Slow dynamics of dense colloidal suspensions in two and three dimensions

A Thesis

submitted to

Indian Institute of Science Education and Research Pune

in partial fulfillment of the requirements for the

BS-MS Dual Degree Programme

by

Atharva Pandit

Indian Institute of Science Education and Research Pune

Dr. Homi Bhabha Road,

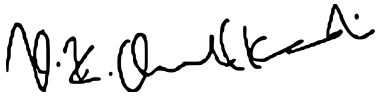
Pashan, Pune 411008, INDIA.

May, 2021

Supervisor: Dr. Vijayakumar Chikkadi

Certificate

This is to certify that this dissertation entitled “Slow dynamics of dense colloidal suspensions in two and three dimensions” towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Atharva Pandit at Indian Institute of Science Education and Research under the supervision of Dr. Vijayakumar Chikkadi, Assistant Professor, Department of Physics , during the academic year 2020-2021.



Dr. Vijayakumar Chikkadi

Committee:

Dr. Vijayakumar Chikkadi

Dr. Apratim Chatterji

This thesis is dedicated to COVID-19

for making everything harder than it should have been.

Declaration

I hereby declare that the matter embodied in the report entitled “Slow dynamics of dense colloidal suspensions in two and three dimensions” are the results of the work carried out by me at the Department of Physics, Indian Institute of Science Education and Research, Pune, under the supervision of Dr. Vijayakumar Chikkadi and the same has not been submitted elsewhere for any other degree.

Atharva Pandit

Acknowledgments

I would like to thank IISER-Pune for providing me with the infrastructure and the opportunity to carry out my research without any constraints. I would also like to thank Dr. Vijayakumar Chikkadi for his guidance and mentorship. Finally, I would like to thank my friends and lab-mates, for their help and support.

Abstract

We develop computer codes to efficiently & accurately locate and track particles in 2D & 3D images obtained from confocal microscopy. We prepare 2D & 3D colloidal systems with area/volume fractions (ϕ) close to the glass transition and compare their properties, behaviour and dynamics in this regime. These dynamics are characterised by structural quantities such as mean square displacement, pair correlation function, bond orientational order & intermediate scattering function. We show that our 3D systems have strong icosahedral order with very weak tendencies towards HCP and FCC orders, and that these structures do not change with ϕ. We quantify characteristic spatial and bond angle relaxation times, in order to compare their behaviours as we approach glass transition in 2D & 3D. Similarly, we describe how to quantify a dynamic correlation length for our systems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Literature Review

As molecular liquids are cooled down, the relaxation time for structural rearrangements increases drastically. If crystallization is avoided by cooling very slowly, then most liquids fall out of equilibrium to enter a metastable phase of supercooled liquids, and subsequently form glasses at sufficiently low temperatures below T_g. We model two and three dimensional glasses using Brownian colloidal suspensions of the same dimension [7][10]. The colloidal glass transition is controlled by the concentration of particles ϕ, as is the temperature or the pressure in molecular systems. Therefore, an analysis of glassy dynamics involves varying ϕ in regimes where glass transition is expected to occur while observing how various static and dynamic structural quantities evolve. Static quantities such as Bond Orientational Order quantify the types of crystalline order in the system [12] [13] [15]. Other dynamic structural quantities have been shown to vary differently in two and three dimensions using computer simulations [4], or by using very small systems ($\sim$ 2000 particles) [17].

1.2 Aim of the thesis

The dynamics of colloidal glasses are studied by observing them under a confocal microscope for long periods of time. The microscope gathers data by photographing the system at a constant frame rate. For 3D systems, the focal plane changes in the z-direction, thereby creating a stack of 2D images which collectively form a 3D image. Computer programs/softwares are used to locate the particles within the images and then track them over time. There are multiple codes and softwares available online which are designed for this [2][5][9]. However, we found that these were unsuitable for our large systems due to multiple reasons, elaborated in chapter 2. Therefore, our first task is to develop a robust and efficient program for locating and tracking particles accurately, which is capable of handling our large datasets.

The tracking software will provide us with coordinates of the particles at all points of time, using which we can calculate the structural quantities. The structural quantities we will be looking at are Pair Correlation Function $(g(r))$, Mean Square Displacement (MSD), Bond Orientational Order (BOO) and Intermediate Scattering Function (ISF). We observe how these quantities vary with ϕ in the regime where glass transition is expected to occur. BOO in 3D provides us a way to check the symmetries and crystalline orders in our systems. Comparing our BOO-3D plots with literature, will give us an idea whether our systems exhibit BCC, FCC, liquid or icosahedral order [12]. ISF and BOO also provide us with characteristic spatial and bond angle relaxation times respectively [8]. Similarly we also plan to find the dynamic correlation length of each system [4]. Our plan is to calculate and then compare the behaviour of all of the above listed quantities in 2D and 3D and see if there are any differences as we approach the glass transition.

Chapter 2

Methods

2.1 Preparing the systems

For 2D hard sphere colloidal glasses, we used silica particles of radii $2.32\mu m$ and $3.34\mu m$. A suspension of $35\mu L$ of large particles and $65\mu L$ of small particles was washed and added to $750\mu L$ of deionized (DI) water and mixed together by sonication. The 35 : 65 ratio was to ensure that the crystallisation of the system was prevented. Sample cells were made by stacking multiple $100\mu m$ laser-cut double-sided tapes onto a cover slip. The depth of the sample well could be changed by changing the number of double-sided tapes. The area fraction was be determined by counting the number of particles in the image after sedimentation. So by varying the density of the suspension as well as the depth of the wells, we were able to create samples of area fractions between $0.55-0.75$.

The 3D colloidal glasses were prepared by suspending sterically stabilized fluorescent polymethylmethacrylate (PMMA) particles in a density and refractive index matching mixture of cycloheptylbromide (CHB) and cis-decalin. All particles have a diameter of $\sigma = 1.3\mu m$ with a polydispersity of 7% to prevent crystallization. The suspensions were centrifuged at high speeds and temperature i.e. at 5000 rpm and 35°C to obtain a dense and homogeneous sediment. This

sediment was diluted to volume fractions ranging from 0.55 to 0.60 in steps of 0.01.

All measurements were taken after the systems had reached a steady state where there was no stimulus for the particles other than their own Brownian motion. These systems were then imaged over time with a confocal microscope using a 488 nm laser.

2.2 Particle Tracking

The microscope records the data as 8-bit precision images i.e. each pixel in the image has a grayscale value between $0-255$ with 0 being completely black and 255 being completely white. A standard image consists of 512 x 512 pixels in the x-y plane covering $92.44\mu m$ in each direction and 1 pixel per $0.15\mu m$ on the z-axis. The Crocker-Grier algorithm is used to locate the centroid pixel for each particle in the images [2]. The Kilfoil & Gao algorithm improves upon this by introducing subpixel accuracy [5]. Finally, the Jensen & Nakamura algorithm performs the tracking iteratively i.e. the particles found in each iteration are deleted from the original image and the entire process is repeated on the new image until a sufficient number of particles are found [9]. This is helpful in 3D as the scattering of the laser beam off colloids causes a brightness gradient along the z-axis making particles in the darker regions harder to detect.

Let us consider a 2D image represented by the matrix $A(x,y)$. A has 512 rows and columns, with each entry between $0-255$. Each digital image has some amount of noise which is unavoidable, such noise tends to be purely random with a correlation length of $\lambda_n = 1$.
Define a background matrix $A_w(x,y)$ as :

$$A_w(x,y) = \frac{1}{(2w+1)^2} \sum_{i,j=-w}^{w} A(x+i,y+j) \tag{2.1}$$

i.e. The background matrix is modelled by a boxcar average over a region of extent $2w+1$ where w is an integer larger than a sphere's radius in pixels, but smaller than the average separation between two spheres.

In order to supress the noise, we convolve the image with a Gaussian surface of revolution of half width equal to λ_n. Convolution of a Gaussian mask onto the image can be represented as :

$$A_{\lambda_n}(x,y) = \frac{1}{B}\sum_{i,j=-w}^{w} A(x+i,y+j)\exp\left(-\frac{i^2+j^2}{4\lambda_n^2}\right) \tag{2.2}$$

where $B = \left[\sum_{i=-w}^{w}\exp\left(-\left(i^2/4\lambda_n^2\right)\right)\right]^2$ is the normalisation constant.
The ideal image is the Gaussian blurred image from which the background is subtracted. This is known as *bandpass filtering*. This can be done in one step as :

$$K(i,j) = \frac{1}{K_0}\left[\frac{1}{B}\exp\left(-\frac{i^2+j^2}{4\lambda_{\mathrm{n}}^2}\right) - \frac{1}{(2w+1)^2}\right] \tag{2.3}$$

$$K_0 = \frac{1}{B}\left[\sum_{i=-w}^{w} exp(-\frac{i^2}{2\lambda_n^2})\right]^2 - \frac{B}{(2w+1)^2} \tag{2.4}$$

Once we have the bandpass filtered image, we find the local brightness maxima and mark them as candidates for particle centres. A pixel is considered as a local maxima if no other pixel within distance w of it is brighter. To speed up the process, we could also search only amongst the top 30 percentile of brightest pixels of the image. However, we noticed this trick performs poorly in 3D images where the bottom sections are significantly darker. We then perform a Gray-Scale Dilation Operation to further narrow down the list of potential candidates. A Gray-Scale Dilation is an elementary morphological operation where in we set the value of pixel $A(x,y)$ to the highest value of any pixel within a distance w of point (x,y). If a pixel has the same value before and after the gray-scale dilation, then it is marked as a candidate point.

This process gives us the brightest pixel within a locality, which is presumably the closest pixel to the sphere's geometrical centre. The offset from the pixel grid to the actual centre is found by performing a brightness-weighted average in a region w around the pixels. This process is

analogous to finding the centre of mass of a body, here we call it the *centre of brightness*.

$$\begin{pmatrix} \varepsilon_x \\ \varepsilon_y \end{pmatrix} = \frac{1}{m_0} \sum_{i^2+j^2 \leqslant w^2} \begin{pmatrix} i \\ j \end{pmatrix} A(x+i, y+j) \tag{2.5}$$

$$m_0 = \sum_{i^2+j^2 \leqslant w^2} A(x+i, y+j) \tag{2.6}$$

m_0 is the integrated brightness of the sphere's image (equivalent to total mass of a body in the analogy). If the local maxima was at (x', y') then we calculate the offset $(\varepsilon_x, \varepsilon_y)$ to give the final refined estimate of the centre (x_0, y_0) :

$$(x_0, y_0) = \left(x' + \varepsilon_x, y' + \varepsilon_y\right) \tag{2.7}$$

If either of $|\varepsilon_x|$ or $|\varepsilon_y| > 0.5$, then we shift the centroid location to the nearest candidate pixel in that direction and recalculate the offset until both values are less than 0.5.

This technique reduces the standard deviation of position measurement to better than about 0.1 pixels. The above discussion was for 2D images, it is quite easy to extrapolate this to 3D by adding a z-component to all of the above equations. However, Kilfoil & Gao pointed out that this method usually results in z-coordinates that are either integers or half-integers [5]. This is due to the fact that almost always, the confocal microscopy systems being used will have a much lower resolution in the z-direction than the x or y directions. This results in particles looking like ellipsoids that are either stretched or squashed along the z-axis. This is known as sub-pixel biasing or pixel locking, as it locks the coordinates to the underlying discrete pixel grid, statistically the positions should appear with equal probability at any fractional pixel value and be completely uncorrelated to the grid. A clear indication of sub-pixel bias in the tracked locations is the broadening of the nearest-neighbour peak in the radial distribution function $g(r)$, due to the density of particles being concentrated around integer or half-integer value planes instead of having an even spread.

To introduce subpixel accuracy of ~ 0.1 pixels in all 3 dimensions, we introduce a refinement step known as *fracshift*. Consider the 3D pixel brightness matrix $A(x, y, z)$, then we perform per-

form the operations mentioned above on it to finally arrive at the step :

$$\begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{pmatrix} = \frac{1}{M} \sum_{i^2+j^2+k^2 \leq w^2} \begin{pmatrix} i \\ j \\ k \end{pmatrix} A(x+i, y+j, z+k) \tag{2.8}$$

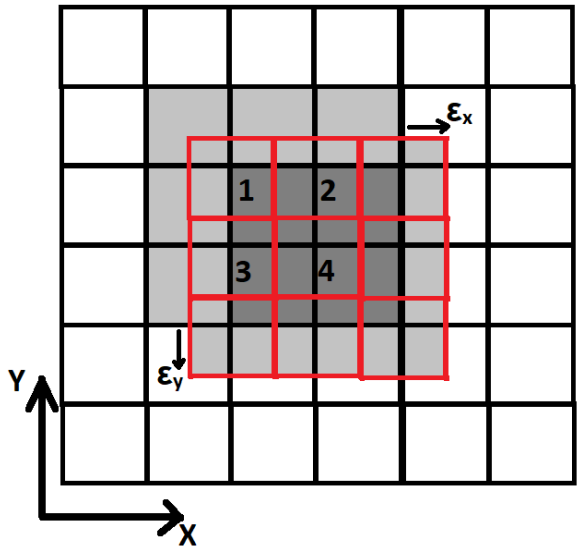


Figure 2.1: Illustration of fracshift. The new 3x3 mask (red) is shifted by $(\varepsilon_x, \varepsilon_y)$, the pixels which are completed covered by the new mask are marked in dark grey and numbered 1-4.

We update the feature centre to $(x_0, y_0, z_0) = (x' + \varepsilon_x, y' + \varepsilon_y, z' + \varepsilon_z)$ as before. Then we create a temporary mask of $3x3$ pixels and overlay it on top of the original grid with the central pixel at (x_0, y_0, z_0) (see Fig.(2.1)). We select the pixels that are completely covered by the new mask and then we update the brightness matrix by a linear interpolation between these pixels weighted by their areal contribution (overlapping area). In Fig.(2.1) we see pixels $1, 2, 3\&4$ are completely overlapped by the mask with offset $(\varepsilon_x, \varepsilon_y)$, if I_i represents the brightness value of the i^{th} pixel, we calculate the area weighted fracshift as :

$$I = I_1 * (1 - |\varepsilon_x|)\left(1 - |\varepsilon_y|\right) + I_2 * |\varepsilon_x|\left(1 - |\varepsilon_y|\right) + I_3 * (1 - |\varepsilon_x|)\,|\varepsilon_y| + I_4 * |\varepsilon_x|\,|\varepsilon_y| \tag{2.9}$$

In 3D, the new mask will involve 8 pixels, 4 from the plane above and 4 from the one below.

We calculate I as before (Eqn 2.9) for both planes, and then repeat the same with ε_z for the pixel sandwiched in between. If I_q and I_{q+1} are the updated values of the pixels in the different z-planes :

$$I = \begin{cases} I_{\mathrm{q}} * |\varepsilon_z| + I_{\mathrm{q+1}} * (1 - |\varepsilon_z|), q = z_0 \text{ if } \varepsilon_z \geq 0 \\ I_{\mathrm{q}} * (1 - |\varepsilon_z|) + I_{\mathrm{q+1}} * |\varepsilon_z|, q = z_0 - 1 \text{ if } \varepsilon_z < 0 \end{cases} \tag{2.10}$$

Kilfoil & Gao claim twenty iterations of fracshifts improves the $x-y$ accuracy four fold and the z accuracy five fold. This level of subpixel accuracy is more than sufficient for our purposes.

For 2D images, we can stop here and begin our analysis, however in 3D we observe that the number of particles being found drops with increasing depth, even though the sample is prepared to be uniformly dense. We suspect this happens due to the scattering of light as we go deeper, causing the pixels at higher z values to be dimmer compared to the top regions. This causes the algorithm to reject the particles at the bottom and we end up with a false density gradient within the system. To counter this, we employ the iterative tracking algorithm developed by Jensen & Nakamura [9].

This algorithm uses the same techniques as Kilfoil & Gao iteratively, however in between each iteration we replace the particles we have found with false particles of brightness value zero, essentially deleting them from the raw image. Once again on the modified raw image, we run the Kilfoil algorithm and this time we find a new set of points which were earlier obscured by the brighter particles. Now we delete these newly found particles and keep iterating until either we have found a sufficient number of particles or there are no new particles to be found within the image. This technique vastly improves the performance the tracking of 3D images with brightness gradients, providing the expected number of particles in at most 2 to 4 iterations. Some of these codes were taken from openly available software sources, some were written by us. The comparison of the performance of each algorithm and it's implementations is presented in chapter 3.

2.3 Finding Trajectories

In order to study the dynamics of the systems, we need to observe the trajectories of the particles, i.e. know the spatial coordinates of each particle at every timestep. Even though most quantities that we calculate are statistical averages, we need to keep track of individual particles in subsequent timeframes so that we can observe it's behaviour and interactions with it's neighbours. This is done by assigning a unique identity number to each particle, so that even if the particles move between frames, we don't mix them up. Crocker, Kilfoil & Jensen provide their own algorithms and codes for temporal tracking, they are designed for systems that have at most $\sim 10,000$ particles per timestep, therefore we can implement them directly for our 2D systems which have $\sim 2,000$ particles per timestep. However our 3D systems have $\sim 2,00,000$ particles per timestep which creates a problem. Every single tracking software/code we were able to find online ran into an error due to excessive combinatorics i.e. the dataset was too large for the program to handle. Hence, we had to come up with our own method of temporal tracking which had to be much more computationally efficient.

The obvious way to calculate displacements between frames is to measure the distance between points in frame no.1 with each and every point in frame no.2 and assign the same unique identity numbers to the particles that have the smallest separations. Say we have N particles in each frame, then the number of pairings possible is $N!$, if we have T number of frames, then the total number of calculations needed is $N! * (T-1)$. Also worth noting is that the formula for distance ($r = \sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}$) itself is computationally heavy. There are many ways to optimise this process, and most of them have been implemented in the popular algorithms mentioned above, however they still lead us to a program that has $\sim \mathcal{O}(N^2)$ complexity. Considering our 3D datasets had a total of $\sim 10^7$ particles for every ϕ, even with all the optimisations a $\sim \mathcal{O}(N^2)$ program was not going to work.

One of the most computationally efficient ways to generate a list of nearest neighbours of all points is to partition the dataset into a *k-dimensional tree* or *k-d tree* for short. A k-d tree

is a special case of binary space partitioning trees in which every leaf node is a k-dimensional point. The construction of the tree is fairly straightforward, each non-leaf node generates a splitting hyperplane that divides the space into two parts. The top most point is generally selected to be the median or close to the median of the dataset (although this is not necessary), then at each level of the tree we insert two points while cycling through the dimensions. For example (see Fig.(2.2)), for a 3 dimensional dataset, the root would have an x-aligned plane i.e. points with x-coordinate less than the root's x-coordinates will be put in the left subtree, points with larger x-coordinates will be put in the right subtree. The next level (root's children) will have a y-aligned plane, their children would be z-aligned and then again x-aligned and so on until all points have been added to the tree.

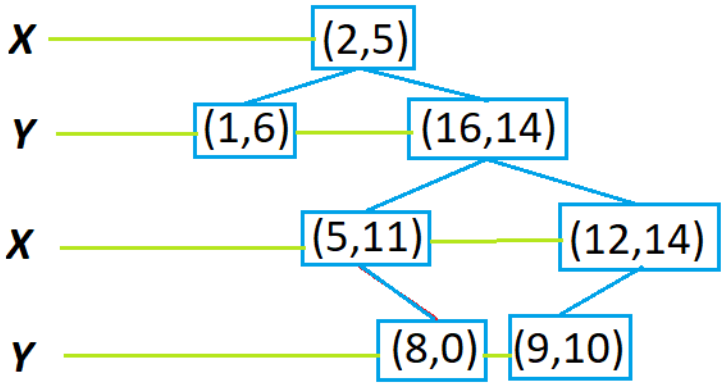


Figure 2.2: k-d tree for a small 2 dimensional data set. At each level the splitting dimension is written on the left (connected by green line). The parent nodes are connected to their children nodes by blue lines. As the root node is not the median of the data set, this is an example of an unbalanced tree.

Once the tree is constructed, finding the nearest neighbours can be done quickly, say we have a query point (which may or may not belong to the tree), we proceed with it from the root as if we are planning to insert it i.e. compare it's x-coordinates at the x-aligned planes, move to the right subtree if the x-value is greater and so on. Finally once we reach a leaf node, we calculate the distance between it and the query point and save it as our current best guess. Then we move back up the tree and check the distance again, if it is smaller than the previous one then we update the current node as new best guess. Check if any point on the other half of the subtree rooted at

the current node could be a better guess, if not eliminate that half of the subtree. In mathematical terms, this is equivalent to checking if the hypersphere centred at our query point is intersecting the splitting hyperplane. If it does intersect, then there could be nearer points on the other side of the plane, hence we move down the other branch and check distances. If it does not intersect, then we move up and repeat the same process. Once we have reached the root node, whichever node was the last to be marked as best guess is our nearest neighbour. This can be modified to include k-best guesses to find the k-nearest neighbours.

The worst case time complexity for building a k-d tree is $\sim \mathcal{O}(nlog(n))$. For making a query and finding nearest neighbours, the average time complexity is $\sim \mathcal{O}(log(n))$ and worst case complexity is $\sim \mathcal{O}(n)$. In any case, using the k-d tree approach is exponentially faster than any conventional approach of finding nearest neighbours. We used the *cKDTree* library from *SciPy* to construct a k-d tree our dataset, as it is a *C* library it has an incredibly fast execution time. For our system of $\sim 2 * 10^5$ particles with 3D coordinates, constructing and querying the k-d tree never took more than ~ 3 seconds.

Let's say we focus on particle i at timestep t, our task is to find it in the frame at timestep $t+1$ and assign it the same unique identity number. To do this we construct a k-d tree of all the points present in frame $t+1$. We query it with the coordinates of particle i at t and find it's $8-12$ nearest neighbours (depending on the system). As our systems are densely packed, a particle is not expected to move more than it's own radius on average, therefore our query can be restricted to a ball of $\sigma/2$. We assign the closest neighbour the same unique id as particle i and move on to the next particle. Once assigned a unique identity number, the particle is removed from the list so that it does not get assigned to any other particle, this ensures a one-one pairing between frames. Some particles may drift in and out near the boundaries, this means every frame has an unequal number of particles. Any particles which have not been paired up with a particle from the previous frame are given completely new identity numbers. Then we impose a cutoff parameter known as 'goodenough' i.e. if the number of frames a particle is present in is less than goodenough, then we delete it from the list. In all of our datasets, we selected goodenough to be equal to the number of

frames. That means any particle not present in every single frame is removed from the dataset.

Finally, after processing the images and finding the coordinates of the particle centroids and then tracking them over time, our result is an array with columns x, y, z, t, id. The final step is to remove the system drift from the data. This is done by subtracting the average displacement vector of the centre of brightness of the entire image from each particle's position vector, the average is a rolling average of usually $10-15$ frames. This resulting array is then used to perform analysis and to calculate the quantities mentioned in the next section.

2.4 Structural Quantities

Once the particles have been successfully located and tracked, the centroid coordinates are used to calculate four major structural quantities that define the phase, behaviour and dynamics of the system.

1. Mean Square Displacement (MSD) :

Describes the average deviation of particles with respect to their mean positions over time. Densely packed systems create cages and confine the particles, reducing their mobility. The MSD is defined as an ensemble average for N number of particles with $\mathbf{x^{(i)}}(t)$ being the position of particle i at time t.

$$\mathrm{MSD}(\Delta t) \equiv \left\langle \left| x(t+\Delta t) - x(t) \right|^2 \right\rangle = \frac{1}{N} \sum_{i=1}^{N} \left| x^i(t+\Delta t) - x^i(t) \right|^2 \tag{2.11}$$

Δt is known as the lag time, t iterates over all possible starting points within the data.

Cage-relative displacement was introduced as a means to study the melting of 2D crystals and to get rid of Mermin-Wagner Fluctuations. However, it also finds use in the analysis of 2D glasses. Cage refers to a restriction in movement imposed upon a particle by it's neighbours, who are in

turn restricted by their neighbours. MSD in glasses displays a two-step relaxation behaviour due to intermittent jump motion. We see an increase in MSD proportional to t, then a plateau region, followed by another increase proportional to t. To minimise the effects of the jump motions and long-wavelength fluctuations we define cage-relative displacement as [14]

$$\Delta r^i_{\mathrm{CR}}(\Delta t) = \Delta r^i(t+\Delta t) - \frac{1}{N^i_{\mathrm{n.n}}} \sum_{j \in \mathrm{n.n.}} \Delta r^j(t) \tag{2.12}$$

n.n represents the nearest neighbours of the particle, Δt represents lag time. Using Eqn.(2.12) we define cage-relative MSD as

$$MSD_{\mathrm{CR}}(\Delta t) = \frac{1}{N} \sum_{i=1}^{N} \left| \Delta r^i_{\mathrm{CR}}(\Delta t) \right|^2 \tag{2.13}$$

2. Pair Correlation Function (g(r)) :

$g(r)$ is a measure related to the probability of finding the centre of a particle at a distance r from the centre of another particle. Let V be the volume of the system, N be the number of particles in it with diameter σ, and ρ be it's density. The probability $g(r)$ is normalised by the density of the system ρ, therefore at larger distances, we expect to see $g(r)$ to be close to 1 i.e. the probability of finding a particle at $r >> \sigma$ is equal to the probability of finding particles at r if the system was homogeneous and uniform with density ρ.

$$\rho = \frac{4\pi(\sigma/2)^3}{3} \frac{N}{V} \tag{2.14}$$

$$g(r) = \frac{V}{4\pi r^2 N^2} \left\langle \sum_i \sum_{j \neq i} \delta\left(r - r_{ij}\right) \right\rangle \tag{2.15}$$

For hard sphere systems, the particles cannot overlap or squish into each other. Assuming the particles are jammed together, the first maxima of $g(r)$ is taken as the diameter of the particles. The first minima is taken as the average separation between nearest neighbours. We also use $g(r)$ as a tool to verify whether the tracking programs have functioned correctly, if we see a small peak

before σ or if the graph is too jagged or if the peak is too broad, it is an indication that we have tracked noise or some other false particles.

Eqn.(2.15) is the standard definition of $g(r)$, however there is a caveat; it assumes all particles to be equivalent which is true only in monodisperse systems. For polydisperse systems, we generalise eqn.(2.15) to include all possible particle-particle interactions and account for them. As we have a bidisperse system in 2D, we define a two-component $g(r)$ as [6]

$$g_{\alpha,\beta}(r) = \frac{V}{4\pi r^2 N_\alpha N_\beta} \left\langle \sum_{\substack{i\in\alpha \\ i=1}}^{N_\alpha} \sum_{\substack{j\in\beta \\ j=1}}^{N_\beta} \delta\left(r - r_i + r_j\right) \right\rangle \tag{2.16}$$

Here α,β represent the two types of particles. Using this eqn., we expect to see 3 peaks in our graph; one each for small-small, small-big and big-big particle interactions.

3. Bond Orientational Order (BOO) :

Also known as the hexatic order parameter ψ_6 in 2D, is a measure of local as well as extended orientational symmetry within a system [15]. First introduced by Steinhardt et al. [15], it has now become the standard tool for identification of crystalline phases, mainly FCC, BCC, HCP and iscohedral symmetries in glasses, crystals, liquids, etc. It is also widely used to study glass transitions and melting transitions in crystals. We define rotationally invariant quantities q_ℓ & q_ℓ which describe local symmetries and orders and can be coarse-grained into a global variables Q_ℓ & W_ℓ for detecting single crystalline order across the system.

'Bonds' refers to imaginary lines between nearest neighbours. For every bond whose midpoint is at $\vec{r}$ we assign a set of numbers q_{lm} which are equal to the spherical harmonics Y_{lm}.

$$q_{lm} \equiv Y_{lm}(\theta(\vec{r}), \phi(\vec{r})) \tag{2.17}$$

$\theta(\vec{r})$ and $\phi(\vec{r})$ are polar angles at $\vec{r}$ measured with respect to some coordinate axis. The spher-

ical harmonics Y_{lm} form a $(2l+1)$ dimensional representation of the rotational group SO(3)[15]. As the Y_{lm} can be changed by simply rotating the axes, we define two quantities q_ℓ and w_ℓ that are rotationally invariant.

$$q_l \equiv \left(\frac{4\pi}{2\ell+1} \sum_{m=-\ell}^{\ell} |q_{\ell m}|^2 \right)^{1/2} \tag{2.18}$$

$$w_l \equiv \sum_{\substack{m_1,m_2,m_3 \\ m_1+m_2+m_3=0}} \begin{bmatrix} \ell & \ell & \ell \\ m_1 & m_2 & m_3 \end{bmatrix} \times q_{\ell m_1}, q_{\ell m_2}, q_{\ell m_3} \tag{2.19}$$

The coefficient $\begin{bmatrix} \ell & \ell & \ell \\ m_1 & m_2 & m_3 \end{bmatrix}$ in Eq.(2.5) are known as Wigner-3j symbols [18]. We calculate these two invariants and compare the results to literature values to identify the cluster symmetries and orders within our systems.

It is helpful to even out the local variations in $q_{\ell m}$ and take into account the second shell of neighbours by coarse-graining the quantities. We average the $q_{\ell m}$ for each particle over it's neighbours, as their $q_{\ell m}$ are determined by the bonds between them and their own neighbours and so on, we get a more accurate understanding of the broader structures at the cost of spatial resolution [11]. Define $Q_{\ell m}$ as the average of $q_{\ell m}$ over all it's neighbours. If particle i has N_b neighbours then :

$$Q_{\ell m} = \frac{1}{N_b} \sum_{\text{bonds}} q_{\ell m}(\vec{r}) \tag{2.20}$$

We calculate Eqn.(2.17) and Eqn.(2.18) again using $Q_{\ell m}$ to define coarse-grained Q_ℓ and W_ℓ.

BOO in 3D is extremely sensitive to the choice of neighbours [13]. There are multiple ways of calculating nearest neighbours, each of which gives a slightly different set of neighbours. Therefore, it is important to understand the different ways of finding the nearest neighbours of a point within a given set of points. The most common methods are :

1. **Range search** : We define an area/volume of radius 1.2σ or 1.4σ (1.2 or 1.4 is chosen by convention, any distance less than the twice the diameter of the particle is reasonable). σ is

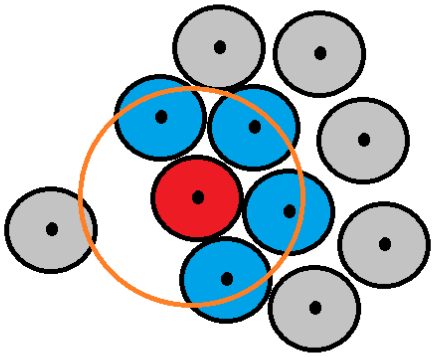
defined as the point where the first peak in pair correlation function occurs. The first minima of the pair correlation function is the average inter-particle distance, we could choose this as our radius as well. We count all particles within the shell as neighbours. (See Fig.(2.3(a))

2. **k-nearest** : If we have a reasonable estimate of the number of neighbours each particle should have (say k), then we can select the k closest points as neighbours for each point. This forces a fixed number of neighbours on each point, works well if the system is densely packed and homogeneous. (See Fig.(2.2(b))

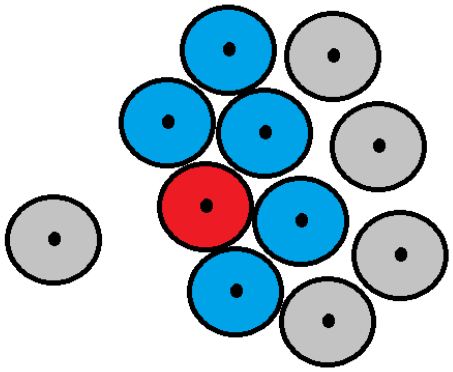
3. **Voronoi Tesselation** : Partition space into regions where each Voronoi Cell consists of all points closer to the seed point (particle centre) than any other seed (other particles). Regions that share the same cell boundary (line in 2D, plane in 3D) are marked as neighbours. (See Fig.(2.2(c))

4. **Delaunay Triangulation** : Is a triangulation of points such that no points lie within the circumcircle of any triangle. Each edge of a triangle represents a bond between neighbours. Voronoi Triangulation and Delaunay Triangulation are duals of each other i.e. they are equivalent methods of finding neighbours. (See Fig.(2.2(d))

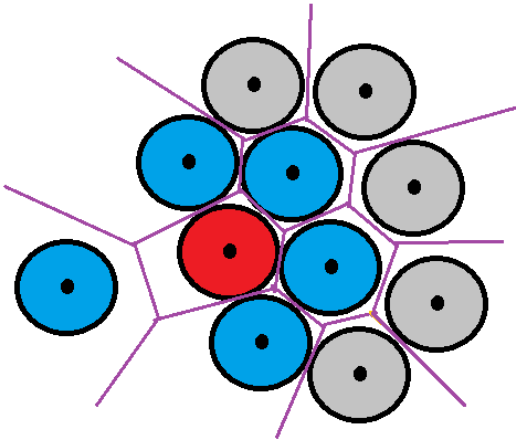
Each method has it's advantages and drawbacks and in general any one of them cannot be considered more accurate than the others. BOO in 3D is extremely sensitive to small changes in the choice of neighbours. Therefore it is necessary to construct a robust method of calculating BOO which takes care of these discrepancies. A new morphometric approach, where we construct Minkowski structure metrics (MSM) Q_l' which are similar to BOO and mathematically equivalent to the Minkowski tensors handles these problems [13].



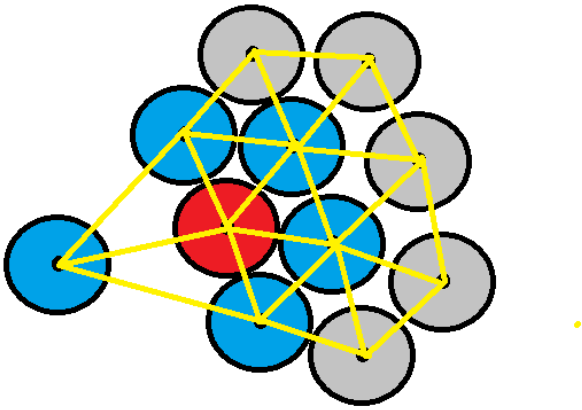
(a) Range Search



(b) k-nearest $k = 5$



(c) Voronoi Tesselation



(d) Delaunay Triangulation

Figure 2.3: Discrepancies arising due to the different ways of finding nearest neighbours.Particle of interest is marked in red, neighbours are marked in blue. (a) Particles inside orange circle are marked as neighbours, radius =1.2σ. (b) $k = 5$ nearest neighbours according to distance. (c) Purple lines are Voronoi cell walls (lines in 2D, planes in 3D). (d) Yellow lines form edges of the triangulation.

We modify Eq.(2.4) by weighing the contribution of each neighbour to the structure metric by associated relative area factor $A(f)/A$. Here $A(f)$ is the surface area of the Voronoi cell facet separating the neighbouring regions associated with each bond. The total surface area of each Voronoi cell is represented by A i.e. the sum of $A(f)$ over all faces is equal to A. Thus, we define MSM Q'_l at particle i as:

$$q'_\ell(i) = \sqrt{\frac{4\pi}{2\ell+1}\sum_{m=-l}^{l}\left|\sum_{f\in F(a)}\frac{A(f)}{A}q_{\ell m}\left(\theta_f,\varphi_f\right)\right|^2} \tag{2.21}$$

where θ_f and ϕ_f are the polar angles on the outer normal vector n_f of facet f. This simple change leads to robust, continuous, and parameter-free structure metrics Q'_l that avoid the shortcomings of the conventional Q_ℓ discussed above [13]. We can use the coarse-grained $Q_{\ell m}$ (eqn.(2.20)) in Eqn.(2.21) to coarse-grain q'_ℓ into Q'_ℓ.

In 2D, BOO is called hexatic order parameter ψ_6 and is used as a means to characterise the short range positional and quasi-long-range orientational order of a system in hexatic phase. The hexatic phase is any phase that has a sixfold orientational order, it exists between the solid and isotropic liquid phases of 2D systems only [16]. For a particle i at $\vec{r}$, which has N_i number of neighbours and θ_{ij} is the angle between particle i and j with respect to a coordinate axis, then ψ_6 is given by

$$\Psi\left(\vec{r}_i\right) = \frac{1}{N_i}\sum_{j=1}^{N_i} exp(i6\theta_{ij}) \tag{2.22}$$

It is also possible to measure the characteristic bond angle relaxation time for our system by time correlating $q_{\ell m}$ and ψ_6. Define $q^i_{lm}(t)$ and $\psi^n_6(t)$ according to Eqn.(2.17,22) respectively. Then we define their time correlations as

$$C_\Psi(\Delta t) = \frac{\left\langle\sum_n \Psi^n_6(t+\Delta t)\left[\Psi^n_6(t)\right]^*\right\rangle}{\left\langle\sum_n\left|\Psi^n_6(t)\right|^2\right\rangle} \tag{2.23}$$

$$C_Q(\Delta t) = \frac{(4\pi)}{(2\ell+1)} \left\langle \sum_i \sum_{m=-l}^{l} q^i_{\ell m}(t+\Delta t) \left[q^i_{\ell m}(t)\right]^* \right\rangle \tag{2.24}$$

Here $*$ represents the complex conjugates. Δt is known as the lag time, t iterates over all possible starting points within the data. We define the characteristic bond angle relaxation time τ_θ, as the time taken for C_Q and C_ψ to decorrelate by a factor of **e**.

$$C_Q(\tau_\theta) = C_\psi(\tau_\theta) = \frac{1}{e} \tag{2.25}$$

4. Intermediate Scattering Function (ISF) :

The radial distribution function $g(\vec{r})$ describes the spatial correlations as a function of distance from a reference particle. On the other hand, the ISF studies correlations between particles in reciprocal space i.e. in Fourier components. It is denoted by $F(\vec{k},t)$ where $\vec{k}$ is a vector in reciprocal space and t is time. It is defined as the Fourier transform of the *van Hove function* (VHF) [1]. The VHF for a spatially uniform system containing N point particles is defined as :

$$G(\vec{r},t) = \left\langle \frac{1}{N} \int \sum_{i=1}^{N} \sum_{j=1}^{N} \delta\left[\vec{r}' + \vec{r} - \vec{r}_j(t)\right] \delta\left[\vec{r}' - \vec{r}_i(0)\right] d\vec{r}' \right\rangle \tag{2.26}$$

which can be simplified to

$$G(\vec{r},t) = \left\langle \frac{1}{N} \int \rho\left(\vec{r}' + \vec{r}, t\right) \rho\left(\vec{r}', 0\right) d\vec{r}' \right\rangle \tag{2.27}$$

Fourier transform of VHF :

$$F(\vec{k},t) = \int G(\vec{r},t) e^{-i\vec{k}\cdot\vec{r}t} d\vec{k} \tag{2.28}$$

Out of which, we can define a self $F_s(\vec{k},t)$ and dependent part $F_d(\vec{k},t)$.

$$F_s(\vec{k},t) = \int G_s(\vec{r},t) e^{-i\vec{k}\cdot\vec{r}t} d\vec{k} \tag{2.29}$$

$$F_d(\vec{k},t) = \int G_d(\vec{r},t) e^{-i\vec{k}\cdot\vec{r}t} d\vec{k} \tag{2.30}$$

Instead of Fourier transforms, we can also compute ISF directly from the particle trajectories.

$$F_S(\vec{k},t) = \frac{1}{N}\langle \sum_{j=1}^{N} \exp\left[i\vec{k}\cdot\left(\vec{r}_j(t) - \vec{r}_j(0)\right]\right\rangle \tag{2.31}$$

$$F_d(\vec{k},t) = \frac{1}{N}\sum_{k=1}^{N}\sum_{j=1}^{N}\left\langle \exp\left[i\vec{k}\cdot\left(\vec{r}_j(t) - \vec{r}_k(0)\right]\right\rangle \tag{2.32}$$

For our analysis, Eq.(2.12) shall be the only relevant equation. $F_s(\vec{k},t)$ characterises the mean relaxation time of the system (area under the curve of $F_s(k,t)$ can be used to define a relaxation time). Spatial fluctuations of $F_s(k,t)$ provide information on dynamic heterogeneities.

As we defined cage-relative MSD for 2D glasses (eqn.(2.13)), we can also define cage-relative ISF to overcome the same problems of jump motions and long-wavelength fluctuations. Using the same $\Delta\vec{r}^{i}_{CR}(t)$ from eqn.(2.12), we define cage-relative ISF $F_{S-\mathrm{CR}}(k,t)$ as [8]

$$F_{S-\mathrm{CR}}(\vec{q},t) = \left\langle \frac{1}{N}\sum_{j=1}^{N} exp(-i\vec{q}\cdot\left(\Delta\vec{r}^{j}(t) - \Delta\vec{r}^{j}_{\mathrm{CR}}(t)\right)) \right\rangle \tag{2.33}$$

Similar to eqn.(2.24), we define a characteristic spatial relaxation time τ_α as the time taken for ISF to decorrelate by a factor of **e**.

$$F_s(\vec{k},\tau_\alpha) = \frac{1}{e} \tag{2.34}$$

By plotting τ_α/τ_θ vs $1/\phi$, where ϕ is the area (or volume) fraction, we get to compare the decoupling rates of spatial and orientational order near the glass transition.

In our analysis, we do not vary the wavevector $\vec{k}$, we keep it fixed at $\vec{k} = \frac{2\pi}{\sigma}$, to probe at the relevant timescales.

Chapter 3

Results and Discussion

3.1 Particle Tracking

Eqn.(2.3) and Eqn.(2.4) describe the process of bandpass filtering, Fig.(3.1) provides a comparison about what happens to an image of a 2D crystal after passing it through the bandpass filter. We can see that due to the gaussian blurring and subtraction of the background, the features appear to be more sharp and localised. This bandpassed image is then processed using Eqn.(2.5)-Eqn.(2.10) giving us a list of x, y, z, t coordinates. Fig(3.2) and Fig(3.3) illustrate the tracking results by plotting an xy cross-section of our system with the coordinates (blue dots) overlaid on top of the original image. This also provides an easy visual check to see if our tracking program has worked correctly.

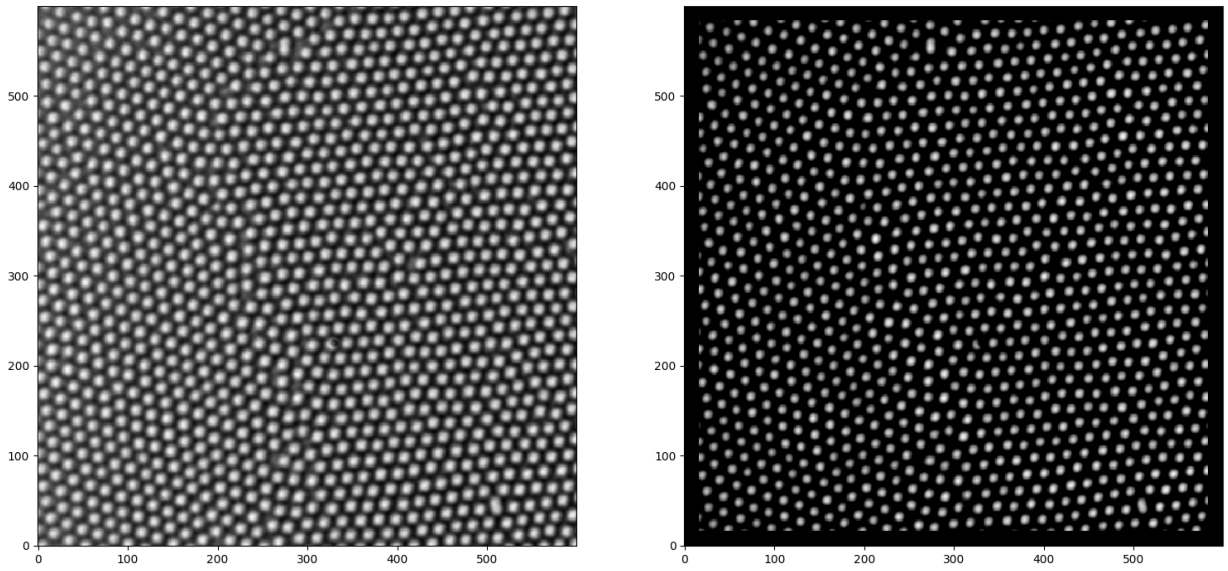


Figure 3.1: Before and after Bandpass Filter, 2D crystal

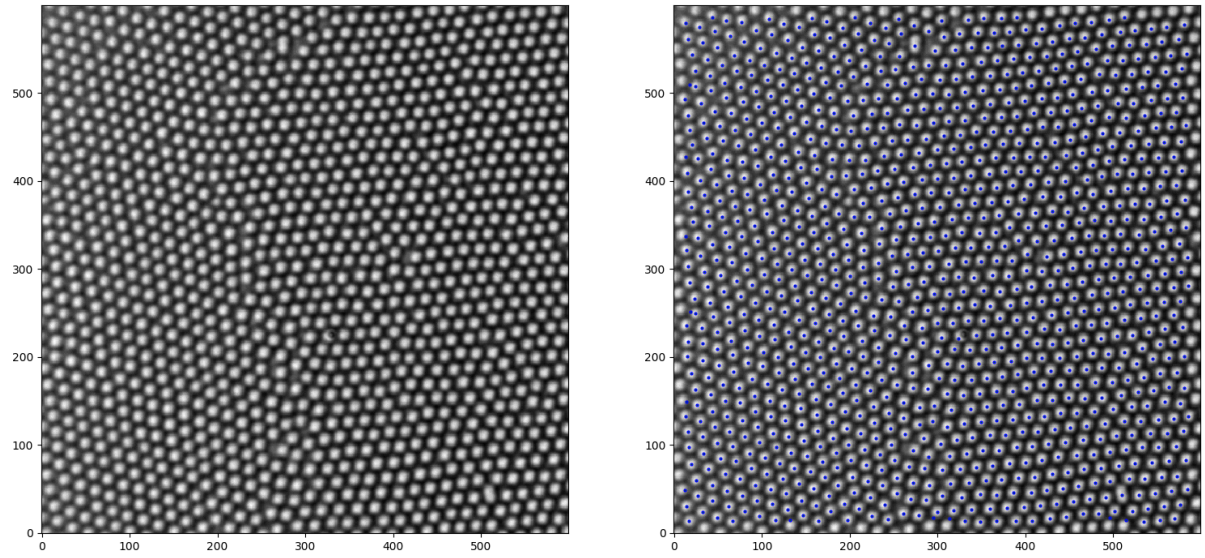


Figure 3.2: Tracked points overlaid on 2D crystal

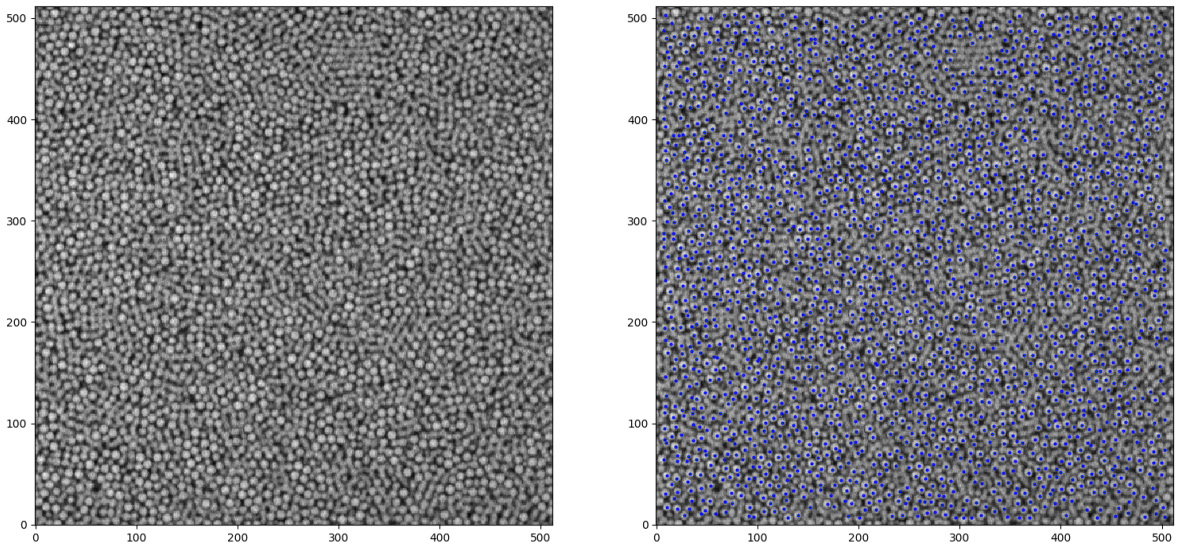


Figure 3.3: Tracked points overlaid on dense 3D glass

In Fig.(3.3), some particles seem to be unmarked, this because they are not in same z-plane as the points being plotted. Overall, the program gives satisfactory results with regards to the number of particles being found and their accuracy. Table (3.1) provides a comparison of the different algorithms (see Ch.2) and their implementations. The algorithm used by us is the same as Jensen & Nakamura, however it is significantly faster due to being implemented in Python which has better execution speeds and has plenty of highly optimised libraries and functions one can call upon. Our program provides both accuracy and speed and hence is ideal for the systems we are trying to analyse.

Also important to note that we parallelly processed multiple files at once using multithreading, which could easily be done as our programs were written in Python. Using 4 nodes with 4 CPUs each on the *Parambrahma* supercomputer, we were able to analyse 45 stacks of 3D data usually in under 2 hours, which otherwise would have taken ~ 40 hours on a single core. The other pre-written programs such as TrackPy or the codes available online do not have the provision of multithreading. Even if we were to add it to the programs ourselves, it would be quite difficult to execute in IDL or Matlab. Table (3.1) compares execution times for a single time stack i.e. multithreading is not used here, all codes were run on the same computer with 16 Gb RAM and

Intel i5 processor. If we were to scale it up to multiple stacks, our code would still take roughly the same amount of time to run (assuming those many CPU cores are available) whereas the runtime of the other codes would scale linearly.

Author	Language	No. of Particles	Accuracy	Time
estimate	—	$\sim 2,40,000$	—	—
TrackPy	Python	$44,263$	18.4%	26.4 sec
Crocker & Weeks	IDL	$1,46,408$	61.0%	52.9 sec
Kilfoil & Gao	Python	$1,67,992$	69.9%	30 min 25 sec
Jensen & Nakamura	Matlab	$2,28,964$	95.4%	1 hr 47 min
Me	Python	$2,28,717$	95.4%	44 min 56 sec

Table 3.1: Execution speed and accuracy of different programs for a single 3D stack

3.2 Finding Trajectories

We implement the procedure described in section (2.3) and construct k-d trees for each stack of images, find unique nearest neighbours for each particle between consecutive frames, assign unique identity numbers to each particle and then finally delete all the particles that are not present in every single frame. We had to develop our own code for this as none of the codes available online worked for dense 3D systems with a total of $\sim 10^7$ particles. Our code also works for 2D systems and gives similar results as the online codes, thereby confirming it's accuracy.

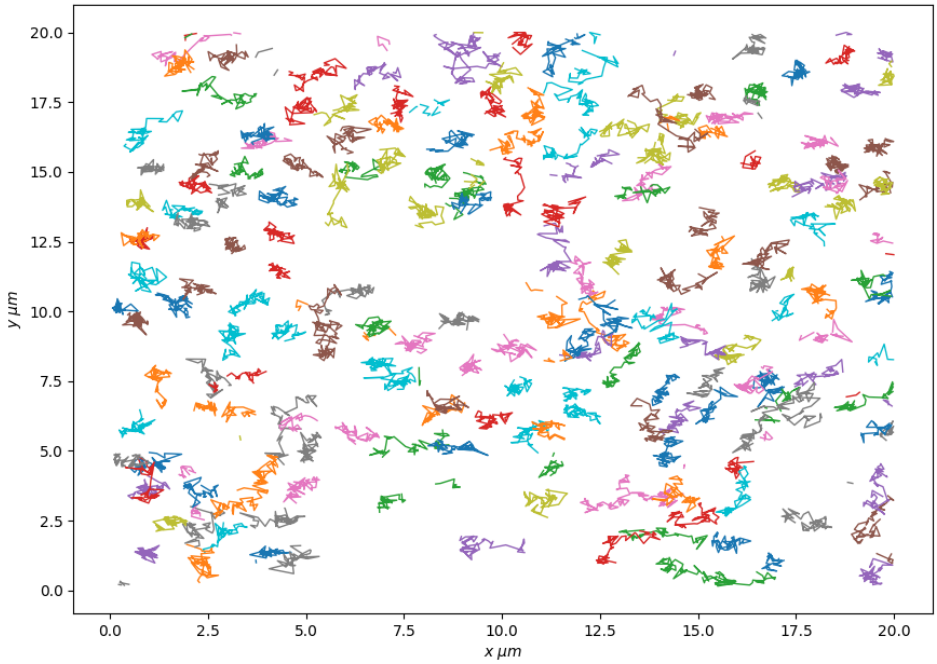


Figure 3.4: Trajectories of a few particles over 45 timesteps

As we can see in Fig.(3.4), the particles execute Brownian motion about their mean position. In table (3.2) we compare how many particles are lost due to drifting out of the edges or due to the inability of the code to catch extremely fasting moving particles i.e. the particles for which $\Delta x >> \sigma/2$. As our system is densely packed such particles will be very few as the motion is restricted due to cage formations by neighbouring particles, however the possibility cannot be completely ruled out. In 2D we hardly lose any particles due to the negligible drift in the system i.e. all particles stay within the field of view throughout the measurements, no leakage near the boundaries. Therefore, it is important to look at the average number of particles per timestep after tracking, and then the number of particles per timestep left after the goodenough filter. Average particles per stack is calculated by summing all the tracked particles in all frames and dividing by the total number of timesteps. Goodenough parameter is equal to the number of frames, i.e. we delete all those particles that are not present in every single frame.

Vol. Frac. ϕ	Est. no. of part.	Avg. part. per stack	After goodenough filter	Part. lost
0.55	$\sim 2,30,500$	$2,01,454.6$	$1,46,054$	27.5%
0.56	$\sim 2,38,000$	$2,10,683.0$	$1,26,232$	40.1%
0.57	$\sim 2,50,000$	$2,32,063.5$	$1,72,835$	25.5%
0.58	$\sim 2,07,500$	$1,83,122.2$	$1,34,535$	26.5%
0.59	$\sim 2,30,000$	$2,08,849.9$	$1,55,864$	25.4%
0.60	$\sim 2,14,500$	$1,89,838.9$	$1,55,483$	18.1%

Table 3.2: Particles lost after time tracking - 3D

For the 3D data, the z-size (height) was not the same for all volume fractions. The *xy* plane always had dimensions $92.4\mu m * 92.4\mu m$, however the z-size varied between $50-60\mu m$ (see Table (3.3)). Therefore in Table (3.2), the number of particles per stack does not increase with increasing volume fraction. Another factor to consider here is that not all measurements are done over the same time period. For higher volume fractions, the particles move slowly, therefore it is necessary to conduct the measurements over long periods of time to observe movement. We did not strictly follow this as there were other variables to consider such as drift or evaporation of the solvent due to heat from the laser, etc. Through trial and error we recorded the data which gave us the best results. In 2D, the measurements were consistent with *xy* dimensions $145\mu m * 145\mu m$ and $10,000$ frames at 21 fps for all area fractions.

Volume Fraction ϕ	z-size (μm)	No. of stacks	Time per stack (sec)	Total time
0.55	60.00	45	38.632	28 min 58 sec
0.56	60.75	45	39.116	29 min 20 sec
0.57	62.85	45	22.037	16 min 31 sec
0.58	51.60	45	17.725	13 min 17 sec
0.59	55.50	45	60.0	45 min
0.60	51.60	30	180.0	1 hr 30 min

Table 3.3: Information about 3D datasets

3.3 Structural Quantities

1. Mean Square Displacement :

MSD is calculated according to Eqn.(2.11) for both 2D and 3D data. Cage-relative MSD for 2D is defined according to Eqn.(2.12). In Fig.(3.6), we see that the cage-relative MSD is less jagged and much smoother compared to the standard MSD, this means that there are no sudden jumps and fluctuations have been successfully suppressed. In Fig.(3.5), we see no plateau region for the smaller $\phi = 0.57, 0.62, 0.65$. For $\phi = 0.68, 0.71, 0.73$ we can see the plateau region beginning to form. For $\phi = 0.75$ we see the longest plateau region, followed by a linear increase. This gradual change in behaviour is an indication of glass transition occuring in this regime.

For 3D (Fig.(3.7)), we do not see any plateaus or any distinct variations between the different volume fractions. It looks as if we are still in the linear domain, a longer measurement where $(\Delta t) \sim 10^4$ sec would be ideal. Also, a larger variation in ϕ, perhaps between $0.45 - 0.65$ may make the glass transition even clearer.

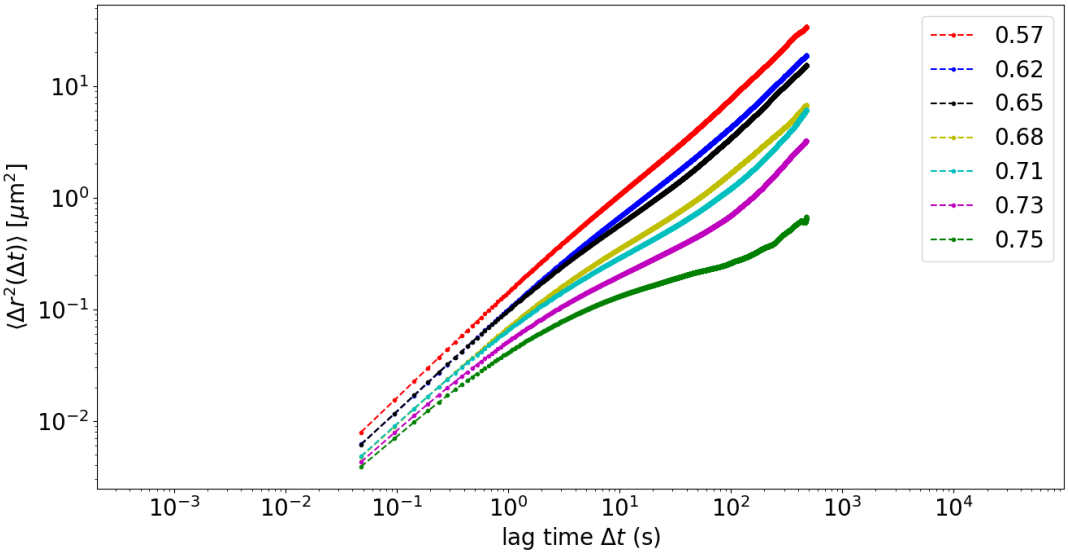


Figure 3.5: MSD for all ϕ - 2D

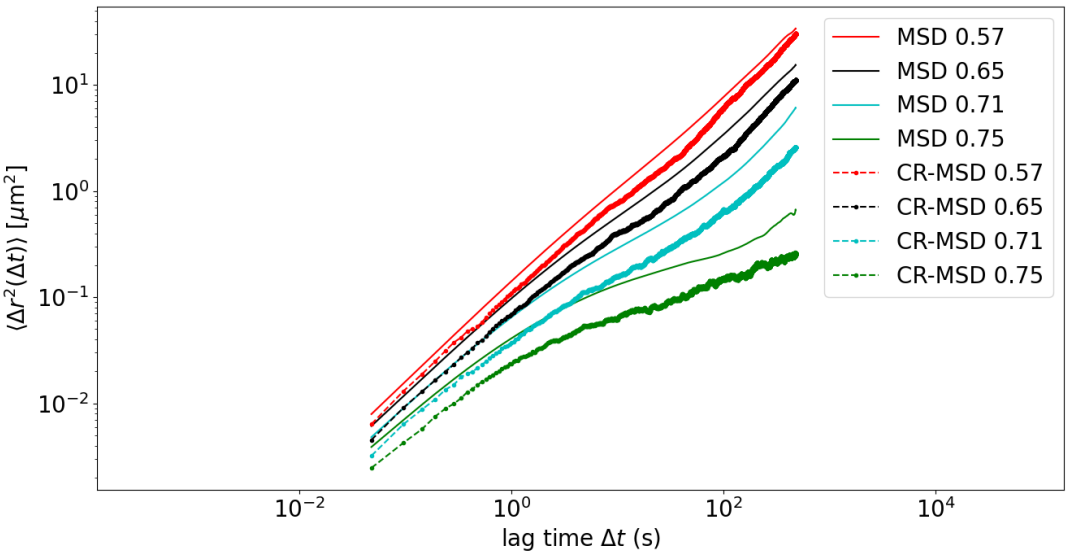


Figure 3.6: MSD & Cage Relative MSD - 2D

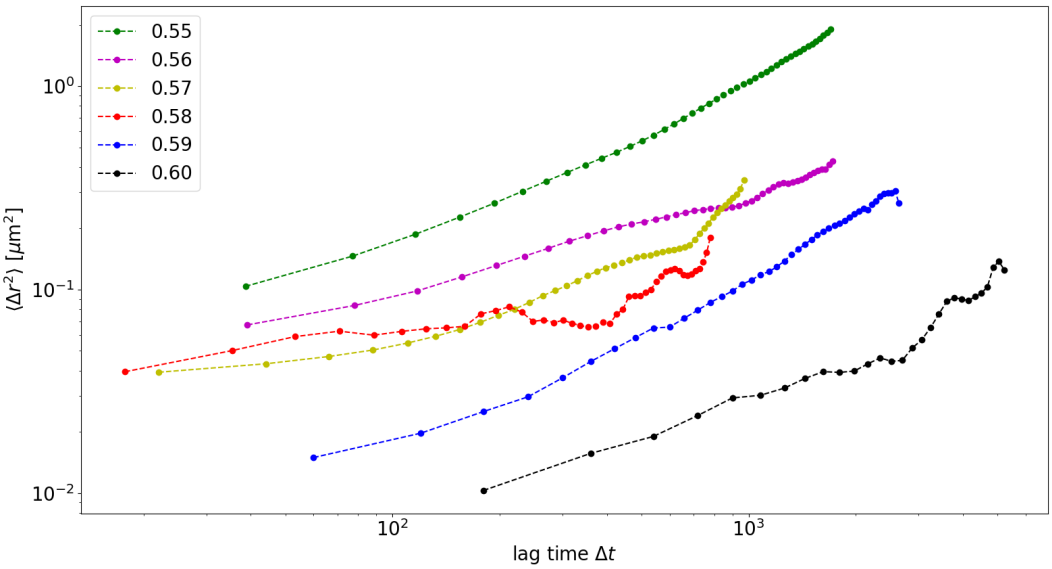


Figure 3.7: MSD for all ϕ - 3D

2. Pair Correlation Function ($g(r)$) :

We calculate $g(r)$ using Eqn.(3.15) and the 3-peak $g(r)$ using the equation for bidisperse systems Eqn.(2.16). For 3D we used colloids with diameter = $1.3\mu m$ and 7% polydispersity and for 2D we used particles of diameter $2.32\mu m$ & $3.34\mu m$. This is exactly where we see peaks in Fig.$(3.10-13)$. We have marked vertical lines at these points in the graph, the peaks appear to be shifted fractionally to the right, this is due to the 7% polydispersity, also sometimes the PMMA particles may swell up slightly after being kept in CHB for too long. In Fig.$(3.8-11)$, we see the peak due to the small-big particle interaction is at $2.83\mu m$. The first peak is considered to be the diameter σ and the first minima is considered as the average interparticle distance. Only for 3D systems do we need average distance between neighbours, from the graph we see that it is roughly equal to $1.3*\sigma$.

The graphs are smooth with sharp peaks where we expect them to be, there are no bumps before the first peak and then finally all graphs oscillate and settle to $y=1$. This gives us assurance about the tracking programs working correctly. If we had detected noise or some other false particles, it generally shows up as an aberration in $g(r)$.

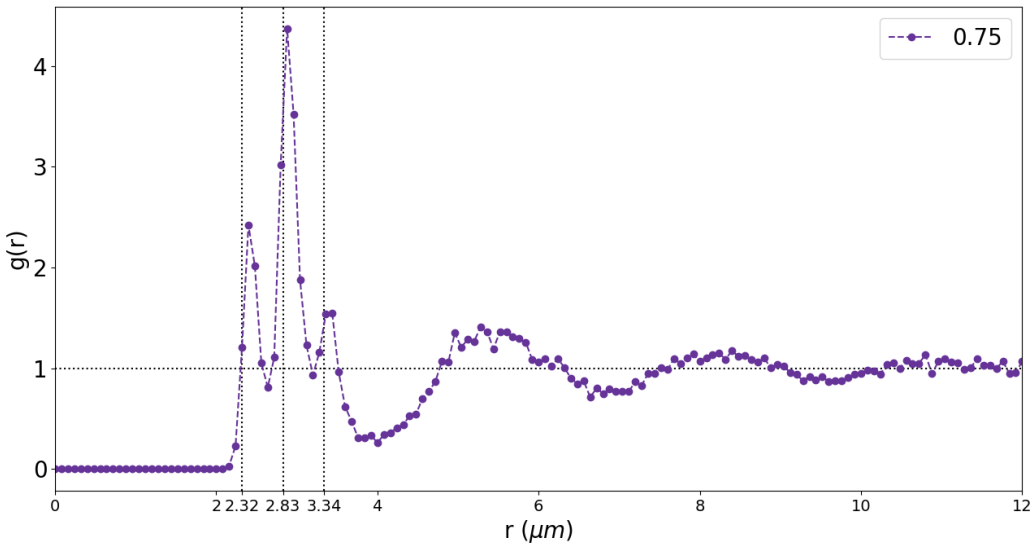


Figure 3.8: Two component $g(\vec{r})$ 2D - $\phi = 0.75$

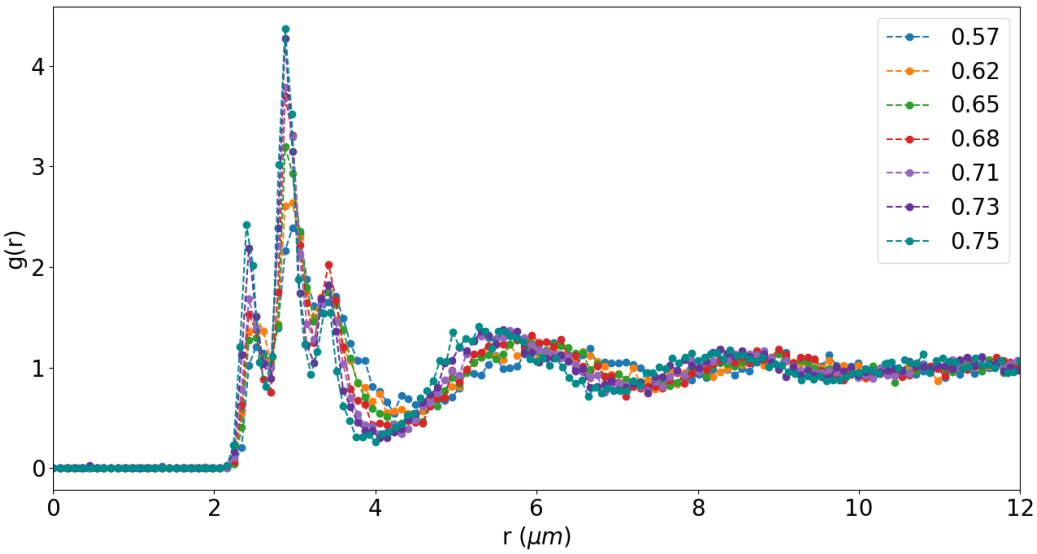


Figure 3.9: Two component $g(\vec{r})$ 2D - all ϕ

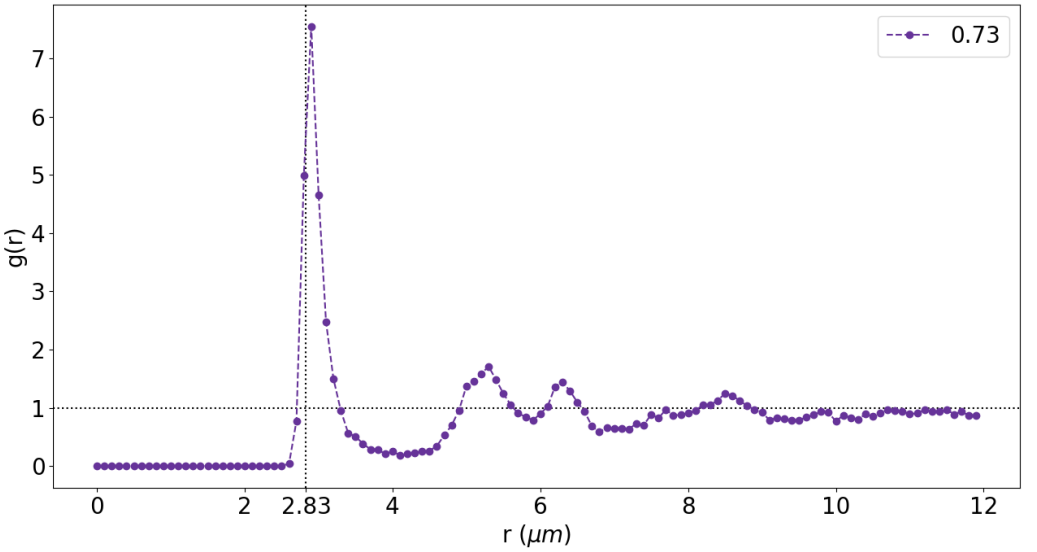


Figure 3.10: Standard $g(\vec{r})$ 2D - $\phi = 0.73$

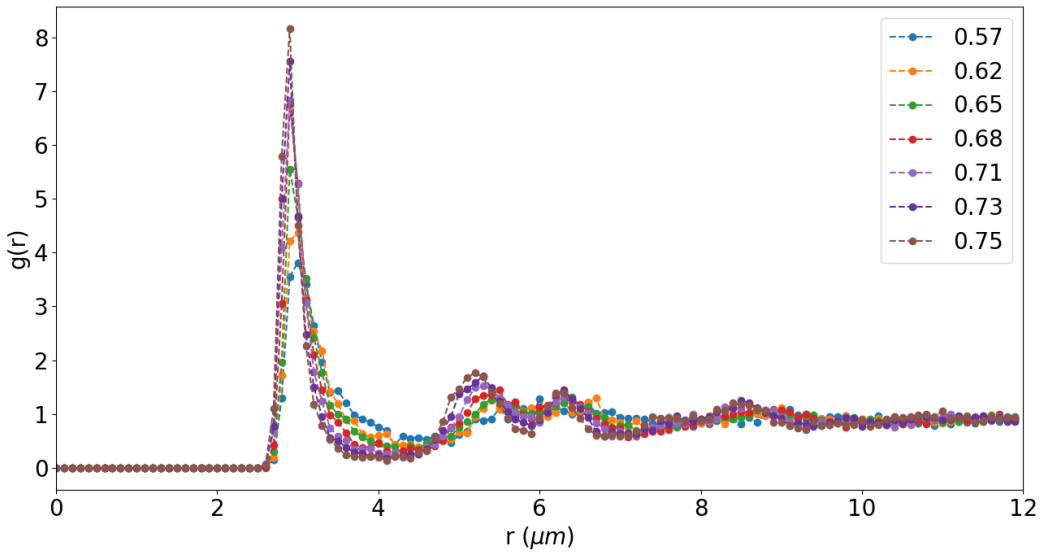


Figure 3.11: Standard $g(\vec{r})$ 2D - all ϕ

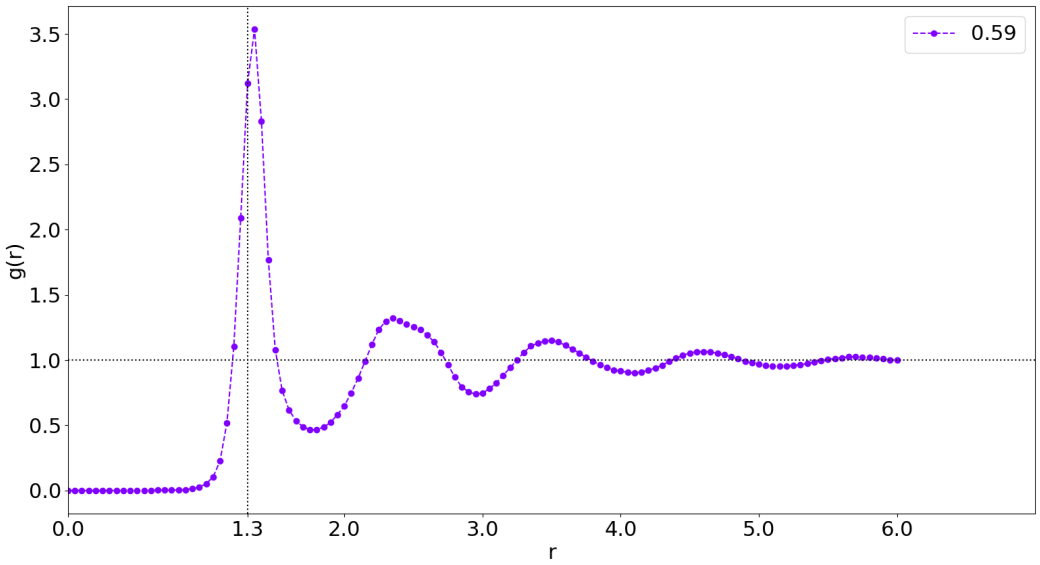


Figure 3.12: $g(\vec{r})$ 3D - $\phi = 0.59$

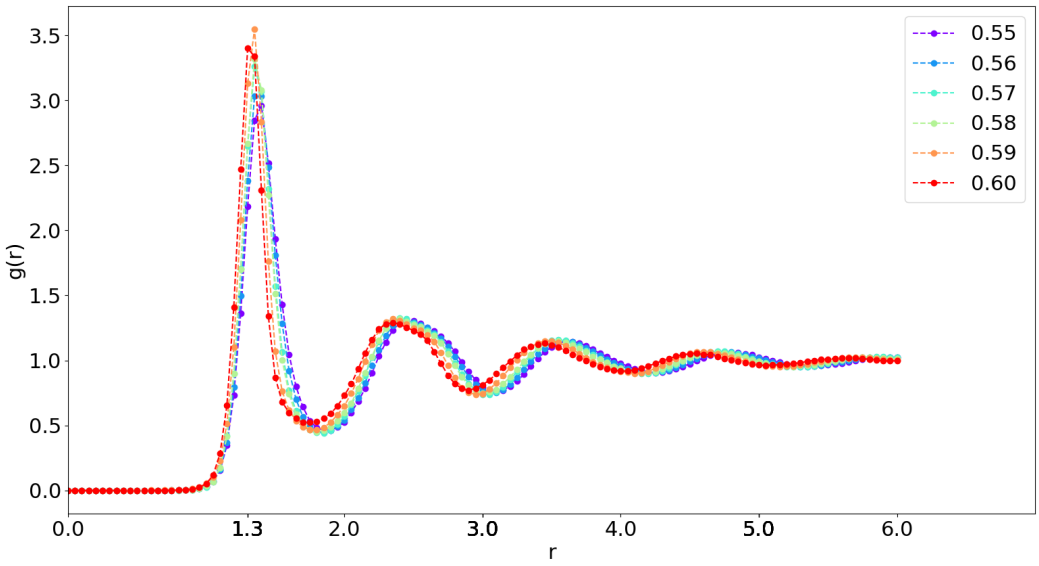


Figure 3.13: $g(\vec{r})$ 3D - all ϕ

3. Bond Order Orientation (BOO) :

Firstly, we calculate and compare BOO-3D using different neighbour counting methods (Fig.(2.2)). Table 3.4 serves as an example of how much BOO-3D is sensitive to slight variations of neighbours. The values of this table were calculated on the same dataset of $\phi = 0.55$. We see that q_2 has a minimum value of 0.13097 and a maximum at 0.62625, a difference by a factor of almost 5. This emphasises the need to use Minkowski Structure Metrics to create a consistent and reliable way to calculate BOO.

Fig.(3.14) shows the variation of q_6 over ϕ when different methods are used. The k-nearest neighbours method seems to be in close agreement with the MSM. This is a coincidence and won't always happen, we wouldn't have seen for q_2 or q_4 or if we had chosen $k = 15$ instead of $k = 13$. Choosing k itself can be a difficult job as we do not know on an average how many neighbours each particle has for different ϕ. MSM eliminates the need for this kind of guesswork.

Method	q_2	q_4	q_6	q_8	q_{10}
range : $r = 1.1\sigma$	0.40892	0.47635	0.58089	0.47427	0.50562
range : $r = 1.2\sigma$	0.20906	0.28780	0.47032	0.31868	0.33873
range : $r = 1.3\sigma$	0.15121	0.21806	0.43526	0.29206	0.28339
range : $r = 1.4\sigma$	0.13097	0.18078	0.40846	0.29060	0.25921
k-nearest : $k = 12$	0.41357	0.35455	0.42389	0.44695	0.47873
k-nearest : $k = 13$	0.41031	0.34335	0.40653	0.42976	0.46047
k-nearest : $k = 14$	0.40797	0.33349	0.38876	0.41247	0.44267
k-nearest : $k = 15$	0.40640	0.32533	0.37200	0.39603	0.42638
Delaunay Tri./Voronoi Tess.	0.62625	0.34567	0.32818	0.29729	0.28389
Minkowski Structure Metric	0.12793	0.24308	0.40356	0.29296	0.28791

Table 3.4: q_l using different ways to find nearest neighbours

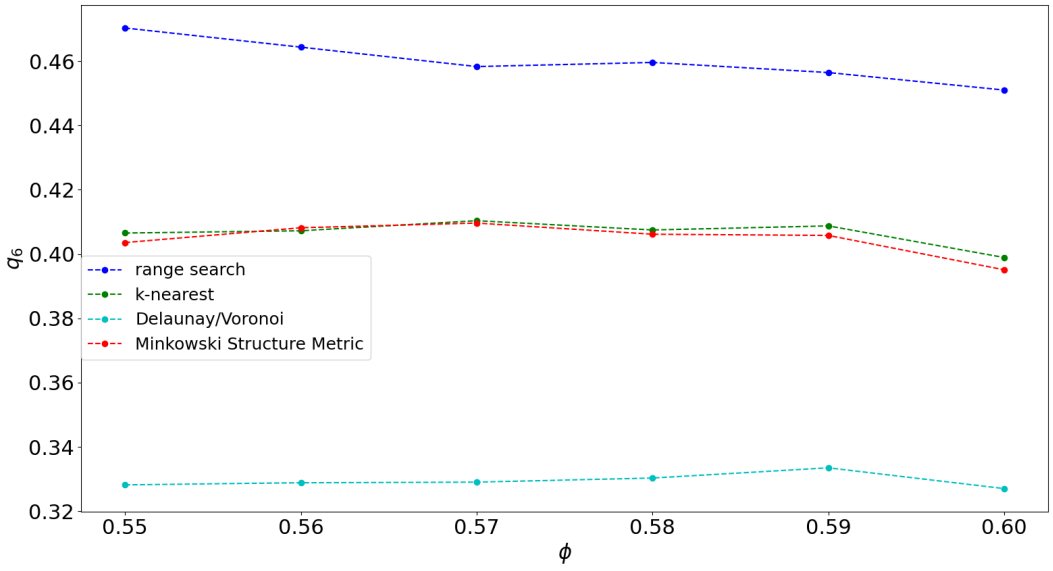
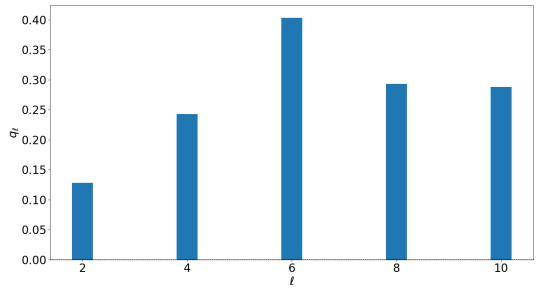
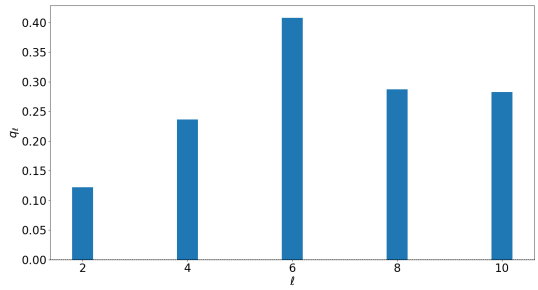


Figure 3.14: q_6 vs ϕ, for each method of counting neighbours. $\phi = 0.55$, $r = 1.3\sigma$ and $k = 13$.

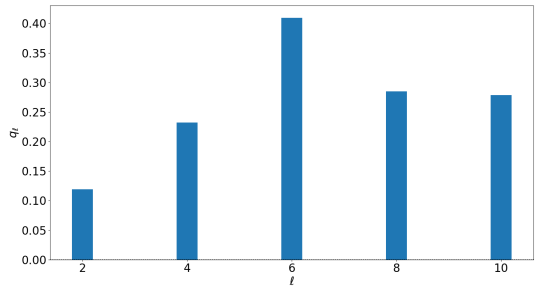
Fig.(3.15-3.19) are plots of Q_ℓ & w_ℓ calculated using Eqn.$(2.19, 21)$. We compare the shapes and nature of these plots to those found in literature [12]. We observe that all the ϕ show very strong icosahedral order, with weak tendencies towards HCP and FCC. The icosahedral order does not grow while approaching glass transition.



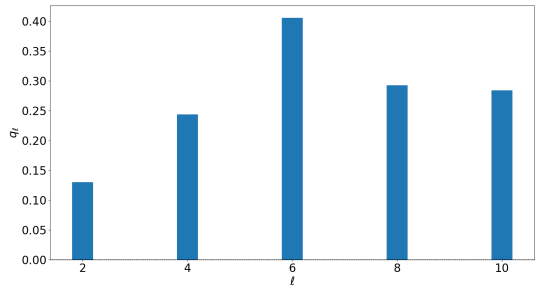
(a) 0.55



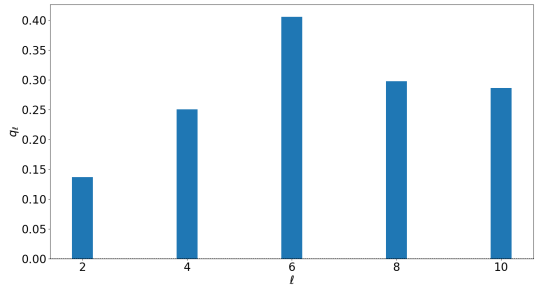
(b) 0.56



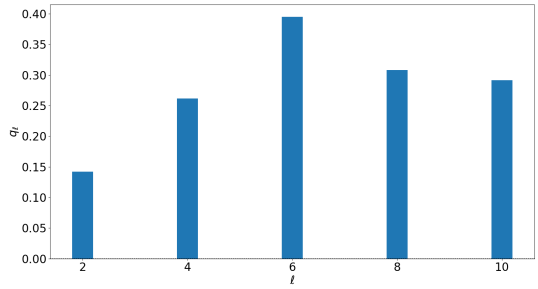
(c) 0.57



(d) 0.58

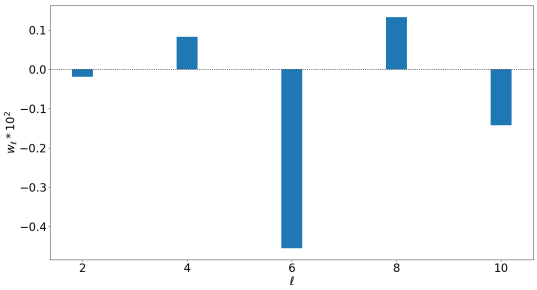


(e) 0.59

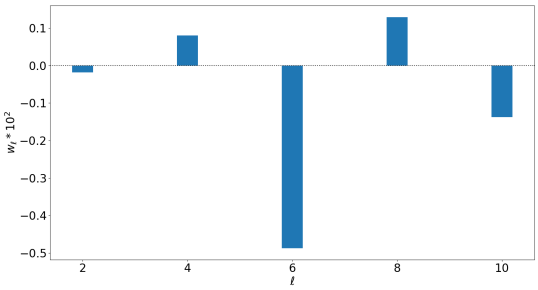


(f) 0.60

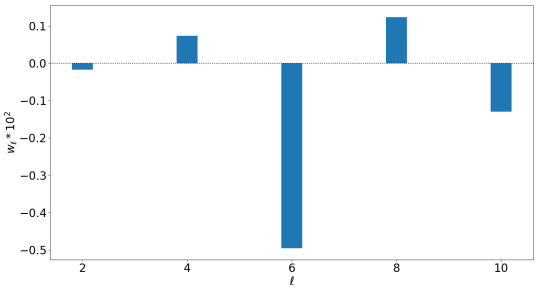
Figure 3.15: Q_ℓ vs ℓ for all ϕ - 3D



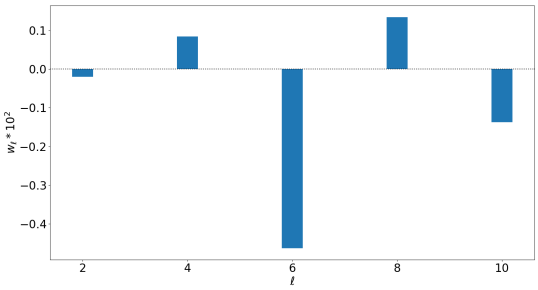
(a) 0.55



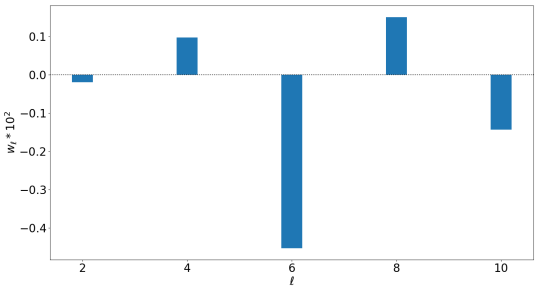
(b) 0.56



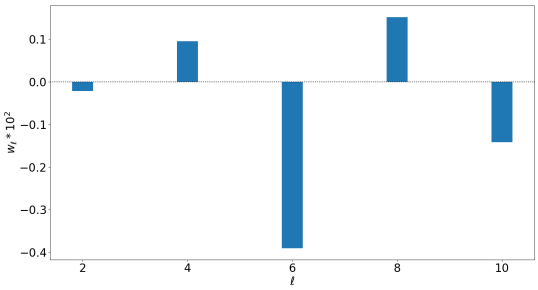
(c) 0.57



(d) 0.58

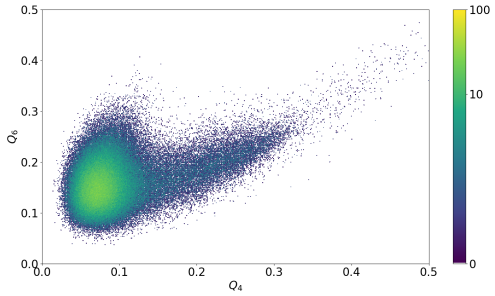


(e) 0.59

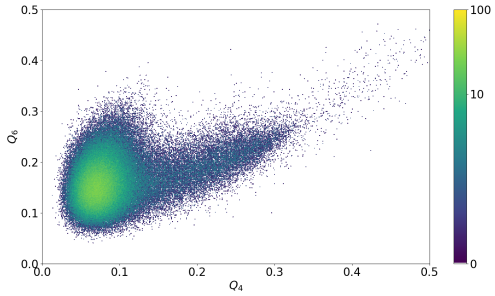


(f) 0.60

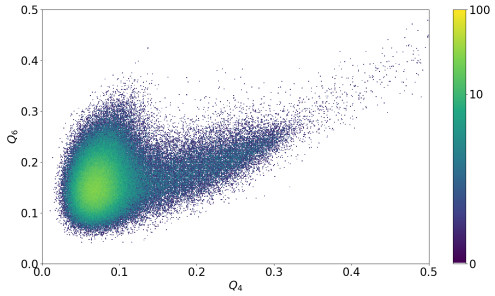
Figure 3.16: w_ℓ vs ℓ for all ϕ - 3D



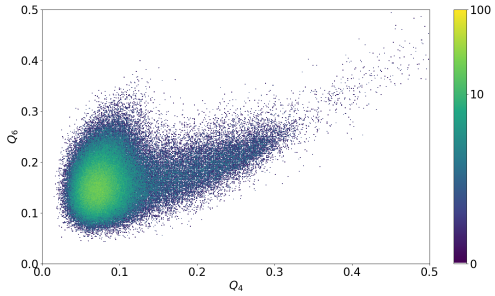
(a) 0.55



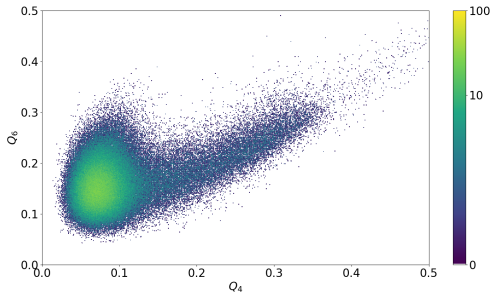
(b) 0.56



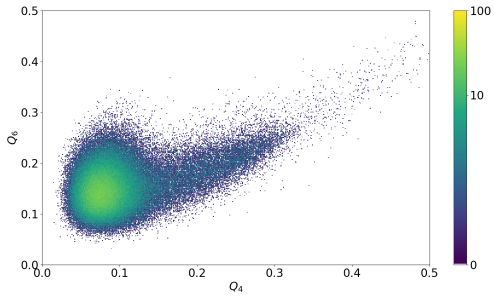
(c) 0.57



(d) 0.58

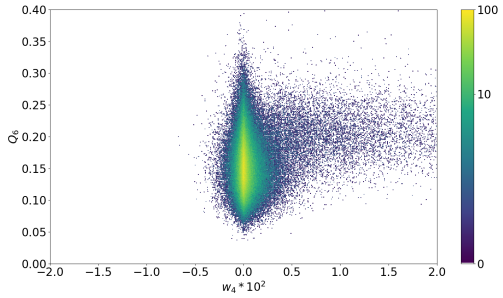


(e) 0.59

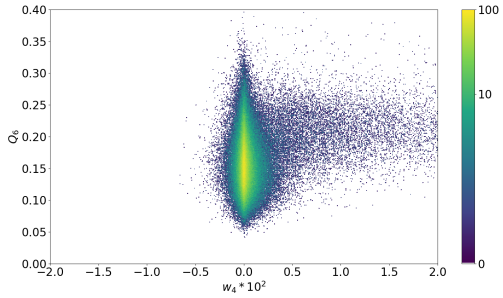


(f) 0.60

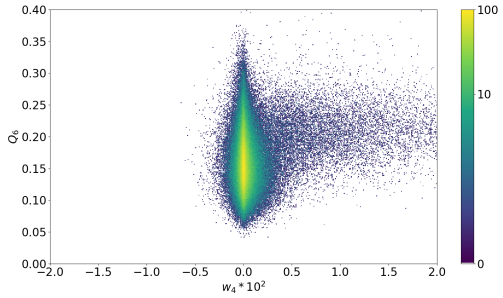
Figure 3.17: Q_6 vs Q_4 for all ϕ - 3D. Colour represents number of points per pixel.



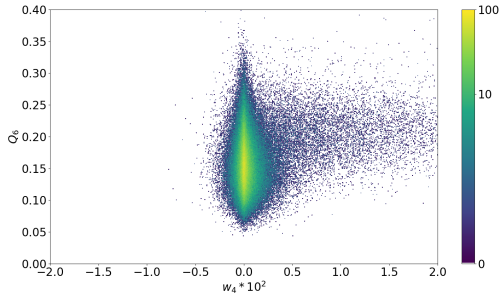
(a) 0.55



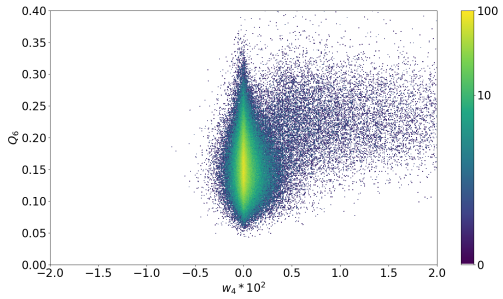
(b) 0.56



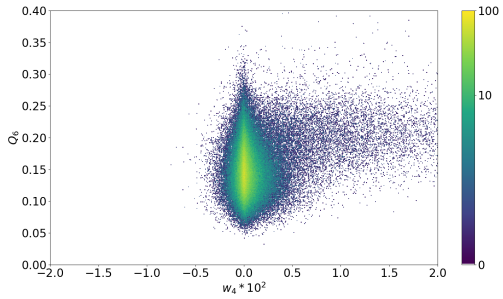
(c) 0.57



(d) 0.58

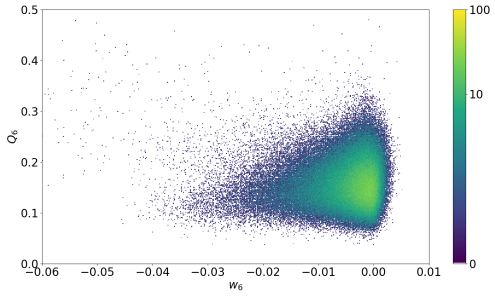


(e) 0.59

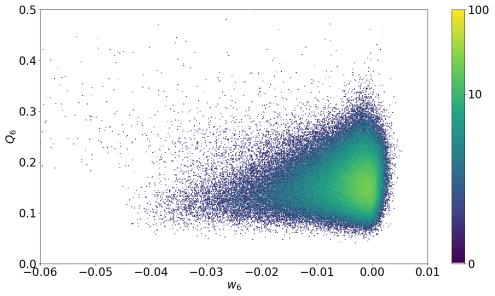


(f) 0.60

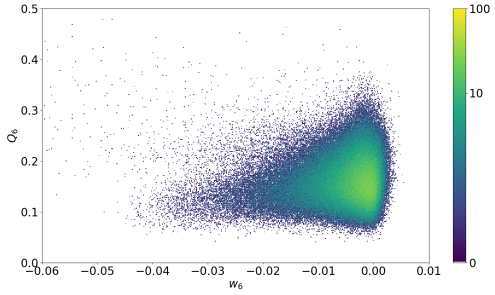
Figure 3.18: Q_6 vs w_4 for all ϕ - 3D. Colour represents number of points per pixel.



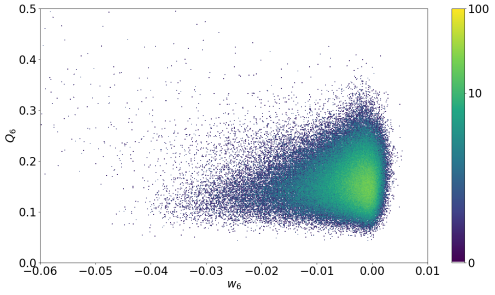
(a) 0.55



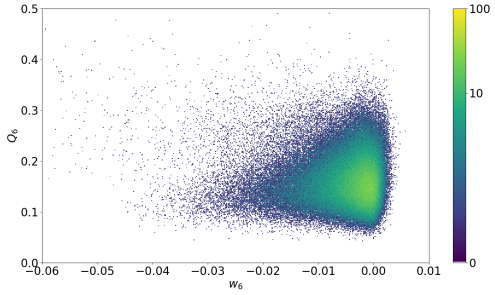
(b) 0.56



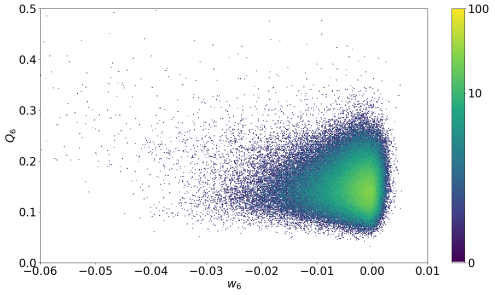
(c) 0.57



(d) 0.58



(e) 0.59



(f) 0.60

Figure 3.19: Q_6 vs w_6 for all ϕ - 3D. Colour represents number of points per pixel.

To find the characteristic bond angle relaxation time τ_θ, we calculate C_Q and C_ψ according to Eqn.(2.23,24). $\tau\theta$ is found using Eqn.(2.25). For 3D systems, at higher values of ϕ we do not see a significant decorrelation. Therefore, we cannot calculate $\tau\theta$ with our current data, we need to conduct much longer measurements in order to find the characteristic timescales for 3D.

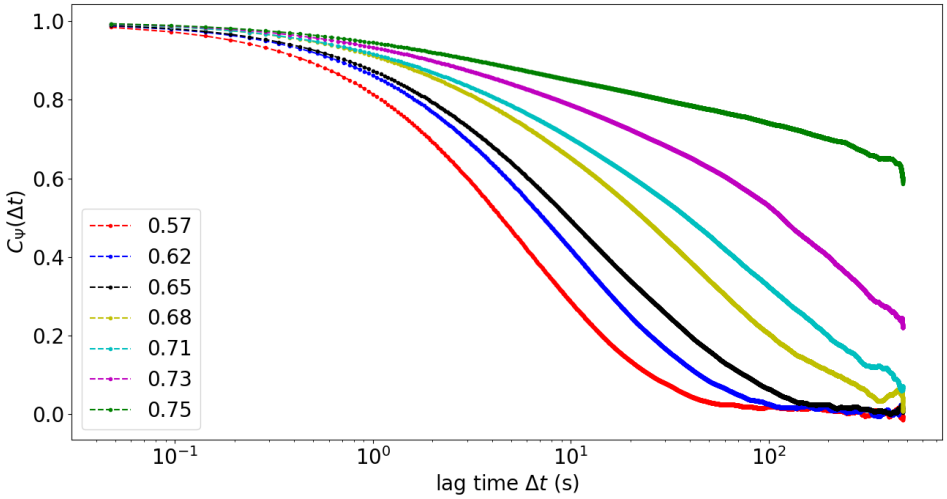


Figure 3.20: Bond Orientational Correlation Function - 2D

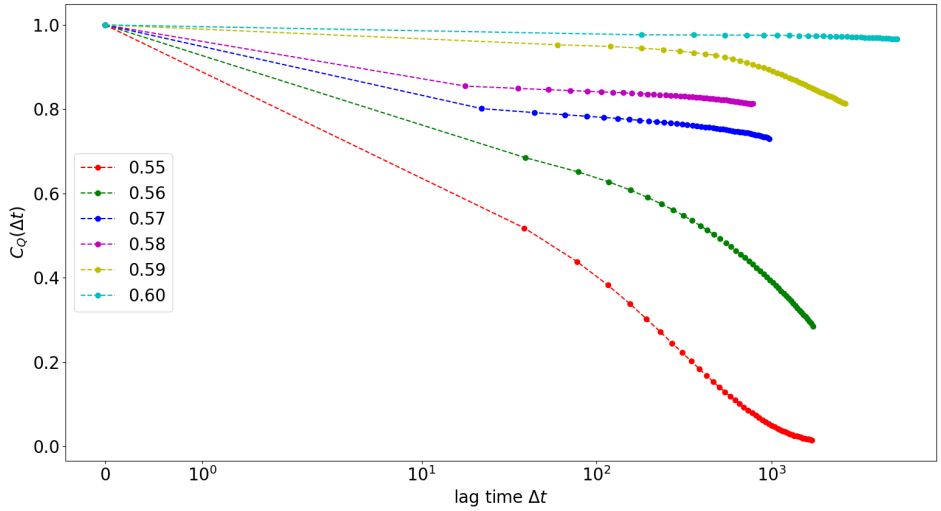


Figure 3.21: Bond Orientational Correlation Function - 3D

4. Intermediate Scattering Function (ISF) :

ISF and cage-relative ISF are calculated according to Eqn.$(2.31, 33)$ respectively. The graph for 3D data looks erratic, perhaps a smoother and more consistent plot could be achieved by taking multiple measurements over a longer time and averaging them.

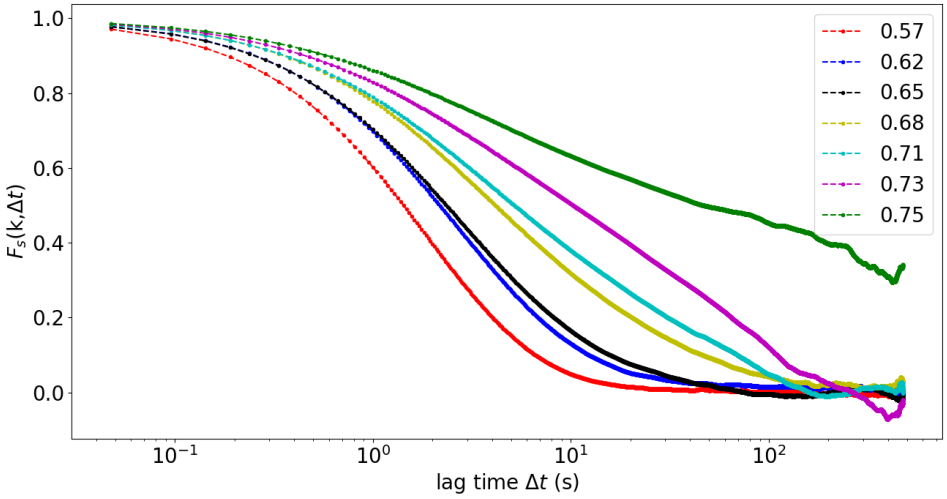


Figure 3.22: Intermediate Scattering Function - 2D

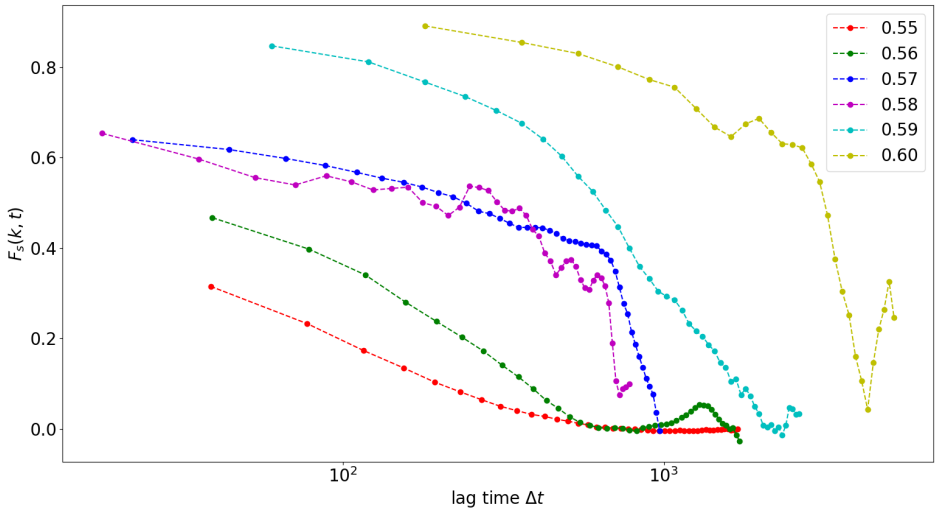


Figure 3.23: Intermediate Scattering Function - 3D

We compare the relaxations of $F_S(\vec{k},t)$ and $C_\psi(t)$ (Fig.$(3.24-26)$). It is a direct way to compare the spatial and orientational relaxations. Cage-relative $F_S(k,t)$ (Fig.(3.25)) follows C_ψ more closely as compared to regular $F_S(k,t)$.

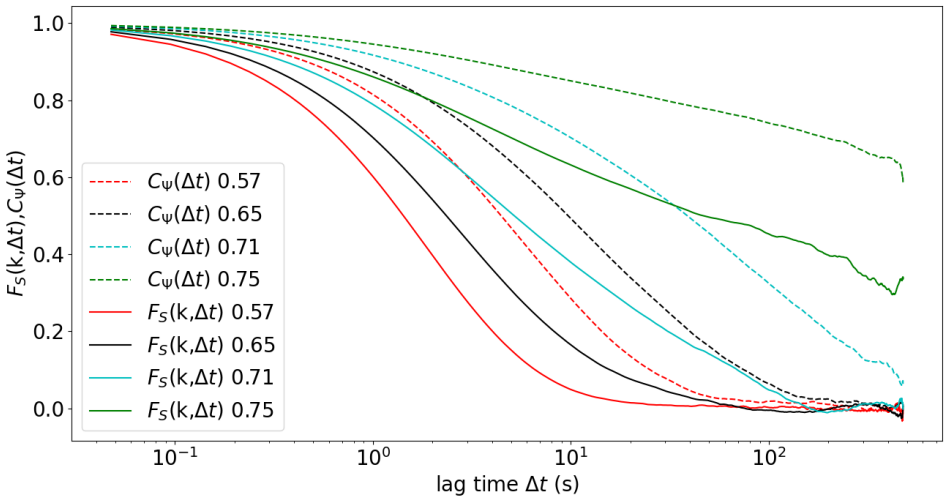


Figure 3.24: $F_s(k,t)$ & $C_\psi(t)$ - 2D

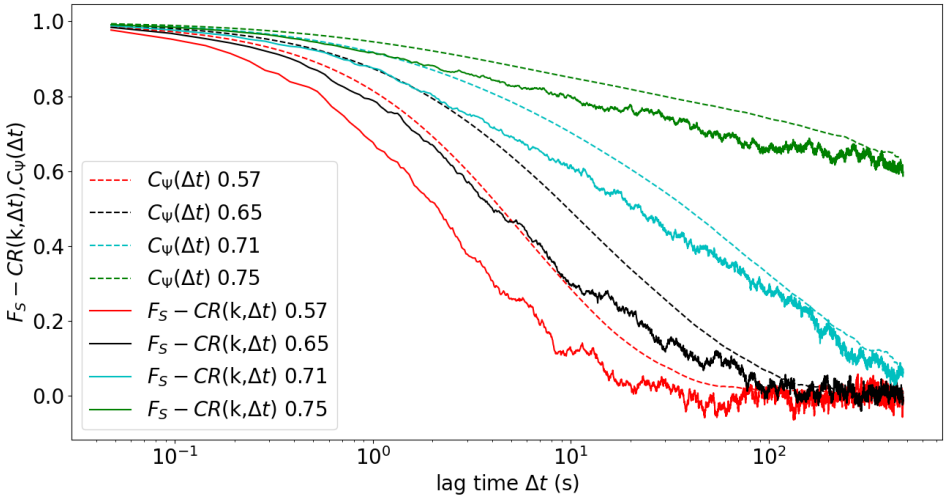


Figure 3.25: Cage relative $F_s(k,t)$ & $C_\psi(t)$ - 2D

Characteristic spatial relaxation time τ_α is found using Eqn.(2.34). We can use τ_α as a normalisation factor for $F_S(\vec{k},t)$ and $C_\psi(t)$ (Fig.(3.26)). These comparisons are quantified in Fig.(3.27) as $\frac{\tau_\theta}{\tau\alpha}$ tells us the exact relation between the different relaxations as we approach glass transition. As mentioned above, as of now we cannot perform the τ_θ,τ_α analysis for 3D, we need to perform multiple long-time measurements for a wider range of ϕ and average over them.

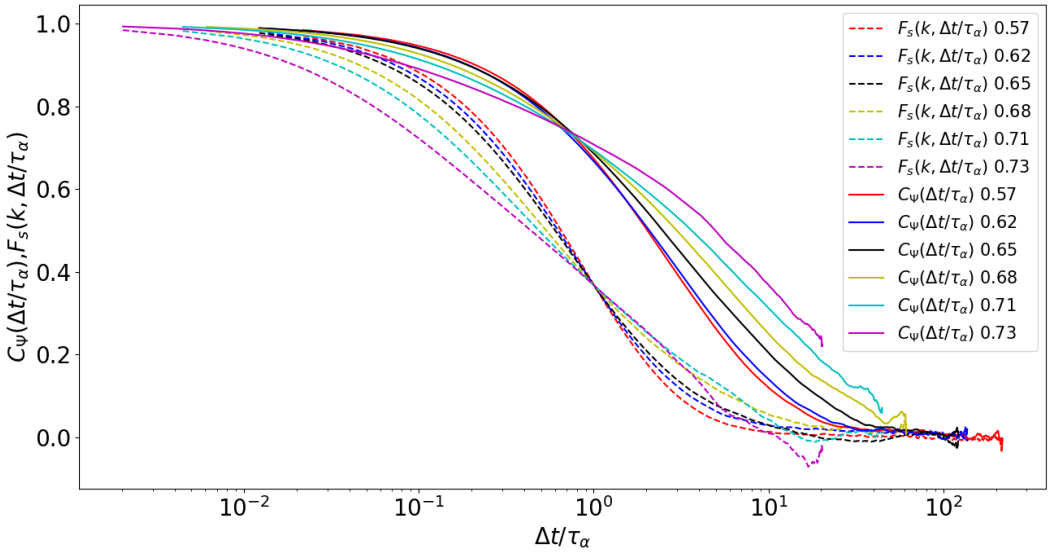


Figure 3.26: Normalised $F_s(k,t)$ & $C_\psi(t)$ - 2D

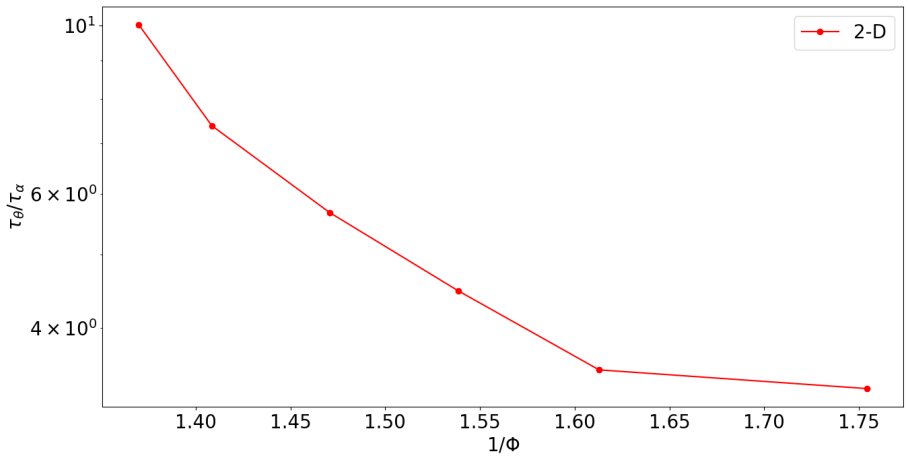


Figure 3.27: $\frac{\tau_\theta}{\tau_\alpha}$ vs $\frac{1}{\phi}$ - 2D

Chapter 4

Conclusions and Outlook

We have developed an efficient, robust and accurate method for locating and tracking a huge number of particles in 3D. For locating the particles, we achieve the same accuracy as Jensen & Nakamura [9] while being more than twice as fast for a single stack (see Table 3.1) and by implementing multithreading routines, we are able to scale our system size to multiple stacks without increasing the execution time, which is not implemented in any of the popular programs. For time tracking, compared to other softwares and programs available online, the number of particles our code can handle is greater by atleast two orders of magnitude, if not more. In 2D, we execute a locating and tracking routine with almost 100% accuracy, while in 3D we end up losing $\sim 20\% - 25\%$ of the particles due to particles drifting in & out of the field of view, and due to extremely fasting moving particles which jump more than a distance σ within a single timestep. There are ways to circumvent the latter problem, as suggested by Kilfoil & Gao [5], first we time-track the slowest moving particles, remove them from the data, then track the slightly faster moving particles and so on until most particles are tracked. Our program could be adapted to implement this, we may see an improvement of upto $\sim 5\% - 10\%$ by capturing the faster particles.

We used these programs to locate and track $10,000$ frames of 2D data with area fractions between $0.57 - 0.75$ taken at 21 frames per second, and 45 stacks of 3D data for volume fractions

between $0.55-0.59$ (and 30 stacks for 0.60) taken at various fps (see Table 3.3). Using the tracked data, we calculated 4 major structural quantities: *MSD, g($\vec{r}$), ISF, BOO* (see Ch.2.4). By comparing these quantities as ϕ varied, we observed the differences between the dynamics of 2D and 3D glasses near the glass transition.

For 2D systems, we get to see a transition in MSD as went from lower area fractions to higher ones. The highest $\phi = 0.75$ shows two distinct regions of linear growth with a plateau in the middle. This is the expected behaviour from colloidal glasses following Newtonian dynamics. In 3D, we do not see any such plateau regions. In literature, we often see these measurements go on for 10^4-10^5 seconds and cover a wide range of volume fractions, our experiments were conducted on a timescale of $\sim 10^3$ seconds (see Table 3.3) with $\phi = 0.55-60$. We plan to conduct longer measurements with a bigger range of ϕ in the future.

The pair correlation functions do not have any kinks or sudden jumps and have sharp peaks exactly where we expect to see them (Fig. 3.8 - 3.12). This gives us confidence that the particle tracking was done properly and there is negligible noise in the data. The first peak of $g(\vec{r})$ gives us the apparent diameter of the particles, the first minima gives us the average interparticle separation. In the bidisperse $g(r)$ (eqn.(2.13), fig(3.5)), the middle peak gives us an idea of how strongly the two types of particles are interacting.

Bond Orientational Order provides us with a measure of local as well as extended orientational symmetries within a system. It is straightforward to calculate BOO in 2D, however in 3D we show that BOO is extremely sensitive to the way in which we count nearest neighbours (Table 3.4, Fig.2.2, Fig.3.30). We employ Minkowski Structure Metrics to create a robust and parameter free method to calculate BOO [13]. We calculate rotationally invariant quantities such as Q_ℓ and w_ℓ for different values of ℓ. We plot their histograms, Q_4 vs Q_6, w_4 vs Q_6 and w_6 vs Q_6 and compare them to literature [15, 13, 17]. We observe that all volume fractions display very strong icosahedral order, with weak tendencies towards HCP and FCC [12]. The icosahedral order does not grow while approaching glass transition. We also calculate the time correlation of BOO, C_ψ and C_Q, which gives us a characteristic bond angle relaxation time (t_θ) for the system. For our

3D systems, only $\phi = 0.55, 0.56$ decorrelated by a factor of e. Therefore, it was not possible to calculate t_θ for all the ϕ. This is further indication that we need to carry out longer measurements especially for the higher ϕ.

The ISF defines a correlation between particles in reciprocal space. We also define a cage-relative ISF for 2D systems (eqn.(2.33)), which suppresses the jump motions and long-wavelength fluctuations within the system. Similar to BOO (eqn(2.24)), we define a characteristic spatial relaxation time (eqn(2.34)) for the system τ_α. Here we do see decorrelation by a factor of e for all the ϕ, we normalise the timescale by dividing by τ_α so that we can compare the decorrelation rates of the various ϕ. The extent of spatial and orientational decorrelation can be compared by the ratio τ_θ/τ_α (fig.(3.27)).

The results for 2D glasses are satisfactory and we need no further data. For 3D glasses, we need to make longer measurements on a wider range of volume fractions so that we get to see relaxations of the systems. In order to quantify the dynamic heterogenities displayed in Fig.$(3.22-27)$, we plan to calculate the four-point Structure Factor $S_4(q,t)$. We then fit $S_4(q,t)$ to the Ornstein-Zernicke form $\chi_4(t)/\left\{1+[q\xi_4(t)]^2\right\}$, here ξ_4 is the dynamic susceptibility which characterises the overall strength of the dynamic heterogenity [3]. This gives us $\chi_4(t)$ which is the dynamic correlation length of the system. It has been shown using simulations, that for 2D : $\chi_4(\tau_\alpha) \sim \xi_4(\tau_\alpha)^{1.5}$ whereas for 3D : $\chi_4(\tau_\alpha) \sim \xi_4(\tau_\alpha)^3$ [4]. We plan to verify the same using data collected from our experiments, thus demonstrating another way in which two and three dimensional glass transitions differ.

Bibliography

[1] Neil W Ashcroft and N David Mermin. "Solid state". In: *Physics (New York: Holt, Rinehart and Winston) Appendix C* (1976).

[2] John C Crocker and David G Grier. "Methods of digital video microscopy for colloidal studies". In: *Journal of colloid and interface science* 179.1 (1996), pp. 298–310.

[3] Elijah Flenner, Hannah Staley, and Grzegorz Szamel. "Universal features of dynamic heterogeneity in supercooled liquids". In: *Physical review letters* 112.9 (2014), p. 097801.

[4] Elijah Flenner and Grzegorz Szamel. "Fundamental differences between glassy dynamics in two and three dimensions". In: *Nature communications* 6.1 (2015), pp. 1–6.

[5] Yongxiang Gao and Maria L Kilfoil. "Accurate detection and complete tracking of large populations of features in three dimensions". In: *Optics express* 17.6 (2009), pp. 4685–4704.

[6] Jean-Pierre Hansen and Ian R McDonald. *Theory of simple liquids*. Elsevier, 1990.

[7] Gary L Hunter and Eric R Weeks. "The physics of the colloidal glass transition". In: *Reports on progress in physics* 75.6 (2012), p. 066501.

[8] Bernd Illing et al. "Mermin–Wagner fluctuations in 2D amorphous solids". In: *Proceedings of the National Academy of Sciences* 114.8 (2017), pp. 1856–1861.

[9] Katharine E Jensen and Nobutomo Nakamura. "Note: An iterative algorithm to improve colloidal particle locating". In: *Review of Scientific Instruments* 87.6 (2016), p. 066103.

[10] Willem K Kegel and Alfons van Blaaderen. “Direct observation of dynamical heterogeneities in colloidal hard-sphere suspensions”. In: *Science* 287.5451 (2000), pp. 290–293.

[11] Wolfgang Lechner and Christoph Dellago. “Accurate determination of crystal structures based on averaged local bond order parameters”. In: *The Journal of chemical physics* 129.11 (2008), p. 114707.

[12] Mathieu Leocmach and Hajime Tanaka. “Roles of icosahedral and crystal-like order in the hard spheres glass transition”. In: *Nature communications* 3.1 (2012), pp. 1–8.

[13] Walter Mickel et al. “Shortcomings of the bond orientational order parameters for the analysis of disordered particulate matter”. In: *The Journal of chemical physics* 138.4 (2013), p. 044501.

[14] Hayato Shiba, Peter Keim, and Takeshi Kawasaki. “Isolating long-wavelength fluctuation from structural relaxation in two-dimensional glass: Cage-relative displacement”. In: *Journal of Physics: Condensed Matter* 30.9 (2018), p. 094004.

[15] Paul J Steinhardt, David R Nelson, and Marco Ronchetti. “Bond-orientational order in liquids and glasses”. In: *Physical Review B* 28.2 (1983), p. 784.

[16] Katherine J Strandburg. “Two-dimensional melting”. In: *Reviews of modern physics* 60.1 (1988), p. 161.

[17] Skanda Vivek et al. “Long-wavelength fluctuations and the glass transition in two dimensions and three dimensions”. In: *Proceedings of the National Academy of Sciences* 114.8 (2017), pp. 1850–1855.

[18] Eugene P Wigner. “On the matrices which reduce the Kronecker products of representations of SR groups”. In: *The Collected Works of Eugene Paul Wigner*. Springer, 1993, pp. 608–654.