3.1 Optische Pinzette

SICHERHEITSHINWEIS: Der im Versuch verwendete Laser hat die Laserklasse 3B und darf daher nie ohne geeignete Schutzvorkehrungen verwendet werden. Der optische Aufbau ist von einem Schutzgehäuse vollständig umschlossen. Dieses Gehäuse darf nur geöffnet werden, wenn der Laser außer Betrieb ist!
Die Sicherheitsschalter dürfen auf keinen Fall manipuliert werden!

Hinweise:

- Diese Versuchsanleitung basiert in Teilen auf der Staatsexamensarbeit von Antje Tönnies, die im Jahr 2005 eine optische Pinzette für das Anfängerpraktikum der Universität Stuttgart aufgebaut hat [Tön05]. Die Staatsexamensarbeit ist im internen Bereich des AP-Servers abrufbar (Zugriff nur nach Login).
- Wenn Sie während der Versuchsdurchführung schon mit der Auswertung beginnen und/oder Fragen zur Programmierung an Ihren Tutor stellen wollen, bringen Sie möglichst Ihren Laptop zum Praktikum mit und installieren Sie die von Ihnen präferierte Programmierumgebung (`Python`, `MATLAB`) bitte bereits vor dem Praktikumstag.

Ziel

- Beobachtung und quantitative Auswertung der Bewegung von kleinen Teilchen in einer wässrigen Suspension
- gezielte Beeinflussung der Teilchenbewegung durch Laserstrahlung
- Bestimmung der dynamischen Scherviskosität einer Kolloidsuspension

Zubehör

- Kompaktaufbau „Optische Pinzette“ der Fa. Thorlabs mit PC-Steuerung (siehe [tho])
- wässrige Kolloidprobe mit kugelförmigen SiO_2-Partikeln (Radius 2,1 µm) in versiegelter Glasküvette (Querschnitt 0,05 mm × 1,00 mm) auf Objektträger
- Software zur Programmierung der Auswerteroutinen, z. B.
  - `Python`
    kostenlos erhältlich für Windows/macOS/Linux z. B. unter
    - (Anaconda) `https://www.anaconda.com/products/distribution`, (Stand 24.01.2024) oder
    - (Visual Studio Code) `https://code.visualstudio.com/`, (Stand 24.01.2024),
  - `MATLAB`
    - im Rahmen einer Landeslizenz kostenlos erhältlich für Windows/macOS/Linux unter `https://www.kim.uni-konstanz.de/services/software-und-hardware/wissenschaftliche-software/MATLAB/`, (Stand 24.01.2024)

3.1.1 Grundlagen

Eine optische Pinzette dient dazu, kleine Teilchen mit Hilfe von Lichtstrahlen zu „fangen“ und zu bewegen. Die Erstveröffentlichung der Methode erfolgte bereits im Jahr 1970 durch Arthur

Ashkin [Ash70], eine Weiterentwicklung wurde im Jahr 1986 ebenfalls durch Ashkin zusammen mit anderen Autoren publiziert. Die Methode entwickelte sich schnell zu einem Standardwerkzeug, so dass Ashkin dafür im Jahr 2018 den Nobelpreis erhielt (genauer gesagt die Hälfte des Physiknobelpreises des Jahres 2018, siehe z. B. [Sch18]).

3.1.1.1 Kolloid

Unter einem Kolloid bzw. einer Kolloidsuspension (von ⟨altgr.⟩ *κόλλα kólla* „Leim“, ⟨altgr.⟩ *εἶδος eidos* „Form, Aussehen“ und ⟨lat.⟩ *suspendere* „aufhängen, in der Schwebe lassen“) versteht man allgemein feste oder flüssige Teilchen mit einer Größe zwischen wenigen Nanometern und einigen Mikrometern, die in einem flüssigen oder gasförmigen Medium relativ gleichmäßig verteilt sind. Oft werden etwas ungenau auch die einzelnen Partikel selbst als Kolloide bezeichnet.

3.1.1.2 Viskosität

Unter der dynamischen Scherviskosität η versteht man die Zähigkeit einer Flüssigkeit, genauer das Verhältnis aus Schubspannung und Geschwindigkeitsgradient.[1]

3.1.1.3 Optische Pinzette / Fallenpotenzial

Bei der barometrischen Höhenformel ergibt sich aus dem Zusammenwirken von Gravitationspotenzial (Newton) und zufälliger Gleichverteilung eine Exponentialfunktion (Boltzmann) für die Aufenthaltswahrscheinlichkeit eines Teilchens als Funktion der Höhe. Im Fall der optischen Pinzette ist es sehr ähnlich, allerdings wirkt hier nicht das Gravitationspotenzial, sondern ein Potenzial, das abhängig ist von der räumlichen Intensitätsverteilung des Laserstrahls und der Position des Partikels relativ zum Fokus des Laserstrahls. Meistens wird ein rotationssymmetrischer Strahl mit gaußförmigem Profil verwendet und oft (auch im Praktikumsversuch) außerdem die Beweglichkeit des Partikels auf zwei Dimensionen eingeschränkt, so dass die Angabe des Potenzials als Funktion des Radius von der Strahlmitte ausreicht. Dann erhält man eine Exponentialfunktion für die Aufenthaltswahrscheinlichkeit als Funktion des Abstands r. Aus dieser kann das Potenzial der Form $V(r) = \kappa \cdot r^2$ mit der „Federkonstante“ κ der optischen Pinzette für die gefangenen Teilchen berechnet werden.

Im ersten Teil des Praktikumsversuchs wird die Viskosität der Suspension dadurch bestimmt, dass ein Teilchen mit Hilfe der optischen Pinzette festgehalten wird, während die Probe mit einer definierten konstanten Geschwindigkeit relativ dazu bewegt wird. Durch die Bewegung wird das Teilchen im Fallenpotenzial so weit aus der Mitte heraus verschoben, dass sich ein Kräftegleichgewicht zwischen der Kraft im Potenzial und der viskosen Reibungskraft (Stokes) einstellt. Aus der Reibungskraft, der Geschwindigkeit der Bewegung und der Größe des Partikels kann die Viskosität berechnet werden.

3.1.1.4 Brown'sche Bewegung

Der schottische Botaniker Robert Brown machte im Jahr 1827 bei der mikroskopischen Beobachtung einer Pollensuspension die Beobachtung, dass darin enthaltene kleine Teilchen (z. B.

[1] Hiervon zu unterscheiden ist die kinematische Viskosität ν. Die beiden Größen hängen zusammen über die Massendichte ϱ nach $\eta = \nu \cdot \varrho$.

Zell-Organellen, nicht so sehr die relativ großen Pollenkörner selbst, denn deren Bewegung ist viel geringer) eine unregelmäßige Zitterbewegung ausführten. Wie erst viel später von Albert Einstein (1905) und Marian Smoluchowski (1906) erklärt und quantitativ beschrieben werden konnte, beruht diese Bewegung auf den zufälligen Stößen der Flüssigkeitsmoleküle auf die Partikel.[2]

3.1.1.5 Diffusion / mittlere quadratische Verschiebung (*mean squared displacement*, MSD)

Die Brown'sche Bewegung führt zu einer ungerichteten Bewegung der Teilchen, die sich im Laufe der Zeit immer weiter vom Startpunkt weg bewegen (siehe Abbildung 3.1) und dabei gleichmäßig in der Flüssigkeit verteilen (Diffusion). Um die Bewegung der Teilchen zu charakterisieren, könnte man prinzipiell ihre Geschwindigkeit betrachten, z. B. den über die Zeit und viele Teilchen gemittelten Geschwindigkeitsbetrag. Allerdings ist dies kein gutes Maß, da die Bewegung selbstähnlich ist und deshalb auf verschiedenen Längenskalen betrachtet zu unterschiedlichen Geschwindigkeiten führt.

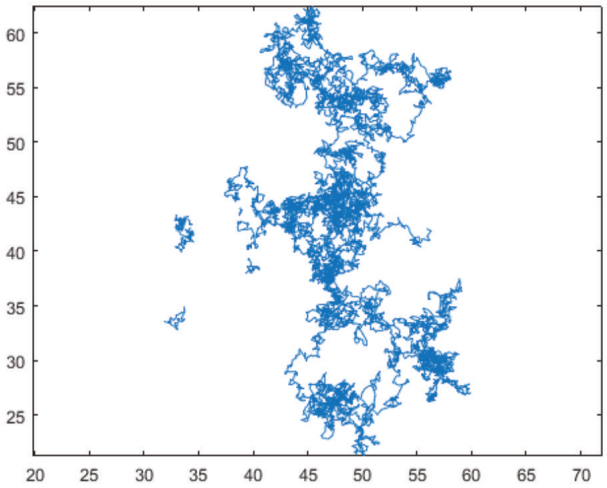

Abbildung 3.1: Beispiel für Trajektorie eines Teilchens in der Kolloidsuspension. Fehlende Punkte entstehen durch Schwierigkeiten bei der automatischen Erkennung der Teilchenposition.

Besser geeignet ist die Betrachtung der zeitlichen Entwicklung der sog. mittleren quadratischen Verschiebung, das ist der Erwartungswert des quadrierten Abstands eines Teilchens von seinem Startpunkt:

$$\mathrm{MSD}(\tau) = \langle (\triangle r(\tau))^2 \rangle = \langle |\vec{r}(t+\tau) - \vec{r}(t)|^2 \rangle \tag{3.1}$$

Im zweiten Teil des Praktikumsversuchs wird die Viskosität der Suspension durch Beobachtung der freien Diffusion eines Teilchens in der Kolloidsuspension bestimmt.

[2] Der häufig verwendete Begriff Brown'sche Molekularbewegung bezieht sich allerdings nicht auf die stoßenden Flüssigkeitsmoleküle, sondern kommt daher, dass man Anfang des 19. Jahrhunderts den Begriff Molekül ganz allgemein für alle kleinen Teilchen verwendete.

3.1.2 Versuchsdurchführung

3.1.2.1 Aktive Mikrorheologie

Die notwendigen Schritte sind hier nur knapp beschrieben. Fragen Sie im Fall von Unklarheiten bei Ihrem Tutor nach.

Hinweise:

- Die Videos müssen im avi-Format[3] gespeichert werden, damit die Auswertung anschließend ohne Probleme funktioniert. Wählen Sie dazu in den Programmeinstellungen die Option „quick save / avi / 8bit“.
- Damit die Auswertung gut klappt, muss das Bild sowohl kontrastreich als auch hell genug sein. Fragen Sie Ihren Tutor nach der richtigen Beleuchtungs-Einstellung.

a) Programme `Kinesis` und `ThorCam` starten
b) Probe einbauen, richtig positionieren und Schrittmotorzähler im Programm zurücksetzen
c) Ein Teilchen fangen und bei verschiedenen Laserintensitäten[4] jeweils ein Video aufnehmen. 5000 bis 10 000 Frames (ca. 10 min) pro Messung ergeben ausreichend Statistik zur Bestimmung des Fallenpotenzials.
d) Eine (eher hohe) Laserintensität auswählen und gefangenes Teilchen durch die Suspension bewegen. Die Bewegungsgeschwindigkeit kann im Thorlabs-GUI als Probentisch-Geschwindigkeit eingegeben werden. Sie sollte niedrig genug sein, dass das Teilchen gehalten werden kann (bis ca. $10\,\frac{\mu m}{s}$ geht das auf jeden Fall gut).
e) Für die Auswertung wird dann die Veränderung der Teilchenposition innerhalb der Falle durch Reibung gemessen. Dazu wird z. B. ein Video aufgenommen, während dem der Tisch zunächst in Ruhe ist und dann bewegt wird (jeweils z. B. ca. 500 Frames). Die resultierenden Histogramme weisen dann zwei Maxima auf (einmal für Tisch in Ruhe, einmal für bewegten Tisch), zwischen denen die Distanz ausgewertet werden kann. Die Position der Maxima kann und muss dabei mit Subpixel-Auflflflösung bestimmt werden.
f) Messung mehrfach wiederholen, dauert pro Messung ca. 5 min

3.1.2.2 Passive Mikrorheologie

g) Ein Teilchen in die Mitte des Gesichtsfeldes und in die Mitte der Probezelle bringen. Dabei darauf achten, dass ein möglichst großer Bereich rund um das Teilchen frei von anderen Teilchen ist.
h) Laser ausschalten
i) Video bei 5 fps mit ca. 10 000 Frames aufnehmen (Dauer ca. 30 min). Mehr fps sind problematisch, weil der Kameraspeicher vollläuft und dadurch zwischendurch Bilder verloren gehen. Es wird allerdings angezeigt, wie viel Prozent der Bilder verloren gegangen sind.

3.1.3 Auswertung

Bestimmung der Partikelpositionen aus den Videodateien mit dem Programm `VideoTracker`. Dabei kann der Schwellwert mit Hilfe eines Schiebereglers eingestellt werden, um ihn an un-

[3] *audio video interleave*, ein weit verbreitetes Video-Containerformat

[4] Die Laserintensität wird über den Pumpstrom durch die Laserdiode gesteuert. Dabei darf ein Maximalstrom von 110 mA nicht überschritten werden, um die Diode nicht zu zerstören.

terschiedliche Beleuchtungsverhältnisse anzupassen (Default-Wert 175, Werte unter 100 sind problematisch). Das Programm erstellt aus dem `avi`-Film eine `csv`-Datei (*comma separated values*).

3.1.3.1 Aktive Mikrorheologie

3.1.3.1.1 Bestimmung der Fallenpotenziale

Führen für jede verwendete Laserintensität folgende Schritte durch:

a) 2D-Wahrscheinlichkeits-Histogramm $p(x, y)$ der Teilchenpositionen erstellen (keine Darstellung nötig)
b) Umrechnen von $p(x, y)$ in $p(r)$ (Wahrscheinlichkeit als Funktion des Abstands vom Zentrum)
c) Bestimmen Sie das Fallenpotenzial $V(r)$ als Funktion des Abstands vom Zentrum des Laserstrahls für alle verwendeten Stromstärken.

$$p(r) = \exp\left(\frac{-V(r)}{kT}\right) \tag{3.2}$$

$$\Rightarrow V(r) = -kT \cdot \ln(p(r)) \tag{3.3}$$

Stellen Sie alle berechneten Fallenpotenziale $V(r)$ in einem gemeinsamen Diagramm dar.

3.1.3.1.2 Bestimmung der Viskosität

d) Bestimmen Sie die „Federkonstante“ κ als Vorfaktor in der Anpassung der quadratischen Näherung $V(r) = \kappa \cdot r^2$ des für die weiteren Versuche verwendeten Potenzials.

Bestimmen sie jeweils

e) den Abstand r_1 zwischen den beiden Maxima für ruhenden und bewegten Probentisch und
f) die Viskosität aus dem Kräftegleichgewicht bei Bewegung mit der Geschwindigkeit v nach

$$-\frac{\mathrm{d}}{\mathrm{d}r}V(r)\bigg|_{r_1} = 6\pi\eta a v \tag{3.4}$$

mit

$$a = \text{Teilchenradius}$$

$$\Rightarrow 2\kappa r_1 = 6\pi\eta a v \tag{3.5}$$

$$\eta = \frac{2\kappa r_1}{6\pi a v} \tag{3.6}$$

Berechnen Sie den Mittelwert der Viskosität.

3.1.3.2 Passive Mikrorheologie

g) Stellen Sie die Trajektorie in einem xy-Diagramm dar.
h) Erstellen Sie ein Diagramm des MSD als Funktion der Zeit, d. h. MSD(t).

i) Passen Sie eine Funktion der Form

$$\mathrm{MSD}(t) = s \cdot t + c_2 \cdot t^2 \tag{3.7}$$

an die Daten an, um mögliche Drift korrigieren zu können. Zeichnen Sie auch diese Anpassungsfunktion in das Diagramm ein.

j) Aus der Steigung s (linearer Anteil) ergibt sich die Diffusionskonstante

$$D = \frac{s}{4} \tag{3.8}$$

k) Berechnen Sie die Viskosität nach der Stokes-Einstein-Formel:

$$\eta = \frac{kT}{6\pi a D} \tag{3.9}$$

mit

$$a = \text{Teilchenradius}$$

3.1.4 Code-Fragmente für die Erstellung von Auswertungsprogrammen

Die folgenden Code-Fragmente sind als Anregung für die selbstständige Programmierung in `Python` oder `MATLAB` zu verstehen.

Sie finden auch fertige Auswerteroutinen auf dem AP-Server auf der Seite mit der Versuchsanleitung zu diesem Versuch:
`https://ap.physik.uni-konstanz.de/index.php?option=setuplayout=setupid=94`

3.1.4.1 Benötigte Bibliotheken:

Python:
Für die meisten mathematischen Methoden, Plotten, und Fitten wird jeweils eine Bibliothek benötigt. Diese können vor Ausführen der Befehle oder am Beginn eines jeden Scripts importiert werden, mit den Befehlen

```
## import libraries
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

Die benötigten Behle können dann mit np.Befehl bzw plt.Befehl ausgeführt werden.

MATLAB:
Um in MATLAB Kurven fitten zu können, wird die „Curve-Fitting-Toolbox“ benötigt, die ebenfalls in der Uni-Lizenz enthalten ist. Zur Installation in der Werkzeugleiste im Reiter Home → Add-Ons → Get Add-Ons nach „Curve-Fitting-Toolbox“ suchen und auf install klicken.

3.1.4.2 CSV-Dateien einlesen

Spyder (Anaconda, Python):
Datei per Drag-and-Drop in das Fenster „Variable Explorer“ ziehen. Es öffnet sich ein Import-Wizard, in dem Spaltentrennzeichen (,) und Zeilentrennzeichen (end of line) ausgewählt werden können. Denken Sie daran, die erste Zeile (Spalten-Namen) nicht mit zu importieren. Im Feld „Variable Name“ kann der Datensatz benannt werden (in den Folgenden Beispielen: pos). Um beim Ausführen von Python-Code automatisch auf die Datensätze im „Variable Explorer“ zugreifen zu können muss dies einmalig in Spyder als Präferenz eingerichtet werden. Klicke dazu auf den Schraubenschlüssel in der Werkzeugleiste → Run → General Settings → Run in console's namespace instead of an empty one.

MATLAB:
Datei per Drag-and-Drop in das „Command Window“ ziehen. Es öffnet sich ein Import-Wizard, in dem Spaltentrennzeichen (,) und Zeilentrennzeichen (end of line) ausgewählt werden können. Außerdem kann im Dropdown-Menü (mittig oben) der Datentyp gewählt werden, den der importierte Datensatz annehmen soll. Wähle hier „Numeric Matrix“. Standardmäßig ist „Table“ ausgewählt, damit kommen die hier verwendeten Befehle allerdings nicht zurecht. Denken Sie daran, die erste Zeile (Spalten-Namen) nicht mit zu importieren. Im fettgedruckten Feld in der Datenansicht kann der Datensatz außerdem benannt werden (in den folgenden Beispielen: pos).

3.1.4.3 Mean-squared displacement

pos : Die Koordinaten von der CSV Datei eingelesen (1. Spalte x-Koordinaten, 2. Spalte y-Koordinaten)

Python:
Zunächst die benötigten Matrizen initialisieren, die am Ende die Ergebnisse beinhalten sollen:

```
numberOfFrames = np.size(pos,0);
numberOfTimesteps = round( numberOfFrames / 2);

msd = np.zeros( numberOfTimesteps );
```

Jetzt durch alle $\triangle t$ iterieren, dabei jeweils zugehörige Abstände berechnen und über alle Abstände zu diesem $\triangle t$ mitteln, um MSD($\triangle t$) zu bestimmen. Dann mit nächstem Wert für $\triangle t$ fortfahren.

```
for dt in np.arange(1, numberOfTimesteps, 1):
    displacementX = (pos[ (1+dt):, 0 ] - pos[ 1:(-dt), 0 ])
       *0.19*10**(-6);
    displacementY = (pos[ (1+dt):, 1 ] - pos[ 1:(-dt), 1 ])
       *0.19*10**(-6);
    squaredDisplacement = displacementX**2 + displacementY**2;

    msd[dt] = np.nanmean( squaredDisplacement );
```

MATLAB:
Zunächst die benötigten Matrizen initialisieren, die am Ende die Ergebnisse beinhalten sollen:

```
numberOfFrames = size(pos,1); % Anzahl der Messpunkte (Anzahl
   Frames)

% Nur ersten Teil der Daten auswerten (Statistik):
numberOfTimesteps = round(numberOfFrames / 4);

msd = zeros( numberOfTimesteps, 1 ); % Matrix mit Ergebnis
   initialisieren
```

Jetzt durch alle $\triangle t$ iterieren, dabei jeweils zugehörige Abstände berechnen und über alle Abstände zu diesem $\triangle t$ mitteln, um MSD($\triangle t$) zu bestimmen. Dann mit nächstem Wert für $\triangle t$ fortfahren.

```
for dt = 1:numberOfTimesteps
    displacementX = pos( 1+dt:end, 1) - pos( 1:end-dt, 1);
    displacementY = pos( 1+dt:end, 2) - pos( 1:end-dt, 2);
    squaredDisplacement = displacementX.^2 + displacementY.^2 ;

    msd(dt) = mean( squaredDisplacement, 'omitnan');
end
```

3.1.4.4 2D - Histogramme

Aus dem Plot können z. B. die Position der maximalen Aufenthaltswahrscheinlichkeit (zum Verschieben des Koordinatensystems (sodass der Fallenmittelpunkt im Ursprung liegt) vor Koordinatentransformation zu Zylinderkoordinaten), oder der Abstand der beiden Peaks bei den Messungen mit bewegten Teilchen abgelesen werden.

pos : Die Koordinaten von der CSV Datei eingelesen (1. Spalte x-Koordinaten, 2. Spalte y-Koordinaten)

Python:

```
# min und max Werte fuer binning
minX = np.amin( pos[:,0] )
maxX = np.amax( pos[:,0] )
minY = np.amin( pos[:,1] )
maxY = np.amax( pos[:,1] )
# Arrays der Bins
xBins =  np.arange(minX, maxX, 0.1)
yBins =  np.arange(minY, maxY, 0.1)

# Zaehlen, wie haeufig bestimmte x- und y-Positionen vorkommen (
   abgetastet in Rastergroesse bestimmt durch bins):
counts, xEdges, yEdges = np.histogram2d( pos[:,0], pos[:,1],
   bins=( xBins, yBins ), density = True )
plt.hist2d( pos[:,0], pos[:,1], bins=( xBins, yBins ), density =
    True )
# Die Kanten zu den Mittelpunkten der Bins transformieren:
xEdges = xEdges[1:]- ( xEdges[1] - xEdges[0] )/2
```

```
yEdges = yEdges[1:]- ( yEdges[1] - yEdges[0] )/2
```

MATLAB:

```
% Zaehlen, wie haeufig bestimmte x- und y-Positionen vorkommen (
   abgetastet in Rastergroesse 'BinWidth' 0.1):
[counts, xEdges, yEdges] = histogram2( pos(:,1), pos(:,2),  '
   BinWidth', 0.1);
% Die Kanten zu den Mittelpunkten der Bins transformieren:
xEdges = xEdges(2:end)-( xEdges(2)-xEdges(1) )./2;
yEdges = yEdges(2:end)-( yEdges(2)-yEdges(1) )./2;
```

3.1.4.5 mittlere Geschwindigkeit

pos : Die Koordinaten von der CSV Datei eingelesen (1. Spalte x-Koordinaten, 2. Spalte y-Koordinaten)

Python:

```
delta_t = 100; # Hier eine beliebige Zeitdauer in Frames
   einfuegen
v = np.empty( np.size(pos,0) );
v[:] = np.nan; # mit NaN initialisieren

for timeCounter in np.arange(delta_t+1, np.size(pos,0), 1):
    # Distanz:
    s = ( ( pos[timeCounter, 0] - pos[timeCounter-delta_t, 0] )
       **2 + ( pos[timeCounter, 1] - pos[timeCounter-delta_t, 1]
        )**2 )**0.5
    # Geschwindigkeit in px/fr
    v[timeCounter] = s/delta_t;
# mittlere Geschwindigkeit:
v_mittl = nanmean(v)
```

MATLAB:

```
delta_t = 100; % Hier eine beliebige Zeitdauer in Frames
   einfuegen
v = nan( size(pos,1), 1 ); % mit NaN initialisieren

for timeCounter = (delta_t+1):1:size(pos,1)
    % Distanz:
    s = sqrt( ( pos(timeCounter, 1) - pos(timeCounter-delta_t,
       1) ).^2 + ( pos(timeCounter, 1) - pos(timeCounter-delta_t
       , 1) ).^2 );
    % Geschwindigkeit in px/fr :
    v(timeCounter) = s/delta_t;

end
% mittlere Geschwindigkeit:
```

```
12 v_mittl = mean( v, 'omitnan' );
```

3.1.4.6 radiale Wahrscheinlichkeitsverteilung

pos : Die Koordinaten von der CSV Datei eingelesen (1. Spalte x-Koordinaten, 2. Spalte y-Koordinaten)
xCent : x-Stelle der maximalen Aufenthaltswahrscheinlichkeit
yCent : y-Stelle der maximalen Aufenthaltswahrscheinlichkeit

Python:
Zunächst den Ursprung der Datenpunkte auf das Maximum legen. Dann zu Zylinderkoordinaten transformieren.

```
1 xPos = pos[:,0] - xCent;
2 yPos = pos[:,1] - yCent;
3 rPos = ( xPos**2 + yPos**2 )**0.5;
```

Jetzt ein Histogramm aus den Datenpunkten erstellen und normieren

```
1 rCounts, rEdges = np.histogram( rPos, bins= np.arange(0, np.amax
     (rPos), 0.1), density = True)
2 # Wieder: Bin-Kanten in Mittelpunkte transformieren:
3 rEdges = rEdges[1:] - ( rEdges[1] - rEdges[0] )/2;
4 # Auf Flaechenelement normieren:
5 rCountsNorm = rCounts / (2*np.pi*rEdges);
6 # symmetrische Verteilung ( prob(-r) := prob(r) )
7 radius = np.concatenate( (-rEdges, rEdges), axis = 0);
8 prob = np.concatenate( (rCountsNorm , rCountsNorm), axis = 0)
```

Damit es keine Probleme beim Ziehen des Logarithmus gibt: alle Einträge mit einer Aufenthaltswahrscheinlichkeit von 0 entweder entfernen, oder durch die minimale Aufenthaltswahrscheinlichkeit ersetzen.

```
1 radius = radius[ prob != 0 ];
2 prob = prob[ prob != 0 ];
```

MATLAB:
Zunächst den Ursprung der Datenpunkte auf das Maximum legen. Dann zu Zylinderkoordinaten transformieren.

```
1 xPos = pos(:,1) - xCent;
2 yPos = pos(:,2) - yCent;
3 rPos = sqrt( xPos.^2 + yPos.^2 );
```

Jetzt ein Histogramm aus den Datenpunkten erstellen und normieren

```
1 [rCounts, rEdges] = histcounts( rPos, 'BinWidth', 0.1, '
     Normalization', 'probability');
2 % Wieder: Bin-Kanten in Mittelpunkte transformieren:
3 rEdges = rEdges(2:end)-( rEdges(2)-rEdges(1) )./2;
4 % Auf Flaechenelement normieren:
5 rCountsNorm = rCounts ./ (2*pi*rEdges);
```

```
% symmetrische Verteilung ( prob(-r) := prob(r) )
radius = [-rEdges, rEdges];
prob = [rCountsNorm rCountsNorm];
```

Damit es keine Probleme beim Ziehen des Logarithmus gibt: Alle Einträge mit einer Aufenthaltswahrscheinlichkeit von 0 entweder entfernen, oder durch die minimale Aufenthaltswahrscheinlichkeit ersetzen.

```
radius = radius( prob ~= 0 );
prob = prob( prob ~= 0 );
```

3.1.4.7 Fitten

V: Array mit den Werten des Potenzials an den im Array radius gespeicherten radialen Koordinaten.

Python:

```
# Funktion definieren, die gefittet werdem soll:
def fitFunction(x, v0, k, a):
    return v0 + k*(x-a)*(x-a)

# Startwerte der Parameter definieren (v0, k, a):
init_values = [0, 1, 0]
# fitten:
fitparams, fitCovariances = curve_fit(fitFunction, radius, V, p0
   =init_values)
# Die Fitparameter sind der Reihe nach im Array fitparams
# Die Unsicherheit der Fitparameter sind die Diagonalelemente
   der Matrix fitCovariances
```

MATLAB:

```
% Funktion definieren, die gefittet werden soll
fitTypeHandle = fittype('v + k*(x-a)^2','independent',{'x'},'
   coefficients',{'v','a','k'});
% fitten :
fitHandle = fit(radius', V', fitTypeHandle, 'StartPoint', [0 0
   1])
% Fitparameter und Unsicherheiten werden in der Variable
   fitHandle gespeichert.
```

Literaturverzeichnis

[Ash70] Ashkin, A.: Acceleration and Trapping of Particles by Radiation Pressure. Phys. Rev. Lett., 24:156–159, https://doi.org/10.1103/PhysRevLett.24.156.

[Sch18] Schirber, Michael: http://doi.org/10.1103/Physics.11.100.

[tho] https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=6966.

[Tön05] Tönnies, Antje: Staatsexamensarbeit. https://ap.physik.uni-konstanz.de/AP-intern/Ressourcen/Literatur/Optische-Pinzette/toennies2005_Aufbau%20und%20Konzeptionierung%20einer%20optischen%20Pinzette%20fuer%20das%20Anfaengerpraktikum.pdf.