BACHELOR

Brownian motion analysis on tethered particles to probe particle states

van Haaften, Rik

Award date:
2019

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Applied Physics
Molecular Biosensing for Medical Diagnostics

Brownian motion analysis of tethered particles to probe particle states

Bachelor End Project

H.B. van Haaften (0947108)

Supervisors:
dr. ir. A.M. de Jong
Y. Lin

Final Version

Eindhoven, July 2019

Abstract

Micro- to nanosized particles tethered to a surface are used for multiple applications in biophysics. One application is to measure the presence of target molecules, which is done by probing association and dissociation of tethered particles to molecules located at the surface. Dissociation and association only takes place when the target is present, resulting in single molecule detection of the target. The main goal of this project is to study a new method for probing bound and unbound states based on the determination of the diffusion coefficient of a tethered particle over time. This study shows that while considering small displacements Brownian motion analysis can be used for this purpose. Increments and decrements of the diffusion coefficient were directly linked to the association and dissociation of the tethered particle. By using a threshold on the diffusion coefficient, bound and unbound state lifetimes can be measured with a time resolution ranging from 0.33 to 1 second. Brownian motion analysis was compared to the algorithm which is currently used for detecting binding and unbinding event. It was found that both methods show false positive events which would not be detected by using the other method. Furthermore, a different tethered particle configuration is used in which the bead does not bind to molecules located at the surface, but to a molecule attached to the tether at the anchoring point. Brownian motion analysis turned out to be not suitable to this tethered particle configuration for detecting binding and unbinding events. Finally, a study was done on how the diffusion coefficient behavior depends on using different components in the tethered particle system. It is shown that the increase of diffusion coefficient per increase of motion area differs when changing bead type, surface coating, particle coating or tether length.

Contents

1 Introduction **1**

2 Tethered Particle Motion **2**

2.1 Systems and methods 2

3 Brownian Motion **6**

3.1 Free motion 6

3.2 Confined motion 8

4 Methods for determining the diffusion coefficient **9**

4.1 Freely moving particles 9

4.2 Tethered particles 12

4.3 Conclusion 15

5 Probing bound and unbound state **16**

5.1 Specifications time span 16

5.2 First result and uncertainty analysis 17

5.3 Probing particle states 20

5.4 Comparison to other methods 25

5.5 Conclusion 28

6 Surface Interactions **29**

6.1 Specifications method 30

6.2 Discussion 30

6.3 Conclusion 32

7 Conclusions **33**

Bibliography **35**

Appendix **36**

A Matlab codes **37**

A.1 Diffusion coefficient of free moving particles using mean square displacement . . . 37

A.2 Diffusion coefficient of free moving particles using standard deviation 38

A.3 Diffusion coefficient of free moving particles using the empirical cumulative distribution 40

A.4 Diffusion coefficient of tethered particles by linear fit through mean square displacement 42

A.5 Diffusion coefficient of tethered particles by using the standard deviation 43

A.6 Diffusion coefficient of tethered particles by using the empirical cumulative distribution 44

A.7 Diffusion coefficient of tethered particles by using a non-linear fit through the mean square displacement 45

A.8 Diffusion coefficient over time with error bar 47

A.9 Diffusion coefficient threshold 49

A.10 Diffusion coefficient plotted against amplitudes 58

B Correlations diffusion coefficient and motion amplitudes for different systems **61**

Chapter 1

Introduction

Tethered particle motion (TPM) has found many applications in biophysics. Particles with a size of less than a micron tethered to a surface are the basic concept of the system [1] [2]. Single molecule interactions in the environment can be rendered by analyzing the lateral motion of the beads. For instance, TPM has found applications in determining DNA lengths [1] [2], viscous wall effects [3], bond characteristics [4] etc. One main point of interest is the usage of the TPM system for biosensing [5]. By measuring the lateral movement of a tethered particle the presence of a target can be detected. When the target molecule is present, the tethered particle is able to form an additional bond to the surface. Therefore, detecting binding and unbinding events gives information on the presence of target molecules. Measuring the presence of these targets gives potentially possibility to monitor drug levels, organ failure, infections, etc. An overview of the tethered particle system will be given in chapter 2.

More than one method is used for detecting binding and unbinding events [6] [7]. The main goal of this project is to study a method for probing particle stated based on Brownian motion analysis. An attempt will be done to link the diffusion coefficient value of a tethered particle to a particle state. Theoretical background on Brownian motion will be presented in chapter 3. It will be shown that by analyzing the correlation between displacements of the tethered particle the diffusion coefficient can be calculated [8]. The hypothesis is that in a bound state the particle moves more confined, which should result in a decrease in the diffusion coefficient value.

Different methods for calculating the diffusion coefficient are known and it depends on the system and application which method is suitable [9] [10] [11]. Studies focused on the diffusion coefficient were used to measure motion dependency on wall-distance [12], buffer content [3], presence of membranes [10] [13], etc. In chapter 4 a comparison will be presented between different methods for analyzing the tethered particle motion. The goal will be to find the most accurate method for determining the diffusion coefficient. In chapter 5 this method will be used to calculate the diffusion coefficient of a tethered particle over time. This signal will be used to probe tethered particle states. Results will be validated and compared to currently used methods which were illustrated in chapter 2.

Finally, while designing the TPM system there is freedom of choice in different components. It is found that colloidal particles differ in motion behavior in the presence of different buffers [3] or surface types [14]. Therefore it is concluded that the behavior of tethered particles can differ when using different components in the TPM system. In chapter 6 an attempt will be done to link the behavior of the diffusion coefficient of tethered particles to different tethered particle systems. Not enough data was analyzed to make strong statements, yet correlations were found indicating what could be studied next.

Chapter 2

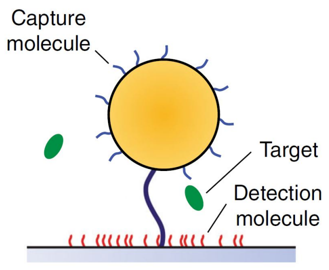
Tethered Particle Motion

In biophysics less than micron-sized particles tethered by a flexible molecule to a surface, the so-called tethered particle motion (TPM) system, have found purpose in many applications [1] [2] [3]. One main point of interest is to detect binding and unbinding events of the tethered particle to the surface [5]. Applications of the TPM system make use of the measurement of the particle's motion in the lateral direction. By analyzing the motion of the particle, conclusions can be drawn about interactions of single molecules in the environment. In this chapter, an overview will be given on components of the TPM system and possible behavior of the system relevant for detecting binding and unbinding events.

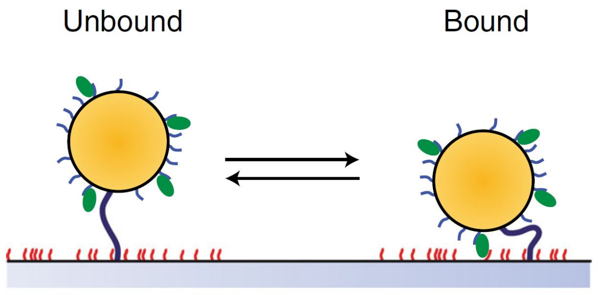
2.1 Systems and methods

One application is to detect binding and unbinding events of the tethered particle such that the TPM system can function as a biosensor [5]. By monitoring the mobility of the tethered particles, the formation of an extra bond to the surface caused by the presence of a target can be detected. Therefore, the presence of the target can be detected with a single-molecule resolution which could result in the possibility to monitor drug levels, organ failures, infections, etc. One method for constructing this system is by putting a coating on both the particle and the surface, between which an additional bond can be formed when a target is present. In figure 2.1 the sensing process is illustrated: Capture molecules (blue) are put on the bead. These capture molecules can form specific bonds with the target (green). Detection molecules (red) on the surface can form specific bonds with the target as shown in figure 2.1a. When both the detection molecule as the capture molecule binds to the target, the tethered particle binds to the surface as shown in figure 2.1b.

Freedom of choice is present in the particle type, surface coating, coupling strategy, particle coating and tether length used in the TPM configuration. Coatings are added to the system to avoid nonspecific bonds of the bead to the surface [6]. The coupling strategy is determined by which pair of capture and detection molecules are used. When a particle undergoes switching behavior between bound and unbound state it is called active. Accurate methods for detecting binding and unbinding events are relevant since there is a relation between the unbound state lifetime and target concentration [5].



(a)



(b)

Figure 2.1: The process of dissociation and association of the TPM system is illustrated. (a) The relevant components of the TPM system are illustrated. When the target (green) is bound to a capture molecule (blue) the particle can bind to the surface by forming a bond with a detection molecule (red). (b) When a target molecule is present, it enables the bead to switch from a bound to unbound state and vice versa. During a measurement of the lateral movement of a tethered particle, it is able to switch multiple times from one state to another [5].

An algorithm is used for the detection of binding and unbinding events which is based upon changes of the motion pattern when a particle dissociates or associates [4] [7]. Figure 2.2 shows an overview of possible motion patterns which can be distinguished while analyzing the lateral motion of a tethered particle. Each dot in a motion pattern corresponds to the position of the tethered particle at some moment in time during a measurement of its lateral movement. Both the number of bonds to the surface as the possible presence of protrusions can result in different motion patterns. When the particle is bound only by the tether to the surface, the motion pattern is either shaped like a disc, bell or ring. This is called the single tethered state. When an additional bond is present, a stripe pattern should be distinguishable. Finally, when more than one additional bond or a non-specif bond is present a triangular or spot pattern should be recognizable.

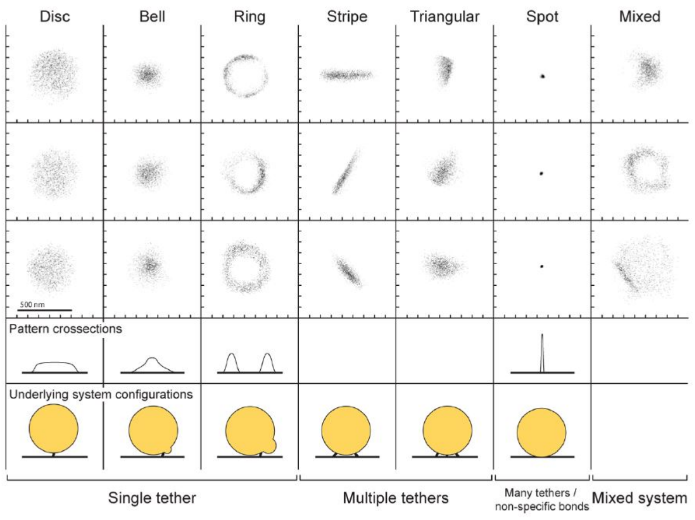


Figure 2.2: An overview is shown of possible motion patterns which can be found by analyzing the lateral movement of a tethered particle. Each dot in a motion pattern corresponds to the location of the tethered particle at some moment in time. The presence of a protrusion on the particle may cause disc-, bell- or ring-formed patterns. Information on the number of bonds to the surface is found when stripe, triangular or spot patterns are distinguishable [4].

The current algorithm monitors both Cartesian coordinates and polar coordinates to detect binding and unbinding events [7]. The coordinate values $x(t)$, $y(t)$, $r(t)$ and $|\theta(t)|$ and displacement values $\Delta x(t)$, $\Delta y(t)$, $\Delta r(t)$ and Δd_θ are monitored over time. Here $\Delta d_\theta = r\Delta\theta$. The parameters are scanned for significant changes in motion behavior, from which binding and unbinding events are detected. To receive a conceptual understanding of the current algorithm, a simplified explanation is given here on how data is analyzed. Figure 2.3 shows an illustration of such data analysis. In the graph on the right it can be seen that at certain moments in time the value of either $\Delta x(t)$, $\Delta y(t)$, $\Delta r(t)$ or Δd_θ decreases. When at these moments the value of its corresponding coordinate $x(t)$, $y(t)$, $r(t)$ or $|\theta(t)|$ becomes fixed, it is concluded that a binding or unbinding event took place. For instance, at these moments the motion pattern of the particle could transit from a circular to a stripe- or spot-like motion pattern. As seen in figure 2.3 both Cartesian and polar coordinates are considered, since not all events are detected by considering only one of these coordinate systems. The final result combines events which were detected by using either one of these coordinate systems.

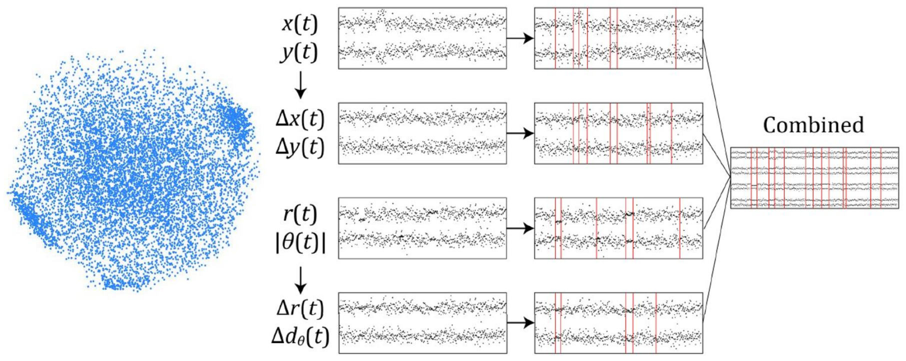


Figure 2.3: In the current algorithm the behavior of coordinate values $x(t)$, $y(t)$, $r(t)$ and $|\theta(t)|$ and displacement values $\Delta x(t)$, $\Delta y(t)$, $\Delta r(t)$ and Δd_θ are monitored to detect significant changes in motion behavior. The red vertical lines in the graphs on the right depict binding and unbinding events. When the value of either $\Delta x(t)$, $\Delta y(t)$, $\Delta r(t)$ or Δd_θ decreases and its corresponding coordinate $x(t)$, $y(t)$, $r(t)$ and $|\theta(t)|$ becomes fixed at the same moment in time, binding and unbinding events are detected. Since using only Cartesian coordinates or polar coordinates will not detect all events, events which were detected by each coordinate system are combined in the final result [7].

Another configuration exists for tethered particles which is used to measure the presence of target molecules. In this configuration detection molecules are not attached to the surface, but to the tether near the anchoring point at the surface [6]. Figure 2.4 illustrates this configuration: Near the anchoring point a detection molecule is attached to the tether. A capture molecule on the bead can form a bond to this detection molecule when a target molecule is present. The lateral movement in a bound state will differ compared to the previous configuration since the bead will not form a bond to the surface. In this configuration, the particle will show a circular motion pattern both in the bound state as the unbound state. The difference is that in a bound state the radius of the motion pattern is smaller. Therefore, the radial distance of the bead compared to the center of its motion can be monitored for detecting binding and unbinding events. Since in a bound state the radius of the circular motion pattern is smaller, a threshold on the radial displacement can be used to probe particle state. The plot on the right in figure 2.4 illustrates such analysis. It is shown how a low or high value of the radial distance is directly linked to the tethered particle state.

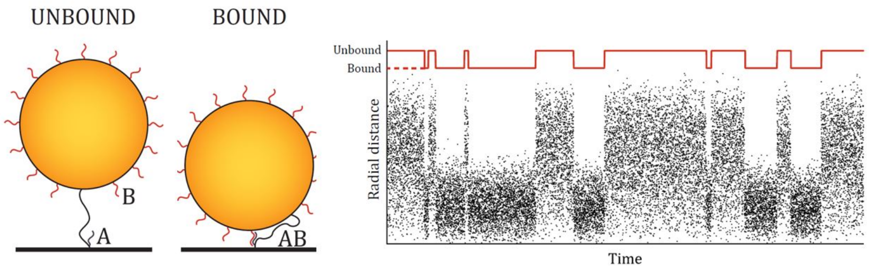


Figure 2.4: Another configuration of tethered particles can be used for measuring the presence of target molecules. A detection molecule (A) is added to the tether at the anchoring point to the surface. The detection molecule can form a bond with the capture molecules (B) on the bead when a target is present. Both in the bound state as the unbound state the motion pattern of the particle is shaped like a circle, but with a lower radius while being in a bound state. Therefore, the radial distance of a tethered particle compared to its center of motion is measured, which results in lower values during periods in which the particle is in a bound state [6].

A final property of the motion patterns of all tethered particle configurations is the major axis of motion A_{major} and perpendicular minor axis of motion A_{minor}[6] [7]. Figure 2.5 illustrates the definitions of these two quantities. The value of A_{major} and A_{minor} are determined by half the value of the maximum and minimum diameter of the motion pattern respectively. The symmetry S_{sym} of the motion pattern is defined by the ratio between the minor and major amplitude. It is defined as $S_{sym} = A_{minor}/A_{major}$. The value of S_{sym} can range from 0 for a line motion pattern to 1 for a total circular motion pattern.

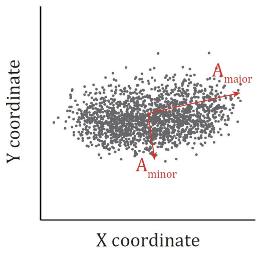


Figure 2.5: The motion pattern illustrates the definitions of A_{major} and A_{minor}. The value of A_{major} and A_{minor} are determined by half the value of the maximum and minimum diameter of the motion pattern respectively [6].

Chapter 3

Brownian Motion

3.1 Free motion

The theory used for the analysis of the motion of colloidal particles is focused on Brownian motion. Due to thermal excitation micro- to nano-scale particles collide, which in the case of free movement results in trajectories named Brownian walks [8]. During the walk, a particle is taking steps from which the values are determined by a Gaussian distribution. Figure 3.1 shows an example of a particle undergoing a Brownian walk which started at position (x_0, y_0) and ended at (x_t, y_t) after time t. Per time step Δt the particle's displacement in the horizontal and vertical direction is determined by a Gaussian probability distribution [12].

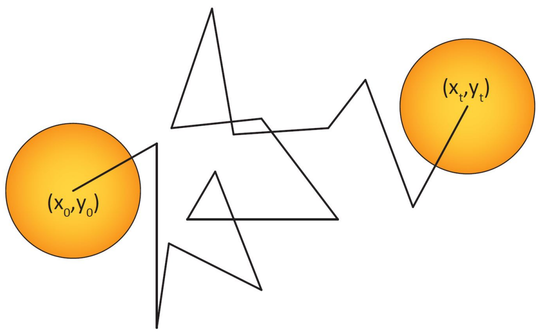


Figure 3.1: Example of a two dimensional Brownian walk in which a particle over time t moves in steps from position (x_0, y_0) to (x_t, y_t). Each step is taken over a time step Δt in which the horizontal and vertical displacement value is determined by a Gaussian probability distribution [6].

The movement of the particle is a stochastic process, which depends on the size and environment of the particle. This is reflected in the value of the diffusion coefficient D, since the probability distribution of a one-dimensional displacement Δx over a time step Δt is given by [12]

$$P_{1D}(\Delta x, \Delta t) = \frac{1}{\sqrt{4\pi D_x \Delta t}} \exp\Big(- \frac{\Delta x^2}{4 D_x \Delta t}\Big). \tag{3.1}$$

Here D_x equals the diffusion coefficient in the direction of Δx. While increasing the value of D_x, the width of the Gaussian probability function increases, meaning that the particle undergoes larger displacements Δx during a time step Δt. When the movement is isotropic it can be stated that $D_x = D$. For a two-dimensional isotropic Brownian walk the probability of finding a particle in a two-dimensional surface element $dA = 2\pi r dr$ at position r, after time step Δt is given by [10]

$$P_{2D}(r, \Delta t) = \frac{1}{4\pi D \Delta t} \exp\Big(\frac{-r^2}{4D\Delta t}\Big). \tag{3.2}$$

From equation (3.2) an expression can be derived which represents the probability $F(r, \Delta t)$ of finding the particle at a distance r or smaller after time step Δt. By integrating it is found that

$$F(r, \Delta t) = 2\pi \int_0^r P(r', \Delta t) r' dr' = 1 - \exp\Big(\frac{-r^2}{4D\Delta t}\Big). \tag{3.3}$$

From equations (3.1) and (3.2) expressions for the mean and mean square value of the displacement can be calculated. First, for the mean displacement it can be found that $\langle \Delta x \rangle = \langle r \rangle = 0$. Moreover, by calculating the variance it can be found that $\langle \Delta x^2 \rangle = \int_{-\infty}^{\infty} \Delta x^2 P_{1D}(\Delta x, \Delta t) d(\Delta x) = 2D_x \Delta t$ and $\langle r^2 \rangle = \int_0^{2\pi} \int_0^{\infty} r^2 P_{2D}(r, \Delta t) r dr d\theta = 4D\Delta t$ [10]. More generally, for the mean square displacement $\langle r_d^2 \rangle$ in d dimensions it is found that

$$\langle r_d^2 \rangle = 2dD\Delta t. \tag{3.4}$$

When looking at the trajectory of a particle the mean square displacement can be calculated by using [3] [13]

$$\begin{aligned} \langle r^2 \rangle(t) = \langle r^2 \rangle(n\Delta t) = \langle x^2 \rangle(n\Delta t) + \langle y^2 \rangle(n\Delta t) = \\ \frac{1}{N-1-n} \sum_{j=1}^{N-1-n} \Big([x(j\Delta t + n\Delta t) - x(j\Delta t)]^2 + [y(j\Delta t + n\Delta t) - y(j\Delta t)]^2 \Big) \end{aligned} \tag{3.5}$$

with

$$t = n\Delta t. \tag{3.6}$$

Here $x(j\Delta t + t)$ and $y(j\Delta t + t)$ describe the particle coordinates after a time interval $t = n\Delta t$ compared to starting coordinates $x(j\Delta t)$ and $y(j\Delta t)$. N equals to total number of positions which are considered. The one-dimensional mean square displacements in the x and y direction are given by $\langle x^2 \rangle$ and $\langle y^2 \rangle$.

Other relations exist for the diffusion coefficient which do not directly depend on the statistics of a Brownian walk. For freely moving colloidal particles a relation exists between the particle radius r, temperature T, viscosity η and the diffusion coefficient D [15]. The diffusion coefficient is given by the Stokes-Einstein equation, which states

$$D = \frac{k_B T}{6\pi \eta r}, \tag{3.7}$$

in which k_B equals the Boltzmann constant. Furthermore, at close proximity to a surface the value of D is affected by viscous effects. To correct the diffusion coefficient value of a particle located near a wall correction factors given by Faxén's Law need to be used [16].

3.2 Confined motion

The apparent diffusion coefficient of tethered particles which undergo confined motion will be determined. From a particle trajectory the displacements will be determined, each over the same time step. A value of the diffusion coefficient will be calculated from the underlying correlation of these displacements. For particles which do not move freely, displacement distributions with different variance than equation (3.4) are expected. A distinction is made between free diffusion, diffusion with drift, anomalous diffusion and confined diffusion [15]. When the mean square displacement in d dimensions is calculated for confined motion, a different relationship than equation (3.4) is found. For these particles it is found that [15] [17]

$$\langle r_d^2 \rangle = R^2(1 - \exp\frac{-\Delta t}{\tau_d}). \tag{3.8}$$

Here τ_d equals the time of confinement which depends on the diffusion coefficient D, confined radius R and motion dimension d.

One of the main principles which will be used in the determination of the diffusion coefficient of tethered particles is that for small Δt confined motion can be approximated as it were Brownian motion. Therefore, in the limit of $\Delta t = 0$, by setting the diffusive speed of a confined particle equal to the diffusive speed of a freely moving particle, it can be found that $\tau_d = R^2/2dD$ [7]. By rewriting the exponent term of equation (3.8) by a Taylor expansion it can be understood why for small Δt confined motion can be approximated by Brownian motion. It follows that equation (3.8) can be approximated by

$$\langle r_d^2 \rangle \approx R^2(1 - 1 + \frac{\Delta t}{\tau} - \mathcal{O}(\frac{\Delta t^2}{\tau^2})) = 2dD\Delta t - \mathcal{O}(D^2\Delta t^2). \tag{3.9}$$

By taking small values of Δt, equation (3.9) can be linearized by neglecting the error term. It follows that equation (3.4) reappears. This result can be interpreted that for small time steps Δt confined motion can be analyzed as it were regular Brownian motion since under this limit the variance of both motion types are equal. Therefore, the correlations following from equations (3.1) and (3.2) can be used to determine the diffusion coefficient of tethered particles when small time steps Δt are considered.

Finally, an equation has been derived which represents the mean square displacement for confined motion. For confined motion in one dimension in a infinite square potential well with dimensions $[L_x, L_y]$ it is derived that [3] [13]

$$\langle x^2 \rangle(t) = \frac{L_x^2}{6} - \frac{16L_x^2}{\pi^4} \sum_{n=1(odd)}^{\infty} \frac{1}{n^4} \exp\Big(\frac{n^2\pi^2 D_x t}{L_x}\Big), \tag{3.10}$$

where D_x represents the diffusion coefficient in the x-direction [13]. When the particle motion is isotropic, the diffusion coefficient is equal in all directions.

Chapter 4

Methods for determining the diffusion coefficient

Data sets which resulted from measurements of the movement of tethered and freely moving particles were made available by the group of Molecular Biosensing for Medical Diagnostics. In these data sets, the trajectories of particles during experiments were stored with a frame rate of 30 frames per second. In this chapter, multiple methods for determining the diffusion coefficient from these trajectories will be discussed and compared. The methods follow from the theory presented in chapter 3. For each trajectory in a data set a value of the diffusion coefficient will be determined, which results in a distribution of diffusion coefficients for the whole data set. One of the goals of the discussion is to determine which of the considered methods is best suitable for detecting binding and unbinding events of active particles. One important constraint for a method is to find a small width in the diffusion coefficient distribution, such that there is a higher chance that bound state and unbound state are distinguishable.

Matlab codes for each method described in this chapter can be found in appendix A.1 until A.7.

4.1 Freely moving particles

Multiple methods were used to calculate the diffusion coefficient of freely moving particles. To be able to compare methods, the following methods were used on a data set which resulted from a measurement of freely moving particles. Each method is used on each trajectory in the data set, which results in a diffusion coefficient value for each particle.

1. From the trajectory of each particle, the two-dimensional mean square displacement $\langle r^2 \rangle(t)$ was determined by using equation (3.5). N was set equal to 1800 frames, such that a trajectory of 60 seconds would be considered. For freely moving particles the value of D can be calculated by linearly fitting equation (3.4) with $d = 2$ to $\langle r^2 \rangle(t)$. Figure 4.1 shows an example of such a fit.

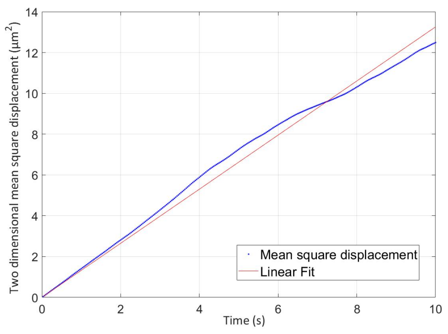


Figure 4.1: A plot is shown of the two-dimensional mean square displacement over time which was determined from the trajectory of a freely moving particle. The two-dimensional mean square displacement is calculated by using equation (3.5). The diffusion coefficient of freely moving particles can be calculated by fitting equation (3.4) with $d = 2$ to the two-dimensional mean square displacement. An example of such a fit is shown here by the red line.

2. The standard deviation $\sigma_{\Delta x}$ will be calculated of the one-dimensional displacements Δx in a particle trajectory. The time between displacements is given by Δt. Figure 4.4 shows two distributions of Δx with a different value of Δt between the displacements. It can be seen that the width of the distribution increases when Δt increases, which results in a higher value of $\sigma_{\Delta x}$. This is reflected by the variance of the displacements, which is given by $\langle \Delta x^2 \rangle = \sigma^2_{\Delta x} = 2D\Delta t$. Five different values of Δt will be used for which each $\sigma_{\Delta x}$ will be determined. Therefore the value of D can be calculated for each particle trajectory by linearly fitting the found values of $\sigma^2_{\Delta x}$ with respect to the corresponding values of Δt. Figure 4.2 shows an example of such a linear fit.

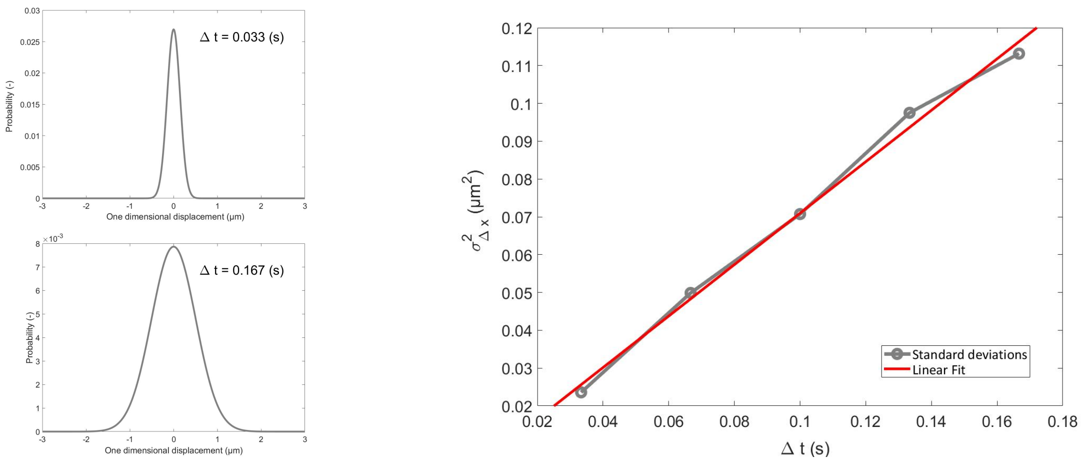


Figure 4.2: The distributions of one-dimensional displacements for two values of Δt are shown. By increasing the time Δt between displacements, the distribution of one-dimensional displacements becomes wider. The variance of the distribution of the one-dimensional displacements is given by $\langle \Delta x^2 \rangle = \sigma^2_{\Delta x} = 2D\Delta t$. Therefore, by calculating $\sigma^2_{\Delta x}$ for five different values of Δt a linear fit can be used to calculate the diffusion coefficient value. An example of such a fit is is shown by the red line plotted in the graph on the right.

3. Finally, from each particle trajectory the empirical cumulative distribution will be determined of radial square displacements r^2. The time between each displacement is taken equal to a fixed value of Δt. Since the empirical cumulative distribution is an estimator of the probability of finding the particle at a distance r or smaller after a time step Δt, the diffusion coefficient can be calculated by fitting equation (3.3) over the graph. Figure 4.3 shows an example of such a fit. By constructing an empirical cumulative distribution for each particle trajectory and using a fit according to equation (3.3), a value of the diffusion coefficient is determined for each trajectory in the data set.

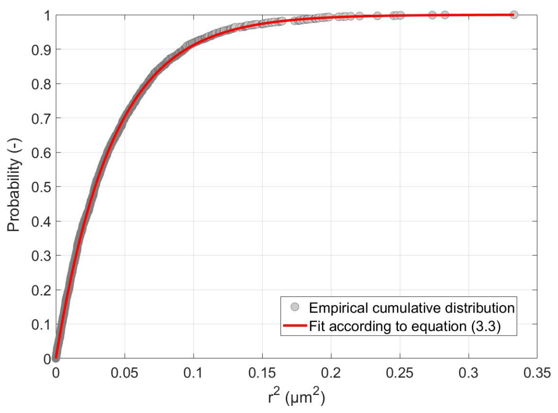
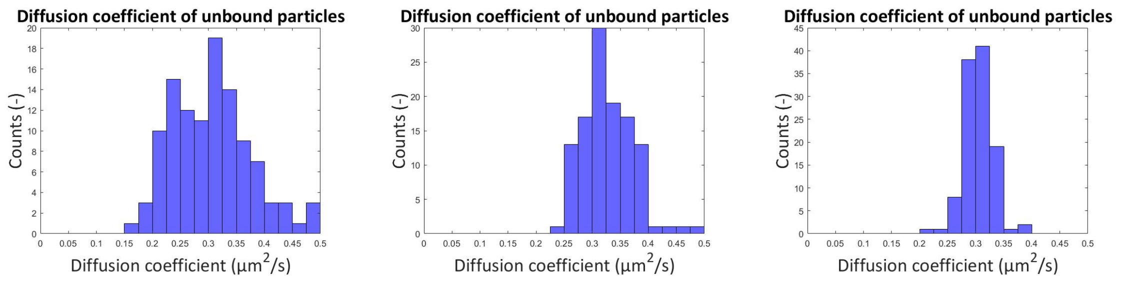


Figure 4.3: The empirical cumulative distribution is plotted of the square displacements r^2 in a particle trajectory with a fixed time step Δt between the displacements. This distribution is an estimator of the probability of finding the particle at a distance r or smaller after a time step Δt. Therefore, equation (3.3) can be fitted over the distribution to find a value of the diffusion coefficient.

The methods are based on statistics, which means that adding more displacements to a method will automatically result in more accurate determination of the diffusion coefficient. Therefore, for each method a total trajectory time of 60 seconds was used with the same value of Δt between each displacement. Overlapping trajectories were filtered out to ensure valid results.

Figure 4.4 shows the diffusion coefficient distributions which were constructed by using the three methods. The main difference was found in the width of the distributions. It can be seen that fitting equation (3.3) over the empirical cumulative distribution of square displacements results in the sharpest distribution. This indicates that this method is the most accurate. Similar results were found in literature, where the diffusion coefficient was determined of simulated particle diffusion [11]. From these simulations it was also found that linearly fitting equation (3.4) to $\langle r^2 \rangle(t)$ for each particle trajectory gives a wider distribution than fitting equation (3.3) over the empirical cumulative distribution of square displacements.



(a) (b) (c)

Figure 4.4: (a) Diffusion coefficient values determined, each for each particle trajectory, by fitting equation (3.4) with $d = 2$ to the two-dimensional mean square displacement. (b) Diffusion coefficient values determined, each for each particle trajectory, by calculating the variance of one-dimensional displacements with different values of Δt between the displacements. A linear fit is used between the variance values and the corresponding values of Δt. (c) Diffusion coefficient values determined, each for each particle trajectory, by fitting equation (3.3) to the empirical cumulative distribution of the square displacements in a particle trajectory.

4.2 Tethered particles

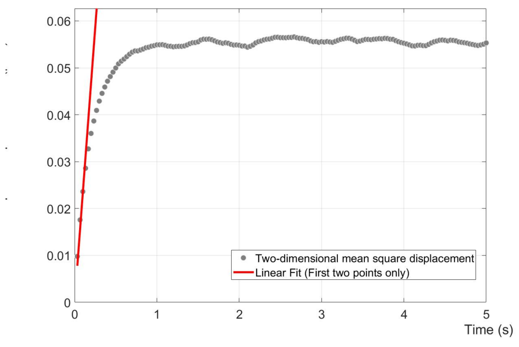
The diffusion coefficient was also determined for the trajectories of tethered particles by using different methods. The result of equation (3.9) will be used, such that the same methods as for freely moving particles can be used as long as displacements are considered between small values of Δt only. The smallest possible value of Δt will be used, which is limited by the frame rate. In total four methods were used to calculate the diffusion coefficient of each trajectory in a data set.

1. Again, the two-dimensional mean square displacement $\langle r^2\rangle(t)$ was determined by using equation (3.5) for each trajectory. In this case the value of D will be determined by fitting equation (3.4) with $d = 2$ over the first two points of $\langle r^2\rangle(t)$ only. By doing so, only a small value of Δt is considered. Figure 4.5a shows a graph of the two-dimensional mean square displacement of a tethered particle. An example of a linear fit through the first two data points is illustrated by the red line.

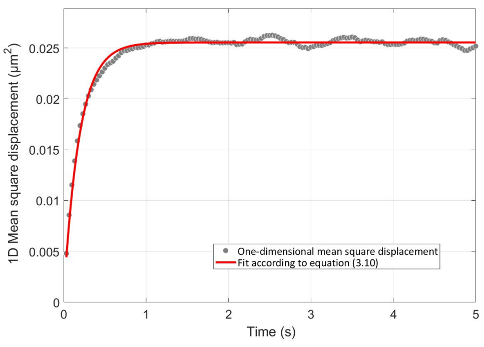
2. The standard deviation $\sigma_{\Delta x}$ of the one-dimensional displacements Δx with time Δt between the displacements will be calculated for each particle trajectory. In contrast to what was done for freely moving particles, only the smallest possible value of Δt will be used to find one value of $\sigma_{\Delta x}$. Again it is used that $\sigma^2_{\Delta x} = \langle \Delta x^2\rangle = 2D\Delta t$, which means that dividing $\sigma^2_{\Delta x}$ through $2\Delta t$ results into the value of D.

3. As done for the trajectories of freely moving particles, the empirical cumulative distribution of r^2 will be constructed over which a fit of equation (3.3) will be done to determine the value of D. For tethered particles, the smallest possible value of Δt will be used as the time between displacements. Fits which result from this method were similar to the fit shown in figure 4.3.

4. Finally, equation (3.10) with $n = 5$ will be fitted over the one-dimensional mean square displacement $\langle x^2\rangle$. The one-dimensional mean square displacement is calculated by using an one-dimensional form of equation (3.5) on a particle trajectory. Since equation (3.10) does not make use of the result of equation (3.9), the fit will be done over the whole graph of $\langle x^2\rangle$. Figure 4.5b shows an example of a graph of the one-dimensional mean square displacement of a tethered particle. The red graph illustrates a fit which follows from equation (3.10), which results in a value of D.



(a)



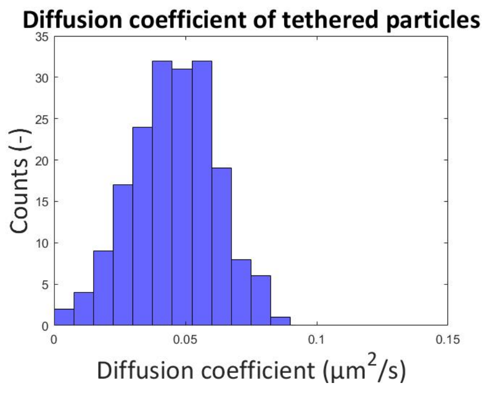
(b)

Figure 4.5: (a) A plot is shown of the two-dimensional mean square displacement over time, which results from a trajectory of a tethered particle. A linear fit was done through the first two points of the plot to determine a value of D. It was used that $\langle r^2(t)\rangle = 2dD\Delta t$ in this region. (b) A plot is shown of the one-dimensional mean square displacement over time, which results from a trajectory of a tethered particle. A fit according to equation (3.10) is used to determine a value of D.

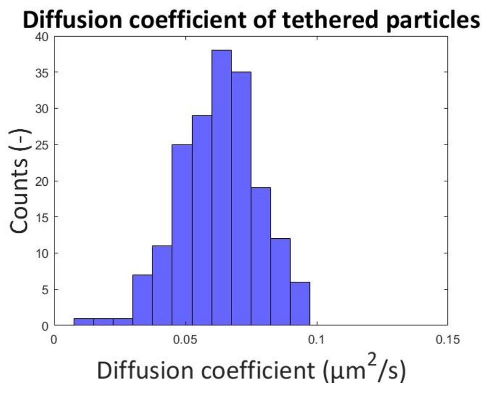
The same number of displacements is used for each method: each method analyzed a trajectory of 60 seconds with the same minimum time step $\Delta t = 0.033s$ between displacements. Data following from measurements of nonactive TPM systems were used for the analysis. Tethered particles would not form bonds to the surface during the measurements since no target molecules were added to the system. To ensure only single tethered particle would be considered, particles which showed sufficient symmetry ($S_{sym} \geq 0.75$) and from which the minor amplitude value fell into the correct range (50 nm $\leq A_{minor} \leq$ 150 nm) were selected [6]. Therefore, particles which were stuck or had more than one attached tether were filtered out.

All four methods were used to determine the diffusion coefficient distribution of ten different TPM systems. For each trajectory which resulted from a measurement, the value of the diffusion coefficient was determined which results in histograms as shown in figure 4.6. Each measurement used myone particles and a 221 base pair double-stranded DNA tether. The particle coating, surface coating and coupling strategy differed between the ten measurements. Either BSA or PLL-g-PEG was used as surface coatings and either DBCO-azide or DIG-AntiDig was used as coupling strategy. For the particle coating either biotinPEG, 11 base pair single-stranded DNA or 20 base pair single-stranded DNA was used. Since the results of all ten systems were similar, figure 4.6 shows only the diffusion coefficient distributions which resulted from one of these systems.

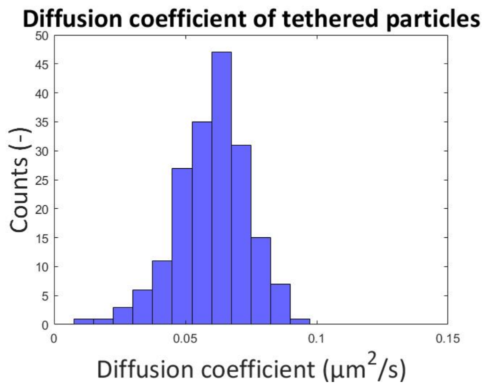
As can be seen in figure 4.6d, using a fit of equation (3.10) to the one-dimensional mean square displacement of trajectories of tethered particle results in the widest distribution. It implies that this method gives the most inaccurate result compared to the other methods. This might be caused by the fact that equation (3.10) was derived for a square potential well, while the confinement of a tethered particle follows different geometry. Another cause of the wide distribution which could be proposed is that the motion was not isotropic. The diffusion coefficient was also determined for each trajectory by calculating the standard deviation of one-dimensional displacements, from which the diffusion coefficient was determined by using $\langle x^2(t)\rangle = 2dD\Delta t$. This was done in both the x- as the y-direction, which resulted in similar distributions of D in both directions. It was therefore concluded that the motion was isotropic, which means that isotropy cannot be the cause of the wide distribution in figure 4.6a. Yet, using a fit of equation (3.10) over the one-dimensional mean square displacement is the only method which does not make use of the result which follows from equation (3.9). Only this method is valid for large values of Δt, which indicates that the other methods give results in the right order size.



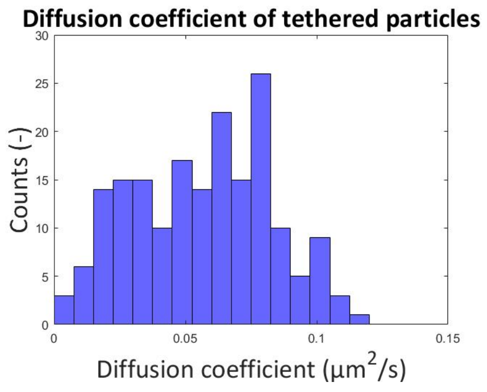
(a)



(b)



(c)



(d)

Figure 4.6: (a) Diffusion coefficient distribution which is constructed by using a linear fit only on the first two points of the two-dimensional mean square displacement which is determined of each trajectory. (b) Diffusion coefficient distribution which is constructed by calculating the variance of the one-dimensional displacements in each trajectory, after which each variance is divided trough $2D\Delta t$. (c) Diffusion coefficient distribution which is constructed by fitting equation (3.3) to the empirical cumulative distribution of square displacements over a time step Δt in each trajectory. (d) Diffusion coefficient distribution which is constructed by fitting equation (3.10) with $n = 5$ to the one-dimensional mean square displacement which is determined of each trajectory.

The distribution in figure 4.6a resulted from using a linear fit over the first two points of the two-dimensional mean square displacement. The mean value of the distribution shown by figure 4.6a seems shifted compared to the mean values of the distributions shown by figures 4.6b and 4.6c. The same observation was made for all ten TPM systems. A possible cause is that although only the first two points of the mean square displacement were considered, the time interval over which the slope was determined was not as small as the time step between the displacements which were used for calculating the standard deviation or empirical cumulative distribution. The first two points of $\langle r^2(t)\rangle$ span a time interval of 0.067 seconds, while the minimum value of Δt between displacements equals 0.033 seconds. The error term in equation (3.9) therefore results in a higher value, which could result in a smaller value of the diffusion coefficient.

Finally, it can be seen that the distribution which is shown by figure 4.6c is slightly sharper than the distribution which is shown by figure 4.6b. This means that fitting equation (3.3) to the empirical cumulative distribution of the square displacements results into the most accurate determination of the diffusion coefficient.

4.3 Conclusion

An overview and comparison are shown of possible methods for determining the diffusion coefficient of freely moving or tethered particles. It was observed that fitting equation (3.3) over the empirical cumulative distribution of square displacements r^2 results in the sharpest diffusion coefficient distributions. In chapter 5 the state of tethered particles will be probed by monitoring the diffusion coefficient of over time. Since the diffusion coefficient distribution of bound and unbound tethered particles might overlap, it was chosen to use this method to determine the diffusion coefficient over time. When it is possible to construct sharper distributions, there is a smaller chance of overlap, which results in more accurate detection of binding and unbinding events.

Chapter 5

Probing bound and unbound state

In the previous chapter, an overview was given of possible methods for determining the diffusion coefficient of particles. From the considered methods it was found that fitting equation (3.3) over the empirical cumulative distribution of square displacements in a trajectory results in the sharpest diffusion coefficient distributions. This method will be used in this chapter to detect binding and unbinding events of the TPM system. First, a closer look will be taken on what time span is needed for calculating the diffusion coefficient with sufficient certainty. Thereafter, results will be shown of the detection of binding and unbinding events by monitoring the diffusion coefficient in time. These results will be validated by considering the motion patterns of tethered particles. Finally, monitoring the diffusion coefficient will be compared to currently used methods for detecting binding and unbinding events.

5.1 Specifications time span

By using the chosen method one has freedom of choice over which time interval the value of D is determined and which time step Δt between displacements is taken. Together these quantities determine the number of square displacements which are used to construct the empirical cumulative distribution. The last will impact how well equation (3.3) will fit over the distribution, since when little data is used the estimator will not completely correspond with the true cumulative distribution [18]. To illustrate the impact of these quantities, the diffusion coefficient of each nonactive tethered particle trajectory in a data set was determined while altering one of their values.

Figure 5.1a shows how changing the value of Δt impacts the diffusion coefficient distribution while keeping the number of square displacements in the empirical cumulative distribution constant. Both the mean as the mean plus or minus the standard deviation are plotted to illustrate the width of the distribution. The graph shows that the diffusion coefficient value decreases for increasing Δt. This corresponds with the result of equation (3.9). It is therefore validated that while calculating the diffusion coefficient of tethered particles, the time step Δt between displacements should be taken as low as possible. This value is limited by the frame rate of the measurement.

Furthermore, figure 5.1b shows how the number of square displacements per fit impacts the diffusion coefficient distribution while keeping the value of Δt fixed. Following the previous result, the lowest possible value of Δt was used for constructing this graph. It can be seen how the diffusion coefficient distribution becomes sharper when the number of square displacements increases. It is favorable to achieve a high time resolution while calculating the diffusion coefficient since then the bound and unbound state lifetime can be determined more precisely. Therefore, a low number of square displacements should be used per calculation of the diffusion coefficient per time interval. Figure 5.1b shows that after adding 30 square displacements in the empirical cumulative distribution the width of the diffusion coefficient distribution starts to remain constant. Also, the biggest conversion of the width of the distribution seems to happen while adding the first ten square displacements to the calculation. Therefore the number of square displacements used per calculation is proposed to equal a number between 10 and 30.

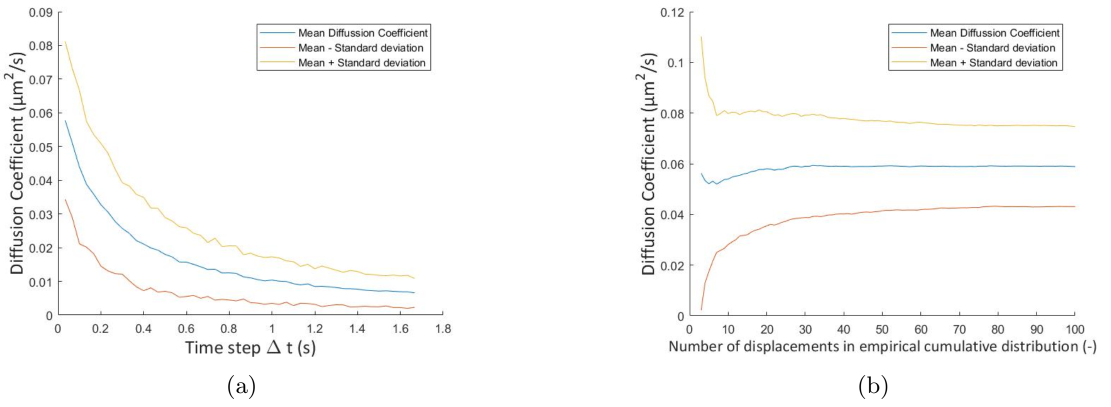
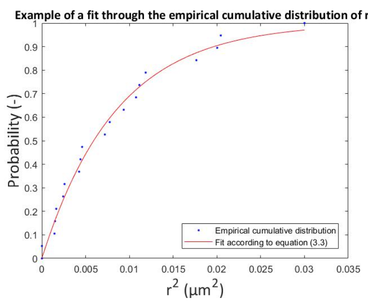


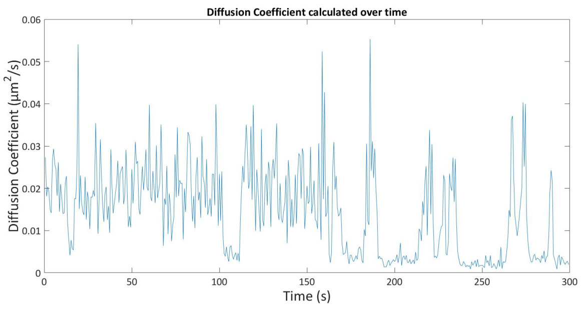
Figure 5.1: The diffusion coefficient can be calculated of each trajectory in a data set, resulting in a distribution of diffusion coefficients. This is done for a data set which resulted from a measurement of nonactive tethered particles. (a) The mean, mean plus the standard deviation and mean minus the standard deviation of the diffusion coefficient distribution are plotted to show how the distribution alters for increasing value of Δt. The number of square displacements per empirical cumulative distribution remained constant for determining each diffusion coefficient distribution. (b) The mean, mean plus the standard deviation and mean minus the standard deviation of the diffusion coefficient distribution are plotted to show how the distribution alters for an increasing number of square displacements per empirical cumulative distribution. The time step between displacements remained constant for determining each diffusion coefficient value.

5.2 First result and uncertainty analysis

Based on the discussion above it was chosen to calculate the diffusion coefficient of a tethered particle once every 0.67 seconds. This means that after every 20 frames a fit will be done over the empirical cumulative distribution of the square displacements in that interval. An example of such a fit is shown in figure 5.2a. Figure 5.2b shows an example of the diffusion coefficient plotted in time of an active particle while using 20 frames per calculation. At certain moments, for instance around $t = 100$ seconds, a decrease of the diffusion coefficient can be distinguished. This decrease indicates a binding event, which will be validated later in this chapter. For the diffusion coefficient plots in this chapter, the next 20 frames will be considered after a calculation of the diffusion coefficient. How this choice impacts the plot of the diffusion coefficient in time will be discussed in section 5.3.



(a)



(b)

Figure 5.2: (a) Over a time interval of 0.67 seconds, 20 square displacements are determined from which an empirical cumulative distribution is constructed. Equation (3.3) is fitted over the distribution to determine the value of D. After each interval of 20 frames, the next 20 frames are selected to determine the diffusion coefficient, which results in a graph of the diffusion coefficient in time. (b) A plot is shown of the diffusion coefficient in time. The value of the diffusion coefficient was calculated every 0.67 seconds. At certain moments in the graph, a decrease of the diffusion coefficient is observed, which indicate binding events of the particle.

A derivation was done of the uncertainty of the diffusion coefficient per time interval. First, uncertainty bounds are calculated for the empirical cumulative distribution by using the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality [18]. Because limited data is used each interval for constructing the empirical cumulative distribution, the true cumulative distribution lays between these bounds. The true cumulative distribution lays with probability $1 - \alpha$ within the range of the empirical cumulative distribution plus ϵ and the empirical cumulative distribution minus ϵ, where $\epsilon = \sqrt{\frac{\ln \frac{\alpha}{2}}{2n}}$. Here n equals the number of data points used for constructing the empirical cumulative distribution. In the calculations here a value of 0.32 for α was chosen to receive 68%–interval bounds.

Secondly, equation (3.3) will be rewritten in the linear form $y = Ax$ such that linear regression can be used for the determination of the value of D. When data $y = (y_1, ..., y_n)$ and $x = (x_1, ..., x_n)$ show a linear correlation in which y_i has uncertainties S_{y_i} the value of A and its uncertainty can be calculated by using

$$A = \frac{\sum w \sum wxy - \sum wx \sum wy}{\sum w \sum wx^2 - (\sum wx)^2} \tag{5.1}$$

and

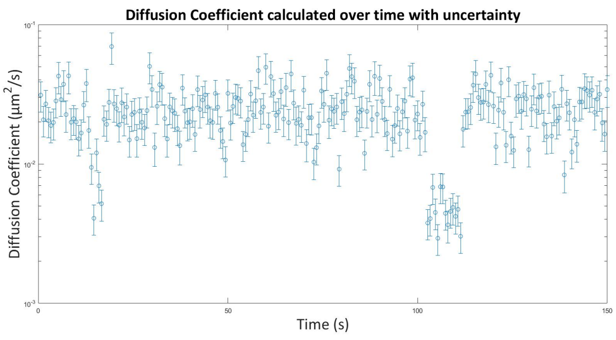
$$S_A = \sqrt{\frac{\sum w}{\sum w \sum wx^2 - (\sum wx)^2}}, \tag{5.2}$$

where $w_i = 1/S_{y_i}^2$ [19]. Rewriting equation (3.3) in a linear form results into

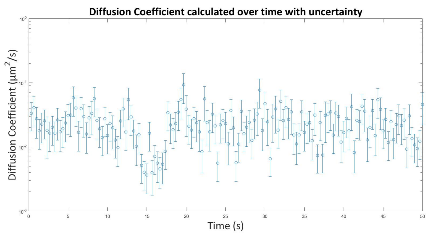
$$\ln\left(1 - F(r^2, \Delta t)\right) = \frac{1}{4D\Delta t} r^2. \tag{5.3}$$

Since the uncertainty of $F(r^2, \Delta t)$ is known by using the previously mentioned DKW-inequality, the uncertainty of the left-hand side of this equation can be calculated. The uncertainty of a function $f(x)$ in which x has uncertainty S_x is given by $S_f = \sqrt{(\frac{\partial f}{\partial x})^2 S_x^2}$ [19]. Therefore the uncertainty of the left-hand side of equation (5.3) is given by $\sqrt{\frac{\epsilon^2}{(1-F(r^2,\Delta t))^2}}$.

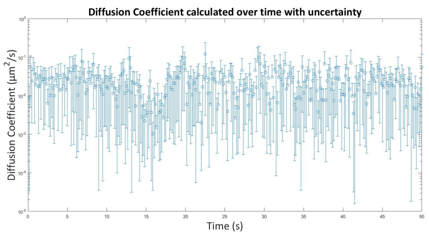
Putting the above together, a method is found for calculating the uncertainty of D per time interval. First, the uncertainty of the empirical cumulative distribution is calculated by using



(a)



(b)



(c)

Figure 5.3: The diffusion coefficient is plotted in time while including uncertainty bounds. The uncertainty is calculated following the method as discussed in section 5.2. (a) The diffusion coefficient calculated once per 0.67 seconds. (b) The diffusion coefficient calculated once per 0.33 seconds. (c) The diffusion coefficient calculated once per 0.17 seconds.

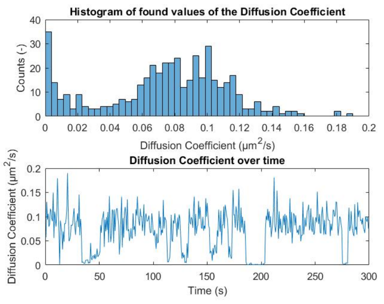
the DKW-inequality. Next, the uncertainty of the left-hand side of equation (5.3) is calculated which equals S_y. The left-hand side of equation (5.3) itself equals y, $\frac{1}{4D\Delta t} = A$ and $r^2 = x$ of the linear equation $y = Ax$. Therefore, the values of A and S_A can be calculated by using linear regression following from equations (5.1) and (5.2) after which the value of D and its uncertainty are calculated by multiplication with $4\Delta t$.

Examples of graphs of the diffusion coefficient in time including error bars are shown in figure 5.3. The diffusion coefficient was calculated once per 0.67, 0.33 and 0.17 seconds, in which the number of square displacements per calculation equaled 20, 10 and 5 respectively. By comparing figure 5.3c to figures 5.3a and 5.3b it can be seen that using a time interval equal to 0.17 seconds results in unusable results. This indicates the same conclusion that was made from figure 5.1b: At least 10 displacements per calculation of D are proposed to be used. The Matlab code used for calculating the graphs in figure 5.3 was added to appendix A.8.

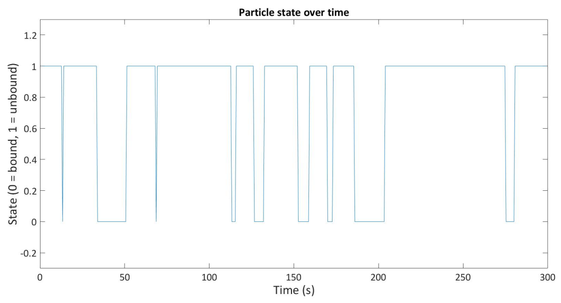
The same kind of uncertainty analysis was done while taking into account the uncertainty of the particle position, which equals 3 nm [7], instead of taking into account the uncertainty in the empirical cumulative distribution. Using the uncertainty in the position, while using 10 frames per calculation of D, would result in an uncertainty value in the diffusion coefficient which was three times lower than when uncertainty in the empirical cumulative distribution was considered. Therefore it was chosen to neglect the uncertainty in the position and only focus on the uncertainty which results from the DKW-inequality while using linear regression. When one also wants to include the uncertainty of the position in the determination of the diffusion coefficient, equation (5.3) would have uncertainty in both x and y when writing it in the linear form $y = Ax$. Total least squares should be used for taking into account the uncertainty in both x and y [20].

5.3 Probing particle states

A threshold is used on the diffusion coefficient to detect binding and unbinding events. In this section tethered particles are considered which form additional bonds to the surface. Myone particles tethered by a 221 base pair double-stranded DNA tether were used in the corresponding measurements. PLL-g-PEG was used as surface coating, 11 base pair single-stranded DNA was used as particle coating and DBCO-azide was used as coupling strategy. Until stated otherwise, the upcoming results correspond to measurements of this tethered particle configuration. Since the behavior per particle differs, a general threshold for all particles was not found. Therefore a threshold was selected manually for each particle. First, the diffusion coefficient will be calculated per time interval over the whole time span. Secondly, a histogram will be constructed of all determined diffusion coefficient values. Based on these graphs, as shown in figure 5.4a, a threshold can be selected. To take into account overlap of the bound and unbound state diffusion coefficient distributions an event will be detected only when the threshold is crossed plus an additional percentage of its value. This percentage was found by fine-tuning manually and might differ per TPM system. For the upcoming results, this percentage was fine-tuned to a value of 25%. When requiring a higher time resolution the number of frames per calculation can be set equal to lower numbers, but it needs to be taken into account that the diffusion coefficient distributions of the bound and unbound state do not start to overlap too significantly. When requiring less overlap of the bound and unbound state diffusion coefficient distributions, the number of frames should increase, which results in a loss of time resolution. Figure 5.4b shows an example of a plot of the particle state in time found by using a threshold on the diffusion coefficient.



(a)



(b)

Figure 5.4: (a) The diffusion coefficient in time and histogram of all determined diffusion coefficient values are plotted which resulted from a trajectory of an active tethered particle. A distinction can be seen between two diffusion coefficient distributions. This distinction is used to select a threshold which is used for detecting binding and unbinding events. (b) A plot is shown of the particle state in time. After selecting a threshold, the bound and unbound state of the particle can be probed in time. When the diffusion coefficient crosses the threshold plus an additional percentage a binding or unbinding event is detected.

Every 20 frames a value of the diffusion coefficient was calculated for constructing the plots of the diffusion coefficient in time. This means that after every 20 frames, the next 20 frames were used for calculating the diffusion coefficient. A closer look was taken on what impact a binding or unbinding event during a time interval has on the fit as shown in figure 5.2a. Figure 5.5 shows a fit of equation (3.3) to the empirical cumulative distribution of square displacements which were taken from an interval in which a binding event took place. Although correlation was lost, it was found that such fits result in a value of D laying between the bound and unbound state diffusion coefficient distribution. Therefore, using a threshold may not result in detecting a binding or unbinding event until the next interval is analyzed. This means that the uncertainty of detecting binding events in 100%-interval bounds is proportional to the time taken per calculation.

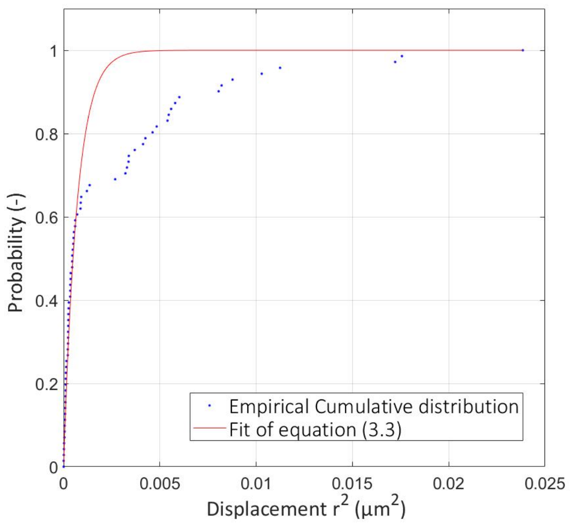


Figure 5.5: The empirical cumulative distribution is plotted of square displacements which were taken from a time interval in which a binding event took place. A fit is shown of equation (3.3) to the empirical cumulative distribution from which a value of the diffusion coefficient was determined. Although correlation was lost, a diffusion coefficient value would be determined between the bound and unbound state diffusion coefficient value.

A method to decrease this uncertainty in the bound and unbound state lifetime would be to move the frame interval of 20 frames by steps of one frame. This way for each frame a value of the diffusion coefficient will be calculated. This would result in a higher resolution in the plot of the diffusion coefficient in time. The disadvantage would be that the time needed for the calculation of the diffusion coefficient plot increases significantly. Optimization of the code in appendix A.8 would be required for following this method. For instance, by calculating the diffusion coefficient by using linear regression as derived in section 5.2 instead of using a non-linear fit might decrease the time needed per calculation of the diffusion coefficient.

A better understanding was searched for the need of different thresholds for different tethered particles. Furthermore, particles were found from which more than two different states seemed to be distinguishable. An explanation was found by taking into account the minor amplitude, major amplitude and motion pattern of the particle trajectory. Particles were distinguished from which the difference between the maximum and minimum value of the diffusion coefficient was relatively high compared to other particles. Figure 5.6 shows that the motion patterns of these particles on average have a larger minor and major amplitude. Inactive particles were filtered out while constructing these histograms. The result can be understood by knowing that a larger minor or major amplitude correlates with the area the particle was able to reach. Reaching a larger area means that the tethered particle can move more freely, which results in a higher value of the diffusion coefficient. Moreover, having a small major or minor amplitude might also indicate that the particle switched between bound states during the whole measurement.

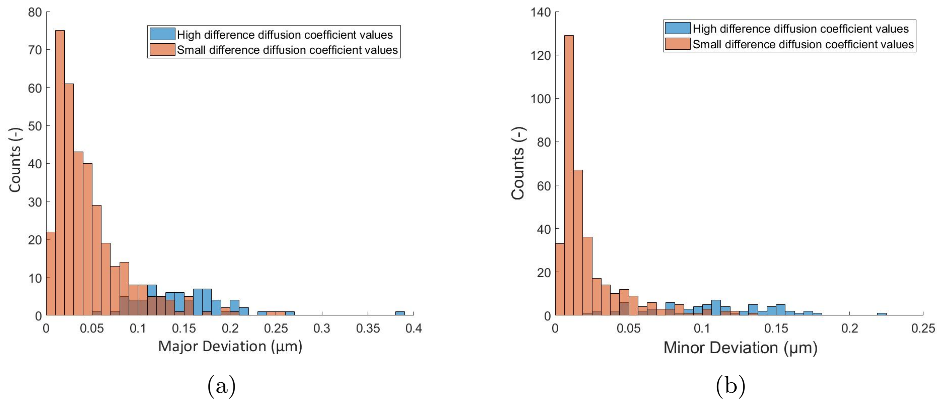
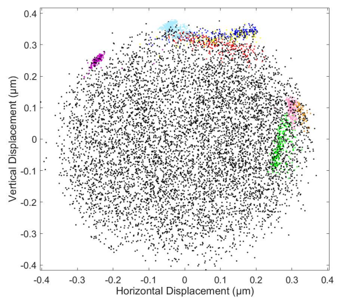


Figure 5.6: Two distributions were plotted of either the (a) major deviation or (b) minor deviation of tethered particles. The orange distributions represent active tethered particles with a small difference between the maximum and minimum value of the diffusion coefficient, while the blue distributions represent active tethered particles with a high difference between the maximum and minimum value of the diffusion coefficient.

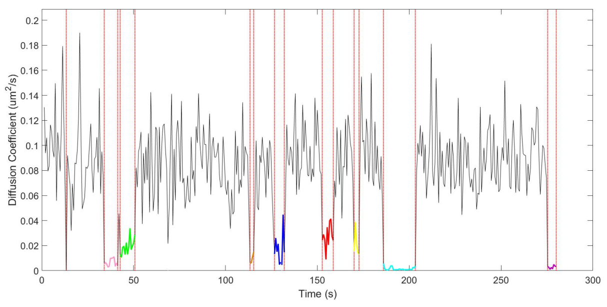
More information was found by analyzing the motion patterns of tethered particles. It was seen that active tethered particles with a high difference in the diffusion coefficient values showed circular motion patterns. Such circular motion patterns indicate that at some moment during the measurement the particle was in a single tethered state. A low difference in the diffusion coefficient value was linked to motion patterns with stripe or spot motion patterns, which can be linked to tethered particles which switch from one bound state to another bound state. To illustrate, figures 5.7, 5.8 and 5.9 shows for three particles how the diffusion coefficient changed in time. To show how the particle behaved during the measurement, the corresponding motion patterns were also plotted. By using colors, motion patterns can be linked to certain diffusion coefficient values at certain moments in time. The motion patterns will be linked to the number of bonds of the particle to the surface as discussed in chapter 2. The Matlab code which was used to detect binding and unbinding events and plot the motion pattern of tethered particles was added to appendix A.9.

Figure 5.7 shows the result of a particle from which the difference between the minimum and maximum value of the diffusion coefficient was relatively high compared to other particles. It can be seen how low values of the diffusion coefficient are linked to stripe or spot motion patterns. Therefore it is validated that during these moments the particle was in a bound state. It was found that a large difference in diffusion coefficient in time is linked to circular motion patterns. This means that a big difference in the diffusion coefficient value in time can be linked to particles switching from a single tethered state to bound states.

Figure 5.8 shows the result of a particle with a low difference between the minimum and maximum value of the diffusion coefficient. It is shown that the particle switched from a stripe to a spot motion pattern and backward. Other particles with a relatively small difference between the highest and lowest diffusion coefficient value also showed switching behavior from one to another bound state. Therefore, measuring a low difference in the maximum and minimum value of the diffusion coefficient is linked to particles switching from one bound state to another bound state.

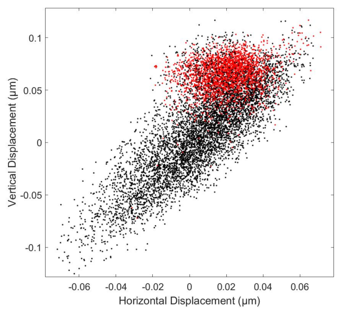


(a)

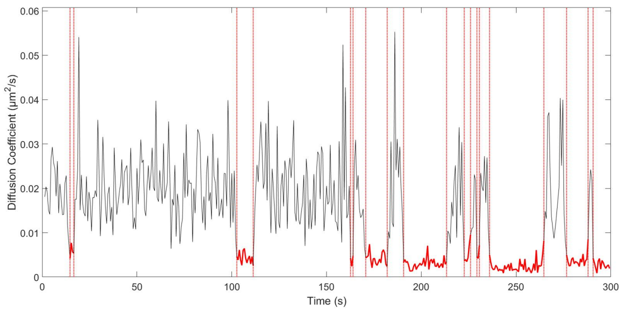


(b)

Figure 5.7: (a) The motion pattern is plotted of a tethered particle which switched from a single tethered state to bound states. By using colors stripe and spot motion patterns were directly linked to decreases in the diffusion coefficient in figure 5.7b. (b) The diffusion coefficient in time is plotted corresponding to the motion pattern shown by figure 5.7a. Binding and unbinding events detected by using a threshold on the diffusion coefficient are indicated by vertical red lines. Colored segments of the plot are linked to colored motion patterns in figure 5.7a.



(a)

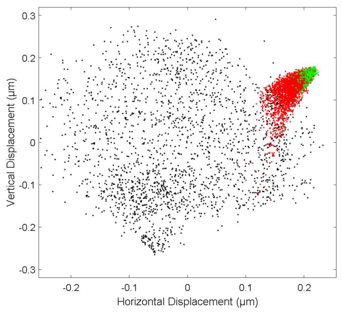


(b)

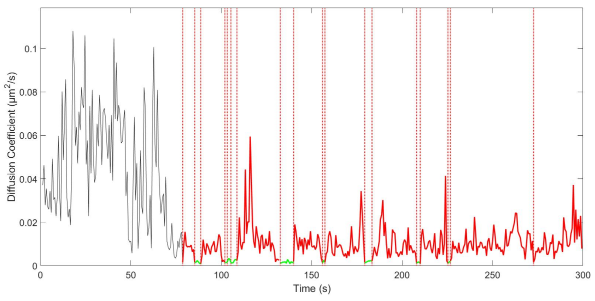
Figure 5.8: (a) The motion pattern is plotted of a tethered particle which switched between two different bound states. The two bound states are plotted as a red spot motion pattern and a black stripe motion pattern. The spot motion pattern corresponds to lower diffusion coefficient values than the stripe motion pattern. (b) The diffusion in time coefficient is plotted for a particle which switched between two different bound states. The red vertical lines indicate binding and unbinding events which were detected by putting a threshold on the diffusion coefficient. The red segments of the diffusion coefficient plot correspond to time intervals in which the particle is in a more confined bound state.

Earlier this section, it was stated that there is uncertainty in the bound and unbound state lifetime. When the value of D is calculated over a time interval in which the particle switched state, the event might not be detected by using a threshold on the diffusion coefficient until the next time interval. Figure 5.8b shows a plot in which this inaccuracy can be observed since some red segments overlap with increasing parts of the diffusion coefficient. This can for instance be observed between $t = 200$ and $t = 250$ seconds. Moreover, the motion pattern which is shown by figure 5.8a shows some red spots plotted in an area where only black spots would be expected. This is also caused by the error in the bound and unbound state lifetime.

Finally, figure 5.9 shows that more than two states can be distinguished while probing states by using the diffusion coefficient signal. At the beginning of the measurement, the particle is single tethered to the surface which is recognized by the circular motion pattern. Thereafter the particle switches between two bound states, which is indicated with red and green colors. A threshold for the analysis was set between these two bound states, which results in binding and unbinding events given by the red vertical lines. The more confined green spot motion pattern is linked to lower diffusion coefficient values. At these low diffusion coefficient values it is hard to select a correct threshold, if there is any, since the distributions of the two bound states overlap. Therefore some false positive events were detected in figure 5.9b. For instance, false positive events are shown around $t = 100$ seconds.



(a)



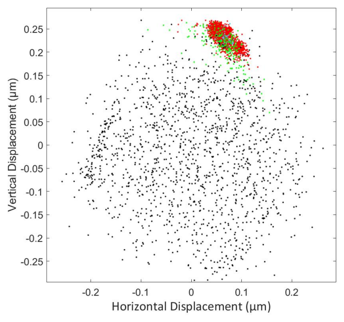
(b)

Figure 5.9: (a) The motion pattern is plotted of a tethered particle from which the motion corresponded to more than two states. The black motion pattern corresponds to a single tethered state, while the red and green motion patterns correspond to two different bound states. (b) The diffusion coefficient is plotted in time. Only one threshold was set between the two bound states, which results in the binding and unbinding events indicated by the red vertical lines. The black, red and green segments of the diffusion coefficient correspond to the black, red and green motion patterns in figure 5.9a.

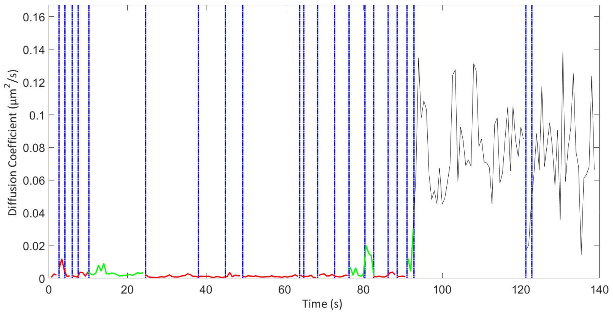
5.4 Comparison to other methods

As discussed in chapter 2, another algorithm is used to detect binding and unbinding events. The algorithm monitors the particle location and displacements to detect significant changes in the motion pattern. For instance, by using the algorithm the same events were detected for the particle measurement which was shown in figure 5.7. Moreover, both systems have a similar time resolution of around 20 frames (0.67 seconds) for finding events [7].

In contrast, also different results were found while using both methods to the same tethered particle measurement. Since the diffusion coefficient distributions of bound states can overlap, using a threshold on the diffusion coefficient might result in false positive events. These false positive events would not be detected by using the current algorithm. For example, the false positive events which are shown in figure 5.9b would not have been detected by monitoring the particle coordinates for significant changes in the motion pattern. False positive events were also detected by monitoring the particle location which would not have been detected by using a threshold on the diffusion coefficient in time. For instance, figure 5.10 shows how the current algorithm detects events while the diffusion coefficient remains at a low value. At these moments the motion pattern of the particle remains formed as a spot focused at the same location. Therefore it can be concluded that these events are wrong.



(a)



(b)

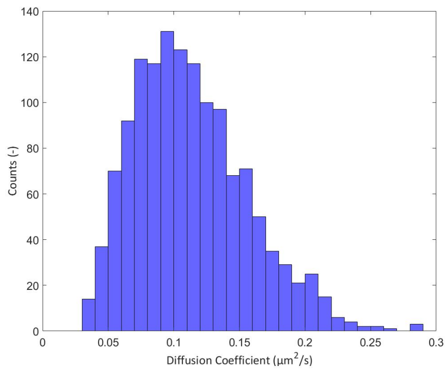
Figure 5.10: (a) The motion pattern is plotted of a tethered particle in which the particle was stuck in the red motion pattern. Green stripe motion patterns can be distinguished. (b) The diffusion coefficient is plotted in time. The red segments of the diffusion coefficient correspond to the red motion pattern in figure 5.10a. The vertical blue lines indicate binding and unbinding events resulted by monitoring the particle location for finding changes in motion patterns. Although events are detected, the red diffusion coefficient segments remain at low values and the particle remains stuck at the red spot motion pattern.

A possible cause for finding the false positive events in figure 5.10 could be that the current algorithm searches for relative changes of a coordinate value compared to previous values [7]. When a particle is in a bound state, a coordinate value of the particle is fixed. A small change of the coordinate value is therefore sufficient for the detection of an event, even though the particle remains stuck in the same area. The diffusion coefficient remains at low values at these false positive events since the particle still undergoes small displacements per frame. Therefore, it is proposed to investigate the usage of the diffusion coefficient signal to filter out false positive events. When the diffusion coefficient does not change significantly at an event which was detected by monitoring the particle position, this information could be used to mark the event as invalid.

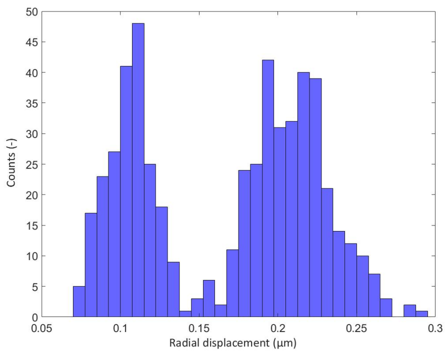
Finally, the current algorithm only indicates at which moment in time a binding or unbinding event occurred. It gives no information on the bond type or whether the particle switched from a bound state to another bound state. This information can be found while monitoring the diffusion coefficient, as shown in figures 5.7, 5.8 and 5.9. Therefore, it is again proposed to investigate the possibility to combine both methods. By doing so it is expected that information can be obtained of the bond type of tethered particles to the surface in the analysis of tethered particle motion.

In chapter 2 another configuration of tethered particles was discussed. In this system, a detection molecule would be attached to the tether at the anchoring point to the surface. Instead of forming a bond with detection molecules at the surface, the particle would form a bond with the detection molecule attached to the tether while measuring the presence of a target. Binding and unbinding events are detected by monitoring the radial displacement of the tethered particle compared to the center of its motion. To take into account noise in the measurement of the radial displacement, it is averaged every 20 frames. Therefore, both methods should be able to detect events with the same time resolution.

An attempt was done for detecting binding and unbinding events of this tethered particle configuration by using a threshold on the diffusion coefficient. Figure 5.11a shows a histogram of all diffusion coefficient values which were determined by analyzing a trajectory from a tethered particle with this configuration. It can be seen that there is a high overlap between the bound and unbound diffusion coefficient distributions. This might be caused by the remaining mobility of the tethered particle while being in a bound state. Therefore it is hard to select a correct threshold in the diffusion coefficient for detecting binding and unbinding events. Of 58 active particles which were analyzed only for two particles a distinction was observed in the diffusion coefficient distribution. None of these distinctions were as clear as the two peaks in the distribution of all measured radial displacements. An example of such a distribution is shown in figure 5.11b. Figure 5.11b shows that a clear threshold can be selected between the bound and unbound state when looking at the radial displacement distribution. Similar distributions of the radial displacement were found for all tethered particles which were analyzed.



(a)

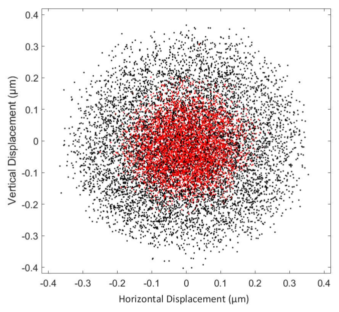


(b)

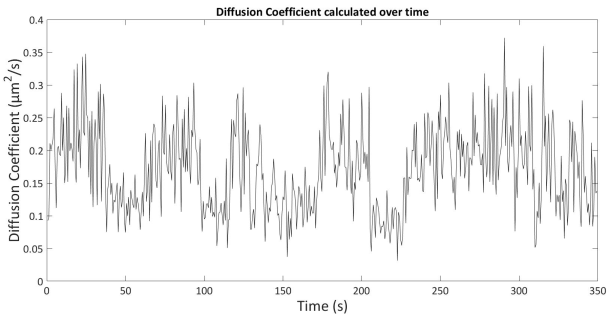
Figure 5.11: (a) A histogram is shown of all the diffusion coefficient values which were determined from a trajectory of a tethered particle. The corresponding tethered particle did attach to a detection molecule which is located at the anchoring point of the tether. Although the particle is able to switch between two states during the measurement, there is only one peak distinguishable. (b) A histogram is shown of all measured radial displacements which were determined from a trajectory of a tethered particle. Two peaks can be distinguished, which correspond to the bound and unbound state of the tethered particle.

Figure 5.12c shows how using a threshold on either the diffusion coefficient or the radial displacement resulted in different detection of binding and unbinding events. The radial displacement in time is plotted in figure 5.12c, in which the red segments correspond to the red motion patterns in figure 5.12a. The red and green vertical lines indicate events which were detected by using a threshold on the diffusion coefficient or radial displacement respectively. For illustration, figure

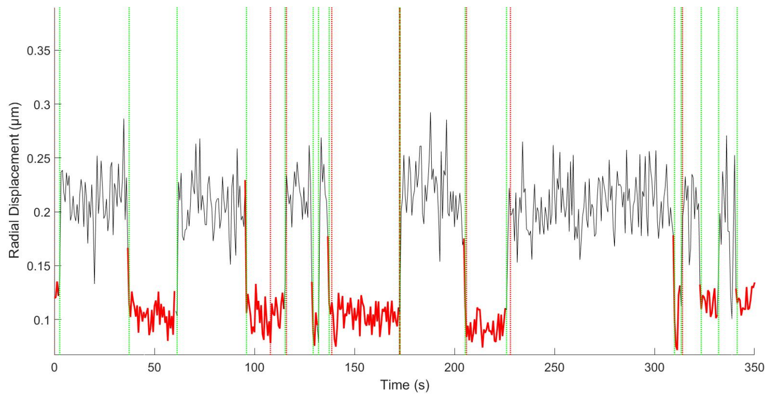
5.12b shows a plot of the diffusion coefficient in time of the same particle. Since the bound and unbound state diffusion coefficient distributions overlap significantly, the value of the diffusion coefficient needs to pass the threshold plus an additional 50 % for detecting binding and unbinding events. Between $t = 200$ seconds and $t = 250$ seconds it can be seen that both methods resulted in similar events. In contrast, around $t = 50$ seconds it can be seen that using a threshold on the diffusion coefficient would not result in the detection of binding and unbinding events, while the particle did switch between states. More missing and false events by using a threshold on the diffusion coefficient can be observed in the same graph. For this reason it is concluded that monitoring the diffusion coefficient is not applicable to the analysis of this configuration of tethered particles.



(a)



(b)



(c)

Figure 5.12: (a) The motion pattern is plotted of a tethered particle which associated and dissociated to a detection molecule which was attached to the tether. The red patterns correspond to the red segments in the plot of the radial displacement in time in figure 5.12c. The small red circular pattern corresponds to the bound state, while the large black circular pattern corresponds to the unbound state. (b) The diffusion coefficient in time is plotted of the tethered particle from which the motion pattern is shown in figure 5.12a. Some decreases and increases in the diffusion coefficient can be distinguished in the graph, which indicate binding and unbinding events. Yet, the overlap of the diffusion coefficient values was too significant, such that using a threshold would result in wrong event detection. (c) The radial distance of a tethered particle is plotted in time. A threshold is used on this quantity for the detection of binding and unbinding events of the tethered particle. These binding and unbinding events are depicted by vertical green lines. The red vertical lines correspond with binding and unbinding events which were detected by using a threshold on the diffusion coefficient.

5.5 Conclusion

The diffusion coefficient of active particles was calculated per time interval. First, it was shown how the uncertainty of the diffusion coefficient per time interval can be calculated. This uncertainty is mostly influenced by the number of square displacements which is used per calculation. It is shown that by using a threshold on the diffusion coefficient in time, binding and unbinding events can be detected of tethered particles which form a bond with detection molecules located at the surface. For this analysis, a trade-off needs to be made between time resolution and the possibility of finding false positive events. By decreasing the number of frames per calculation, the overlap of the bound and unbound state diffusion coefficient distribution increases, which results in possible wrong detection of events. Results were validated by linking the diffusion coefficient value at certain moments in time directly to motion patterns of tethered particles. It was found that by monitoring the diffusion coefficient in time information can be obtained on the bond type of a tethered particle to the surface.

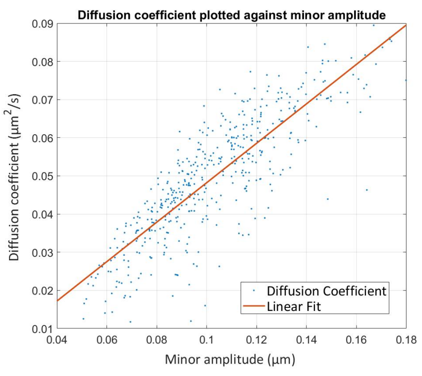
A comparison was done between monitoring the diffusion coefficient or the particle position for detecting events. It was found that both methods would result in false positive events which would not be detected by using the other method. Therefore it is proposed to combine both methods for more precise event detection. Moreover, monitoring the diffusion coefficient would give information on the bond type of the particle to the surface, while monitoring the position of the particle would not. Finally, it was tried to detect binding and unbinding events of another tethered particle configuration. In this configuration, the particle forms a bond to a detection molecule attached to the tether. It was found that monitoring the diffusion coefficient of these tethered particles is not suitable for detecting binding and unbinding events. This was caused by too significant overlap of the bound and unbound state diffusion coefficient distributions.

Chapter 6

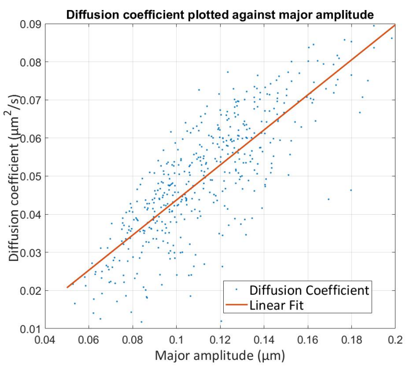
Surface Interactions

In chapter 2 it was stated that while constructing a TPM system different particle coating, coupling strategy, surface coating, tether length and particle type can be used. It is expected that the motion of tethered particles differs when altering these components. For instance, using a different coating might result in different viscosity near the surface. As seen in equation (3.7), the viscosity of the medium affects the diffusion coefficient value of a particle [3]. A small study was done on how the diffusion coefficient depends on different components in the TPM system.

When the motion pattern of a tethered particle consists of a higher minor or major axis, the particle is able to cross a larger area. Therefore, the motion seems less confined, which should result in a larger value of the diffusion coefficient. This hypothesis was validated by analyzing trajectories resulting from experiments with nonactive single tethered particles. A fit of equation (3.3) to the empirical cumulative distribution of the square displacements was used to analyze the trajectory of each particle in a data set. Figure 6.1 shows that plotting the diffusion coefficient of each particle against the corresponding value of the motion amplitudes results in a positive linear correlation. It was found that the slope of these graphs differed when different components were used in the TPM system, indicating that the diffusion coefficient increase per increase of motion area differs per TPM system.



(a)



(b)

Figure 6.1: The diffusion coefficients of nonactive tethered particles are plotted against the (a) minor amplitude and (b) major amplitude of the motion pattern of the corresponding particle. A linear correlation is found between the diffusion coefficient values and motion amplitudes. The slope-value of a linear fit as shown by the red line differs when different components are used.

6.1 Specifications method

In chapter 5 it was found that using the lowest possible value of Δt would result in the highest value of the diffusion coefficient. A different result was found for calculating the slope-value between the diffusion coefficient and motion pattern amplitude. It is found that using a time step of 0.1 seconds (3 frames) between displacements would result in the highest slope-value. One of the goals would be to link slope-values to a particular type of TPM system. Therefore it was chosen to use this value of Δt for analyzing different data sets. Since trajectories of 60 seconds were analyzed the number of square displacements in the empirical cumulative distribution per calculation of the diffusion coefficient equaled 600. As discussed in section 5.1, such a large number of displacements should result in a good determination of the diffusion coefficient per particle. Finally, motion patterns were filtered out based on symmetry and minor amplitude value. When using myone particles, motion patterns with symmetry less than 0.75 and minor amplitude outside the range of 50 to 150 nm were excluded. When silica particles were used, the same requirement was used on the symmetry and a minor amplitude between 40 to 100 nm was required [6]. The Matlab code used for this analysis was added to appendix A.10.

Five components of the TPM system can be altered to see how it affects motion behavior. These components were shortly discussed in chapter 2. Table 6.1 shows an overview of the options which were used for each component. In appendix B an overview is given of which slope–values were found for 34 systems. Without taking into account compatibility, in total 72 combinations of systems can be constructed from the number of options as shown in table 6.1. Not enough measurements were analyzed to make hard statements on relations between the slope-value and the TPM components which were used.

Table 6.1: The components of the TPM system were altered between measurements which were analyzed for finding slope-values as illustrated in figure 6.1. An overview is shown of the options per component which were considered in this project.

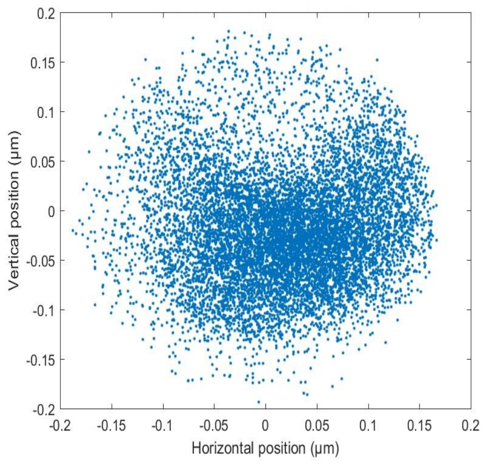
Particle type	Particle coating	Surface coating	Tether length	Coupling strategy
myone	Particle Binder ssDNA 20 base pair	BSA	120 base pair	DIG-AntiDIG (antigen-antibody)
silica	Particle Binder ssDNA 11 base pair	Casein	221 base pair	DBCO-azide click chemistry
	biotinPEG	PLL-g-PEG		

6.2 Discussion

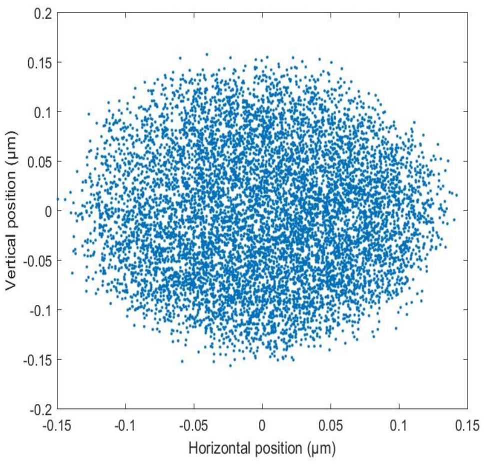
Several explanations are possible for the different behavior of the tethered particles when components are altered of the TPM system. First of all, different buffers might result in different viscous effects near the surface [3]. As seen in equation (3.7), different values of viscosity leads to different values of the diffusion coefficient. Furthermore, Faxén's Law states that near a surface the diffusion coefficient value of a particle changes based on the distance to the wall [16]. By using a different tether length, the possible distances to the wall are altered, which might result in different values of the diffusion coefficient. The stiffness of the tether and smoothness of the particle can also differ per TPM system [7]. Finally, using different coatings might result in different electric potentials near the surface [14]. For instance, when a particle is repulsed away from the surface, drag might decrease which results in a higher value of the diffusion coefficient.

Correlations were found between the slope-values and components of the TPM system. It was found that either using silica particles or varying the tether length seemed to result in different slope-values. Measurements with silica particles would result in slope-values ranging from 0.6 to 0.7 μm/s, while using myone particles would result in values ranging from 0.5 to 0.55 μm/s. Since between these systems also the tether length could differ, no conclusion could be drawn of the direct cause of this difference. The cause might be that silica particles have a smoother surface than myone particles, which could result in a higher increase of diffusivity by decreasing the drag on the particle. It was also observed that the measurement which used a casein coating resulted in the highest slope–value. For a casein measurement a slope-value of 1.40 μm/s was found, while other slope-values would range between 0.5 and 0.9 μm/s. It is shown that the presence of casein can result in electric repulsion of a particle to the surface [14]. This could result in a higher distance of the particle to the surface, decreasing drag, resulting in a higher increase in the diffusion coefficient per motion area.

The motion patterns of tethered particles corresponding to the measurements with the highest and lowest slope–value were compared. These values equaled 1.4 μm/s and 0.5 μm/s respectively. Figure 6.2 illustrates differences which were observed in the motion patterns. From the system with the highest slope–value, 88% of the particles showed a uniform motion pattern with the average position located at the center as shown in figure 6.2b. From the system with the lowest slope-value, 45% of the particles showed a nonuniform motion pattern with the average position shifted from the center as shown in figure 6.2a. Furthermore, figure 6.3 shows histograms of the mean one-dimensional position of each tethered particle in a system. It is observed that the mean position values corresponding to the lowest slope-value are shifted and widely distributed compared to the mean position values corresponding to the highest slope-value. Tethered particles corresponding to figure 6.2a seem to favor to locate at a certain region in the motion pattern. This means that the motion is more confined, which results in lower displacement values, which results in a lower increase in the diffusion coefficient value when a motion amplitude increases.



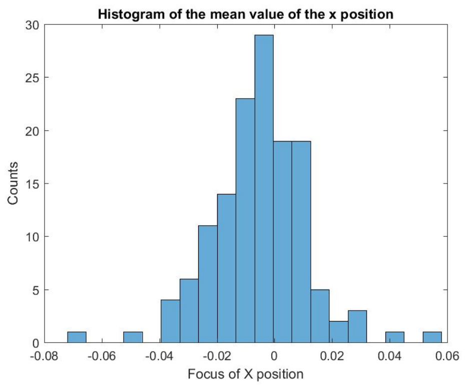
(a)



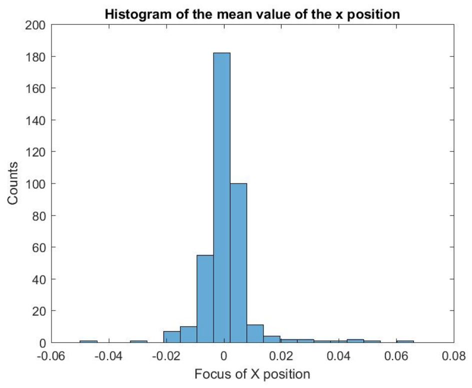
(b)

Figure 6.2: (a) A motion pattern is plotted to illustrate a nonuniform distribution from which the average position is shifted from the center. These type of motion patterns were linked to low slope-values between the diffusion coefficient and motion amplitudes. (b) A motion pattern is plotted to illustrate a uniform distribution from which the average position can be found at the center. These type of motion patterns were linked to high slope-values between the diffusion coefficient and motion amplitudes.

Finally, it was stated that altering components of the TPM system might alter the stiffness of the tether. It is shown that by taking the Fourier transform of the equation of motion of a particle trapped in a parabolic well potential, the trap stiffness can be calculated by analyzing the power spectrum of motion [14] [21]. This method could be used to receive more information on the possible effects of altering components in the TPM system.



(a)



(b)

Figure 6.3: The mean value of the x-position of each particle trajectory in a data set is plotted in a histogram. (a) Trajectories of tethered particles are considered corresponding to a measurement with a low increase of the diffusion coefficient per increase of the motion amplitudes. (b) Trajectories of tethered particles are considered corresponding to a TPM system with a high increase of the diffusion coefficient per increase of the motion amplitudes.

6.3 Conclusion

It is shown that by using different components in the TPM system, the increase of diffusion coefficient per increase of motion amplitude differs. An attempt was done to explain this difference in behavior, but since multiple components were changed per data set, no hard conclusions were drawn on direct causes. Possible correlations were found between slope-values and TPM configurations which might be used for further research.

For instance, there is a strong indication that changing either the tether length or particle type impacts the increase of diffusion coefficient per area size. Experiments could be done in which either one of these two components would be altered, from which the linear correlation between diffusion coefficient and motion amplitudes would be studied. A first goal would be to find whether the particle type, the tether length or both affect the tethered particle behavior. Furthermore, by only altering the tether length between experiments and analyzing the data according to section 6.1, a correlation might be found between the tether length and the increase of diffusion coefficient per increase of motion amplitude.

Moreover, using casein as a coating in the TPM system seems to result in a relatively high increase in the diffusion coefficient per increase of motion amplitude. A possible explanation could be that this coating electrically repulses the particle farther from the surface, decreasing drag on the particle. A closer look needs to be taken on how the system behaves while only altering the surface coating per TPM system. Theory should be studied on how different surface coatings could alter viscous and electrostatic effects.

In short, in the future more experiments should be analyzed to find more understanding of how using different components in the TPM system alters the increase in diffusion coefficient per increase in motion amplitude. Two additional analysis tools could be useful to get more understanding. First, there is an indication that the motion patterns of single tethered particles per TPM system differ, indicating that particles tend to focus more at one position in one system compared to another. Finally, by analyzing the power spectrum of a particle position the tether stiffness could be determinable, which might give additional information on varying interactions when using different components in the TPM system.

Chapter 7

Conclusions

The main goal of the project was to study the usage of Brownian motion analysis for probing particle states in the TPM system. The hypothesis was that in a bound state the particle would show more confined motion, resulting in a lower diffusion coefficient value. This hypothesis was validated by calculating the diffusion coefficient in time of active tethered particles and considering the corresponding motion patterns. Not only information can be obtained on the occurrence of binding and unbinding events, but also on the type of bond the particle forms to the surface.

An overview of different methods for calculating the diffusion coefficient of freely moving or tethered particles was presented in chapter 4. A comparison was made to find which of the considered methods would give the most accurate results. The main principle used for confined motion was that it can be approximated by Brownian motion while taking small time steps between displacements. It was shown that fitting equation (3.3) over the empirical cumulative distribution of the square displacements in a time interval would result in the most accurate determination of the diffusion coefficient. Therefore this method was used to calculate the diffusion coefficient in time, from which binding and unbinding events can be detected.

For probing the particle state the diffusion coefficient was calculated every 20 frames. This means that after every 20 frames, the next 20 frames are used for calculating the next value of the diffusion coefficient. This results in an uncertainty in the bound and unbound state lifetime which is proportional to the time interval used for each calculation of the diffusion coefficient. This uncertainty can be reduced by moving the frame interval over which the diffusion coefficient is calculated by steps of one frame. Following that method, for each frame a value of the diffusion coefficient would be calculated. A disadvantage would be that the time needed for the analysis would increase significantly.

A trade-off was found between time resolution and overlap of the bound and unbound state diffusion coefficient distributions. A low overlap is needed to detect events accurately by using a threshold. When a low number of frames is used, one can detect binding and unbinding events with a higher time resolution, but it needs to be taken into account that event detection becomes less accurate. Finally, it was proposed to use 10 to 30 frames per calculation of the diffusion coefficient, resulting in a time resolution from 0.33 to 1 second.

Furthermore, a derivation was done of the uncertainty of each value of the diffusion coefficient in a time span. This derivation was based on the DKW-inequality and linear regression. The uncertainty in the particle position was not taken into account since the impact of the position uncertainty was lower than the impact of the uncertainty in the empirical cumulative distribution. Following the uncertainty analysis, it was again proposed to use 10 to 30 frames per calculation of the diffusion coefficient.

Event detection by using Brownian motion analysis was compared to event detection by monitoring the particle position. Using a threshold on the diffusion coefficient could result in false positive events when low values of the diffusion coefficient are considered. This is caused by a high overlap of two different bound state diffusion coefficient distributions. By monitoring the particle position also false positives were detected when the particle is in a bound state. False positive events which result from one method would not be detected by using the other method. Therefore it is proposed to combine both methods for more accurate detection of binding and unbinding events. For instance, by monitoring the diffusion coefficient a filter could be constructed for selecting invalid events which were detected by monitoring the particle position. Moreover, it is shown that monitoring the diffusion coefficient gives information on the type of bond formed by the particle to the surface, while monitoring the particle position does not.

Another tethered particle configuration was considered which does not bind to detection molecules located at the surface, but to a detection molecule added to the tether close to the anchoring point. An attempt was done to detect binding and unbinding events of this configuration by monitoring the diffusion coefficient over time. Although some decreases and increases in the diffusion coefficient were observed, it was found that this method would not result in the correct detection of binding and unbinding events. Therefore it is concluded that monitoring the diffusion coefficient in time is not suitable for probing the particle state of this type of tethered particle configuration.

A small study is done on how altering the particle coating, surface coating, particle type, tether length or coupling strategy of a TPM system affects the behavior of the diffusion coefficient of tethered particles. It was found that there is a linear correlation between the diffusion coefficient of a tethered particle and its minor or major amplitude. The slope of this linear correlation differs when different components of the TPM system are used and might be used to explain different particle behavior per system. Yet, not enough data sets were analyzed in this project to draw hard conclusions on the causes of different behavior. Possible future research has been proposed. It is not only proposed to analyze the diffusion coefficient of tethered particles, but information might also be obtained by analyzing motion patterns or by determining the tether stiffness.

Finally, one possible lookout needs to be addressed. It was shown in chapter 5 that using a smaller value of the time step between displacements results in higher values of the diffusion coefficient of tethered particles. Yet, the minimum value of the time step is limited by the frame rate which was used during experiments. Looking back at figure 5.1a, it can be observed that while decreasing the time step value even below the current limit might still result in an increase in the diffusion coefficient. This indicates that the error term in the approximation of equation (3.9) might not be totally neglected while measuring with the currently possible range of time steps between displacements. In future research, it might be interesting to see whether a higher frame rate might result in higher diffusion coefficient values. One limitation might be that the uncertainty in particle position will increase while using a higher frame rate. Yet, an increase of frame rate might not only result in a higher time resolution but also to less overlap between the bound and unbound state diffusion coefficient distributions. This would conclude that Brownian motion analysis would become more powerful since the determination of bound and unbound state lifetimes would become more accurate and precise.

Bibliography

[1] P.C. nelson; C. Zurla; D. Brogioli; J.F. Beausang; L. Finzi; D. Dunlap. Tethered Particle Motion as a Diagnostic of DNA Tether Length. *The Journal of Physical Chemistry*, 110(34), 2006. 1, 2

[2] S. Brinkers; H.R.C. Dietrich; F.H. de Groote; I.T. Young; B. Rieger. The persistence length of double stranded DNA determined using dark field tethered particle motion. *The Journal of Physical Chemistry*, 130(21):17260–17267, 2009. 1, 2

[3] S. Kumar; C. Manzo; C. Zurla; S. Ucuncuoglu; L. Finzi; D. Dunlap. Enhanced Tethered-Particle Motion Analysis Reveals Viscous Effects. *Biophysical Journal*, 106:399–409, 2014. 1, 2, 7, 8, 29, 30

[4] E.W.A. Visser; L.J. van IJzendoorn; M.W.J. Prins. Particle Motion Analysis Reveals Nanoscale Bond Characteristics and Enhances Dynamic Range for Biosensing. *ACS Nano*, 10(3):3093–3101, 2016. 1, 3

[5] E.W.A. Visser; J.Y. Yan; L.J. van IJzendoorn; M.W.J. Prins. Continuous biomarker monitoring by particle mobility sensing with single molecule resolution. *Nature Communications*, 9(2541), 2018. 1, 2, 3

[6] R.M. Lubken. Design of a Two-State Molecular Switch for Continuous Single-Molecule Biosensing: a TPM Feasibility Study, 2017. 1, 2, 4, 5, 6, 13, 30

[7] E.W.A. Visser. *Biosensing Based on Tethered Particle Motion.* PhD thesis, Eindhoven University of Technology, 2017. 1, 3, 4, 5, 8, 19, 25, 30

[8] P. Nelson. *Biological Physics.* W.H. Freeman and Company, 2014. 1, 6

[9] A.V. Weigel; B. Simon; M.M. Tamkun; Diego Krapf. Ergodic and nonergodic processes coexist in the plasma membrane as observed by single-molecule tracking. *PNAS*, 108(16):6438–6443, 2011. 1

[10] A.V. Weigel; S. Ragi; M.L. Reid; E.K. Chong; M.M. Tamkun; D. Krapf. Obstructed diffusion propagator analysis for single-particle tracking. *Physical Review E*, 85(041924), 2012. 1, 7

[11] K.L. Hartman; S. Kim; K. Kim; J.M. Nam. Supported lipid bilayers as dynamic platforms for tethered particles. *Nanoscale*, 10(7):66–76, 2015. 1, 11

[12] B. Lin; J. Yu; S.A. Rice. Direct measurements of constrained Brownian motion of an isolated sphere between two walls. *Physical Review*, 62(3):62–76, 2000. 1, 6

[13] A. Kusumi; Y. Sako; M. Yamamoto. Confined Lateral Diffusion of Membrane Receptors as Studied by Single Particle Tracking (Nanovid Microscopy). Effects of Calcium-induced Differentiation in Cultured Epithelial Cells. *Biophysical Journal*, 65:2021–2040, 1993. 1, 7, 8

[14] E. Schäffer; S.F. Norrelykke; J. Howard. Surface forces and drag coefficients of microspheres near a plane surface measured with optical tweezers. *American Chemical Society*, 23:3654–3665, 2007. 1, 30, 31

[15] M.J. Saxton. *Fundamental Concepts in Biophysics.* Humana Press, 2009. 7, 8

[16] H. Faxén. Die Bewegung Einer Starren Kugel Längs Der Achse Eines Mit Zaner Flusigkeit Gefullten Rohres. *Arkiv för Matematik, Astronomi och Fysik*, 17:667–6685, 1923. 7, 30

[17] N. Ruthardt; D.C. Lamb; C. Bräuchle. Single-particle Tracking as a Quantitative Microscopy-based Approach to Unravel Cell Entry Mechanisms of Viruses and Pharmaceutical Nanoparticles. *The American Society of Gene & Cell Therapy*, 19(7):1199–1211, 2011. 8

[18] A. Vaart. *Asymptotic Statistics.* Cambridge University Press., 1 edition, 1998. 16, 18

[19] J.R. Taylor. *An introduction to error analysis.* University Science Books, U.S., 2 edition, 1997. 18

[20] H.W. Coleman; W.G. Steele. *Experimentation and Uncertainty Analysis for Engineers.* John Wiley Sons Inc., 2 edition, 1999. 19

[21] F.S. Pavone; F. Jülicher; H. Flyvbjerg S.F. Norrelykke; E. Schäffer, J. Howard. Calibration of optical tweezers with positional detection in the back focal plane. *Review of Scientific Instruments*, 77(103101):3654–3665, 2006. 31

Appendix A

Matlab codes

A.1 Diffusion coefficient of free moving particles using mean square displacement

```
 1 %% Setting some parameters
 2 % Parameter 1 : Time interval(current:EndingT = 300 frames;)
 3 EndingT = 100;
 4 % Get all particle number
 5 test_n          =       (1:size(Result,2));
 6 % Parameter : Throw away overlapped particles (current threshold: <= 0)
 7 
 8 test_eliminatej = [];
 9 for i = 1:size(Result,2)
10     test_eliminatej = [];
11     for j = size(Result,2):-1:i+1
12         if any(sqrt( (Result(i).DriftCorrectedTrajectory(1,:) - Result(j).
               DriftCorrectedTrajectory(1,:)).^2 + ...
13             (Result(i).DriftCorrectedTrajectory(2,:) - Result(j).
                   DriftCorrectedTrajectory(2,:)).^2 ) <= 0)
14             %Set these overlapped particles as zero(so can be deleted later at line
                   43)
15             test_n(i) = 0;
16             test_eliminatej = [test_eliminatej,j];
17         end
18     end
19     test_n(test_eliminatej) = 0;
20 end
21 
22 % Get the nonzero number
23 test_n = nonzeros(test_n)';
24 
25 MSD     =   zeros(1,size(EndingT,2));
26 slope   =   zeros(size(test_n,2),2);
27 
28 %% Calculate 1.displacement 2.MSD 3.slope of MSD(For two dimension, slope/4=
       diffusion coefficient)
29 % Run loop for every particle No.i
30 for i = test_n
31     X=Result(i).DriftCorrectedTrajectory(1,:);
32     Y=Result(i).DriftCorrectedTrajectory(2,:);
33 
34     [frames]=size(X);
35     % define the number of frames
36     %% Get information of each position for each bead in pixels.
37     % Parameter 2 : if use different magnification or microscope
```

```
38     %Transfer positions from pixels into um. (1 pixel equals 227?? nm, for 40x magnification)
39     %Transfer positions from pixels into um. (1 pixel equals 794 nm, for 20x magnification)
40     %Change to 460 nm when Leica was used (Leica at 20x magnification).
41     X2=X*.794;
42     Y2=Y*.794;
43 
44     %   j is time interval
45     for j = 1:EndingT
46         % Calculating distance(displacement)
47         % Note: remember to add script for deleting outliers (while calculating mean(X2) and mean(Y2))
48         squaredisp = zeros(1,frames(1,2)-j);
49         for l = 1:frames(1,2)-j
50             % Squared displacement
51             squaredisp(l) = (X2(l+j)-X2(l))^2+(Y2(l+j)-Y2(l))^2;
52         end
53         % Mean
54         MSD(j) = sum(squaredisp(:))/(frames(1,2)-j);
55     end
56 
57 t = (1:EndingT)/30;
58 
59 % Linear fitting of MSD curve
60 slope(i,:) = polyfit(t,MSD,1);
61 end
62 
63 DC = slope(:,1)/4; %Calculate DiffCoef
64 DiffCoef = DC(any(DC,2),:); %Get rid of elements equal to zero
65 
66 %% Plot Histogram
67 close all
68 
69 figure(1)
70 histogram(DiffCoef,0:0.0250:0.5,'FaceColor','b')
71     title('Diffusion coefficient of unbound particles','FontSize',22,'FontName',"Calibri")
72     xlabel('Diffusion coefficient ( m^2/s)','FontSize',22, 'FontName',"Calibri")
73     ylabel('Counts (-)','FontSize',22, 'FontName',"Calibri")
```

A.2 Diffusion coefficient of free moving particles using standard deviation

```
1  %% Set Parameters concerning time
2  TotalFrames         =       1800; %Total number of frames being considered
3  FrameIntervalSteps  =       5;    %Number of framesteps which will be used
4  FrameRate           =       30;
5  
6  %% Read Position Data
7  %Skip overlapping trajectories
8  test_n          =       (1:size(Result,2)); %Get all particle numbers
9  
10 for i = 1:size(Result,2)
11     test_eliminatej = [];
12     for j = size(Result,2):-1:i+1
13         if any(sqrt( (Result(i).DriftCorrectedTrajectory(1,:) - Result(j).DriftCorrectedTrajectory(1,:)).^2 + ...
14             (Result(i).DriftCorrectedTrajectory(2,:) - Result(j).DriftCorrectedTrajectory(2,:)).^2 ) <= 0)
15             %Set these overlapped particles as zero(so can be deleted later at line 43)
16             test_n(i) = 0;
```

```matlab
            test_eliminatej = [test_eliminatej,j];
        end
    end
    test_n(test_eliminatej) = 0;
end

% Get the nonzero number
test_n = nonzeros(test_n)';

X = zeros(size(test_n,2),TotalFrames); %Prelocate matrixes for faster computation
Y = zeros(size(test_n,2),TotalFrames);

for i = test_n %Reed out and process position data
    X(i,:)=Result(i).DriftCorrectedTrajectory(1,1:TotalFrames);
    Y(i,:)=Result(i).DriftCorrectedTrajectory(2,1:TotalFrames);
end
%Get information of each position for each bead in pixels.
    % If use different magnification or microscope
    %Transfer positions from pixels into um. (1 pixel equals 227?? nm, for 40x magnification)
    %Transfer positions from pixels into um. (1 pixel equals 794 nm, for 20x magnification)
    %Change to 460 nm when Leica was used (Leica at 20x magnification).
    X2=X(any(X,2),:)*.794; %Also, get rid of elements equal to zero.
    Y2=Y(any(X,2),:)*.794;

%Prelocate vectors for faster computation
PDFx            =       zeros(1,size(test_n,2));
PDFy            =       zeros(1,size(test_n,2));
RMS_x           =       zeros(size(test_n,2),FrameIntervalSteps);
RMS_y           =       zeros(size(test_n,2),FrameIntervalSteps);
FrameStep   =       zeros(1,FrameIntervalSteps);

%% Calculate Step made particle in x and y direction per FrameInterval.
%Set values for loop
for k = 1:1:FrameIntervalSteps
FrameStep(k) = k;

s               =       2*FrameStep(k);
m               =       FrameStep(k);
e               =       TotalFrames;

dX              =       zeros(size(test_n,2),TotalFrames);
dY              =       zeros(size(test_n,2),TotalFrames);

for i = 1:size(test_n,2) %Calculate for all valid particles particles

    for j = s:m:e %Loop should take intervals of steps equal to the FrameInterval
       dX(i,j) = X2(i,j)-X2(i,j-m);
       dY(i,j) = Y2(i,j)-Y2(i,j-m);
    end

dX2                 =   dX(:,any(dX,1)); % Get rid of zero collumns
dY2                 =   dY(:,any(dY,1));

RMS_x(i,k)          =   std(dX2(i,:))^2; %From paper, RMS in a direction is determined using the standard deviation
RMS_y(i,k)          =   std(dY2(i,:))^2;
end


end
% Result, RMS for ith particle in k timesteps (TimeInterval)

%% SET UP LINEAR FITTING
TimeInterval = (1/FrameRate).*FrameStep;
fit_x = zeros(size(RMS_x,2),2);
fit_y = zeros(size(RMS_x,2),2);
```

```

for i = 1:1:size(RMS_x,1) % The slope of each rms plot should give an expression for diffcoeff
    fit_x(i,:) = polyfit(TimeInterval,RMS_x(i,:),1);
    fit_y(i,:) = polyfit(TimeInterval,RMS_y(i,:),1);
end

% MSD = 2D*t, so devide by 2
DiffCoeff_x = (1/2).*fit_x(:,1);
DiffCoeff_y = (1/2).*fit_y(:,1);

%% PLOT FIGURES
close all

figure(1)
histogram(DiffCoeff_x,0:0.0250:0.5,'FaceColor','b')
    title('Diffusion coefficient of unbound particles in x-direction','FontSize',22,'FontName',"Calibri")
    xlabel('Diffusion coefficient ( m^2/s)','FontSize',22, 'FontName',"Calibri")
    ylabel('Counts (-)','FontSize',22, 'FontName',"Calibri")

figure(2)
histogram(DiffCoeff_y,0:0.0250:0.5,'FaceColor','b')
    title('Diffusion coefficient of unbound particles in y-direction','FontSize',22,'FontName',"Calibri")
    xlabel('Diffusion coefficient ( m^2/s)','FontSize',12, 'FontName',"Calibri")
    ylabel('Counts (-)','FontSize',12, 'FontName',"Calibri")
```

A.3 Diffusion coefficient of free moving particles using the empirical cumulative distribution

```
%% Parameters
FrameStep      =      1;                  % FrameInterval between which the displacement will be considered
TotalFrames    =      1800;               % Set to 1800 frames for analyzing each particle for one minute
FrameRate      =      30;
TimeStep       =      FrameStep/FrameRate;   %Small time interval in which the diffusion coefficient will be determined later in script

% Set up fittype and options.
ft = fittype( '1-exp(-x/(4*a))', 'independent', 'x', 'dependent', 'y' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = 0.0344460805029088;

% Get all particle number
test_n         =      (1:size(Result,2));

%% dummy variables for calculating the displacement in for loop
s              =       2*FrameStep;
m              =       FrameStep;
e              =       TotalFrames;
r              =       zeros(1,(e-s)/m);     %Define matrix for faster computation

% Parameter : Throw away overlapped particles (current threshold: <= 0)
for i = 1:size(Result,2)
    test_eliminatej = [];
    for j = size(Result,2):-1:i+1
        if any(sqrt( (Result(i).DriftCorrectedTrajectory(1,:) - Result(j).DriftCorrectedTrajectory(1,:)).^2 + ...
27             (Result(i).DriftCorrectedTrajectory(2,:) - Result(j).DriftCorrectedTrajectory(2,:)).^2 ) <= 0)
28             %Set these overlapped particles as zero(so can be deleted later at line 43)
29             test_n(i) = 0;
30             test_eliminatej = [test_eliminatej,j];
31         end
32     end
33     test_n(test_eliminatej) = 0;
34 end
35
36 % Get the nonzero number
37 test_n = nonzeros(test_n)';
38
39 X = zeros(size(test_n,2),TotalFrames); %Define matrixes for faster computation
40 Y = zeros(size(test_n,2),TotalFrames);
41
42 for i = test_n %Reed out and process position data
43     X(i,:)=Result(i).DriftCorrectedTrajectory(1,1:TotalFrames);
44     Y(i,:)=Result(i).DriftCorrectedTrajectory(2,1:TotalFrames);
45 end
46
47  %% Get information of each position for each bead in pixels.
48     % If use different magnification or microscope
49     %Transfer positions from pixels into um. (1 pixel equals 227?? nm, for 40x magnification)
50     %Transfer positions from pixels into um. (1 pixel equals 794 nm, for 20x magnification)
51     %Change to 460 nm when Leica was used (Leica at 20x magnification).
52     X2=X(any(X,2),:)*.794; %Also, get rid of elements equal to zero.
53     Y2=Y(any(X,2),:)*.794;
54
55 DiffCoeff = zeros(1,size(test_n,2)); %Define Matrix for faster computation
56
57 for i = 1:size(test_n,2)
58     for j = s:m:e %Loop should take intervals of steps equal to the FrameInterval
59         r(j) = (X2(i,j)-X2(i,j-m))^2+(Y2(i,j)-Y2(i,j-m))^2; %Distance for
60     end
61   r_distance = r(:,any(r,1)); %Get rid of elements equal to zero
62
63 %% The probability for finding the particle at a certain distance or smaller is determined (CDF)
64 [CDF,D] = ecdf(r_distance); %Calculate CDF
65
66 %% Fit: 'Extract data for diffussioncoefficient through fit from CDF
67 [xData, yData] = prepareCurveData(D, CDF );
68
69 % Fit model to data.
70 try %When fit gives an error, skip the fit and  continue with next iteration in for loop.
71 [fitresult, gof] = fit( xData, yData, ft, opts );
72 a = coeffvalues(fitresult); %receive result from fit 1-exp(x/-4a)
73 DiffCoeff(i) = a/TimeStep; %Since a = D*t, devide through the timeinterval
74 catch
75      disp('A Wild False Fit Appeared')
76 end
77 end
78
79 %% Pot histogram of all DifCoeffiecents per dataset
80 close all
81
82 figure(1)
83 histogram(DiffCoeff,0:0.0250:0.5,'FaceColor','b')
84     title('Diffusion coefficient of unbound particles','FontSize',22,'FontName',"Calibri")
85     xlabel('Diffusion coefficient ( m^2/s)','FontSize',22, 'FontName',"Calibri")
86     ylabel('Counts (-)','FontSize',22, 'FontName',"Calibri")
```

A.4 Diffusion coefficient of tethered particles by linear fit through mean square displacement

```
%% Setting some parameters
% Parameter 1 : Time interval(current:EndingT = 300 frames;)
EndingT = 2;
% Get all particle number

%Ignore particles which do not show sufficient symmetry
Pixel = .794;
% Nikon: 794
% M1 microscope: 641
test_n = find([Result.Sym] >= 0.75 & ([Result.MinorDev]*Pixel >= 0.050) & ([Result.
    MinorDev]*Pixel <= .180) );

MSD     =   zeros(1,size(EndingT,2));
slope   =   zeros(size(test_n,2),2);

%% Calculate 1.displacement 2.MSD 3.slope of MSD(For two dimension, slope/4=
    diffusion coefficient)
% Run loop for every particle No.i
for i = test_n
    X=Result(i).DriftCorrectedTrajectory(1,:);
    Y=Result(i).DriftCorrectedTrajectory(2,:);

    [frames]=size(X);
    % define the number of frames
    %% Get information of each position for each bead in pixels.
    % Parameter 2 : if use different magnification or microscope
    %Transfer positions from pixels into um. (1 pixel equals 227?? nm, for 40x
        magnification)
    %Transfer positions from pixels into um. (1 pixel equals 794 nm, for 20x
        magnification)
    %Change to 460 nm when Leica was used (Leica at 20x magnification).
    X2=X*.794;
    Y2=Y*.794;

    %   j is time interval
    for j = 1:EndingT
        % Calculating distance(displacement)
        % Note: remember to add script for deleting outliers (while calculating
            mean(X2) and mean(Y2))
        squaredisp = zeros(1,frames(1,2)-j);
        for l = 1:frames(1,2)-j
            % Squared displacement
            squaredisp(l) = (X2(l+j)-X2(l))^2+(Y2(l+j)-Y2(l))^2;
        end
        % Mean
        MSD(j) = sum(squaredisp(:))/(frames(1,2)-j);
    end

t = (1:EndingT)/30;

% Linear fitting of MSD curve
slope(i,:) = polyfit(t,MSD,1);
end

DC = slope(:,1)/4; %Calculate DiffCoef
DiffCoef = DC(any(DC,2),:); %Get rid of elements equal to zero

%% Plot Histogram
histogram(DiffCoef,0:0.0075:0.15,'FaceColor','b')
    title('Diffusion coefficient of tethered particles','FontSize',22,'FontName',"
        Calibri")
    xlabel('Diffusion coefficient ( m^2/s)','FontSize',22, 'FontName',"Calibri")
    ylabel('Counts (-)','FontSize',22, 'FontName',"Calibri")
```

A.5 Diffusion coefficient of tethered particles by using the standard deviation

```
%% Set Parameters concerning time
TotalFrames         =       1800; %Total number of frames being considered
FrameStep           =       1;
FrameRate           =       30;
TimeStep            =       FrameStep/FrameRate;

%% Read Position Data
%Ignore particles which do not show sufficient symmetry
Pixel = .794;
% Nikon: 794
% M1 microscope: 641
test_n = find([Result.Sym] >= 0.75 & ([Result.MinorDev]*Pixel >= 0.050) & ([Result.
    MinorDev]*Pixel <= .180) );
X = zeros(size(test_n,2),TotalFrames); %Prelocate matrixes for faster computation
Y = zeros(size(test_n,2),TotalFrames);

for i = test_n %Reed out and process position data
    X(i,:)=Result(i).DriftCorrectedTrajectory(1,1:TotalFrames);
    Y(i,:)=Result(i).DriftCorrectedTrajectory(2,1:TotalFrames);
end
%Get information of each position for each bead in pixels.
    % If use different magnification or microscope
    %Transfer positions from pixels into um. (1 pixel equals 227?? nm, for 40x
        magnification)
    %Transfer positions from pixels into um. (1 pixel equals 794 nm, for 20x
        magnification)
    %Change to 460 nm when Leica was used (Leica at 20x magnification).
    X2=X(any(X,2),:)*.794; %Also, get rid of elements equal to zero.
    Y2=Y(any(X,2),:)*.794;

%Prelocate vectors for faster computation
PDFx            =       zeros(1,size(test_n,2));
PDFy            =       zeros(1,size(test_n,2));
MSD_x           =       zeros(size(test_n,2),1);
MSD_y           =       zeros(size(test_n,2),1);

%% Calculate Step made particle in x and y direction per FrameInterval.
%Set values for loop


s               =       2*FrameStep;
m               =       FrameStep;
e               =       TotalFrames;

dX              =       zeros(size(test_n,2),TotalFrames);
dY              =       zeros(size(test_n,2),TotalFrames);
DiffCoeff_x     =       zeros(1,size(test_n,2));
DiffCoeff_y     =       zeros(1,size(test_n,2));

for i = 1:size(test_n,2) %Calculate for all valid particles particles

    for j = s:m:e %Loop should take intervals of steps equal to the FrameInterval
       dX(i,j) = X2(i,j)-X2(i,j-m);
       dY(i,j) = Y2(i,j)-Y2(i,j-m);
    end

dX2                 =   dX(:,any(dX,1)); % Get rid of zero collumns
dY2                 =   dY(:,any(dY,1));

MSD_x(i)            =   std(dX2(i,:))^2; %From paper, RMS in a direction is
    determined using the standard deviation
MSD_y(i)            =   std(dY2(i,:))^2;

```

```
DiffCoeff_x(i)          =    MSD_x(i)/(2*TimeStep);
DiffCoeff_y(i)          =    MSD_y(i)/(2*TimeStep);
end


%% PLOT FIGURES
figure(1)
histogram(DiffCoeff_x,0:0.0075:0.15,'FaceColor','b')
    title('Diffusion coefficient of tethered particles','FontSize',22,'FontName',"
        Calibri")
    xlabel('Diffusion coefficient ( m^2/s)','FontSize',22, 'FontName',"Calibri")
    ylabel('Counts (-)','FontSize',22, 'FontName',"Calibri")

figure(2)
histogram(DiffCoeff_y,0:0.00375:0.15)
xlabel('Diffusion Coefficient (um^2/s)')
ylabel('Number of Counts (-)')
title('Diffusion Coefficient in the y-direction')
```

A.6 Diffusion coefficient of tethered particles by using the empirical cumulative distribution

```
%% Parameters
FrameInterval    =        1;                      % FrameInterval in which the
    displacement will be considered
TotalFrames      =        1800;                   % Set to 1800 frames for analyzing
    each particle for one minute
FrameRate        =        30;
TimeInterval    =         FrameInterval/FrameRate; %Small time interval in which the
    diffusion coefficient will be determined later in script


% Set up fittype and options.
ft = fittype( '1-exp(-x/(4*a))', 'independent', 'x', 'dependent', 'y' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = 0.003;

%% dummy variables for calculating the displacement in for loop
s                =         2*FrameInterval;
m                =         FrameInterval;
e                =         TotalFrames;
r                =         zeros(1,(e-s)/m);        %Define matrix for faster
    computation

%Ignore particles which do not show sufficient symmetry
Pixel = .794;
% Nikon: 794
% M1 microscope: 641
test_n = find([Result.Sym] >= 0.75 & ([Result.MinorDev]*Pixel >= 0.050) & ([Result.
    MinorDev]*Pixel <= .180) );

X = zeros(size(test_n,2),TotalFrames); %Define matrixes for faster computation
Y = zeros(size(test_n,2),TotalFrames);

for i = test_n %Reed out and process position data
    X(i,:)=Result(i).DriftCorrectedTrajectory(1,1:TotalFrames);
    Y(i,:)=Result(i).DriftCorrectedTrajectory(2,1:TotalFrames);
end

%% Get information of each position for each bead in pixels.
    % If use different magnification or microscope
    %Transfer positions from pixels into um. (1 pixel equals 227?? nm, for 40x magnification)
    %Transfer positions from pixels into um. (1 pixel equals 794 nm, for 20x magnification)
    %Change to 460 nm when Leica was used (Leica at 20x magnification).
    X2=X(any(X,2),:)*.794; %Also, get rid of elements equal to zero.
    Y2=Y(any(X,2),:)*.794;

DiffCoeff = zeros(1,size(test_n,2)); %Prelocate Diffussion Coefficient Matrix for faster computation

for i = 1:size(test_n,2)
    for j = s:m:e %Loop should take intervals of steps equal to the FrameInterval
        r(j) = (X2(i,j)-X2(i,j-m))^2+(Y2(i,j)-Y2(i,j-m))^2; %Distance for
    end
  r_distance = r(:,any(r,1)); %Get rid of elements equal to zero

%% The probability for finding the particle at a certain distance or smaller is determined (CDF)
[CDF,D] = ecdf(r_distance); %Calculate CDF

%% Fit: 'Extract data for diffussioncoefficient through fit from CDF
[xData, yData] = prepareCurveData(D, CDF );

% Fit model to data.
try %When fit gives an error, skip the fit and  continue with next iteration in for loop.
[fitresult, ~] = fit( xData, yData, ft, opts );
catch
    try
        opts.StartPoint = 0.0003;
        [fitresult, ~] = fit( xData, yData, ft, opts );
        opts.StartPoint = 0.003;
    catch
     disp('False Fit!')
    end
end

a = coeffvalues(fitresult); %receive result from fit 1-exp(x/-4a)
DiffCoeff(i) = a/TimeInterval; %Since a = D*t, devide through the timeinterval
end
%%
figure(1)
histogram(DiffCoeff,0:0.0075:0.15,'FaceColor','b')
    title('Diffusion coefficient of tethered particles','FontSize',22,'FontName',"Calibri")
    xlabel('Diffusion coefficient ( m^2/s)','FontSize',22, 'FontName',"Calibri")
    ylabel('Counts (-)','FontSize',22, 'FontName',"Calibri")
```

A.7 Diffusion coefficient of tethered particles by using a non-linear fit through the mean square displacement

```
%% Setting some parameters
% Parameter 1 : Time interval
EndingT = 150; %Don't set too hight since false fits can occur

%% Particle selection

%Ignore particles which do not show sufficient symmetry
Pixel = .794;
% Nikon: 794
% M1 microscope: 641
```

```
test_n = find([Result.Sym] >= 0.75 & ([Result.MinorDev]*Pixel >= 0.050) & ([Result.
    MinorDev]*Pixel <= .180) );

%% Calculate 1.displacement 2.MSD 3.slope of MSD(For two dimension, slope/4=
    diffusion coefficient)

%Prelocating for faster computation
X            = zeros(1,size(test_n,2));
MSD          = zeros(1,size(EndingT,2));
distance_r   = zeros(size(test_n,2),size(EndingT,2));
DiffCoeff    = zeros(1,size(test_n,2));

% Run loop for every particle No.i
for i = test_n
    X=Result(i).DriftCorrectedTrajectory(1,:);
    Y=Result(i).DriftCorrectedTrajectory(2,:);
    % define the number of frames
    [frames]=size(X);

    %% Get information of each position for each bead in pixels.
    % Parameter 2 : if use different magnification or microscope
    %Transfer positions from pixels into um. (1 pixel equals 227?? nm, for 40x
        magnification)
    %Transfer positions from pixels into um. (1 pixel equals 794 nm, for 20x
        magnification)
    %Change to 460 nm when Leica was used (Leica at 20x magnification).
    X2=X*.794;

    %  j is time interval
    for j = 1:EndingT
        % Calculating distance(displacement)
        % Note: remember to add script for deleting outliers (while calculating
            mean(X2) and mean(Y2))
        distance_r(i,j) = sqrt ((X2(j)-mean(X2))^2);
        squaredisp = zeros(1,frames(1,2)-j);
        for l = 1:frames(1,2)-j
            % Squared displacement
            squaredisp(l) = (X2(l+j)-X2(l))^2;
        end
        % Mean
        MSD(j) = sum(squaredisp(:))/(frames(1,2)-j);
    end

t = (1:EndingT)/30;

%% Fit: 'untitled fit 1'.
[xData, yData] = prepareCurveData( t, MSD );

% Set up fittype and options.
ft = fittype( '(L^2/6)*(1-96*((1/625)*exp(-250*(pi^2)*x/(L^2))+(1/81)*exp(-90*(pi
    ^2)*x/(L^2))+exp(-D*(pi^2)*x/(L^2)))/pi^4)', 'independent', 'x', 'dependent', '
    y' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = [0.913375856139019 0.63235924622541];

% Fit model to data.
try
[fitresult, gof] = fit( xData, yData, ft, opts );
catch
disp('false fit')
end

DiffCoeff(i) = fitresult.D;
end

close all
```

```matlab

DiffCoeff = DiffCoeff(:,any(DiffCoeff,1)); % Get Rid of zero elements

figure(1)
histogram(DiffCoeff,0:0.0075:0.15,'FaceColor','b')
    title('Diffusion coefficient of tethered particles','FontSize',22,'FontName',"
        Calibri")
    xlabel('Diffusion coefficient ( m^2/s)','FontSize',22, 'FontName',"Calibri")
    ylabel('Counts (-)','FontSize',22, 'FontName',"Calibri")
```

A.8 Diffusion coefficient over time with error bar

```matlab
%% I: Parameters
FrameStep       =   1;                    %Step over which distances will be
    calculated
FrameRate       =   30;
TotalFrames     =   size(Result(2).DriftCorrectedTrajectory,2);              %
    Total processing time
FrameInterval   =   20;                   % Control number of measurements per
    second
TimeStep        =   FrameStep/FrameRate;  %Timestep used later to calculate
    Diffusion coeficient

prompt = 'Which particle number do you want to consider?';
PN = input(prompt);

% Set Fit options for later in the script:
ft = fittype( '1-exp(-x/(4*a))', 'independent', 'x', 'dependent', 'y' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = 0.0001;

Pixel           =   .794;
%Transfer positions from pixels into um. (1 pixel equals 227?? nm, for 40x
    magnification)
%Transfer positions from pixels into um. (1 pixel equals 794 nm, for 20x
    magnification)
%Change to 460 nm when Leica was used (Leica at 20x magnification).

%Prelocate matrices for faster computation
 r_distance              = zeros(1,FrameInterval);
 dr                      = zeros(1,FrameInterval);
 DiffCoeff               = zeros(1,(TotalFrames-FrameInterval)/FrameInterval);
 Time                    = zeros(1,(TotalFrames-FrameInterval)/FrameInterval);
 DC_LinReg_DKW           = zeros(1,(TotalFrames-FrameInterval)/FrameInterval);
 S_DC_LinReg_DKW         = zeros(1,(TotalFrames-FrameInterval)/FrameInterval);

%Read out position data
X               =   Result(PN).DriftCorrectedTrajectory(1,1:TotalFrames)*Pixel; %
    Import data on position of N'th bead
Y               =   Result(PN).DriftCorrectedTrajectory(2,1:TotalFrames)*Pixel;

%% II: Start Computation of Diffusion Coefficient using nonlinear fitting
for i= FrameInterval : FrameInterval : TotalFrames-FrameInterval

Time(i) = i/FrameRate;

    for j = i:FrameStep:(i+FrameInterval) %Loop should take intervals of steps
        equal to the FrameInterval
        r_distance(j-i+1) = (X(j)-X(j-FrameStep))^2+(Y(j)-Y(j-FrameStep))^2; %
            Distance for
    end
% Get rid of zero elements:
```

```
r_distance = r_distance(:,any(r_distance,1));
dr = dr(:,any(dr,1));

[CDF,D] = ecdf(r_distance); %Calculate ECDF

%Start fitting CDF for extracting Diffusion coefficient
%prepare fit data
[xData, yData] = prepareCurveData( D, CDF );

% receive result from fit 1-exp(x/-4a)
            try %When fit gives an error, adjust options of fit and retry
                [fitresult, gof] = fit( xData, yData, ft, opts );
                DiffCoeff(i) = coeffvalues(fitresult)/TimeStep;
            catch
                try
                    opts.StartPoint = 0.00001; %Set start value to lower value
                    %, since bound particles have way smaller diffusion coefficient
                    [fitresult, gof] = fit( xData, yData, ft, opts );
                    opts.StartPoint = 0.0001;
                    DiffCoeff(i) = coeffvalues(fitresult)/TimeStep;
                catch
                disp(['False Fit at ', num2str(Time(i))]) %If still a bad fit is
                    found, skip iteration and set Diffusion coefficient equal to
                    zero
                DiffCoeff(i) = 0;
                end
            end

%% III Data Analysis
%% Uncertainty Analysis: LINEAR REGRESSION Taking into account variance in ECDF (
    DKW Inequality)

alpha = 0.32; %set alpha equal to 0.32 to find 68%-interval
epsilon = sqrt(log(2/alpha)/(2*size(D,1))); % Determine Epsilon according to DKW
     inequality
y = log(1./(1-CDF));
x = D;
S_CDF = epsilon;
S_y = sqrt( (S_CDF.^2)./((1-CDF).^2) );

x(size(x,1))=[];  %delete the element for which log(1-CDF) equals infinity
y(size(y,1))=[];  %lenght of y needs to match the lenght of x for valid analysis
S_y(size(S_y,1)) =[];
x(1) = [];
y(1) = [];
S_y(1)  = [];

% Start using formula Taylor for linear regression (Determine a and S_a of
% the formula y = ax
w = (1./S_y.^2);
delta = sum(w)*sum(w.*x.*x) - (sum(w.*x))^2;

a = (sum(w)*sum(w.*x.*y)-sum(w.*x)*sum(w.*y))/delta;
S_a = sqrt(sum(w)/delta);

DC_LinReg_DKW(i)        =   abs(1/(4*a*TimeStep));
S_DC_LinReg_DKW(i)      =   sqrt((S_a^2)/(16*(a^4)*(TimeStep^2)) );
end
%% IV: Start plotting the result
% Get rid of zero elements in all relevant matrices:
Time                    =   Time(:,any(Time,1)); % Get rid of zero elements
DiffCoeff               =   DiffCoeff(:,any(DiffCoeff,1)); % Get rid of zero
     elements
DC_LinReg_DKW           =   DC_LinReg_DKW(:, any(DC_LinReg_DKW,1));
S_DC_LinReg_DKW         =   S_DC_LinReg_DKW(:, any(S_DC_LinReg_DKW,1));

close all
```

```

figure(1)
plotbrowser('on')
plot(Time,DiffCoeff);xlim([0 350])
    title('Diffusion Coefficient calculated over time','FontSize',25, 'FontName',"
        Calibri")
    a = get(gca,'XTickLabel');
    set(gca,'XTickLabel',a,'FontName','Calibri','fontsize',18)
    ylabel('Diffusion Coefficient ( m^2/s)','FontSize',25, 'FontName',"Calibri")
    xlabel('Time (s)','FontSize',25, 'FontName',"Calibri")

figure(2)
plotbrowser('on')
errorbar(Time, DC_LinReg_DKW, S_DC_LinReg_DKW, 'o')
set(gca, 'YScale','log')
    title('Diffusion Coefficient calculated over time with uncertainty','FontSize'
        ,25, 'FontName',"Calibri")
    ylabel('Diffusion Coefficient ( m^2/s)','FontSize',25, 'FontName',"Calibri")
    xlabel('Time (s)','FontSize',25, 'FontName',"Calibri")
```

A.9 Diffusion coefficient threshold

For using this code, also use the function vline as written underneath.

```
%% 1: Parameters
FrameStep       =   1;                      %Step over which distances will be
    calculated
FrameRate       =   30;
TotalFrames     =   size(Result(2).DriftCorrectedTrajectory,2);              %
    Total processing time
FrameInterval   =   20;                     % Control number of measurements per
    second
TimeStep        =   FrameStep/FrameRate;    %Timestep used later to calculate
    Diffusion coeficient


prompt = 'Which particle number do you want to consider?';
PN = input(prompt);

% Set Fit options for later in the script:
ft = fittype( '1-exp(-x/(4*a))', 'independent', 'x', 'dependent', 'y' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = 0.0001;

Pixel           =   .794;
%Transfer positions from pixels into um. (1 pixel equals 227?? nm, for 40x
    magnification)
%Transfer positions from pixels into um. (1 pixel equals 794 nm, for 20x
    magnification)
%Change to 460 nm when Leica was used (Leica at 20x magnification).

%% 2. Calculate Diffusion Coefficients
%Prelocate matrices for faster computation
 r_distance                 = zeros(1,FrameInterval);
 dr                         = zeros(1,FrameInterval);
 DiffCoeff                  = zeros(1,(TotalFrames-FrameInterval)/FrameInterval);
 DC_LinReg_SSE              = zeros(1,(TotalFrames-FrameInterval)/FrameInterval);
 S_DC_LinReg_SSE            = zeros(1,(TotalFrames-FrameInterval)/FrameInterval);
 Time                       = zeros(1,(TotalFrames-FrameInterval)/FrameInterval);

%Read out position data
```

```matlab
X                 =    Result(PN).DriftCorrectedTrajectory(1,1:TotalFrames)*Pixel; % Import data on position of N'th bead
Y                 =    Result(PN).DriftCorrectedTrajectory(2,1:TotalFrames)*Pixel;

% Determine middle of motion pattern
        SortedX = sort(X);  % Sort all the x positions in an array from low to high
        SortedY = sort(Y);

        PercentDisregard = 0.003;   % Percentage from the sorted array that is disregarded to determine the center

        X_BottomFivePercent = SortedX(ceil(TotalFrames .* PercentDisregard));  % From the sorted datapoints a certain percentage on the bottom is disregarded, a list is created that does not contain these points
        X_TopFivePercent = SortedX(ceil(TotalFrames .* (1-PercentDisregard)));
        X_Center = (X_BottomFivePercent + X_TopFivePercent) ./ 2;

        Y_BottomFivePercent = SortedY(ceil(TotalFrames .* PercentDisregard));
        Y_TopFivePercent = SortedY(ceil(TotalFrames .* (1-PercentDisregard)));
        Y_Center = (Y_BottomFivePercent + Y_TopFivePercent) ./ 2;

        % Centered X and Y values:
        X_Centered = X - X_Center;
        Y_Centered = Y - Y_Center;

for i= FrameInterval : FrameInterval : TotalFrames-FrameInterval

Time(i) = i/FrameRate;

    for j = i:FrameStep:(i+FrameInterval) %Loop should take intervals of steps equal to the FrameInterval
        r_distance(j-i+1) = (X(j)-X(j-FrameStep))^2+(Y(j)-Y(j-FrameStep))^2; % Distance for
    end
% Get rid of zero elements:
r_distance = r_distance(:,any(r_distance,1));

[CDF,D] = ecdf(r_distance); %Calculate CDF

%Start fitting CDF for extracting Diffusion coefficient
%prepare fit data
[xData, yData] = prepareCurveData( D, CDF );

% receive result from fit 1-exp(x/-4a)
            try %When fit gives an error, adjust options of fit and retry
                [fitresult, gof] = fit( xData, yData, ft, opts );
                DiffCoeff(i) = coeffvalues(fitresult)/TimeStep;
            catch
                try
                    opts.StartPoint = 0.00001; %Set start value to lower value
                    %, since bound particles have way smaller diffusion coefficient
                    [fitresult, gof] = fit( xData, yData, ft, opts );
                    opts.StartPoint = 0.0001;
                    DiffCoeff(i) = coeffvalues(fitresult)/TimeStep;
                catch
                disp(['False Fit at ', num2str(Time(i))]) %If still a bad fit is found, skip iteration and set Diffusion coefficient equal to zero
                DiffCoeff(i) = 0;
                end
            end
end
% Get rid of zero elements in all relevant matrices:
Time                        =    Time(:,any(Time,1)); % Get rid of zero elements
DiffCoeff                   =    DiffCoeff(:,any(DiffCoeff,1)); % Get rid of zero elements

```

```
%% 4. Define Threshold
clear SelectedX
close all;
f = figure;
subplot(2,1,1)
histogram(DiffCoeff,50); hold on
title('Histogram of found values of the Diffusion Coefficient')
xlabel('Diffusion Coefficient ( m^2/s)')
ylabel('Counts (-)')
subplot(2,1,2)
plot(Time,DiffCoeff); hold on
ylabel('Diffusion Coefficient ( m^2/s)')
xlabel('Time (s)')
title('Click on vertical threshold and press ENTER')
[~, SelectedX] = ginput();
hold off
close (f)

%Define needed threshold for analysation
Sigma = 0.25*SelectedX;
Thresholds.Sigma = Sigma;

Thresholds.Threshold = SelectedX;

%% 5. Use Threshold for finding events.

StateTrace = zeros(1,length(DiffCoeff));
% First look at what the begin state of the particle is:
if DiffCoeff(1) >= Thresholds.Threshold
    StateTrace(1) = 1;

else
    StateTrace(1) = 0;
end

%Now find the state changes of the particle on the rest of the events.
for m = 2:length(DiffCoeff)
     if DiffCoeff(m) >= Thresholds.Threshold+Thresholds.Sigma
        StateTrace(m) = 1;
    elseif DiffCoeff(m) < Thresholds.Threshold-Thresholds.Sigma
        StateTrace(m) = 0;
    else
        StateTrace(m) = StateTrace(m-1);
    end
end

%% 6. Calculate the binding times
clear UnbindingEvent BindingEvent
Counter1 = 1;
Counter2 = 1;

if StateTrace(1) == 0
    BindingEvent(Counter1) = 1;
    Counter1 = Counter1 + 1;

elseif StateTrace(1) == 1
    UnbindingEvent(Counter2) = 1;
    Counter2 = Counter2 + 1;
else
    error('error');
end

for i = 1:length(StateTrace)-1

    if StateTrace(i) == 1 && StateTrace(i+1) == 0

        BindingEvent(Counter1) = Time (i+1);
        Counter1 = Counter1 + 1;

    elseif StateTrace(i) == 0 && StateTrace(i+1) == 1

        UnbindingEvent(Counter2) = Time (i);
        Counter2 = Counter2 + 1;

    end

end


if StateTrace(1) == 0 && length(BindingEvent) ~= length(UnbindingEvent)

    BoundTimes = abs(BindingEvent(1:end-1)-UnbindingEvent(1:end));
    UnboundTimes = abs(UnbindingEvent(1:end)-BindingEvent(2:end));

elseif StateTrace(1) == 0 && length(BindingEvent) == length(UnbindingEvent)

    BoundTimes = abs(BindingEvent(1:end)-UnbindingEvent(1:end));
    UnboundTimes = abs(UnbindingEvent(1:end-1)-BindingEvent(2:end));

elseif StateTrace(1) == 1 && length(BindingEvent) ~= length(UnbindingEvent)

    BoundTimes = abs(BindingEvent(1:end)-UnbindingEvent(2:end));
    UnboundTimes = abs(UnbindingEvent(1:end-1)-BindingEvent(1:end));

elseif StateTrace(1) == 1 && length(BindingEvent) == length(UnbindingEvent)

    BoundTimes = abs(BindingEvent(1:end-1)-UnbindingEvent(2:end));
    UnboundTimes = abs(UnbindingEvent(1:end)-BindingEvent(1:end));

end

%% 7. Plot Results
% Read out the events from code Emiel based on motion patterns.
EventFrames = [Result(PN).DetectedEventTimes{1,1}];
LifeTimeFrames = [Result(PN).DetectedEventLifetimes{1,1}];
EventTimes = EventFrames/FrameRate;
LifeTimes  = LifeTimeFrames/FrameRate;

close all

figure(1)
hold on
plot(Time, StateTrace)
vline(EventTimes)
plotbrowser('on')
title('State of particle over time with current threshold (with events from Emiel
    for comparison)')
xlabel('Time (s)')
ylabel('State (0 = bound, 1 = unbound)')
ylim([-0.3 1.3])
hold off

figure(2)
hold on
plot(Time, DiffCoeff)
vline(BindingEvent,'g')
vline(UnbindingEvent,'r')
plotbrowser('on')
title({'Plot of Filtered Diffusion Coefficient with binding events from threshold';
    'g = Binding event & r = Unbinding event'})
xlabel('Time(s)')
ylabel('Filtered Diffusion Coefficient (um^2/s)')

figure(3)
```

```
hold on
plot(Time, DiffCoeff)
plotbrowser('on')
vline(EventTimes)
title('Diffusion Coefficient calculated over time (Events by code Emiel)')
xlabel('Time(s)')
ylabel('Diffusion Coefficient (um^2/s)')
hold off

%% Plot Binding Events in intervals from either 0. Emiels's Result 1. Result
    DiffCoeff
prompt = 'Do you want to use code Emiel or Diffusion Coefficient Thresholding (0 =
    Emiel, 1 = DiffCoeff)';
Opt = input(prompt);

if Opt == 0
    % First plot the first traject
if max(X_Centered(1:EventFrames(1)))-min(X_Centered(1:EventFrames(1))) > 0.8*max(
    X_Centered) || max(Y_Centered(1:EventFrames(1)))-min(Y_Centered(1:EventFrames
    (1))) > 0.8*max(Y_Centered)
figure(4)
plotbrowser on
plot(X_Centered(1:EventFrames(1)),Y_Centered(1:EventFrames(1)),'.','color','k');
    hold on
xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])

figure(5)
plotbrowser on
plot(Time(1:(EventFrames(1)/FrameInterval)),DiffCoeff(1:(EventFrames(1)/
    FrameInterval)),'color','k'); hold on
ylim([0 max(DiffCoeff)*1.1])
vline(EventFrames/FrameRate); hold on
ylim([0 max(DiffCoeff)*1.1])

else

figure(4)
plotbrowser on
plot(X_Centered(1:EventFrames(1)),Y_Centered(1:EventFrames(1)),'.'); hold on
xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])

figure(5)
plotbrowser on
plot(Time(1:(EventFrames(1)/FrameInterval)),DiffCoeff(1:(EventFrames(1)/
    FrameInterval))); hold on
ylim([0 max(DiffCoeff)*1.1])
vline(EventFrames/FrameRate); hold on
ylim([0 max(DiffCoeff)*1.1])
end

pause
% Now start plotting the middle traject using a for loop
for i = 1:1:(size(EventFrames,2))-1
    Start           =   EventFrames(i);
    End             =   EventFrames(i+1);
    plotbrowser on

    if max(X_Centered(Start:End))-min(X_Centered(Start:End)) > 0.8*max(2*X_Centered
        ) || max(Y_Centered(Start:End))-min(Y_Centered(Start:End)) > 0.8*max(2*
        Y_Centered)
    figure(4)
    plotbrowser on
    plot(X_Centered(Start:End),Y_Centered(Start:End),'.','color','k','LineWidth'
        ,0.01); hold on
    xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
    ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])
    xlabel('Horizontal Displacement (um)')
    ylabel('Vertical Displacement (um)')

    figure(5)
    plotbrowser on
    plot(Time((Start/FrameInterval):(End/FrameInterval)),DiffCoeff((Start/
        FrameInterval):(End/FrameInterval)),'color','k'); hold on
    ylim([0 max(DiffCoeff)*1.1])
    xlabel('Time (s)')
    ylabel('Diffusion Coefficient (um^2/s)')
    else
    figure(4)
    plotbrowser on
    plot(X_Centered(Start:End),Y_Centered(Start:End),'.'); hold on
    xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
    ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])
    xlabel('Horizontal Displacement (um)')
    ylabel('Vertical Displacement (um)')

    figure(5)
    plotbrowser on
    plot(Time((Start/FrameInterval):(End/FrameInterval)),DiffCoeff((Start/
        FrameInterval):(End/FrameInterval)),'LineWidth',2); hold on
    ylim([0 max(DiffCoeff)*1.1])
    xlabel('Time (s)')
    ylabel('Diffusion Coefficient (um^2/s)')
    end
    pause
end

% Plot the last section
if max(X_Centered(max(EventFrames):size(X_Centered,2)))-min(X_Centered(max(
    EventFrames):size(X_Centered,2))) > 0.8*max(2*X_Centered) || max(Y_Centered(max
    (EventFrames):size(Y_Centered,2)))-min(Y_Centered(max(EventFrames):size(
    Y_Centered,2))) > 0.8*max(2*Y_Centered)
    figure(4)
    plot(X_Centered(max(EventFrames):size(X_Centered,2)),Y_Centered(max(EventFrames
        ):size(X_Centered,2)),'.','color','k'); hold on
    xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
    ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])
    xlabel('Horizontal Displacement (um)')
    ylabel('Vertical Displacement (um)')

    figure(5)
    plot(Time((max(EventFrames)/FrameInterval):(size(X_Centered,2)/FrameInterval-1)
        ),DiffCoeff((max(EventFrames)/FrameInterval):(size(X_Centered,2)/
        FrameInterval-1)),'color','k'); hold on
    ylim([0 max(DiffCoeff)*1.1])
    xlabel('Time (s)')
    ylabel('Diffusion Coefficient (um^2/s)')
    else
    figure(4)
    plot(X_Centered(max(EventFrames):size(X_Centered,2)),Y_Centered(max(EventFrames
        ):size(X_Centered,2)),'.'); hold on
    xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
    ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])
    xlabel('Horizontal Displacement (um)')
    ylabel('Vertical Displacement (um)')

    figure(5)
    plot(Time((max(EventFrames)/FrameInterval):(size(X_Centered,2)/FrameInterval-1)
        ),DiffCoeff((max(EventFrames)/FrameInterval):(size(X_Centered,2)/
        FrameInterval-1)),'LineWidth',2); hold on
    ylim([0 max(DiffCoeff)*1.1])
    xlabel('Time (s)')
    ylabel('Diffusion Coefficient (um^2/s)')
```

```
end

elseif Opt == 1
    Events = [BindingEvent*FrameRate, UnbindingEvent*FrameRate]; % First calculate
        seconds back to frames
    Events = sort(Events);

    % First plot the first traject
if max(X_Centered(1:Events(1)))-min(X_Centered(1:Events(1))) > 0.8*max(2*X_Centered
    ) || max(Y_Centered(1:Events(1)))-min(Y_Centered(1:Events(1))) > 0.8*max(2*
    Y_Centered)
figure(4)
plotbrowser on
plot(X_Centered(1:Events(1)),Y_Centered(1:Events(1)),'.','color','k'); hold on
xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])

figure(5)
plotbrowser on
plot(Time(1:(Events(1)/FrameInterval)),DiffCoeff(1:(Events(1)/FrameInterval)),'
    color','k'); hold on
ylim([0 max(DiffCoeff)*1.1])
vline(Events/FrameRate); hold on
ylim([0 max(DiffCoeff)*1.1])
else

figure(4)
plotbrowser on
plot(X_Centered(1:Events(1)),Y_Centered(1:Events(1)),'.'); hold on
xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])

figure(5)
plotbrowser on
plot(Time(1:(Events(1)/FrameInterval)),DiffCoeff(1:(Events(1)/FrameInterval)));
    hold on
ylim([0 max(DiffCoeff)*1.1])
vline(Events/FrameRate); hold on
ylim([0 max(DiffCoeff)*1.1])
end
pause


% Continue plotting for the other trajects
for i = 1:1:(size(Events,2)-1)
    Start         =   Events(i);
    End           =   Events(i+1);
    plotbrowser on

    if max(X_Centered(Start:End))-min(X_Centered(Start:End)) > 0.8*max(2*X_Centered
        ) || max(Y_Centered(Start:End))-min(Y_Centered(Start:End)) > 0.8*max(2*
        Y_Centered)
    figure(4)
    plot(X_Centered(Start:End),Y_Centered(Start:End),'.','color','k','LineWidth'
        ,0.01); hold on
    xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
    ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])
    xlabel('Horizontal Displacement (um)')
    ylabel('Vertical Displacement (um)')

    figure(5)
    plot(Time((Start/FrameInterval):(End/FrameInterval)),DiffCoeff((Start/
        FrameInterval):(End/FrameInterval)),'color','k'); hold on
    ylim([0 max(DiffCoeff)*1.1])
    xlabel('Time (s)')
    ylabel('Diffusion Coefficient (um^2/s)')
    pause
```

```matlab
    else
    figure(4)
    plot(X_Centered(Start:End),Y_Centered(Start:End),'.'); hold on
    xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
    ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])
    xlabel('Horizontal Displacement (um)')
    ylabel('Vertical Displacement (um)')

    figure(5)
    plot(Time((Start/FrameInterval):(End/FrameInterval)),DiffCoeff((Start/
        FrameInterval):(End/FrameInterval)),'LineWidth',2); hold on
    ylim([0 max(DiffCoeff)*1.1])
    xlabel('Time (s)')
    ylabel('Diffusion Coefficient (um^2/s)')
    pause
    end
end
    % Plot the last section
if max(X_Centered(max(Events):size(X_Centered,2)))-min(X_Centered(max(Events):size(
    X_Centered,2))) > 0.8*max(2*X_Centered) || max(Y_Centered(max(Events):size(
    Y_Centered,2)))-min(Y_Centered(max(Events):size(Y_Centered,2))) > 0.8*max(2*
    Y_Centered)
    figure(4)
    plot(X_Centered(max(Events):size(X_Centered,2)),Y_Centered(max(Events):size(
        X_Centered,2)),'.','color','k'); hold on
    xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
    ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])
    xlabel('Horizontal Displacement (um)')
    ylabel('Vertical Displacement (um)')

    figure(5)
    plot(Time((max(Events)/FrameRate):(size(X_Centered,2)/FrameInterval-1)),
        DiffCoeff((max(Events)/FrameInterval):(size(X_Centered,2)/FrameInterval-1))
        ,'color','k'); hold on
    ylim([0 max(DiffCoeff)*1.1])
    xlabel('Time (s)')
    ylabel('Diffusion Coefficient (um^2/s)')
    else
    figure(4)
    plot(X_Centered(max(Events):size(X_Centered,2)),Y_Centered(max(Events):size(
        X_Centered,2)),'.'); hold on
    xlim([-1.1*max(X_Centered) 1.1*max(X_Centered)])
    ylim([-1.1*max(Y_Centered) 1.1*max(Y_Centered)])
    xlabel('Horizontal Displacement (um)')
    ylabel('Vertical Displacement (um)')

    figure(5)
    plot(Time((max(Events)/FrameInterval):(size(X_Centered,2)/FrameInterval-1)),
        DiffCoeff((max(Events)/FrameInterval):(size(X_Centered,2)/FrameInterval-1))
        ,'LineWidth',2); hold on
    ylim([0 max(DiffCoeff)*1.1])
    xlabel('Time (s)')
    ylabel('Diffusion Coefficient (um^2/s)')
end

end
```

```matlab
function hhh=vline(x,in1,in2)
% function h=vline(x, linetype, label)
%
% Draws a vertical line on the current axes at the location specified by 'x'.
    Optional arguments are
% 'linetype' (default is 'r:') and 'label', which applies a text label to the graph
     near the line.  The
% label appears in the same color as the line.
%
% The line is held on the current axes, and after plotting the line, the function
```

```
    returns the axes to
% its prior hold state.
%
% The HandleVisibility property of the line object is set to "off", so not only
    does it not appear on
% legends, but it is not findable by using findobj.  Specifying an output argument
    causes the function to
% return a handle to the line, so it can be manipulated or deleted.  Also, the
    HandleVisibility can be
% overridden by setting the root's ShowHiddenHandles property to on.
%
% h = vline(42,'g','The Answer')
%
% returns a handle to a green vertical line on the current axes at x=42, and
    creates a text object on
% the current axes, close to the line, which reads "The Answer".
%
% vline also supports vector inputs to draw multiple lines at once.  For example,
%
% vline([4 8 12],{'g','r','b'},{'l1','lab2','LABELC'})
%
% draws three lines with the appropriate labels and colors.
%
% By Brandon Kuczenski for Kensington Labs.
% brandon_kuczenski@kensingtonlabs.com
% 8 November 2001

if length(x)>1  % vector input
    for I=1:length(x)
        switch nargin
        case 1
            linetype='r:';
            label='';
        case 2
            if ~iscell(in1)
                in1={in1};
            end
            if I>length(in1)
                linetype=in1{end};
            else
                linetype=in1{I};
            end
            label='';
        case 3
            if ~iscell(in1)
                in1={in1};
            end
            if ~iscell(in2)
                in2={in2};
            end
            if I>length(in1)
                linetype=in1{end};
            else
                linetype=in1{I};
            end
            if I>length(in2)
                label=in2{end};
            else
                label=in2{I};
            end
        end
        h(I)=vline(x(I),linetype,label);
    end
else
    switch nargin
    case 1
        linetype='r:';
        label='';
    case 2
        linetype=in1;
        label='';
    case 3
        linetype=in1;
        label=in2;
    end




    g=ishold(gca);
    hold on

    y=get(gca,'ylim');
    h=plot([x x],y,linetype);
    if length(label)
        xx=get(gca,'xlim');
        xrange=xx(2)-xx(1);
        xunit=(x-xx(1))/xrange;
        if xunit<0.8
            text(x+0.01*xrange,y(1)+0.1*(y(2)-y(1)),label,'color',get(h,'color'))
        else
            text(x-.05*xrange,y(1)+0.1*(y(2)-y(1)),label,'color',get(h,'color'))
        end
    end

    if g==0
    hold off
    end
    set(h,'tag','vline','handlevisibility','off')
end % else

if nargout
    hhh=h;
end
```

A.10 Diffusion coefficient plotted against amplitudes

```
function CDF_MinorMajorAxis(Result, filename,FrameStep)
%% 1. Set Parameters for computation
TotalFrames     =       1800;                       % Set to 1800 frames for
    analyzing each particle for one minute
FrameRate       =       30;
TimeInterval    =       FrameStep/FrameRate;    %Small time interval in which the
    diffusion coefficient will be determined later in script

filename = erase(filename,'.mat');

% Set up fittype and options.
ft = fittype( '1-exp(-x/(4*a))', 'independent', 'x', 'dependent', 'y' );
opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
opts.Display = 'Off';
opts.StartPoint = 0.001;

%Ignore particles which do not show sufficient symmetry
Pixel = .794 ; %Put in Resolution
% Nikon: 794 nm
% M1 microscope: 641 nm
% Leica: 460 nm

%% IMPORTANT SELECT CORRECT FILTER BEFORE ANALYSATION
```

```
%For Silica particles use:
test_n = find([Result.Sym] >= 0.75 & ([Result.MinorDev]*Pixel >= 0.04) & ([Result.
    MinorDev]*Pixel <= 0.10) );

%For myone use:
%test_n = find([Result.Sym] >= 0.75 & ([Result.MinorDev]*Pixel >= 0.050) & ([Result
    .MinorDev]*Pixel <= .180) );

r                =         zeros(1,(TotalFrames-1));        %Define matrix for faster
    computation

%Define Matrices for major and minor axis and position for faster computation.
EmielMajorAllum =        zeros(1,size(test_n,2));
EmielMinorAllum =        zeros(1,size(test_n,2));
X                =        zeros(size(test_n,2),TotalFrames); %Define matrixes for
    faster computation
Y                =        zeros(size(test_n,2),TotalFrames);

%% 2. READ OUT DATE FOR MAJOR/MINOR AXIS AND DISPLACEMENT
for i = test_n
    %Ready out the major and major axis for the ith particle
        EmielMajorAllum(i)  =    Result(i).MajorDev.*Pixel; %Compute pixels to
            micrometers
        EmielMinorAllum(i)  =    Result(i).MinorDev.*Pixel;
 % Ready out position of each particle in um.
        X(i,:)              =    Result(i).DriftCorrectedTrajectory(1,1:TotalFrames)
            *Pixel;
        Y(i,:)              =    Result(i).DriftCorrectedTrajectory(2,1:TotalFrames)
            *Pixel;
end

EmielMajorAllum      =        EmielMajorAllum(:,any(EmielMajorAllum,1));
EmielMinorAllum      =        EmielMinorAllum(:,any(EmielMinorAllum,1));

DiffCoeff          =       zeros(1,size(test_n,2)); %Define Matrix for faster
    computation of diffussion coefficient

%% 3. CALCULATE FOR EACH PARTICLE THE DIFFUSSION COEFFICIENT USING CDF
for i = test_n
    for j = 2*FrameStep:FrameStep:TotalFrames %Loop should take intervals of steps
        equal to the FrameInterval
            r(j) = (X(i,j)-X(i,j-FrameStep))^2+(Y(i,j)-Y(i,j-FrameStep))^2; %
                Distance for
    end
  r_distance = r(:,any(r,1)); %Get rid of elements equal to zero

% The probability for finding the particle at a certain distance or smaller is
    determined (CDF)
[CDF,D] = ecdf(r_distance); %Calculate CDF

%% Fit: 'Extract data for diffussioncoefficient through fit from CDF
[xData, yData] = prepareCurveData(D, CDF );

% Fit model to data.
try %When fit gives an error, skip the fit and  continue with next iteration in for
     loop.
[fitresult, ~] = fit( xData, yData, ft, opts );
catch
    try
        opts.StartPoint = 0.0001;
        [fitresult, ~] = fit( xData, yData, ft, opts );
        opts.StartPoint = 0.001;
    catch
        disp('False Fit!')
    end
end

```

```
DiffCoeff(i) = fitresult.a/TimeInterval; %Since a = D*t, devide through the
    timeinterval
% RESULT: CD = Diffussion coefficient of ith particle
end
DiffCoeff = DiffCoeff(:, any(DiffCoeff,1));
%% 4. Plot Minor/DiffCoef
close all
figure('visible','off')
p = polyfit(nonzeros(EmielMinorAllum'), nonzeros(DiffCoeff'),1);
plot(EmielMinorAllum', DiffCoeff', '.')
hold on
plot(xlim, p(1)*xlim+p(2))
grid on
xlabel('Minor amplitude (um)')
ylabel('Diffusion coefficient (um^2/s)')
title(['Diffusion coefficient vs. Minor amplitude',string(filename)])
legend('data',['y = ',num2str(p(1)),'x',' + ',num2str(p(2))],'Location','southeast'
    )
filename1 = [filename,' FI=',num2str(FrameStep),'.MinorDiffCoef.png'];
saveas(gcf,filename1)



figure('visible','off')
FrameStep = polyfit(nonzeros(EmielMajorAllum'),nonzeros(DiffCoeff'),1);
plot(EmielMajorAllum',DiffCoeff','.')
hold on
plot(xlim, FrameStep(1)*xlim+FrameStep(2))
grid on
xlabel('Major amplitude (um)')
ylabel('Diffusion coefficient (um^2/s)')
title(['Diffusion coefficient vs. Major amplitude',string(filename)])
legend('data',['y = ',num2str(FrameStep(1)),'x',' + ',num2str(p(2))],'Location','
    southeast')
filename2 = [filename,' FI=',num2str(FrameStep),'.MajorDiffCoeff.png'];
saveas(gcf,filename2)

disp(string(['For ',filename,' the minor slope equaled ',num2str(p(1)),' and the
    major slope equaled ',num2str(FrameStep(1)),'.']))
end
```

Appendix B

Correlations diffusion coefficient and motion amplitudes for different systems

On the next page table B.1 gives an overview of the found slopes values as discussed in chapter 6.

Table B.1

Surface Coating	Coupling Strategy	dsDNA tether length	Particle	Particle Coating	Minor Slope	Major Slope
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,78212	0,6744
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,65102	0,58735
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,63341	0,55381
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,63417	0,53744
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,66508	0,54276
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,65281	0,55923
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,91077	0,75792
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,66727	0,56798
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,78212	0,6744
BSA	DIG-AntiDIG (antigen-antibody)	120 base pair?	Silica	Particle Binder ssDNA (? bp)	0,76552	0,71332
Casein	DIG-AntiDIG (antigen-antibody)	221 base pair?	Silica	Particle Binder ssDNA (? bp)	1,3848	1,3218
BSA	DIG-AntiDIG (antigen-antibody)	221 base pair	myone	Particle Binder ssDNA (? bp)	0,4887	0,44846
BSA	DIG-AntiDIG (antigen-antibody)	221 base pair	myone	Particle Binder ssDNA (? bp)	0,58182	0,45713
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	Particle Binder ssDNA (11 bp)	0,56053	0,53712
BSA	DIG-AntiDIG (antigen-antibody)	221 base pair	myone	Particle Binder ssDNA (? bp)	0,58106	0,50581
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,54667	0,4939
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	Particle Binder ssDNA (20 bp)	0,50884	0,45139
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	Particle Binder ssDNA (20 bp)	0,50273	0,44175
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	Particle Binder ssDNA (20 bp)	0,51683	0,45994
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	Particle Binder ssDNA (11 bp)	0,51654	0,49593
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	Particle Binder ssDNA (11 bp)	0,49715	0,47508
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	Particle Binder ssDNA (20 bp)	0,52141	0,46498
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	Particle Binder ssDNA (11 bp)	0,54514	0,50968
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	Particle Binder ssDNA (11 bp)	0,51237	0,44581
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,54667	0,4939
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,5338	0,47413
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,57349	0,50277
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,58255	0,52223
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,55487	0,49177
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,52154	0,47279
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,57579	0,51786
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,54268	0,4853
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,55296	0,50915
PLL-g-PEG	DBCO-azide click chemistry	221 base pair	myone	biotinPEG	0,53081	0,45486