Langmuir

Article

pubs.acs.org/Langmuir

Extending Particle Tracking Capability with Delaunay Triangulation

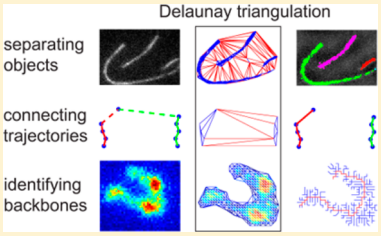
Kejia Chen,[†] Stephen M. Anthony,[#] and Steve Granick*[,†,‡,§,∥]

[†]Departments of Chemical and Biomolecular Engineering, [‡]Materials Science and Engineering, [§]Chemistry, and [∥]Physics, University of Illinois, Urbana, Illinois 61801, United States

[#]Sandia National Laboratory, Albuquerque, New Mexico 87123, United States

Supporting Information

ABSTRACT: Particle tracking, the analysis of individual moving elements in time series of microscopic images, enables burgeoning new applications, but there is need to better resolve conformation and dynamics. Here we describe the advantages of Delaunay triangulation to extend the capabilities of particle tracking in three areas: (1) discriminating irregularly shaped objects, which allows one to track items other than point features; (2) combining time and space to better connect missing frames in trajectories; and (3) identifying shape backbone. To demonstrate the method, specific examples are given, involving analyzing the time-dependent molecular conformations of actin filaments and λ-DNA. The main limitation of this method, shared by all other clustering techniques, is the difficulty to separate objects when they are very close. This can be mitigated by inspecting locally to remove edges that are longer than their neighbors and also edges that link two objects, using methods described here, so that the combination of Delaunay triangulation with edge removal can be robustly applied to processing large data sets. As common software packages, both commercial and open source, can construct Delaunay triangulation on command, the methods described in this paper are both computationally efficient and easy to implement.



INTRODUCTION

We are in the midst of an imaging revolution. Going beyond the older use of images to provide primarily qualitative information (important as that is), rapid developments are underway in *quantitative* imaging—the standardized quantification of large statistical data sets, often from time series of images, and usually enabled by rapid computer processing. While particle tracking of moving objects in time series of images has become routine,[1,2] a great limitation is that the standard particle tracking analyzes diffraction-limited spots, for example in single-molecule studies.[3,4] This paper concerns three significant limitations of the standard methods: first, difficulty to separate reliably objects that are in close proximity; second, difficulty to identify the same moving element in time series in which there are some missing frames; third, difficulty to describe irregular, asymmetric shapes. This paper explains how to extend these capabilities using Delaunay triangulation in ways that are computationally efficient and easy to implement with common software packages.

The technical problem is the following. While to identify different objects in an image often seems obvious to one's eyes, identifying them automatically using computers for quantitative analysis is often hampered by noise in the image, irregularity of shapes, and proximity of objects. The first step to separate objects is to identify the pixels that comprise each object, but in general it is not possible to identify all points correctly, not when there is low signal-to-noise and intensity variation within an object. This precludes simple grouping by connectivity. When dealing with objects of irregular shape, to separate them becomes more difficult. Standard clustering techniques such as K-means and hierarchical have limitations: K-means[5] is only suitable for spherical shapes with similar sizes and hierarchical[6] works only when all the objects have similar shape (detailed comparisons are presented below). Difficulties arise when the objects are highly nonspherical and random; it is a serious problem as this class includes important moving objects such as polymer chains,[7,8] rod-like bacteria,[9] highly branched neurons,[10] and irregular domains in phase separation.[11]

After one succeeds in identifying objects quantitatively and pixel-wise, connecting the position of an object from one frame to the next can reveal useful information about the dynamics of motion.[9,12,13] But this is problematical to do because connecting objects between frames to build up trajectories can be difficult when the object disappears from view for a few frames before becoming visible again, which is common due to blinking of fluorophores[14] and diffusion in and out of the focal plane. If, in analysis, one decides to terminate each trajectory when the moving object becomes invisible, this will improperly bias the data toward short trajectories and will interfere with accumulating information about long time dynamics. There are attempts to connect the gaps in time after linking particles in adjacent frames.[15,16] The new method proposed here simultaneously connects the gaps in space and in time. A better method to bridge time gaps may also further improve the performance of super-resolution imaging techniques, such STORM[17] and PALM,[18] by combining the signal that single fluorophores present during different activation cycles.

Received: January 24, 2014
Revised: March 23, 2014

   dx.doi.org/10.1021/la500323r | *Langmuir* XXXX, XXX, XXX−XXX

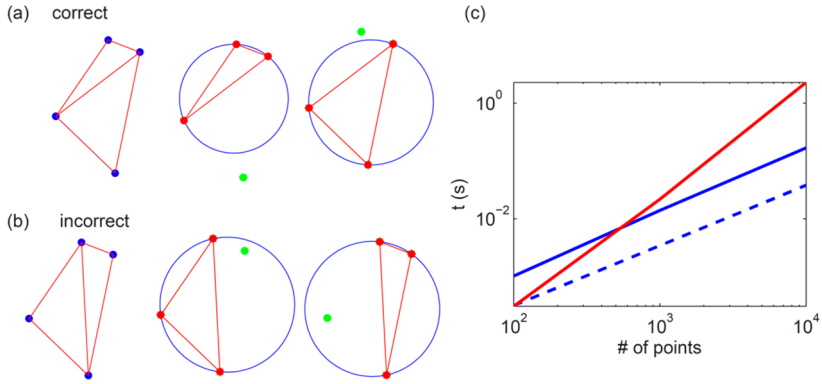


Figure 1. Example of performing Delaunay triangulation on four points in a plane. The triangulation in (a) is correct because the circumcircles (blue) of both triangles (red) exclude the fourth point (green). Case (b) shows an incorrect triangulation where the circumcircles of both triangles contain the fourth point. This tutorial example generalizes to arbitrarily large numbers of points. The text discusses the example of multidimensional space where one axis is position and a second axis is time. (c) Comparison of computation time using Delaunay triangulation (blue) and brute-force strategy (red) of calculating the distance between all points on an image. While the brute-force approach is independent of the dimension, 3D takes longer (blue solid line) than 2D (blue dashed line) for Delaunay triangulation. The brute-force approach is relatively advantageous only for 3D trajectories containing few points (3D: two spatial coordinates plus time), as discussed further in the Supporting Information.

Another application of Delaunay triangulation is in the study of shapes that change with time: for example, the quantification of cell shape[19] and polymer chain configurations.[20] While center-of-mass motion is the standard quantity of which to keep track,[12,13] a good descriptor for shape is less obvious. Principal axes,[7] moments,[21] ellipticity,[22,23] and degree of asymmetry[24] are some of the metrics that have been used to describe shapes. While each captures some important aspects of the shape, such single-number quantifications cannot describe internal degrees of freedom, such as twists and turns within the shape. But it is not helpful to go to the other extreme and retain for analysis all the points in an object; if the information retained is too large, changes of the same shape cannot be discriminated either. The backbone of objects strikes a fine balance between keeping too few and too many details.

Delaunay triangulation, developed in mathematics and computer science of graph theory, contains tools useful for these problems. As we show below and illustrate in detail, it serves to identify objects not only in space but also when the time variable is added. In this argument, the concept of spatial proximity extends to connecting trajectories; one considers time as an additional dimension orthogonal to space, defining the distance between two points as a combination of their distances in space and time, thus augmenting the effectiveness of particle tracking through intervening times. The application to identifying backbones of objects follows similarly, as the minimum spanning tree, defined below, of the graph of all distances for a set of points is a subgraph of their Delaunay triangulation.[25] In this paper, we demonstrate how Delaunay triangulation can be applied to these problems and how the performances compare with existing methods.

METHODS

Delaunay Triangulation Explained. Delaunay triangulation divides a space into subregions according to the rule that the circumcircle of any triangle must be empty (Figure 1). Possessing the unique property that all nearest neighbors are connected, it quantifies spatial proximity very well. As spatial proximity is the physical basis for clustering points, this explains why Delaunay triangulation is widely used in urban and geological studies to aggregate feature points.[26] It amounts to dividing space into a triangular mesh subject to the requirement that no point is inside the circumcircle of any triangle of the mesh.

Popular mathematics software packages offer Delaunay triangulation as a simple command, based on various alternative algorithms that can depend on complex graph theory. However, the method is simple to understand intuitively when one considers just a few points. Figure 1 illustrates it for 4 points in a plane. One observes the correct implementation (Figure 1a), an incorrect triangulation (Figure 1b), and the computation time of this method relative to brute-force calculation (Figure 1c). The Discussion section of this paper elaborates on the information presented in Figure 1c.

Separating Objects. The process of separating objects includes four steps: (1) Selecting an intensity threshold and binarizing the image according to this threshold; the threshold is typically selected as n standard deviations above the average intensity of the image. Here, for physical reasons one selects n depending on the signal-to-noise ratio of the image. (2) Performing Delaunay triangulation (Figure 2c) on the points in the binarized image (Figure 2b); this is done using the built-in function in common software, for example "Delaunay" when one uses Matlab. (3) Selecting a threshold for the maximum distance between two neighboring points in the same object and removing edges in the Delaunay triangulation that exceed the threshold (Figure 2e). (4) Grouping the connected points by the remaining edges (Figure 2h).

To achieve successful separation, the choice of appropriate threshold is critical. The performance of the method depends on two competing factors: the separation between objects and the gap within an object. The higher the ratio between the two, the less sensitive the method is to the choice of threshold. Too low a threshold can cause objects to be artificially fragmented (Figure 2d,g) while too high a threshold can cause nearby objects to artificially cluster together (Figure 2f,i). It is often necessary to iteratively test different candidate threshold values to obtain one that works the best. In Figure 2, we purposely illustrate a very challenging example with two objects being very close at the bottom of the image and with part of the largest object in the bottom left corner of the image not being as bright as the rest so that the object becomes fragmented by binarization. This example gives a ratio close to 1, which approaches the limit of the method. Therefore, we see a high

dx.doi.org/10.1021/la500323r | Langmuir XXXX, XXX, XXX−XXX

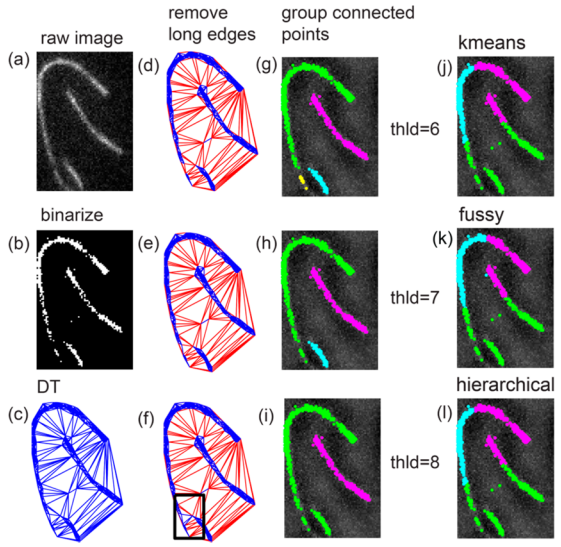


Figure 2. Example of separating actin filaments in a fluorescence image (images taken from ref 7). The raw image (a) is first binarized (b) and a Delaunay triangulation (DT) (c) is constructed on the points that were set to unity in the binarized image. Different edge thresholds were used to remove the long edges (red) and the remaining edges (blue) are shown in (d–f). Finally points that are connected by the remaining edges are grouped together (g–i). Each group is represented by a different color. Proper threshold ("thld") choice (h) is important, as depending upon the threshold chosen, either oversegmentation (g) or undersegmentation (i) segmentation may occur. (j–l) show the results obtained using other clustering functions available in Matlab: *k*-means, fussy *c*-means, and hierarchical.

sensitivity to the threshold value, as expected, but to mitigate this problem, one can exploit the special properties of the edges linking two objects.[26] Zooming in (Figure 3a), one notices that the edges that connect the two objects are substantially longer than neighboring edges. On the basis of this observation, one can remove such edges by comparing edge lengths locally. For each point, if an edge connects to this point exceeds a local threshold, the edge is removed. The local threshold is then determined by[26]

$$\text{threshold}(P_j) = \text{mean}^2_{G_i}(P_j) + \beta\cdot\text{mean\_variation}(G_i) \quad (1)$$

where $\text{mean}^2_{G_i}(P_j)$ is the mean length of the edges formed by the points in the second-order neighbors of point P_j. Second-order neighboring points provide a window to look at local details. Furthermore, β can be tuned to adjust sensitivity to local variations. Since the number of edges formed by second-order neighboring points can be small, the error associated with calculating variations can become significant. Therefore, the mean variation of all the subgraphs $\text{mean\_variation}(G_i)$ is used as the variation indicator.

Another type of local inconsistency can be links between clusters as shown in the example in Figure 3c. Consider the point circled in red in Figure 3d, it has multiple second-order neighbors to the left, but only one to the right. One way to quantitatively capture this is to introduce the idea of local aggregation force,[26] which sums up vectorially the influence of the second-order points on a given point:

$$\vec{F}(P_j, P_k) = \frac{1}{d^2(P_j, P_k)}\vec{e}_{P_j,P_k}, P_k \in N^2_{G_i}(P_j) \quad (2)$$

while the direction of the vector $\vec{e}_{P_j,P_k}$ is determined by the positions of the points. The length of the vector should be inversely proportional to the distance between the two points so that points located far away have lesser influence, hence the normalization by $d^2(P_j,P_k)$. Summing up the individual local aggregation force gives the cohesive local aggregation force on a point:

$$\vec{F}_t(P_j) = \sum \vec{F}(P_j, P_k), P_k \in N^2_{G_i}(P_j) \quad (3)$$

As seen from Figure 3d, the cohesive aggregation force on point located on the border of a cluster point to the interior of that cluster. This causes the point to have a tendency to move into the cluster and implies that the edge pointing to the opposite direction should be removed. Mathematically, this is done by looking at the angles between the cohesive aggregation force vector and the vector between the point and its first-order neighbors. When angles exceed a reasonable threshold, 150° for this example, then the edge is removed.

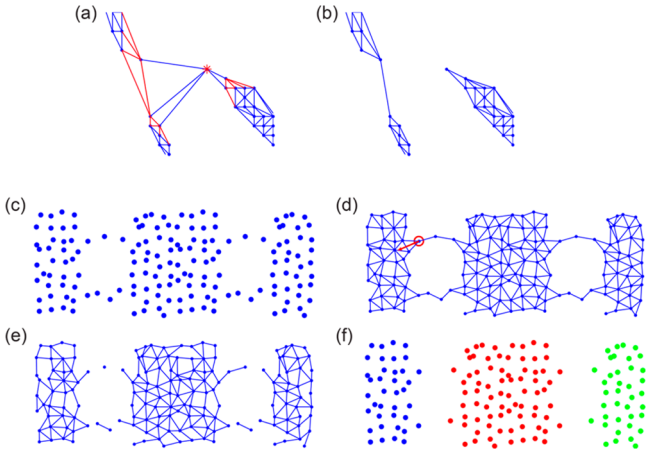


Figure 3. Examples of local removal of inconsistent edges. (a) This shows a zoom-in of the boxed area in Figure 2f. Edges formed by second-order neighbors of the asterisk point are highlighted in red. (b) This shows the result after removal, from (a), of long local edges. (c) This shows an example of local link edges removal. (d) This shows the cohesive aggregation force of the circled point. (e) Edges that point in the opposite direction from the cohesive aggregation forces are removed. (f) This shows the final clustering result.

When To Apply Local Edge Separation. The method is automated to test all points. This is because *a priori* one cannot know which local edges should be removed. The computation time is not expensive as it scales linearly with the number of points. Note that for points in the interior of any cluster the magnitude of the cohesive aggregation force will be small as the forces due to neighbors in opposite directions tend to cancel each other. The direction of the cohesive aggregation force is therefore random, and removing edges can result in arbitrary segmentation of the cluster. In implementing the edge separation method, we avoid this pitfall by applying the angle criterion to only points whose cohesive aggregation force exceeds a set threshold. Figure 3 illustrates a cluster with nonsegmented, well-defined interior.

Connecting Trajectories. The same procedure can connect trajectories, the only difference being that instead of using a threshold to identify feature points as would be seen by one's eyes, a new coordinate system defines trajectory points. The idea is that a trajectory can be viewed as an object whose points lie in a temporal–spatial coordinate system.

dx.doi.org/10.1021/la500323r | *Langmuir* XXXX, XXX, XXX–XXX

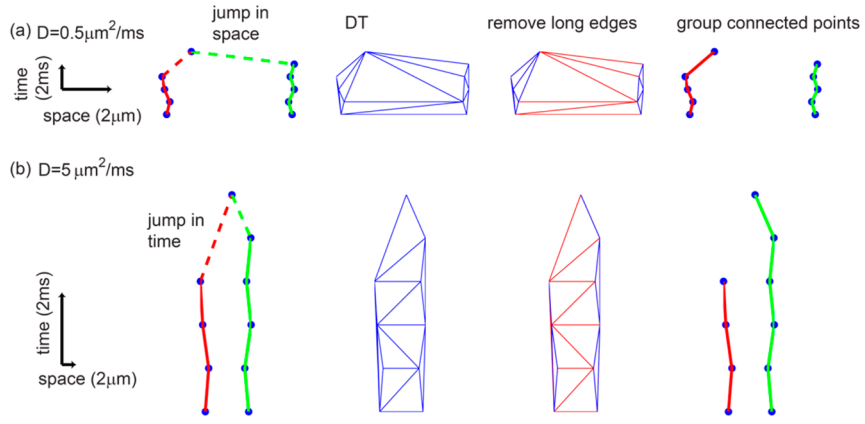


Figure 4. A 1D simulated trajectory to illustrate how Delaunay triangulation applies to trajectories with missing frames. The 1D trajectories (red and green) are plotted in 2D with time as the orthogonal dimension. The problem is to decide, to which trajectory does the last point in time belong? If passage in time is given a relatively small length (a), the jump in space is relatively large and the last point should belong to the red trajectory on physical grounds. Using Delaunay triangulation, after removing the long edges, the last point indeed is connected to the red trajectory. But if the passage in time is given a relatively large length (b), the last point would be assigned to belong to the green trajectory.

Consider the simulated example in Figure 4; two particles diffuse in a micrometer-sized 1D system and are observed every 1 ms. The two trajectories are easily identified by frame-to-frame spatial comparison, except for the problematical last point where there is a time gap. This is an example of the usefulness of Delaunay triangulation, as both of these trajectories can reach the last point through either a jump in space or time. A "slowly" moving object is most likely to move a short distance even after a long time elapses; it is unlikely to move a long distance quickly. Conversely, a "rapidly" moving object is most likely to move a long distance quickly; it is unlikely to remain for a long time near the same location. To determine which is more likely quantitatively, we begin by selecting a conversion factor for time and forming a 2D coordinate system with time as the orthogonal axis. Having established the equivalent scales of the temporal and spatial axes, we then assess, from the relative distance between points in this 2D space, the relative likelihood of the options. But as the distance between orthogonal points depends on the equivalent scales of the axes on which they sit, how to set their equivalent scales requires judicious thinking. The best choices of equivalent scales depend on the physical problem at hand. When the motion is ballistic or has persistent direction, such as active transport of endosomes by molecular motors along microtubules or the electrophoresis of DNA chains where there is a well-defined velocity, the obvious conversion factor between space and time is the velocity. One can multiply the time data with the velocity to convert to spatial units. For example, if the velocity is 100 nm/s, then 10 ms will be converted to 1 nm, so 10 ms in the temporal dimension should have the same length as 1 nm in the spatial dimension.

When the motion is diffusive as the example in Figure 4, the choice of equivalent scale should be guided by the diffusion coefficient. Using the diffusion coefficient is not as straightforward as velocity because distance is less than proportional to linear time. However, in selecting the equivalent scales, it transpires that the exact number does not affect the result significantly, so long as the order of magnitude of equivalent scale selected is reasonable for the physical system one is considering. For example, if the diffusion coefficient is ≈0.5 μm^2/ms, 1 μm in space can be approximated to have the same length as 2 ms in time. With the sample trajectories in Figure 4, equivalent scale in the range of 1 μm to 0.25 ms and 1 μm to 3 ms give the same result. In fact, to select the diffusion coefficient as the base conversion factor will be a better approximation when the time gap is small, than if it is large, and anyway large time gaps are unlikely to occur in trajectories. In the rare cases of long time gaps, the problem can be mitigated during the edge removal step described in the previous section of this paper, by using the correct conversion between time and space in calculating edge lengths:

$$L((x_1, t_1), (x_2, t_2)) = \sqrt{(x_1 - x_2)^2 + D|t_1 - t_2|} \quad (4)$$

In Figure 4, we consider two diffusion coefficients: ≈0.5 and ≈5 μm^2/ms. For the former, considering that on average the particle will move 0.5 μm^2 in 1 ms, an equivalent scale of 2 ms to 1 μm is selected and in this coordinate system, long jumps in space are not likely (Figure 4a). For the latter case, in 1 ms the average distance traveled will be 5 μm^2 on average. Therefore, the equivalent scale of 0.2 ms to 1 μm is selected (Figure 4b), and longer jumps in time are not likely. This done, it is simple to perform Delaunay triangulation in the new coordinate system and remove edges that exceed some reasonable threshold, then to cluster the points into trajectories. In this case, the competing factors are the separation between points that belong to different particles and the gap between points that belong to the same particles. Local edge removal can again be applied to reduce sensitivity to the value of threshold with which the analysis begins. Note that in this example, for all but the last point, the trajectories identified by the Delaunay triangulation using either equivalent scale are the same as those connected by frame-to-frame comparison. Therefore, Delaunay triangulation can connect the trajectories on its own and one does not need to use it together with frame-to-frame comparison. The advantage of Delaunay triangulation is that it provides a rational decision of how to treat the final point of the trajectory. If the edges connecting the final point to both trajectories turn out to be too long, then the final point will be assigned as the start of a new trajectory. Physically, this could happen when a new particle enters the field of view, and hence becomes a useful method to discriminate genuine new particles from older particles that had momentarily disappeared from view.

dx.doi.org/10.1021/la500323r | *Langmuir* XXXX, XXX, XXX−XXX

Identifying Backbones. Whereas after isolating an object, simple spline curve fitting can identify its backbone provided that it does not contain sharp turns, this works poorly for objects with sharp turns, such as the DNA molecule in Figure 5a. Delaunay triangulation is helpful in such cases. Two

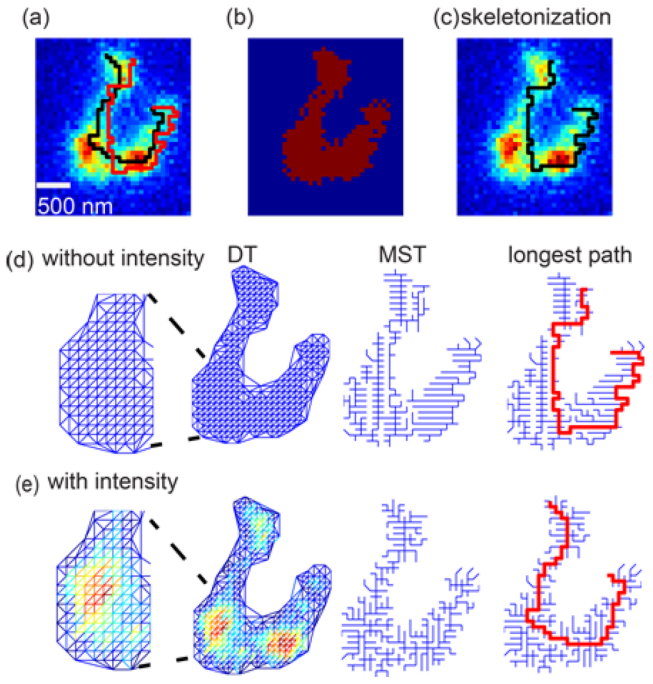


Figure 5. Example of identifying the backbone of fluorescently labeled λ-DNA molecule (images taken from ref 8). The raw image (a) is first binarized (b) and Delaunay triangulation (DT) is constructed on the points that were set to unity in the binarized image. The minimum spanning tree (MST) of the Delaunay triangulation edges were found using either the edge length (d) or the edge length normalized by intensity (e). These alternative ways of defining weight give different MST and different longest paths (red). Colors of the DT edges in (e) represent the intensity of the edge with red being the highest and blue being the lowest. The longest paths of the MSTs, with intensity in (d) and without intensity in (e), are plotted in black and red, respectively, over the original image in (a). (c) Shows the backbone identified with the skeletonization morphological operation.

additional steps are needed after performing Delaunay triangulation on the points of an object: (1) find the minimum spanning tree of the edges from the Delaunay triangulation; (2) locate the longest path in the minimum spanning tree. A minimum spanning tree is a subgraph that connects all the vertices of a graph with the lowest weight. Mathematically, the minimum spanning tree for the complete graph of the distances between the vertices is known to be a subset of the Delaunay triangulation of the same set of vertices.[25] It is part of common software packages. For example, in the MatlabBGL graph library[27] used for the current study, the function called "mst" calculates the minimum spanning tree of a graph.

The backbone of the object should simultaneously minimize both the length of the path connecting the two ends of the object and the extent of the excursions away from the backbone required to reach all other points. The subset of the edges forming the longest path in the minimum spanning tree provides a good approximation of the optimal backbone, tending to balance these two competing optimizations. By definition, the minimum spanning tree between any pair of points will take the shortest path between those points except when adding additional intermediate points results in a reduction in the length of the paths required to connect the remaining points in the graph.

When computing the minimum spanning tree, it is critical to use intensity-weighted edge lengths:

$$L_w(P_j, P_k) = d^2(P_j, P_k) \exp\left(-\frac{I_j + I_k}{\langle I \rangle}\right) \quad (5)$$

where $\langle I \rangle$ is the average intensity of all the points. This is because if simple Euclidean edge lengths were used, as in Figure 5d, in many subregions of the graph the minimum spanning tree would be highly degenerate. Even if the degeneracy could be broken when considering larger graphs, perhaps the entire graph, and even if this produced a unique minimum spanning tree, the result would be artificial for the following reason. The structure of the minimum spanning tree depends most strongly on pixels at the edges of the thresholded region; therefore, the longest path within the minimum spanning tree exhibits significant randomness and is likely to be off center (Figure 5d). In contrast, when considering the intensity warped space, distances between high intensity points are relatively shorter than between dimmer ones. Therefore, the shortest path preferentially includes the brighter pixels (Figure 5e). In addition to producing better and more robust results, weighing algorithms that favor edges between bright pixels are physically reasonable because in experiments based on fluorescence imaging, the local intensity normally depends on the local density of fluorophores.

Note: this result is insensitive to the exact form of the intensity weighting. We tried an alternative approach to define the intensity-weighted edge lengths, eq 6, and obtained the exact same result:

$$L_w(P_j, P_k) = \frac{d^2(P_j, P_k)}{I_j + I_k} \quad (6)$$

DISCUSSION

Comparison with Alternative Methods. *Separating Objects.* Given a set of points corresponding to different objects, while it should be possible in principle to calculate by brute force the pairwise separation distances rather than perform Delaunay triangulation, to do so can be very computationally expensive and easily exceeds the maximum computer memory when the number of points is large.

A quantitative comparison of computational time is presented in Figure 1c. The computational time grows nearly linearly with the number of points for Delaunay triangulation, but for the brute-force strategy it grows quadratically, with more details given in the Supporting Information. The reason for the near linear computational time for Delaunay triangulation is that unlike the brute-force method, this calculation is local. Subgraphs of the triangulation are insensitive to vertices far away from the region. For example, the divide and conquer algorithm recursively draws a line to split the vertices into two sets and computes the Delaunay triangulation of each set. The two sets are then merged along the splitting line, and with fast merge operation, the total running time is $O(n \log n)$.[28] There exist various alternative algorithms to accomplish this. In terms of memory requirements for computation, empirically we observed that in Matlab

dx.doi.org/10.1021/la500323r | Langmuir XXXX, XXX, XXX−XXX

while 100 000 data points require only about 1.6 GB of RAM when using Delaunay triangulation, 30 000 points computed using brute force already require more than 24 GB of RAM, which exceeds what common computers have. To put this into perspective, readers should consider that 30 000 data points are commonly acquired in typical particle tracking experiments. This is evident when one considers that particle tracking typical spans at least 1000 frames with 10's or even 100's of objects in each frame. Brute-force calculation is therefore prohibitively demanding for efficient computation.

To separate objects, an important advantage of Delaunay triangulation is that, unlike other clustering methods, it makes no assumption about the shape, size distribution, or number of objects. For example, *K*-means[5] requires the number of clusters as an input and cannot separate the objects correctly unless this input is correct. It then assigns all points to the cluster centers to which they are closest and iteratively updates the centers until the total distance is minimized. *K*-means effectively divides space into spherical regions; when dealing with nonspherical objects, parts of an object can be unwittingly assigned to the circle of another object. This was clearly seen in Figure 2j. Similarly, if there is a wide distribution of object sizes, parts of a large object might be closer to the center of a nearby small object than to its own center, resulting in mistaken identification. These problems limit the usefulness of *K*-means to spherical objects with similar sizes. In spite of these limitations, *K*-means is used fairly frequently in the current literature because of its ease of implementation. In this paper, we emphasize that Delaunay triangulation can do better. The methods presented here can be implemented easily because most common software packages contain implementations of Delaunay triangulation. For example, to accomplish Delaunay triangulation, we routinely employ Matlab and compute the minimum spanning tree using the MatlabBGL graph library.[27]

One limitation of using Delaunay triangulation to separate objects, which is also a limitation for other clustering techniques, is that when two objects are very close, it becomes difficult to separate them. This can be mitigated by inspecting locally to remove edges that are longer than their neighbors and also edges that link two objects, using the methods that we have described in detail above. By setting a relatively high threshold globally and removing local inconsistencies afterward, the combination of Delaunay triangulation with local edge removal can be robustly applied to processing large data sets.

Connecting Trajectories. Beyond this, Delaunay triangulation is a natural solution to the endemic problem of how to connect missing frames in trajectories. When it comes to connecting trajectories, it is true that frame-to-frame comparison can be extended to include neighboring frames if frequent frames are missing, but this requires arbitrary judgment, such as how many neighboring frames to include, and what to do when a point is closer to the current point in space than another point, yet is further removed in time. Delaunay triangulation condenses all the above considerations into the single question of equivalence scales of the temporal and spatial axes. This makes the problem much easier and more quantitative to approach. For example, for the special case of STORM[17] experiments, the experimental design ensures that the spatial positions of the fluorophores do not change, so the temporal scale can be set to be much smaller than the spatial scale to make jumps in space very unlikely, and this assists in closing gaps in time. This is in the same spirit as the maximum likelihood based statistical method to find the most likely set of fluorophore positions.[29] As in separating objects, when two points are too close, Delaunay triangulation can combine the respective trajectories to which they belong and the same local edge removal steps can be used to separate the trajectories.

To extend this method to higher dimensional data (such as 4D, three spatial dimensions and 1 time dimension) is easily accomplished. In Matlab, one uses the provided *N*-dimensional Delaunay triangulation function "delaunayn". Alternatively, Qhull provides a freely available, open source implementation of the Quickhull algorithm,[30] a commonly used *N*-dimensional Delaunay triangulation algorithm. Within Mathematica, an interface intended to facilitate the use of Qhull is provided by Wolfram Research.

Identifying Backbones. There exist alternative, well-established skeletonization methods that work well when all the pixels of an object can be identified, in other words, when there are no holes and scattered points due to noise or nonuniformity.[31] However, often this is not the case with fluorescence or bright field images because labeling intensity and refractive index can vary from point to point. The current method based on Delaunay triangulation has a higher tolerance of missing points as the general structure of the minimum spanning tree is not easily affected by a few missing points. Another advantage over morphological methods is that the current method is able to take into account the pixel intensities. It is interesting that the morphological operation of skeletonization (Figure 5c) gives the same backbone as the minimum spanning tree without intensity normalization. The comparison highlights the importance of using the intensity information when seeking to identify the backbone of an object.

As an alternative when points in an object are scattered, the most competitive alternative to Delaunay triangulation is probably the B-spline,[32] which fit a smooth curve to the point cloud. Variations of B-spline curve fitting exist,[33] but generally, B-spline-based methods have difficulty when dealing with sharp turns or large thickness variation along the backbone.[34] The minimum spanning tree of Delaunay triangulation is less sensitive to these issues. But a limitation of using backbone as a representation of object shape is that when objects are close to spherical there is not well-defined longest path, so there is no unique backbone. Another limitation is that the method is most accurate within objects, less so toward their ends.

CONCLUSION

The proposed methods can enhance the current capabilities of particle tracking algorithms by allowing irregular shaped objects to be separated and have their shape characterized by the backbone as well as providing an easy way to connect trajectories with frequent missing frames. Applications that can gain the most benefit by adopting these methods include single molecule studies of polymer conformation dynamics,[20] cell morphology studies,[19] and super-resolution imaging techniques that are based on photoactivation cycles.[17]

ASSOCIATED CONTENT

Supporting Information

Matlab code and sample data. This material is available free of charge via the Internet at http://pubs.acs.org.

AUTHOR INFORMATION

Corresponding Author

*E-mail: sgranick@illinois.edu (S.G.).

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, under Award DEFG02-02ER46019.

■ REFERENCES

(1) Saxton, M. J.; Jacobson, K. Single-Particle Tracking: Applications to Membrane Dynamics. *Annu. Rev. Biophys. Biomol. Struct.* **1997**, *26*, 373−399.
(2) Dupont, A.; Lamb, D. C. Nanoscale Three-Dimensional Single Particle Tracking. *Nanoscale* **2011**, *3*, 4532−4541.
(3) Moerner, W. E. New Directions in Single-Molecule Imaging and Analysis. *Proc. Natl. Acad. Sci. U. S. A.* **2007**, *104*, 12596−12602.
(4) Xia, T.; Li, N.; Fang, X. Single-Molecule Fluorescence Imaging in Living Cells. *Annu. Rev. Phys. Chem.* **2013**, *64*, 459−480.
(5) Jain, A. K. Data Clustering: 50 Years beyond K-Means. *Pattern Recognit. Lett.* **2010**, *31*, 651−666.
(6) Estivill-Castro, V.; Lee, I. Multi-Level Clustering and its Visualization for Exploratory Spatial Analysis. *Geoinformatica* **2002**, *6*, 123−152.
(7) Wang, B.; Guan, J.; Anthony, S. M.; Bae, S. C.; Schweizer, K. S.; Granick, S. Confining Potential when a Biopolymer Filament Reptates. *Phys. Rev. Lett.* **2010**, *104*, 118301.
(8) Guan, J.; Wang, B.; Granick, S. Automated Single-Molecule Imaging to Track DNA Shape. *Langmuir* **2011**, *27*, 6149−6154.
(9) Zhao, K.; Tseng, B. S.; Beckerman, B.; Jin, F.; Gibiansky, M. L.; Harrison, J. J.; Luijten, E.; Parsek, M. R.; Wong, G. C. L. Psl Trails Guide Exploration and Microcolony Formation in Pseudomonas Aeruginosa Biofilms. *Nature* **2013**, *497*, 388−391.
(10) Micheva, K. D.; Busse, B.; Weiler, N. C.; O'Rourke, N.; Smith, S. J. Single-Synapse Analysis of a Diverse Synapse Population: Proteomic Imaging Methods and Markers. *Neuron* **2010**, *68*, 639−653.
(11) Jensen, M. H.; Morris, E. J.; Simonsen, A. C. Domain Shapes, Coarsening, and Random Patterns in Ternary Membranes. *Langmuir* **2007**, *23*, 8135−8141.
(12) Yu, C.; Guan, J.; Chen, K.; Bae, S. C.; Granick, S. Single-Molecule Observation of Long Jumps in Polymer Adsorption. *ACS Nano* **2013**, *7*, 9735−9742.
(13) Walder, R.; Kastantin, M.; Schwartz, D. K. High Throughput Single Molecule Tracking for Analysis of Rare Populations and Events. *Analyst* **2012**, *137*, 2987−2996.
(14) Frantsuzov, P.; Kuno, M.; Janko, B.; Marcus, R. A. Universal Emission Intermittency in Quantum Dots, Nanorods and Nanowires. *Nat. Phys.* **2008**, *4*, 519−522.
(15) Saxton, M. J. Single-Particle Tracking: Connecting The Dots. *Nat. Methods* **2008**, *5*, 671−672.
(16) Jaqaman, K.; Loerke, D.; Mettlen, M.; Kuwata, H.; Grinstein, S.; Schmid, S. L.; Danuser, G. Robust Single-Particle Tracking in Live-Cell Time-Lapse Sequences. *Nat. Methods* **2008**, *5*, 695−702.
(17) Rust, M. J.; Bates, M.; Zhuang, X. Sub-Diffraction-Limit Imaging by Stochastic Optical Reconstruction Microscopy (STORM). *Nat. Methods* **2006**, *3*, 793−795.
(18) Betzig, E.; Patterson, G. H.; Sougrat, R.; Lindwasser, O. W.; Olenych, S.; Bonifacino, J. S.; Davidson, M. W.; Lippincott-Schwartz, J.; Hess, H. F. Imaging Intracellular Fluorescent Proteins at Nanometer Resolution. *Science* **2006**, *313*, 1642−1645.
(19) Joshi, S. D.; Kim, H. Y.; Davidson, L. A. Microscopy Tools for Quantifying Developmental Dynamics in Xenopus Embryos. *Methods Mol. Biol.* **2012**, *917*, 477−493.
(20) Hsieh, C.-C.; Balducci, A.; Doyle, P. S. An Experimental Study of Dna Rotational Relaxation Time in Nanoslits. *Macromolecules* **2007**, *40*, 5196−5205.
(21) Shimada, J.; Yamakawa, H. Moments for DNA Topoisomers - The Helical Wormlike Chain. *Biopolymers* **1988**, *27*, 657−673.
(22) Rosin, P. L. Measuring Shape: Ellipticity, Rectangularity, and Triangularity. *Mach. Vision Appl.* **2003**, *14*, 172−184.
(23) Aktas, M. A.; Zunic, J. Measuring Shape Ellipticity. *Computer Analysis of Images and Patterns: 14th International Conference, Caip 2011, Pt I* **2011**, *6854*, 170−177.
(24) Klingenberg, C. P.; Barluenga, M.; Meyer, A. Shape Analysis of Symmetric Structures: Quantifying Variation among Individuals and Asymmetry. *Evolution* **2002**, *56*, 1909−1920.
(25) Aurenhammer, F. Voronoi Diagrams - A Survey of A Fundamental Geometric Data Structure. *Comput. Surv.* **1991**, *23*, 345−405.
(26) Deng, M.; Liu, Q.; Cheng, T.; Shi, Y. An Adaptive Spatial Clustering Algorithm Based on Delaunay Triangulation. *Comput. Environ. Urban Syst.* **2011**, *35*, 320−332.
(27) Gleich, D. *MatlabBGL, version 4.0*, Stanford University: Stanford, CA, 2008.
(28) Cignoni, P.; Montani, C.; Scopigno, R. DeWall: A Fast Divide and Conquer Delaunay Triangulation Algorithm in E-D. *Comput.-Aided Design* **1998**, *30*, 333−341.
(29) Mukamel, E. A.; Babcock, H.; Zhuang, X. Statistical Deconvolution for Superresolution Fluorescence Microscopy. *Biophys. J.* **2012**, *102*, 2391−2400.
(30) Barber, C. B.; Dobkin, D. P.; Huhdanpaa, H. The Quickhull Algorithm for Convex Hulls. *ACM Trans. Math. Software* **1996**, *22*, 469−483.
(31) Lam, L.; Lee, S. W.; Suen, C. Y. Thinning Methodologies - A Comprehensive Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 869−885.
(32) Eilers, P. H. C.; Marx, B. D. Flexible Smoothing wth B-Splines and Penalties. *Stat. Sci.* **1996**, *11*, 89−102.
(33) Lee, E. T. Y. Comments on some B-Spline Algorithms. *Computing* **1986**, *36*, 229−238.
(34) Liu, Y.; Yang, H. P.; Wang, W. P. Society, I. C. Reconstructing B-Spline Curves from Point Clouds - A Tangential Flow Approach Using Least Squares Minimization. *International Conference on Shape Modeling and Applications, Proceedings* **2005**, 4−12.