```
function[PC,diameters] = reposition(Points,nslices)
%Repositioning algorithm for capillary bridges: Places the Origin [0 0 0]
%in the center of the CB (z being the axis of rotational "symmetry" and z=0
%being where the diameter of the CB is smallest, which is called the neck.
%This proved difficult because rotational symmetry is not perfect, circular
%circumference is not perfect and often times the CB is tilted in respect
%to z. The origin is generally placed too high, the neck is almost always
%at z < -100. However, in the figures made from the "perfect" Surface
%Evolver these mistakes did not occur, the center was always at the neck...
%The result of "reposition" is the Pointcloud (nx3 matrix containing x y z
%coordinates) and diameters, which is a nx4 matrix with center coordinates
%of the circular sections and the radius value in the 4th column

%This step is dependent on the previous steps in Amira, Matlab and during
%LSFM Imaging. The data is turned such that z is the rotational axis of the
%Capillary Bridge.
x0 = Points(:,1);
y0 = Points(:,3);
z0 = -Points(:,2);

%The Pointcloud is translated so that all values are positive.
PC_unsorted = [x0-min(x0),y0-min(y0),z0-min(z0)];
%1:n column added to keep original order of the Points.
PC_unsorted = [PC_unsorted, (1:size(PC_unsorted, 1)).'];

%Sorting after z value
PC_sorted = sortrows(PC_unsorted,3);
%Storing the "sorted" 1:n column. When sorting after this later, the
%original Point order is mainained.
fourth_column = PC_sorted(:,4);
%PC_sorted remains, it contains x y z Points sorted in z value.
PC_sorted = PC_sorted(:,1:3);

%Loop to get the diameters dataset, which slices the Pointcloud in nslices
%of equal height and calculates the crossection of each slice and their
% center and stores it

% Define the top and bottom of the point cloud
PCtop = max(PC_sorted(:,3));
PCbot = min(PC_sorted(:,3));

%Make a matrix (nslices+1)x1 with values of equal distance between the top
%and bottom z-value
Slicematrix = linspace(PCbot,PCtop,nslices+1);

% Initialize diameters matrix
diameters = zeros(nslices, 4);

% Loop over each slice with Iterator h
for h = 1:nslices
    %For each slice, define the max and min z value with the Slicematrix
    lowerheight = Slicematrix(h);
    upperheight = Slicematrix(h+1);
    %Find all Points that lie in this z region and store their indices
    sliceindices = PC_sorted(:,3) >= lowerheight & PC_sorted(:,3) <= ↙
upperheight;

    % Get all Points' coordinates at this z-level
```

```
58      cross_section = PC_sorted(sliceindices, :);
59
60      if isempty(cross_section)
61          % Skip if no points in this slice
62          continue;
63      end
64
65      %With differences, all distances in each dimension between all points in
66      %the crosssection are calculated. So all delta x delta y and delta z
67      %values. Then, in distances, the euclidian sum is calculated to get the
68      %Point distances
69      differences = permute(cross_section, [1 3 2]) - cross_section;
70      distances = sqrt(sum(differences(:, :, 1:2).^2, 3));  % 2D distances
71
72      % Find the maximum distance. This is the "diameter" of this
73      % crosssection.
74      max_distance = max(distances(:));
75
76      % Store the diameter and center coordinates (which are the center of
77      % mass of all crosssection points)
78      diameters(h, 1) = max_distance;
79      diameters(h, 2:4) = mean(cross_section, 1);
80  end
81
82  %In the next step, the crosssection with the smallest diameter is found.
83  %First, a lower and upper index from the diameters matrix are defined. This
84  %is needed because the CB often has the lowest diameter value at the top or
85  %bottom, where the sample ends because of the acupuncture needle being
86  %(obviously) thinner than the neck of the sample. By only looking at the
87  %50% in the middle, these regions are ignored. If there are 50 slices, the
88  %first 13 and last 13 slices are ignored.
89  lowerind = round(0.25*size(diameters,1));
90  upperind = round(0.75*size(diameters,1));
91  %The smallest diameter within the defined region lowerind:upperind is
92  %found.
93  centerdiam = min(diameters(lowerind:upperind,1));
94  %The slice with this diameter is found by identifying the index using
95  %find()
96  centerind0 = find(diameters(lowerind:upperind,1) == centerdiam);
97  %If there are more slices with the same smallest diameter for some reason,
98  %centerind is defined to be a single number.
99  centerind = centerind0(1,1);
100 %The center coordinates of the slice with the smallest diameter are stored.
101 center = diameters(centerind,2:4);
102 disp('center diameter: ');
103 disp(centerdiam);
104
105     %Many changes were made in this function specifically, for transparency
106     %i leave it here:
107         % topdiams = diameters(centerind:size(diameters,1),:);
108         % botdiams = diameters(1:centerind,:);
109         % topind = topdiams(:,1) == max(topdiams(:,1));
110         % botind = botdiams(:,1) == max(botdiams(:,1));
111         %
112         % pneck = usedneckpercentage/100;
113         % topval = pneck*topdiams(topind,4);
114         % botval = pneck*botdiams(botind,4);
115         % neckind = diameters(:,4) >= botval & diameters(:,4) <= topval;
```

```
116         %
117         % neck = diameters(neckind,:);
118 
119         %Finding the center(The coordinates of the center of the smallest ↙
diameter)
120         % center = neck(find(neck(:,1) == min(neck(:,1))),2:4);
121 
122         %Translated diameters dataset
123         % neck = [neck(:,1) (neck(:,2:4)-center)];
124 
125 %The coordinates of the center are subtracted from other coordinates to
126 %reposition them. Now, the center is at [0 0 0]. This is a function output.
127 diameters = [diameters(:,1) (diameters(:,2:4)-center)];
128 
129             %Visualization for testing
130             % figure;
131             % plot(neck(:,4),neck(:,1));
132             %  title('diameters along CBridge height, neck region');
133             %     xlabel('z');
134             %     ylabel('diameter');
135             %      ylim ([0 max(neck(:,1))]);
136             %
137             % figure;
138             %  plot(diameters(:,4),diameters(:,1));
139             %  title('diameters along CBridge height');
140             %     xlabel('z');
141             %     ylabel('diameter');
142             %      ylim ([0 max(diameters(:,1))]);
143 
144 %PC_centered is used to store the Rotated and Translated Point Cloud with
145 %original indexing in 4th column, center at the origin now.
146 PC_centered = [PC_sorted - center,fourth_column];
147 
148 %Resorting by the fourth column for original order which is needed for
149 %curvature data gives PC_shuffled, and the function output PC are only
150 %coordinates.
151 PC_shuffled = sortrows(PC_centered,4);
152 PC = PC_shuffled(:,1:3);
153 
154 
155 
```