a stationary laser beam has been presented and quantified in terms of trapping probability contours, estimated by performing several simulations across all parameters of interest, namely position and velocity.
- The trapping behavior of a microsphere trapped in a moving laser beam is also studied. Uniform motion of the trap either along the horizontal plane or along the beam axis are considered separately and trapping probability contours from each case reveal interesting trends that show distortions probability plots from the static case.
- A model simplification technique for fast and accurate online computation of trapping probability estimates at any arbitrary point in the 3D space and for any value of laser speed

using Gaussian radial basis functions has been developed. This is a very useful tool in automated path planning as it enables us to efficiently maneuver components across a workspace in real-time while avoiding potential obstacles.

The trapping force models used in this work are based on a geometric optics approach and are valid for dielectric microspheres ranging in radius between 1 μm and 10 μm. However, the model simplification techniques presented in this work are very flexible and are generally applicable to other trapping force models as well. As part of future work, we plan to utilize these trapping probability estimation methods to develop real-time motion planning algorithms not just for microspheres, but also for metallic nanoparticles as well as nanowires by substituting appropriate force models for these components.

Acknowledgements. This work was supported in part by the Center for Nano Manufacturing and Metrology, a joint venture between the University of Maryland and the National Institute of Standards and Technology.

References

[1] Ashkin, A., 1992, "Forces of a Single-Beam Gradient Laser Trap on a Dielectric Sphere in the Ray Optics Regime," Biophysical Journal, **61**(2), pp. 569-582.

[2] Ashkin, A., 2000, "History of Optical Trapping and Manipulation of Small-Neutral Particle, Atoms, and Molecules," IEEE Journal of Selected Topics in Quantum Electronics, **6**(6), pp. 841-856.

[3] Svoboda, K., Block, S., 1994, "Optical Trapping of Metallic Rayleigh Particles," Optics Letters, **19**(13), pp. 930-932.

[4] Balijepalli, A., LeBrun, T., Gagnon, C., Lee, Y. G., Dagalakis, N., 2005, "A Modular System Architecture for Agile Assembly of Nanocomponents using Optical Tweezers," In Proceedings of the 2005 SPIE Conference on Optics and Photonics, San Diego, CA, **5908**.

[5] Grier, D. G., 2003, "A Revolution in Optical Manipulation," Nature**, 424**(6950), pp. 810-816.

[6] Peng, T., Balijepalli, A., Gupta, S. K., LeBrun, T., 2007, "Algorithms for On-Line Monitoring of Micro Spheres in an Optical Tweezers-Based Assembly Cell," Journal of Computing and Information Science in Engineering, **7**(4), pp. 334-338.

[7] Wright, W. H., Sonek, G. J., 1993, "Radiation Trapping Forces on Microspheres with Optical Tweezers," Applied Physics Letters, **63**(6), pp. 715-717.

[8] Wright, W. H., Sonek, G. J., Berns, M. W., 1994, "Parametric Study of the Forces on Microspheres Held by Optical Tweezers," Applied Optics, **33**(9), pp. 1735-1748.

[9] Weissbluth, M., 1989, Photon-Atom Interactions, Academic Press, Boston, MA.

[10] Allen, M. P., Tildesley, D. J., 1987, Computer Simulation of Liquids, Clarendon Press, New York, NY.

[11] Swope, W. C., Andersen, H. C., Berens, P. H., Wilson, K. R. A Computer Simulation Method for the Calculation of Equilibrium Constants for the Formation of Physical Clusters of Molecules: Application to Small Water Clusters. The Journal of Chemical Physics, **76**(1):637-49, 1982.

[12] Montgomery, D. C, Runger, G. C., 1994, Applied Statistics and Probability for Engineers, John Wiley & Sons, Inc., New York, NY.

[13] Chivate, P. N., Jablokow, A. G., 1995, "Review of Surface Representation and Fitting for Reverse Engineering," Computer Integrated Manufacturing Systems, **8**(3), pp. 193-204.

[14] Amidror, I., 2002, "Scattered Data Interpolation Methods for Electronic Imaging Systems: a Survey," Journal of Electronic Imaging, **11**(2), pp. 157-176.

[15] Juba, D., Varshney, A., 2007, "Modelling and Rendering Large Volume Data with Gaussian Radial Basis Functions," Technical Report, Number UMIACS-TR-2007-22, University of Maryland, College Park, MD.

[16] Weiler, M., Botchen, R., Stegmaier, S., Ertl, T., Jang, Y., Ebert, D. S., Gaither, K. P., 2005, "Hardware-Assisted Feature Analysis and Visualization of Procedurally Encoded Multifield Volumetric Data," IEEE Computer Graphics and Applications, **25**(5), pp. 72-81.

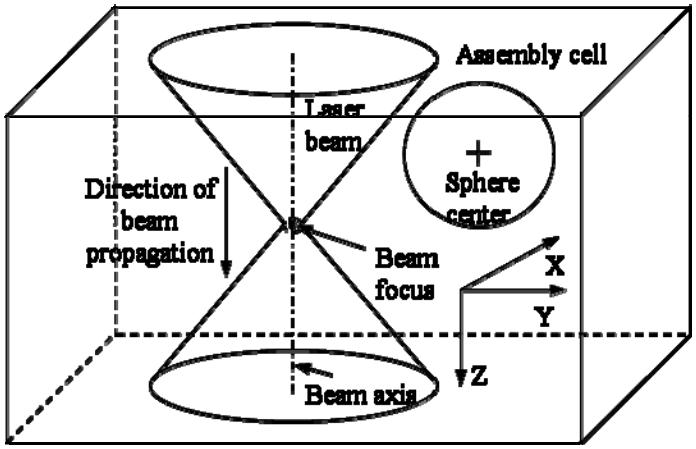


Fig. 1. Schematic illustration of simulation set-up

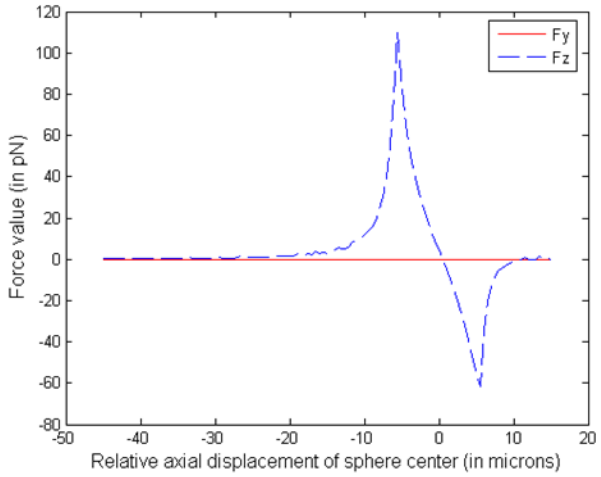


Fig. 2. Axial and transverse force components when sphere center is displaced along the laser beam axis from the focus

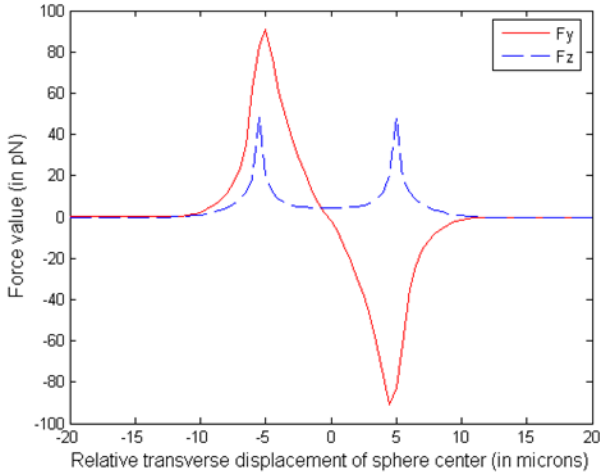
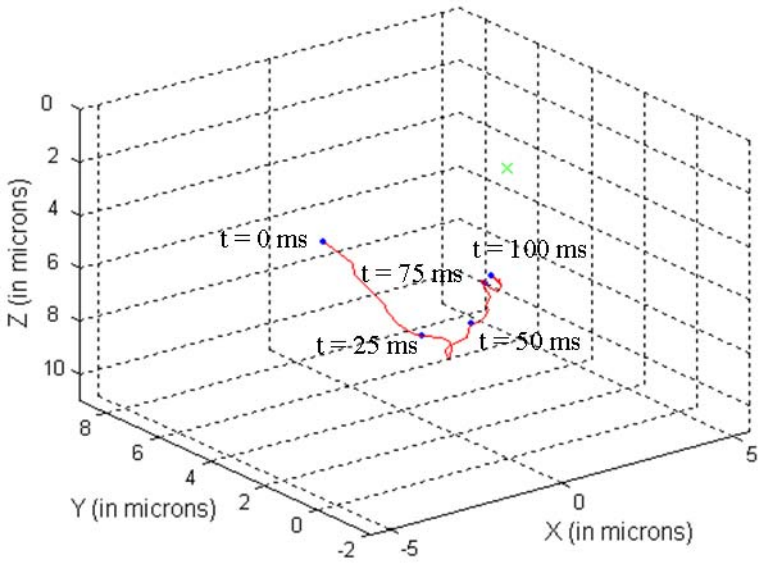
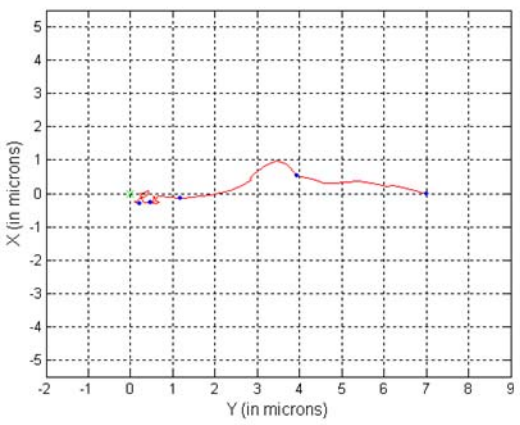


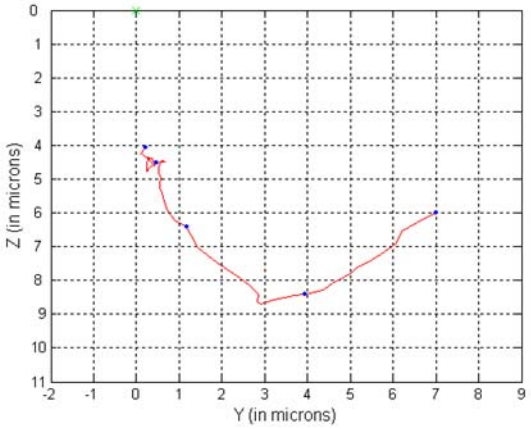
Fig. 3. Axial and transverse force components when sphere center is displaced along the transverse axis from the laser beam focus



(a) Sphere starts below the focal plane at (0 μm, 7 μm, 6 μm); laser beam **stationary with focus at the origin (3D view)**



(b) Above trajectory in XY plane



(c) Above trajectory in YZ plane

Fig. 4. Trajectory of sphere that is trapped by a stationary laser beam

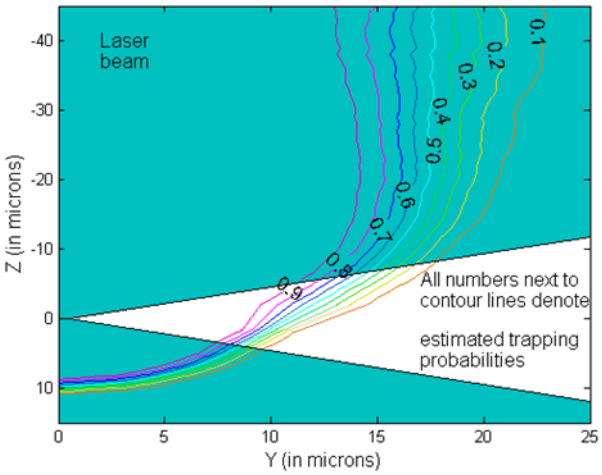


Fig. 5. Estimated trapping probability contours for 7.5 µm radius sphere under influence of a stationary laser beam

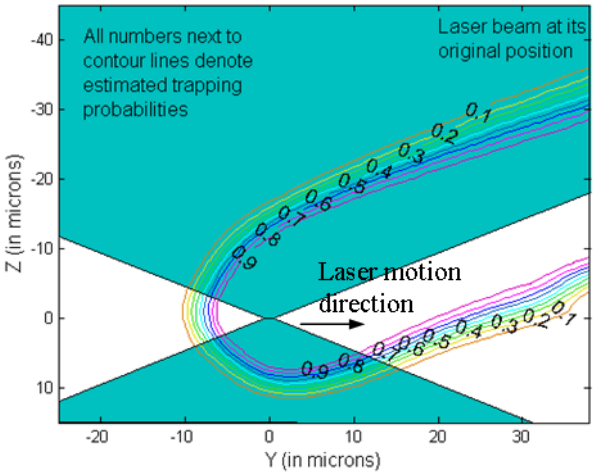
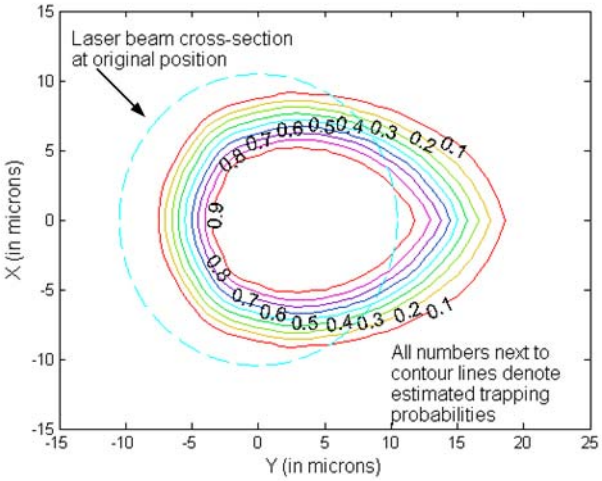
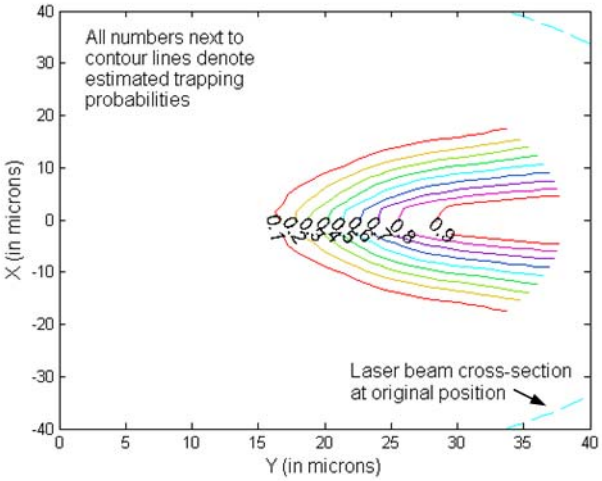


Fig. 6. Estimated trapping probability contours for 7.5 µm radius sphere in Y-Z plane under influence of a laser beam moving along +Y-axis with a speed of 0.352 µm/ms



(a) Plot at z = 5 μm.



(b) Plot at z = -25 μm.

Fig. 7. Estimated trapping probability contours for 7.5 μm radius sphere in X-Y plane under influence of a laser beam moving along +Y-axis with a speed of 0.352 μm/ms

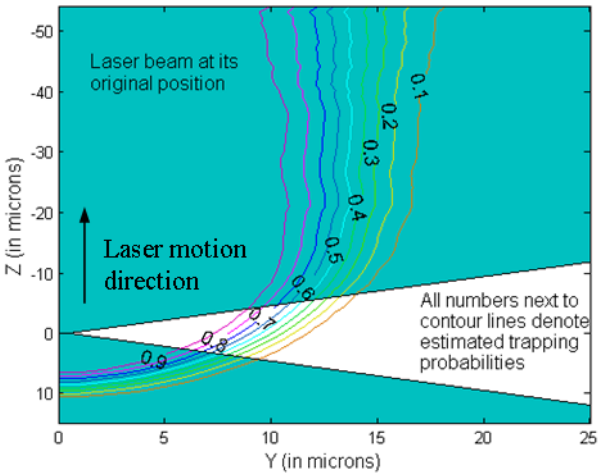


Fig. 8. Estimated trapping probability contours for 7.5 µm radius sphere in Y-Z plane under influence of a laser beam moving along -Z-axis with a speed of 0.176 µm/ms