SI Appendix: Convolutional neural networks automate detection for tracking of submicron scale particles in 2D and 3D

Jay M. Newby* Alison M. Schaefer† Phoebe T. Lee‡
M. Gregory Forest* Samuel K. Lai,3†

May 11, 2018

SI video1 (tracking on experimental videos)

This video is included to illustrate neural network tracking results for experimental test videos, showing a range of challenging conditions. The tracking indicator overlays (red) show locations of particle centers recognized by the neural network tracker.

SI video2 (tracking on synthetic test videos)

This video is included to illustrate neural network tracking results for synthetic test videos. The tracking indicator overlays (red) show locations of particle centers recognized by the neural network tracker.

Simulated videos

The goal of our simulated videos is to approximate the appearance of real videos for training and testing. *These videos are not intended to be accurate simulations of particle videos rooted in optical physics.* As such, they do not expose physical parameters like wavelength, pixel size, refractive index, numerical apature, etc. Instead, we postulate a general form that approximates the shape, with a number of parameters that we can use to randomize over a wide range of possible conditions. The goal is for the neural network to recognize patterns, independent of the precise details of the optics, particles, and camera used.

Given a particle located at $\xi = (0,0,0)$, the observed particle point spread function (PSF) used to generate simulated videos is given by

$$\psi(x,y,z) = I_1(1+0.1(2h_1-1))\left(1-\gamma\left|\tanh(\frac{z}{z_*})\right|\right)\left\{2\exp(-\frac{r^4}{64a^2}) + (1-h_2^4)\left[\exp(-\frac{(r-z)^4}{a^4}) + 0.75h_3\chi[r<z]\sin^2(\left(\frac{\pi r}{z_*}\right)^{3/2})\right]\right\}, \quad (1)$$

where $r = \sqrt{x^2+y^2}$. Here, $I_1 > 0$ sets the intensity scale, z_* determines how the PSF fades as the particle moves in z, and a determines the PSF radius scale. The parameters h_j, $j = 1,2,3$, are values between zero and one, and are intended to randomize the PSF shape and appearance.

*Department of Mathematics and Applied Physical Sciences, University of North Carolina–Chapel Hill, Chapel Hill, NC 27599
†Division of Pharmacoengineering and Molecular Pharmaceutics, Echelon School of Pharmacy, University of North Carolina–Chapel Hill, Chapel Hill, NC 27599
‡UNC-NCSU Joint Department of Biomedical Engineering, University of North Carolina–Chapel Hill, Chapel Hill, NC 27599

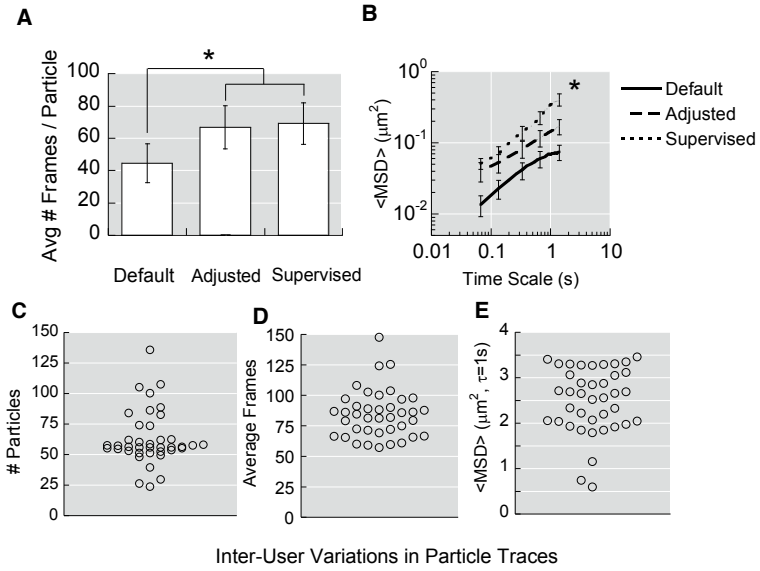

Figure 1: **The need for supervision in particle tracking, and inter-user variations in supervised tracking data.** Data represents the average of 4 movies of muco-inert 200 nm PEGylated polystyrene beads in human cervicovaginal mucus. Data from human supervised tracking (Supervised), which includes manually inspecting paths to remove false positives and minimize false negatives, is compared to results generated under default conditions of the tracking software (Default) and conditions manually adjusted by the user (Adj) to improve tracking accuracy. (A) average frames per particle; (B) ensemble-averaged geometric mean square displacements ($\langle \text{MSD} \rangle$) vs. time scale. Error bars represent standard error of the mean. * indicates statistically significant difference compared to 'Standard' ($p < 0.05$). (C-E) Inter-user variations in particle tracking. Different tracking software users were asked to analyze the same video of 200 nm bead in human cervicovaginal mucus. (A) Total particles tracked; (B) average frames tracked per particle; and (C) ensemble-averaged geometric mean square displacements ($\langle \text{MSD} \rangle$) at a time scale ($\tau$) of 1s.

A number N of random Brownian particle paths $(x_n(t), y_n(t), z_n(t))$ are generated (using Euler's method) to serve as ground truth. Then, the image volume at time t is given by

$$I(x,y,z,t) = \sum_{n=1}^{N} \psi(x - x_n(t), y - y_n(t), z - z_n(t)) + B(x,y,z) + \kappa\Theta(x,y,z,t), \tag{2}$$

where $B(x,y,z)$ is a random background intensity, $\kappa > 0$ scales the noise, and $\Theta(x,y,z,t)$ is comprised of i.i.d normal random variables with mean zero and unit variance. Note that after generating the video using (2), we rounded the output to the nearest integer to more closely represent the integer valued image data most often encountered in experiments. For randomized background we used

$$B(x,y) = I_{\text{back}} I_1 \sin\left(\frac{6\pi}{N_x}\sqrt{g_1(x - g_2 N_x)^2 + g_3(y - g_4 N_y)^2}\right), \tag{3}$$

where I_{back} scales the background intensity relative to the PSF and g_j are uniform random variables.

Neural network architecture

Let the video to be processed by the network be given by $I(x,y,z,t)$, where each dimensional variable is interpreted as indexing discrete pixels (for x, y), slices (for z), and frames (for t). The video dimensions are (N_y, N_x, N_z, N_t).

The CNN input is a single image frame from a video:

$$\text{Input} = I(\cdot, \cdot, z, t), \tag{4}$$

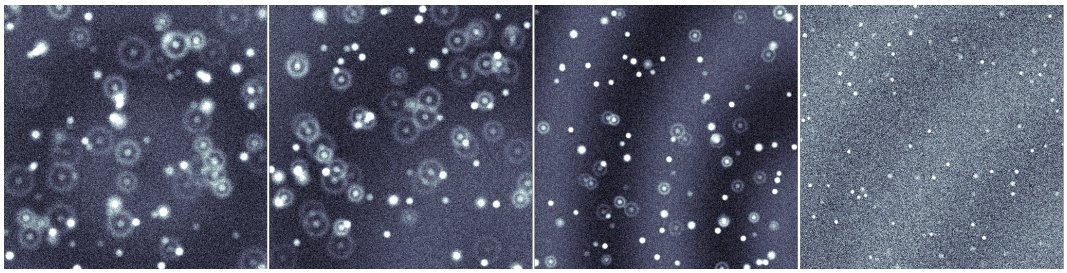

Figure 2: Sample frames from four different synthetic test videos.

for fixed z-axis slice. The input is normalized in order to cope a wide range of possible image intensity values. The image frame $I(\cdot, \cdot, z, t)$ is normalized to have zero mean and unit variance.

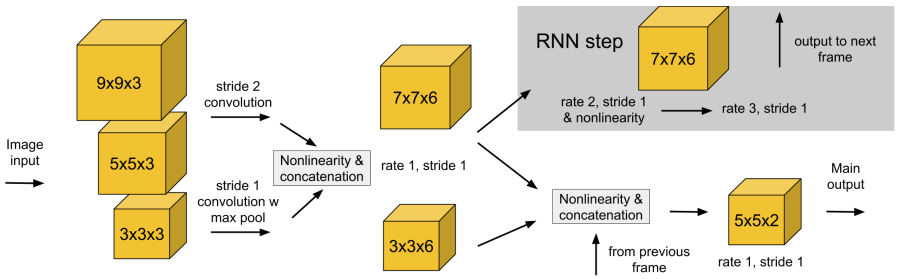

Figure 3: The network architecture of the neural network tracker.

The architecture of the neural network (see Fig. 3) was designed to manage the number of computationally expensive elements, while maintaining prediction accuracy. We used the fully-convolutional segmentation network in Ref. [1] as a starting point for our design. Our first priority was accuracy, followed by evaluation speed. Evaluation time is largely taken up by convolutions. Some remaining constraints we considered were training speed, and memory usage.

The CNN is comprised of three convolutional layers and one recurrent layer. All of the convolution kernels are 4-dimensional arrays whose values are trainable parameters. The sizes of the kernels used for each layer are

- Layer 1: $(9, 9, 1, 3)$, $(5, 5, 1, 3)$, and $(3, 3, 1, 3)$ mapping input images to 3 features each (9 total)
- Layer 2: $(7, 7, 9, 6)$, $(3, 3, 9, 6)$ mapping 9 features to 6 features each (12 total)
- Layer RNN: $(7, 7, 6, 6)$ mapping 6 features to 6 features (this kernel is applied twice successively to the output of kernel 1 of layer 2)
- Layer 3.: $(5, 5, 18, 2)$ mapping 18 features to the final two output log likelihoods (bilinear interpolation is used to upsample to the original image resolution)

The first layer kernels are applied with a stride of two pixels (except for the kernel 3, to which max pooling is applied) so that the layer 1 output has half the x, y resolution as the input (i.e., the layer 2 input tensor has size $(N_y/2, N_x/2, 9)$). In order to maintain a large receptive field with as few trainable parameters as possible, layers 2 and the RNN layer use atrous convolution with a rate of 2 (i.e., they are applied with a stride of 1, but the convolution kernel is applied to a downsampled local patch of the input). The output of the RNN layer is carried forward to the next frame $(t + 1)$ and concatenated with the

output of layer 2 of frame $(t+1)$. The combined 18 features are input into layer 3. Bilinear interpolation is applied after layer 3 to resample the image to its original resolution.

Nonlinearities are applied after each convolution, using

$$\text{output} = F(\sum_{x',y'} K(x',y')\text{input}(x+x',y+y') - b), \quad F(u) = \log(e^u + 1). \tag{5}$$

Each layer has a separate trainable bias b for each output feature.

Let the output of the interpolation layer be denoted as $L_n(x,y)$ for $n = 0, 1$. These outputs are regarded as log likelihoods, at pixel position x, y, for background ($n = 0$) and the presence of a nearby particle ($n = 1$). The final output of the network is the detection probabilities

$$p(x,y,z,t) = \frac{e^{L_1(x,y)}}{e^{L_0(x,y)} + e^{L_1(x,y)}}, \tag{6}$$

Hence, the neural network output, after processing a full video, has the same size and dimension as the video (it may take up more memory since each element is a 32 bit floating point number and videos are typically comprised of 16 bit integers).

Neural network training

Cross entropy is (up to an additive constant that depends on p) a measure of how far the approximated distribution q is from the true distribution p. When $q = p$, the cross entropy reduces to the entropy of the true distribution p. Since p never changes for a given training video, our goal is to minimize $H[p,q]$ with respect to q over the entire training set of videos. At each iteration of the training procedure, a randomly generated training image is processed by the network, the error $H[p,q]$ is computed, and all of the trainable parameters are altered by a small amount (using the gradient decent method explained below) to reduce the observed error. This training procedure is repeated thousands of times until the error is minimized.

Suppose that all of the trainable parameters are arranged into the vector $\boldsymbol{\theta}$. The parameters are adjusted at the end of each training iteration t by computing the gradient of $\mathbf{g}_t = \nabla_\theta H[p_t, q_t]$. The gradient vector points in the direction of steepest rate of increase in the error, so the error can be reduced with $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - r\mathbf{g}_t$, where $r > 0$ is a predefined step size.

Generation of training images was performed in Python and training of the neural network was performed using Google's open source software package, Tensorflow [2]. Training was performed using stochastic gradient descent, with learning rate 0.16. The learning rate was decayed exponentially with decay factor 0.95. Each iteration of training processed a full 256×256 resolution frame from a randomly generated synthetic video, each of which was used for no more than two training iterations. The training was stopped at 100,000 training iterations.

After training, the neural network is deployed using Tensorflow, which executes the most computationally costly elements of the neural net tracker in highly optimized C++ code. Tensorflow can be easily adapted to use multiple cores of a CPU or GPU, depending on available hardware.

Particle path linking

From the neural net output, we extract candidate particles along with their probabilities through thresholding the detection probabilities q_{ijk}, where ijk are the indices for each pixel of a single video frame. The threshold of $q = 0.5$ represents a maximum likelihood classification: everything above $q = 0.5$ represents pixels corresponding to the presence of a nearby particle, and everything below this threshold is most likely part of the image background. The pixels above threshold are grouped into candidate particles using the method of connected components [3]. Connected sets of nearest neighbor pixels $\mathcal{P}_n$ above the threshold are collected as candidate particles. That is, $\mathcal{P}_n$ is a connected set and $q_{ijk} \geq 0.5$ for all $q_{ijk} \in \mathcal{P}_n$.

Each candidate particle is assigned the largest detection probability from its constituent detection probabilities within the connected component, i.e., $\rho_n = \max \mathcal{P}_n$. The position of

each candidate particle is taken to be the center of mass given by, $\mathbf{x}_n = \frac{\sum_{q_{ijk} \in \mathcal{P}_n} (j,i,k) q_{ijk}}{\sum_{q_{ijk} \in \mathcal{P}_n} p_{ijk}}$. Note that there are alternative particle localization methods [4] that may increase accuracy. We have found that the center of mass method yields consistent sub-pixel accuracy of 0.6 pixels on average, which is sufficient for tracking tasks that require high accuracy such as micro-rheology. The next stage is to link candidate particles from one frame to the next.

Let $\mathcal{L}_t$ denote the set of linked particle pairs $(\mathbf{x}_t, \mathbf{x}_{t+1})$ together with their probabilities (ρ_t, ρ_{t+1}) in frame t to $t+1$. We must also consider the possibility that a given particle has just entered or is about to leave the image. Let $\mathcal{N}_t^{\pm}$ be the set of probabilities for particles in frame t that are not linked to a particle in frame $t \pm 1$. Then, the log likelihood cost of the link assignments (or lack of assignment) from frame t to frame $t+1$ is given by

$$\begin{aligned} L_t = &- \sum_{\mathbf{x}_t, \mathbf{x}_{t+1} \in \mathcal{L}_t} \frac{\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2}{2\sigma^2} + \sum_{\rho_t, \rho_{t+1} \in \mathcal{L}_t} [\log \rho_t + \log \rho_{t+1}] \\ &- \sum_{\rho_t \in \mathcal{N}_t^+} \log(1 - \rho_t) - \sum_{\rho_{t+1} \in \mathcal{N}_{t+1}^-} \log(1 - \rho_{t+1}). \end{aligned} \tag{7}$$

The standard deviation σ is a user-specified parameter. Maximization of (7) can be formulated as a linear programming problem, which we solve using the Hungarian-Munkres algorithm [5].

Note that we have made a slight modification to the adaptive linking method developed in [5], where we have made use of the detection probabilities. The standard (non adaptive) approach is to assign a penalty for not assigning a link to a particle, based on a fixed cutoff distance. Our adaptive scheme uses the detection probabilities as a variable cost for not assigning a link to a particle. The lower the detection probability for a particle (due to faint signal or absence in past or future frames), the lower the cost of failing to assign it a link.

We note that σ is the only parameter in our tracking method (there are no adjustable parameters in the neural network localizer). It is reasonable to be concerned about automation when the method contains an adjustable parameter. Because of the adaptive nature of our linking algorithm, which is armed with certainty estimates from the neural network, we have found that in practice, σ rarely needs to be adjusted. In fact, every one of the ~600 videos tracked for testing purposes in this paper used the same value of this parameter ($\sigma = 20$). In the future, it may be possible to eliminate this parameter completely using a more sophisticated linking algorithm, such as a 'particle filter' [6], which is a Bayesian framework that is compatible with the neural network. Moreover, noise in particle localizations can arise from many factors, including low SNR image conditions. Kalman filters have been applied to path linking to reconstruct more accurate paths from noisy localization [7].

Parameter values for tracking software used in the synthetic video tests

Sample frames of the synthetic test videos can be seen in Fig. 2. All of the 2D and 3D videos were tracked using the same set of parameter values. The neural net tracker uses one parameter in its linking method (for collecting particle localizations into paths). The standard deviation for particle displacements was set to $\sigma = 20$.

No method was used with default parameter values. Through experimentation, testing 10-15 parameter sets for each method on the full data set, we chose parameter values that showed the best performance overall. There was no objectively optimal parameter set since we needed to balance false positives and false negatives. We chose parameter sets so that the trackers extracted a reasonable fraction of the particle tracks, while maintaining the lowest possible false postive rate. In practice, paramter values can be tuned to decrease false positives at the expense of fewer extracted tracks.

For Mosaic, we used a custom ImageJ macro to batch process the test videos. The particle detection parameters were

$$\text{radius} = 8, \quad \text{cutoff} = 0, \quad \text{percentile} = 0.8$$

For ICY, we used a custom javascript script for batch processing, which only required parameter values for its partical localization method (the particle linking method is fully au-

tomated). The particle detection parameters were

$$\text{scale}_1 = 0, \quad \text{scale}_2 = 0, \quad \text{scale}_3 = 50, \quad \text{scale}_4 = 100$$

For linking, we specified that particles with PSF radius < 2 (the minimum size in the test videos) be filtered, and that the ICY linker should assume all particles move by standard Brownian motion.

Evaluation against experimental videos

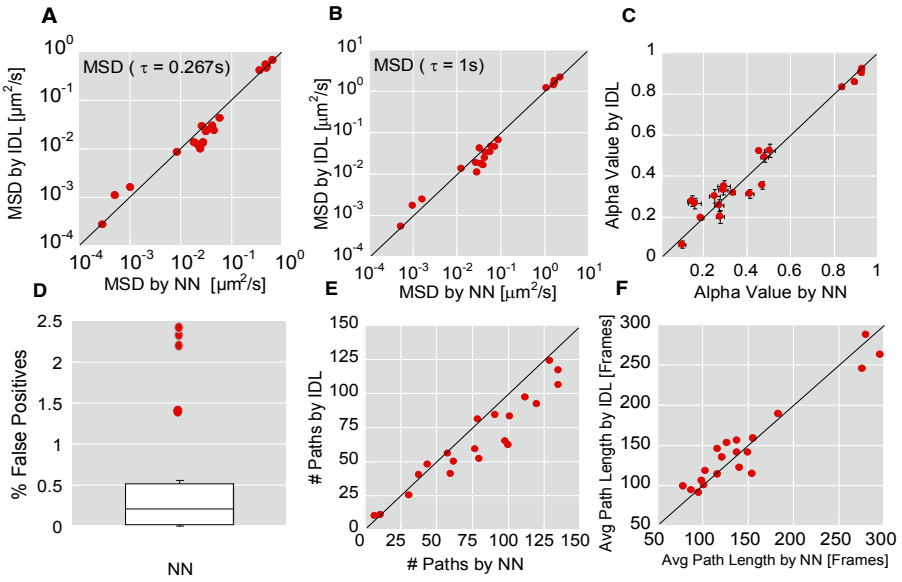

Figure 4: **Comparison of human tracked (assisted by the commercially available software package IDL) and neural network tracked output.** Ensemble-averaged geometric mean square displacements ($\langle$MSD$\rangle$) at a time scale (τ) of (A) $0.267s$ and (B) $1s$. (C) alpha value (D) percentage of false positives normalized by path-length (E) number of particles tracked (F) average path duration per particle. The error bars in (C) represent standard error of the mean. The box plot in (D) shows symbols for the outliers above the 80th percentile of observations. The data set includes 20 different movies encompassing muco-inert 200 nm PEGylated polystyrene beads 200 nm carboxylated beads, HIV virus-like particles and herpes simplex virus in human cervicovaginal mucous. Further details regarding the experimental conditions for the videos used in the test can be found in the Methods Section.

HIV, HSV and nanoparticles were prepared as previously described [8, 9, 10]. Briefly, replication-defective HIV-1, internally labeled with an mCherry-Gag construct to avoid alteration of the viral surface, was prepared by transfection of 293T cells with plasmids encoding NL4-3Luc Vpr-Env-, Gag-mCherry, and YU2 Env in a 4:1:1 ratio [8]. Mucoinert nanoparticles were prepared by conjugating 2 kDa of amine-modified polyethylene glycol to carboxyl-modified nanoparticles via a carboxyl-amine reaction; PEG-grafting was verified using the fluorogenic compound 1-pyrenyldiazomethane (PDAM) to quantify residual unmodified carboxyl groups on the nanoparticles [11]. HSV encoding a VP22-GFP tegument protein packaged into HSV-1 at relatively high copy numbers are produced as previously described [9]. Fluorescent virions or nanoparticles ($\sim 1 \times 10^8$ - 1×10^9 particles per mL) were mixed at 5% vd'ilution into $\sim$20 µL of fresh human cervicovaginal mucus collected as previously described [9], sealed within a custom-made glass chamber. The translational motions of the particles were recorded using an EMCCD camera (Evolve 512; Photometrics, Tucson, AZ) mounted on an inverted epifluorescence microscope (AxioObserver D1; Zeiss, Thornwood, NY), equipped with an Alpha Plan-Apo 100/1.46 NA objective, environmental (temperature and CO2) control chamber, and an LED light source (Lumencor Light En-

gine DAPI/GFP/543/623/690). Videos (512x512, 16-bit image depth) were captured with MetaMorph

imaging software (Molecular Devices, Sunnyvale, CA) at a temporal resolution of 66.7 ms and spatial resolution of 10 nm (nominal pixel resolution 0.156 mm per pixel) for 20 s. Subpixel tracking resolution was obtained by determining the precise location of the particle centroid by light-intensity-weighted averaging of neighboring pixels. Trajectories were analyzed using "frame-by-frame" weighting [12] in which mean squared displacements (MSD) and effective diffusivities (Deff) are first calculated for individual particle traces. Averages and distributions are then calculated at each frame based on only the particles present in that frame before averaging across all frames in the movie. This approach minimizes bias toward faster-moving particle subpopulations.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[3] R. Lumia, "A new three-dimensional connected components algorithm," *Computer Vision, Graphics, and Image Processing*, vol. 23, no. 2, pp. 207–217, 1983.

[4] R. Parthasarathy, "Rapid, accurate particle tracking by calculation of radial symmetry centers," *Nature Methods*, vol. 9, no. 7, pp. 724–726, 2012.

[5] K. Jaqaman, D. Loerke, M. Mettlen, H. Kuwata, S. Grinstein, S. L. Schmid, and G. Danuser, "Robust single-particle tracking in live-cell time-lapse sequences," *Nature methods*, vol. 5, no. 8, pp. 695–702, 2008.

[6] A. Blake and M. Isard, "The condensation algorithm-conditional density propagation and applications to visual tracking," in *Advances in Neural Information Processing Systems*, 1997, pp. 361–367.

[7] P.-H. Wu, A. Agarwal, H. Hess, P. P. Khargonekar, and Y. Tseng, "Analysis of video-based microscopic particle trajectories using kalman filtering," *Biophysical journal*, vol. 98, no. 12, pp. 2822–2830, 2010.

[8] K. L. Nunn, Y.-Y. Wang, D. Harit, M. S. Humphrys, B. Ma, R. Cone, J. Ravel, and S. K. Lai, "Enhanced trapping of hiv-1 by human cervicovaginal mucus is associated with lactobacillus crispatus-dominant microbiota," *MBio*, vol. 6, no. 5, pp. e01 084–15, 2015.

[9] Y.-Y. Wang, A. Kannan, K. L. Nunn, M. A. Murphy, D. B. Subramani, T. Moench, R. Cone, and S. K. Lai, "Igg in cervicovaginal mucus traps hsv and prevents vaginal herpes infections," *Mucosal immunology*, vol. 7, no. 5, pp. 1036–1044, 2014.

[10] S. K. Lai, D. E. O'Hanlon, S. Harrold, S. T. Man, Y.-Y. Wang, R. Cone, and J. Hanes, "Rapid transport of large polymeric nanoparticles in fresh undiluted human mucus," *Proceedings of the National Academy of Sciences*, vol. 104, no. 5, pp. 1482–1487, 2007.

[11] Q. Yang, S. W. Jones, C. L. Parker, W. C. Zamboni, J. E. Bear, and S. K. Lai, "Evading immune cell uptake and clearance requires peg grafting at densities substantially exceeding the minimum for brush conformation," *Molecular pharmaceutics*, vol. 11, no. 4, pp. 1250–1258, 2014.

[12] Y.-Y. Wang, K. L. Nunn, D. Harit, S. A. McKinley, and S. K. Lai, "Minimizing biases associated with tracking analysis of submicron particles in heterogeneous biological fluids," *Journal of Controlled Release*, vol. 220, pp. 37–43, 2015.