# Additional file 1

**Algorithm 1: Intensity Based Clustering (IBC)**

Let be a set of images of the same modality containing the same anatomical structure of various subjects (i.e. particles in the cryo-EM images) and let be the set of all pixels in an image. Each pixel () will be grouped into several consistent “clusters” where the number of clusters is determined according to a specific intensity interval size. To determine the initial number of clusters in the ICB algorithm, for example , if the adjusted intensity range is [0.2, 0.8] as shown in Figure 3(b) and the interval size is 0.15, there are 4 initial cluster levels: the intensity level [0.2-0.35] will be assigned to Cluster 1, [0.35-0.5] to Cluster 2, [0.5-0.65] to Cluster 3, and [0.65-0.8] to Cluster 4. Here, is a real intensity value in a specific range, . Let be the set of the average intensity values of clusters. The centers of clusters are initialized as evenly distributed intervals in the intensity range at the equal step size according to Equation (1):

(1)

where the is the difference between the maximum and the minimum intensity level in each image and 0.15 is the selected interval step size value.

Let be the index of the cluster whose center (is closest to . We denote the cluster assignment of all pixels after the desired centroid and cluster label ( is predicted as. Each pixel is assigned to a specific cluster whose cluster center ( is closest to the according to the absolute intensity difference value using Equation (2):

(2)

The final clusters centroids are updated iteratively according to the average intensity of clusters according to Equation (3):

(3)

The procedure of the clustering algorithm for cryo-EM image clustering is shown below.

|  |
| --- |
| **Algorithm 1** Intensity Based Clustering (IBC) |
| 1. **input:** pre-processed cryo-EM image |
| 1. **return:** clustered image |
| 1. Initialize the minimum and maximum intensity mapping threshold values. |
| 1. Identify the cluster number to be 4 by mapping the input image intensity level to 4 levels based on the intensity adjusted min and max threshold values. |
| 1. Convert the 2-D image into 1-D which has the intensity values of all the pixels |
| 1. ← height  width where is the total number of pixels in the |
| 1. ← /\*maximum values of intensity in the image\*/ |
| 1. ← /\*minimum values of intensity in the image\*/ |
| 1. **for** = 1 to **do** |
| 1. /\*Initialize the cluster centroids by computing the interval step size using Equation (8) \*/ |
| 1. **end for** |
| 1. **repeat** |
| 1. **for** =1 toevery intensity pixel **do** |
| 1. **for** = 1 to each cluster do |
| 1. Assign the cluster whose center ( is closest to according to the absolute intensity difference using Equation (9). |
| 1. **end for** |
| 1. **end for** |
| 1. **for** = 1 to each cluster **do** |
| 1. Recompute the centroid of each cluster according to the average intensity of each cluster using Equation (3). |
| 1. **end for** |
| 1. **until** convergence, i.e. there is no change in cluster centres. |

**Algorithm 2: Image Cleaning and Non-Circular Object Removal**

An intermediate binary image is generated first to enlarge each object in the original binary clustered image by applying the morphological image operation on the original clustered image (binary) using image opening according to Equation (4)

(4)

Where is the original clustered image, is the structural sub image using circular structure , and denote erosion and dilation respectively. Then, the small objects and non-circular ones are removed form the intermediate images based on the object roundness class which is determine by computing the area and perimeters using the connected component pixel index list and the circularity based on the Equation (5):

(5)

The image cleaning and small object removal algorithm is shown below.

|  |
| --- |
| **Algorithm 2** Image Cleaning and Non-Circular Object Removal |
| 1. **input:** /\*cluster cryo-EM image \*/ |
| 1. **return:** /\*cleaned cluster image \*/ |
| 1. /\* Generate an intermediate clustered image by enlarge the small object using the image opening according to Equation (4) \*/. |
| 1. /\* Label each object in the cluster image using MATLAB function () \*/. |
| 1. **for** i=1 to **do** /\* for each object in the intermediate clustered image\*/ |
| 1. /\* determine the connected components (objects) in the image, including a list of indexing pixel locations for each one using MATLAB function (regionprops) \*/. |
| 1. /\*remove the object that has not a fully connected edge using MATLAB function (bwareaopen)\*/. |
| 1. **end for** |
| 1. /\*extract the number of object (particles)\*/ |
| 1. /\*label each object (particle)\*/ |
| 1. **for** =1 to **do** /\* for each object (particles) \*/ |
| 1. Do size filtering and roundness filtering |
| 1. /\* Determine the region area of each connected component (object) using MATLAB function (region props('Area')) \*/ |
| 1. /\* Determine the region perimeters of each connected component (object) using MATLAB function (region props (Perimeter)) \*/ |
| 1. /\* Determine the region circularitiesof each connected component (object) using Equation (5). |
| 1. /\*determine the average objects "roundness" circularities value. \*/ |
| 1. /\* Keep objects that less than or equal to the average object’s "roundness" circularities value using MATLAB function (bwareaopen) \*/. |
| 1. Get actual index numbers instead of a logical vector |
| 1. produce new binary image with only the small, round objects in it |
| 1. /\*remove the object that has not a fully connected edge\*/ |
| 1. **end for** |
| 1. Construct the output image containing only the object circular “roundness” object classes in each image. |

**Algorithm 3: Circular Hough Transformation (CHT)**

The CHT first generates a binary image based on each object’s edges. It then calculates the center and radius of each detected circular object. Each circular object in the binary image is defined by three parameters: the coordinates of the center and the radius as Equations (6) and (7) show [29]:

(6)

(7)

where ranges from 0 to , and is the radius.

In this case, CHT is looking for each circular object of a particular radius based on every boundary point in the clustered image (binary image) using the original coordinated system (xy) space as shown in Equations (6) and (7). Then, every point in the (xy) space is equivalent to a circle in the (ab) space by rearranging Equations (6) and (7) to form the following Equations (8) and (9).

(8)

(9)

Next, all , indexed by theta, are retrieved from Hough space. For each of these R(\theta), a vote is placed in Hough space at . Finally, the cells that receive the greatest number of votes are selected as the centers of the circular objects. In most applications of CHT it is common to use Canny edge detection [29] for the construction of the binary map. In this application Canny edge detection fails to identify sufficient points for CHT to detect the circular object center (top-view particles) as is shown in Figure 10(b) and (e). To overcome this issue, the canny edge detection step is replaced by our IBC algorithm~~.~~ The pixels which makeup the outline of each circular object is extracted to form a boundary pixel list. This is done by removing interior pixels and then treating non-zero pixels as belonging to the object and considering zero valued pixels as the background. In our method a vector , which contains two elements, is extracted by specifying the row and column coordinates of each point on the object boundary by tracing the 4-connected neighbors (setting them to 1). Each direction of the object boundary is traced to specify the direction of the object boundary. The result is a Q-by 2 matrix called the factor, where Q is the number of boundary pixels for the region.

The modified Circular Hough Transform algorithm (CHT) is shown below.

|  |
| --- |
| **Algorithm 3** Circular Hough Transformation (CHT) |
| 1. **input:** /\*cleaned cluster image \*/ |
| 1. **return:** number of circular object (particles) in the cryo-EM |
| 1. center of each circular particle, |
| 1. radius of each one. |
| 1. /\* Determine and extract the boundary pixels list by specifying the row and column coordinates of each point on the object boundary \*/ |
| 1. Construct the output binary image containing only the object circular boundary for each object. |
| 1. /\* *Hough Transform Begin*\*/ |
| 1. **for** 1 to each edge point **do** |
| 1. Draw a circle with centre in the edge point with where is the image pixels with position and , is the circular radius. |
| 1. Increment all coordinates that the perimeter of the circle passes through in the accumulator. |
| 1. Find one or several maxima in the accumulator |
| 1. Map the found parameters corresponding to the maxima back to the original image, where , and is the centre of the maxima. |
| 1. **end for** |
| 1. /\* *Hough Transform End*\*/ |

**Algorithm 4: Circular and Non-Square Object (Particles) Removal**

The main idea of the circular and non-square object removal is illustrated by two steps. First, each object (particle) in the cryo-EM (clustered image) is smoothed using a gaussian filter based on Equation (10) [29]:

(10)

where is the output smoothed image, is the original input image (clustered), and is the gaussian kernel (mask) which is constructed based on using Equation (11) [29]:

(11)

where is the sigma (which represents the signal width), , and is the image dimension. Next, a parallel of the image cleaning and non-circular object removal algorithm is constructed called the circular and non-square object (particles) removal. In this case, first we remove any other objects such as ice artifacts in the binary image (clustered) by removing any object that does not have a square shape. The square object shapes are determined by extracting the square object’s edges using the MATLAB function (*bwareaopen*). Then, instead of keeping all circular objects found in each region by Algorithm 2, objects with circularity value below the computed average are removed. Finally, we implement the reserve objective function of the Algorithm 2 to eliminate any other object that has region’s area size larger that the computed threshold value (average region size). This keeps the squares and removes the circular objects from the final mask (clustered image) of the cryo-EM. The circular and non-square object (particles) removal algorithm is shown below.

|  |
| --- |
| **Algorithm 4** Circular and Non-Square Object (Particles) Removal |
| 1. **input**: /\*cleaned cluster image \*/ |
| 1. **return**: /\*cleaned cluster image with square shapes only\*/ |
| 1. /\* Label each connected components (objects) in the cleaned image, including a list of pixel locations for each one using MATLAB function (bwlabel)\*/ |
| 1. Generate a gaussian kernel (mask) using Equation (11). |
| 1. **for** each object in the cleaned clustered image **do** |
| 1. Smooth each object shape using gaussian filter according Equation (10) with specific kernel size=5x5. |
| 1. **end for** |
| 1. **for** =1 to **do** /\* **for** each object in the cleaned clustered image \*/ |
| 1. Remove the object that has not a fully connected edge using MATLAB function (bwareaopen) |
| 1. /\* Determine the region area of each connected component (object) using MATLAB function (regionprops('Area')) \*/ |
| 1. /\* Determine the region perimeters of each connected component (object) using MATLAB function (regionprops (‘Perimeter’)) \*/ |
| 1. /\* Determine the region circularitiesof each connected component (object) using Equation (5) \*/ |
| 1. /\* Remove all the connected components that are bigger or equal to the average pixels \*/ |
| 1. **end for** |
| 1. Construct an intermediate image containing only the non-circular object “roundness” in each image. |
| 1. **for** =1 to **do** /\* **for** each object in the intermediate clustered image \*/ |
| 1. Determine the region area of each connected component (object) using MATLAB function (*regionprops('Area')*) |
| 1. /\* find the max area for all objects\*/ |
| 1. **end for** |
| 1. **for** each object in the intermediate clustered image **do** |
| 1. Determine the region area of each connected component (object) using MATLAB function (*regionprops('Area')*) |
| 1. Determine the region perimeters of each connected component (object) using MATLAB function (*regionprops(‘Perimeter’))* |
| 1. Determine the region circularitiesof each connected component (object) using Equation (5). |
| 1. **If** < **then** |
| 1. /\* Keep objects that less than the maximum allowable area using MATLAB function (bwareaopen)\*/ |
| 1. /\*Extract each object position using MATLAB function (*ismember*)\*/ |
| 1. **end if** |
| 1. **end for** |
| 1. /\* Construct the cleaned output image containing only the squarest object classes occurring in each image \*/ |

**Algorithm 5: Square (Side View) Particle Detection and Picking**

The particle detection and picking algorithm based on first determining the connected components (for each objects) in the cleaned cryo-EM image, including a list of pixel area and locations for each one. Second, since some of the artifact objects have a fully connected component and almost the same size the side view particles, we determine and extract the smallest rectangle region area. Third, we determine the region area of each connected component (object) and keep the objects that are less than the smallest rectangle region area. Finally, bounding boxes are drawn around each discontinuous region (rectangle region area) after determining the connected components (objects) in the image. A list of pixel locations for each object is returned along with the extracted centroid, defined by the horizontal and vertical coordinates (x,y). The square (side-view) particle detection and picking is shown below.

|  |
| --- |
| **Algorithm 5** Square (Side View) Particle Detection and Picking |
| 1. **input**:  /\*cleaned cluster image with square shapes only\*/ |
| 1. **return**: /\*cleaned cluster image with perfect square shapes\*/ |
| 1. /\* Label each connected components (objects) in the cleaned image and extract the total number of objects including a list of pixel locations for each one using MATLAB function (bwlabel)\*/ |
| 1. **for** i=1 to **do** /\* for each object in the cleaned clustered image\*/ |
| 1. /\* Determine the class measure properties of each connected component using MATLAB function (regionprops). \*/ |
| 1. ] /\* Determine the region area of each connected component (object) using MATLAB function (*regionprops('Area')*) l\*/ |
| 1. /\* Determine and extract the smallest rectangle region area \*/. |
| 1. /\*keep objects that are less that the smallest rectangle region area. \*/ |
| 1. **end for** |
| 1. **for** =1 to **do** /\* for each rectangle region area. in the cleaned clustered image \*/ |
| 1. /\*Determine the connected components (objects) in the image, including a list of pixel locations for each one and extract the centroid is the horizontal coordinate (or x-coordinate) and vertical coordinate (or y-coordinate) using MATLAB function (*regionprops('centroid’*)) \*/. |
| 1. Draw all bounding box for each discontinuous region (rectangle region area). |
| 1. **end for** |

**Algorithm 6: Perfect Square Particles Shape Detection and picking**

The perfect square particles shape detection and picking has three main steps. The first step removes the small attached objects by smoothing each particle. Each particle is convolved with a 50x50 averaging filter kernel. The main particles smoothing (averaging) can be defined in Equation (12) [29]:

(12)

where is defined as each particle sub image, is the smoothing (averaging) kernel, and are the particle sub image dimensions. In the second step, after the small attached object are removed for each particle, we use the Feret diameter measures approach [37] to measure and correct the particle object dimensions. New perfect particle shapes are generated based on the maximum and the minimum Feret diameter. The maximum and minimum dimensions (width) of the particle object are used to identify the antipodal vertex pairs from the convex hull vertices set. Based on the new boundary box dimension, new perfect shapes are generated and inserted above each particle in the clean clustered cryo-EM image. The last step eliminates the outliers object (overlapped particles) by defining the average particles size and eliminating the outliers that have particle size larger than the average size. Then, the new boundary box is drawn based on the dimension of new particle object shapes. The perfect square particles shape detection and picking is shown below.

|  |
| --- |
| **Algorithm 6** Perfect Square Particles Shape Detection and picking |
| 1. **input:** /\*cleaned cluster image with perfect square shapes\*/ |
|  |
| 1. **return**: number of perfect square particles |
| 1. /\*cleaned cluster image with perfect square shapes\*/ |
| 1. /\* Label all connected components (objects) in the cleaned image and extract the total number of objects including a list of pixel locations for each one using MATLAB function (*bwlabel*)\*/. |
| 1. **for** each object in the cleaned clustered image **do** |
| 1. /\* remove the small object by smoothing each object shape using Equation (10) with specific kernel size=5x5 \*/. |
| 1. **end for** |
| 1. **for** =1 to each object (particle object) **do /\*** each object in the cleaned clustered image\*/ |
| 1. /\* get each particle where is the total number of objects in the cluster cryo-EM\*/ |
| 1. /\*measure set of properties specified by properties for each 8-connected component in the binary image using MATLAB function (*regionprops)* \*/. |
| 1. Calculate the ferret properties |
| 1. **for** =1 to each pixel in the cleaned binary mask **do** |
| 1. **/\***convert each object pixel to coordinates as an x-y order including a list of pixel locations for each one using MATLAB function (*PixelList*)\*/ |
| 1. /\*extract the pixel hull diamond shapes using MATLAB function (*PixelHull*) \*/ |
| 1. **/\***Determine the maximum Feret diameter and its orientation (maximum diameter) \*/ |
| 1. /\*computes the minimum ferret diameter\*/ |
| 1. /\*extract the minimum bounding box area\*/ |
| 1. **end for** |
| 1. **/\***Convert each object pixel to coordinates as an x-y order including a list of pixel locations for each one using MATLAB function (*PixelList*)\*/ |
| 1. **for**=1 to size(objects) **do** /\* each object (particle) in the cleaned binary mask\*/ |
| 1. Extract the bounding box dimension |
| 1. Extract the 2D convex hull of the points for each object (particle) /\* and are column-vectors which presents a vector of point indices arranged in a counter-clockwise cycle around the hull \*/ |
| 1. **end for** |
| 1. /\*Generate the final image with perfect squares shape generation\*/ |
| 1. /\*find the connected components of all objects (particles) in binary image\*/ |
| 1. /\* measure properties of particle region\*/ |
| 1. determine and eliminate the outliers object (particles) |
| 1. **for** =1 to **do** /\* each object in the binary image\*/ |
| 1. /\* find the average object area in binary image\*/ |
| 1. **end for** |
| 1. **for** =1 to **do** /\*for each object in the binary image\*/ |
| 1. **if** each  <= **then** |
| 1. keep this object by getting the actual index numbers instead of a logical vector. |
| 1. <= /\* keep objects that are less that the average rectangle region area \*/ |
| 1. /\* Extract each object position using MATLAB function (*ismember*)\*/. |
| 1. **end if** |
| 1. **end for** |
| 1. Determine the connected components (objects) in the image, including a list of pixel locations for each one and extract the centroid is the horizontal coordinate (or x-coordinate) and vertical coordinate (or y-coordinate) using MATLAB function (*regionprops('centroid’*)). |
| 1. Draw all bounding boxes for each discontinuous region (rectangle region area). |