JOURNAL OF
APPLIED
CRYSTALLOGRAPHY

Volume 54 (2021)

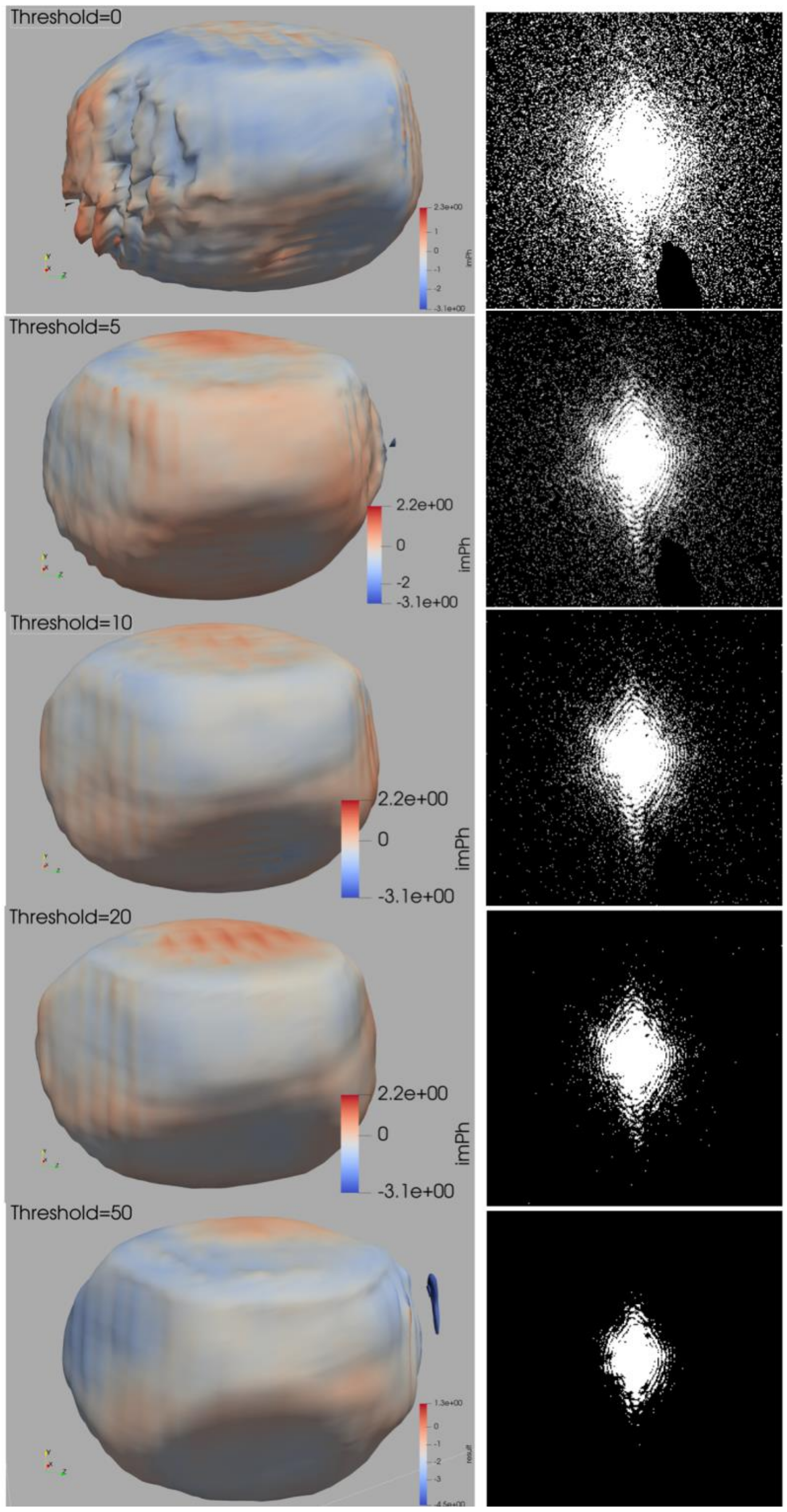
Supporting information for article:

Removal of spurious data in Bragg coherent diffraction imaging: an algorithm for automated data preprocessing

Kenley Pelzer, Nicholas Schwarz and Ross Harder

Supplementary Information

Figure S1. This figure shows diffraction data and reconstructed images using different intensity thresholds (gold nanocrystal data). Reconstructed images are shown on the left. The central slice of the diffraction data is shown on the right. All images were produced with the manually cleaned dataset. The reconstructed images are similar for thresholds ranging from 5 to 50 AIU, while the reconstructed image shows obvious error with a threshold of 0 (no filtering).



Section S1. Discussion of Clustering Methods

Our method can be modified to use any clustering technique for identifying separate diffraction signals. For the rest of this discussion, we will use the word "blobs" (frequently used in discussions of data clustering) to refer to regions of lit-up pixels that human intuition would identify as separate clusters. In our efforts to automate a process that is currently performed using only human intuition, we aim to cluster the data such that one blob is recognized by the algorithm as one cluster. DBSCAN and connected components are both computationally affordable methods that can be applied to this type of blob recognition. Although neither method is guaranteed to achieve complete agreement with human judgment, we believe that both methods are adequate for the algorithm described in this publication (with a human check of final results to identify any significant deviations between human judgment and the automated cleaning). In this section, we discuss these two clustering methods in more detail.

Connected components is simply an approach in which any adjacent pixels are grouped into a single cluster. DBSCAN is somewhat more complicated, and we refer the reader to the original publication for further information (Ester *et al.*, 1996). For the purposes of our discussion, we briefly describe the basics of the DBSCAN algorithm below:

Parameters

minPts: The number of points that must be within a distance ε of a point for it to be classified as a "core point." In counting towards the *minPts* threshold, the point itself is included. (We note that *minPts* is not a minimum cluster size. Because of the treatment of border points, described below, it is possible to have clusters that contain a number of points less than *minPts*.)

ε: This defines the distance between points that is used in definitions of core points and border points.

metric: The metric used for defining the distance between two points.

Algorithm

1. An arbitrary point is selected, and the number of points within distance ε of that point is counted. If the number of points is greater than or equal to *minPts*, the point is classified as a core point p and a new cluster is started. If the number of points is less than *minPts*, the point is classified as noise (although it may be re-classified later as a border point in another cluster).
2. All points q_i within ε of the core point (in the "ε-neighborhood") are assigned to the cluster. If a point q_i is also a core point, the points in its ε-neighborhood are added to the cluster as well. If q_i is not a core point, it is considered a "border point" of the cluster.
3. When this cluster is complete, a new point is selected and the algorithm returns to step 1. This process continues until all data points (in our case, lit-up pixels) have been classified as cluster points or noise points.

For all calculations in this publication, *minPts*=5, $\varepsilon = 1.01\times\max(dx,dy,dz)$, and *metric*=*Euclidean.* (Assigning $\varepsilon = \max(dx,dy,dz)$ would achieve the same effect of including adjacent pixels; the factor of 1.01 is included purely to guard against any issues that could be caused by rounding errors.)

For a small value of ε that defines only adjacent pixels as being within the "ε-neighborhood", DBSCAN and connected components are similar but not exactly equivalent. We discuss the advantages and disadvantages of each approach below.

Advantages of using connected components
We seek a method that is easy for synchrotron users to apply and understand. While connected components is straightforward and opaque for users, DBSCAN is a more complex algorithm. For our purposes, there are two ways in which DBSCAN produces results that may be non-intuitive for users, as described below.

1. The most frequently cited limitation of DBSCAN is that it is a non-deterministic method (the same data can have multiple correct clustering results). This non-determinism is a result of the fact that in some cases, border points on the edge of a cluster *A* can be defined by this algorithm as belonging to either cluster *A* or a nearby cluster *B.* Both possibilities fit the definition of clusters according to DBSCAN, and the assignment to cluster *A* or *B* depends simply on the order in which the data points are processed by an algorithm. This is sometimes referred to as a border point being "shared" between two clusters, but in the final result the point is assigned to only one of the two clusters. The iterative nature of our algorithm makes it possible for an edge point to be assigned to cluster *A* in one iteration and cluster *B* in another iteration (we saw this behavior for 8 points in the example of gold259). This non-deterministic nature of border points is not an inherent problem: We seek to match or exceed the accuracy of human intuition, and human intuition would also be somewhat arbitrary in how to treat edge points in two adjacent clusters. Thus, this non-determinism isn't an inherent inaccuracy, but it is a non-intuitive result that may be confusing to a synchrotron user who is unfamiliar with DBSCAN.

2. Another non-intuitive result that can be caused by DBSCAN clustering (and is influenced by its non-determinism) is the fact that points belonging to a cluster in one iteration can be defined as noise points in the next iteration. This occurs as points belonging to asymmetric or small clusters are deleted from one iteration to the next. We found that this behavior rarely occurred when the parameters were set for the algorithm to remove small clusters. However, when we tested the extreme case of the algorithm in which clusters were deleted purely based on asymmetry considerations rather than size (using a normal filtering threshold), our calculations frequently showed this reassignment of clustered points to noise points. Again, this is not an inherent problem, but simply a feature of the how the DBSCAN algorithm works. This non-intuitive behavior is a result of the way DBSCAN defines core points. If a cluster *A* has only one core point *p*, and core point *p* has only *minPts* points q_i in its ε-neighborhood, this core point only barely meets the conditions to qualify as a core point: If any one of the neighboring points q_i is eliminated, *p* ceases to be a core point. Because edge points can be shared with other clusters, it is possible that a point q_i could be assigned to another cluster *B*, in spite of the fact that *p* depends on q_i to retain its status as a core point. If cluster *B* is then eliminated in one iteration due to size or asymmetry thresholds, in the following iteration point *p* does not have enough points in its ε-neighborhood to qualify as a core point. Because a group of points cannot be classified as a cluster without at least one core point, all points in the original cluster *A* are classified as noise points in the next iteration. Again, DBSCAN gives results that are not inherently inaccurate, but may be non-intuitive for users. The goal of providing users with a straightforward, opaque algorithm is a major advantage of using connected components rather than DBSCAN.

Advantages of using DBSCAN

We see two advantages of using DBSCAN, described below.

1. Unlike connected components, DBSCAN has the ability to define a cluster that includes some blobs separated by dark areas. This can be caused by fringing of data, as discussed in the main text. Although we showed in Section 3.2 that asymmetry constraints were effective in treating fringing, DBSCAN offers the user the option to use a large ε value that would lead to fringed areas being grouped into the same cluster. The problem with a large ε value is that in cases in which an alien signal is nearby the desired data (but not exactly adjacent), the algorithm would group the alien and desired signals into one cluster. For this reason, we recommend a lower value of ε and an asymmetry-based approach, but we wish to give users freedom to make this decision according to the specifics of their system. In principle, other than fringing we would not expect BCDI data to have dark-pixel gaps in the desired data, since the good data in theory should cause a gap-free diffraction signal that decreases monotonically in intensity as it moves away from the Bragg peak. However, again we aim for users to have the freedom to judge for themselves how to interpret and process the diffraction data: DBSCAN offers this flexibility.
2. Another advantage to using a more sophisticated clustering approach as opposed to connected components is that connected components will view blobs that are connected by a "string" of single points as one cluster. DBSCAN, given *minPts>3,* would consider two signals that are connected by a string of single lit-up pixels to be two separate clusters (this a result of the fact that the points in the string cannot be considered as core points if *minPts>3,* and DBSCAN does not allow a string of non-core points to be considered as part of a cluster). We have no reason to expect BCDI data to contain blobs that are connected by long strings, but there may be cases in which blobs that are very near one another show a few lit-up pixels that would function as a "string". In this case, DBSCAN would provide improved accuracy in distinguishing blobs that could be recognized as separate signals by human intuition.

Section S2. Conversion of Pixel Indices to *x,y,z* Coordinates for DBSCAN Algorithm

The concept of the *x,y,z* coordinates processed by DBSCAN differed based on the detector used to gather the data. For data gathered using the photon counting Timepix detector, the *x,y,z* coordinates are the actual positions of the points in reciprocal space. Prior to processing each dataset, we calculated the reciprocal space distances *dx* and *dy* corresponding to the distance between pixels in the detector. The variable *dz* gives the reciprocal space distance between two slices of data. We then used these values of *dx/dy/dz* to calculate the reciprocal space *x,y,z* coordinates based on the position of the pixel. For the Timepix detector, this conversion from pixel positions to reciprocal space *x,y,z* coordinates was fairly trivial, because *dx, dy,* and *dz* have similar values. (For the three examples studied here, the *dx/dy/dz* values vary from one another by less than 10%.) We then assigned the DBSCAN parameter $\varepsilon = 1.01\times\max(dx,dy,dz)$, so that adjacent pixels in the *x,y,z* direction are within the ε-neighborhood.

For the direct detection EEV CCD camera that is used for the silver-gold nanocrystal, using reciprocal space coordinates is less convenient because *dz* is ~4.5 times larger than *dx* and *dy*. This would make it more difficult to define an appropriate value of ε based on reciprocal space coordinates: A value of ε large enough to reach adjacent pixels in the *z* direction would then allow non-adjacent pixels to be clustered in the *x,y* directions. For our example here, we wanted to include only connected points within the same cluster. In this case, it is easiest to simply define *x,y,z* based on the pixel's position on the detector (the *x,y* coordinates) and the slice of data that it belongs to (the *z* coordinate), ignoring actual distances in reciprocal space. The algorithm described above for the Timepix detector is easily modified to take this approach: we simply set *dx=dy=dz=1* when we use pixel positions to create the *x,y,z* coordinates. The ε parameter is still set at $\varepsilon = 1.01\times\max(dx,dy,dz)$ to include adjacent pixels in the ε-neighborhood.

Both approaches to calculating *x,y,z* coordinates are valid for the purposes of our algorithm, and the approach used for the direct detection camera could be just as easily applied with the photon counting detector. We note that for visualization of the intensity volumes in Figures 2, 6, and 7, the data is converted to reciprocal space coordinates regardless of which detector was used. The reciprocal space distances used for this conversion are presented in Section S3 of the Supplementary Information.

Section S3. Conversion to Reciprocal Space Coordinates

The indices of a pixel in the raw data are converted to reciprocal space coordinates for all intensity volume visualizations. Reciprocal space coordinates were also used as input to DBSCAN for data gathered with the Timepix detector, as described in Section S2. To convert to reciprocal space coordinates, it was necessary to derive the appropriate *dx/dy/dz* values corresponding to distances between pixels on the detector (*dx,dy*) and pixels on different slices (*dz*). *dx/dy/dz* are specific to the experiment and are listed below for each system. The Python code that derives these values is available on Github. This code also performs an interpolation of the reciprocal space data into a tif file that can be used for intensity volume visualization in ImageJ.

<u>Gold472</u>
dx = 0.0004943439970174102
dy = 0.0005044893312333975
dz = 0.0004695539481888079

<u>Gold 226</u>
dx = 0.0004975995980932918
dy = 0.0005044893312333975
dz = 0.00047862593801547713

<u>Gold259</u>
dx = 0.0004973213222163641
dy = 0.0005044893312333975
dz = 0.000478039580270565

<u>Ag-Au nanocrystal</u>
dx = 0.0002017438546477028
dy = 0.00020638199914093533
dz = 0.0008998086304684365

Section S4. Application to Gold Nanocrystals

In addition to the gold nanocrystal designated as gold472 presented in the main text, our algorithm was applied to two other gold nanocrystals, designated as gold226 and gold259. Data was gathered using the photon counting Timepix detector described in Section 2.1. The raw data gathered from these nanocrystals could not be successfully phased to produce electron density reconstructions due to overexposure of the pixels in the central bright peak. However, this data is still useful in demonstrating the applicability of our method to a range of datasets that contain large amounts of spurious data. The images below correspond to Figure 2 and Figure 4 of the main text.

Figure S2. Equivalent to Figure 2 of the main text, we show a volume representation of a gold nanocrystal labeled gold226, framed by axes representing the three dimensions of reciprocal space. Coloration is truncated at 100 AIU, as shown by the color bar on the far right. (a) shows the volume rendering of the uncleaned data. (b) shows the same volume after cleaning using C_{max}=2 and S_{max}=0.05.

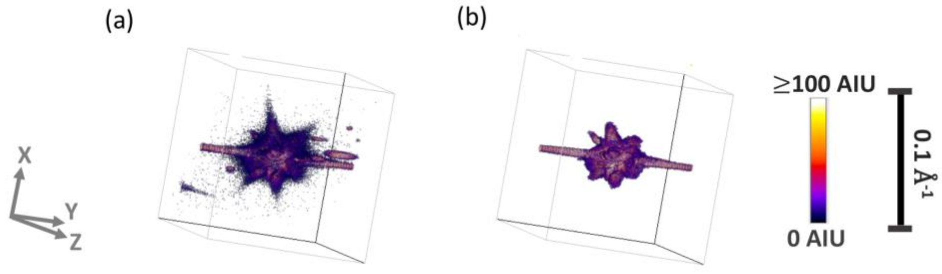


Figure S3. Equivalent to Figure 4 of the main text, we show a two-dimensional slice showing the asymmetry A_i of pixels in a gold nanocrystal labeled gold226. (a) shows the system prior to cleaning. (b) shows the same slice after cleaning using C_{max}=2 and S_{max}=0.05.

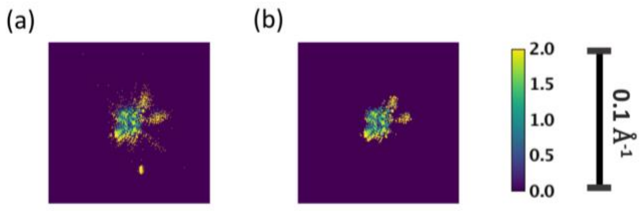


Figure S4. Equivalent to Figure 2 of the main text, we show a volume representation of a gold nanocrystal labeled gold259, framed by axes representing the three dimensions of reciprocal space. Coloration is truncated at 100 AIU, as shown by the color bar on the far right. (a) shows the volume rendering of the uncleaned data. (b) shows the same volume after cleaning using C_{max}=2 and S_{max}=0.05.

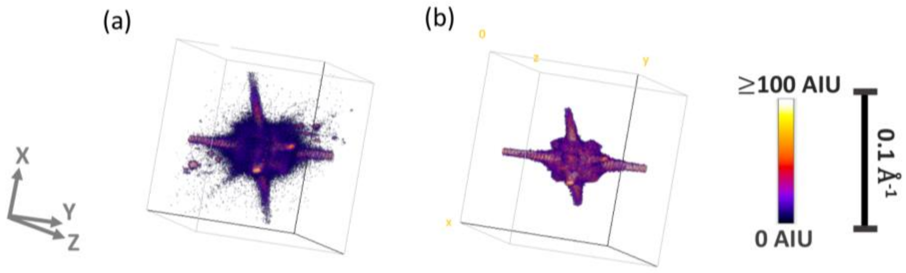


Figure S5. Equivalent to Figure 4 of the main text, we show a two-dimensional slice showing the asymmetry A_i of pixels in a gold nanocrystal labeled gold259. (a) shows the system prior to cleaning. (b) shows the same slice after cleaning using C_{max}=2 and S_{max}=0.05.

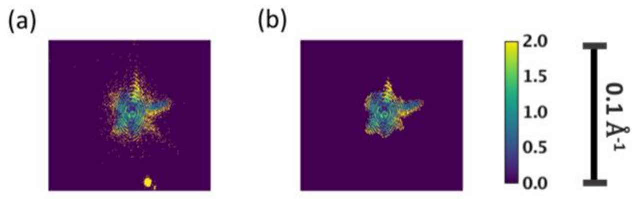
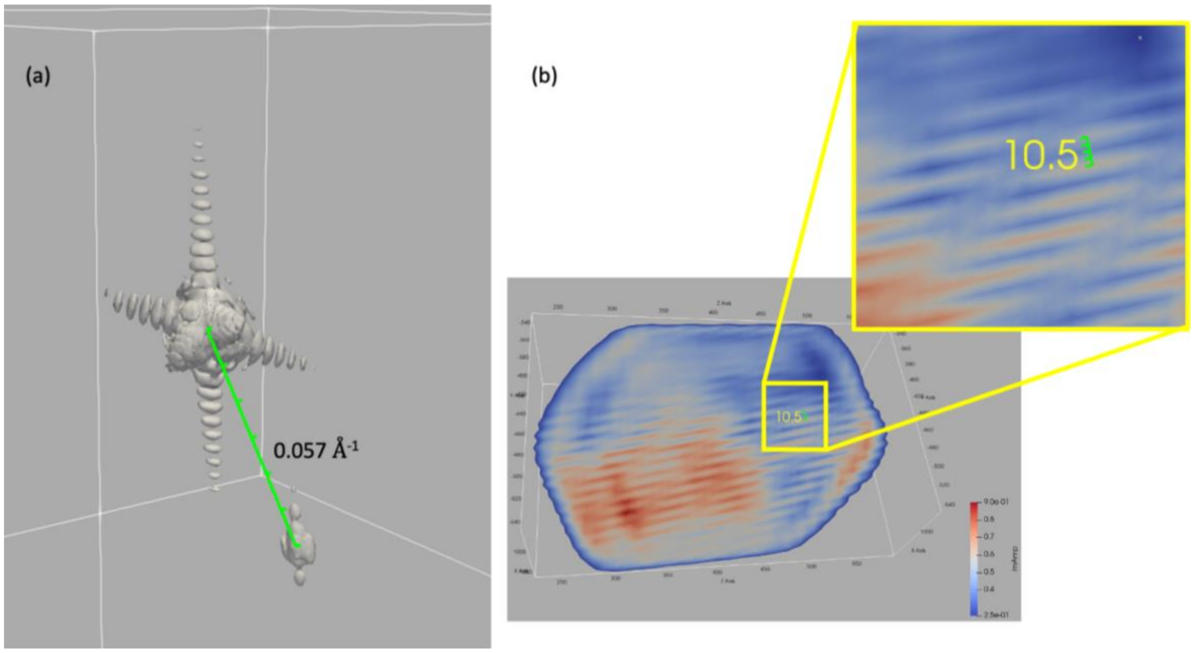


Figure S6. Here we show a closer examination of dataset gold472. (a) The distance between the primary coherent diffraction Bragg peak and the center of the brightest alien signal is about 0.057 inverse Angstrom. This corresponds to a real space distance of 10.9 nm. (b) Upon phase retrieval, the measured spacing between the ripple artifacts seen in the direct space amplitude image is 10.5 nm.



Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. 1996. "A density-based algorithm for discovering clusters in large spatial databases with noise." In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, edited by Evangelos Simoudis, Jiawei Han and Usama Fayyad, 226-231. AAAI Press.