Point Cloud Analysis Methods

Segmentation

```
% Process the data with a "Rolling Mask"
function SegmentationProcessing(obj)

    % Board string and offset
    board_state = "";
    shrink_offset = 0.002;

    % Outer Loop: Rows. Inner loop: Columns.
    for row = 0.148:-0.037:-0.111
        for column = -0.148:0.037:0.111

            % Clip data
            [tmpX, tmpY, tmpZ] = obj.ClipPointCloud(obj.X, obj.Y, obj.Z, ...
                [column+0.037-shrink_offset, column+shrink_offset],...
                [row-shrink_offset, row-0.037+shrink_offset],...
                [0.4, 0.3]);

            % Get number of data points
            d_size = size(tmpX);

            % Append to string if a piece is there or not
            if d_size(1) > 500
                board_state = board_state + "1";
            else
                board_state = board_state + "0";
            end
        end
        board_state = board_state + "/";
    end
    % DEBUG: Display final string
    disp(board_state);
end
```

Segmentation with Clustering Addon

```
%---------------Clustering Addon--------------%
if size(tmpX) > 0
    min_distance = 0.005;

    sizeZ = size(tmpZ);
    newZ = zeros(sizeZ(1), 1);

    p = pointCloud([tmpX, tmpY, newZ]);

    [labels, numClusters] = pcsegdist(p, min_distance);

    largest_cluster = [0 0 0];
    for cluster = 1:numClusters
        index  = find(labels == cluster);
```

```
            if size(index) > size(largest_cluster)
                largest_cluster = index;
            end
        end
    end


    %----------------------END---------------------%
```

Clustering: Color

```
    % Colors Clusters
    function ColorClusters(obj, point_cloud, labels, numClusters)
        labelColorIndex = labels+1;

        % Currently Plot on figure 2. To be changed
        figure(2);
        pcshow(point_cloud.Location, labelColorIndex);

        % Change background to white. Black cluster invisible on black
        % background
        set(gca,'color','w');

        % Make colormap using number of clusters
        colormap([hsv(numClusters+1);[0 0 0]]);
    end
```

Clustering: Basic Approach

```
    % Process data using clustering techniques
    function ClusterProcessing(obj)

        % Min Distance between points for clustering
        min_distance = 0.0050; %0.0075;

        % Use clustering method
        [labels, numClusters] = pcsegdist(obj.point_cloud, min_distance);
        disp(numClusters);

        % Color clusters
        obj.ColorClusters(obj.point_cloud, labels, numClusters);
    end
```

Clustering: Flattening Approach

```
    % Process data using clustering techniques
    function ClusterProcessing(obj)
```

```
        % Min Distance between points for clustering
        min_distance = 0.0050; %0.0075;

        sizeZ = size(obj.Z);
        newZ = zeros(sizeZ(1), 1);

        p = pointCloud([obj.X, obj.Y, newZ]);
        figure(3);
        pcshow(p);

        [labels, numClusters] = pcsegdist(p, min_distance);
        disp(numClusters);

        obj.ColorClusters(p, labels, numClusters);
    end
```

Clustering: Iterative Approach - Active

```
    % This function does the heavy work for conditioning
    % 1st step: all clusters, which are in x-y too large are split with
    % a kmeans clustering in two subclusters
    % 2nd step: a corresponding new list of locs,
    % numCluster, and all class variables is established
    % 3rd step: all clusters, which are vertically too close, are fused
    % in the same cluster
    % 4th step: This gets represted
    function ClusterConditioning(obj, numIter)
    % iterate the cluster conditioning
    for clCounter = 1:numIter
        % 1st step: split too large clusters into subclusters
        labels = obj.labels;
        numClusters = obj.numClusters;
        NewNumClusters = 1; % new ordered number of total clusters
        nloc = zeros(1,3);  % new ordered point location list
        nlabels = [0];      % new corresponding point list

        for i = 1:numClusters
            index = find(labels == i);
            % calculate the x-y extent of the cluster
            cl = obj.loc(index,:);
            sizeX = abs(max(cl(:,1))-min(cl(:,1)));
            sizeY = abs(max(cl(:,2))-min(cl(:,2)));

            % criteria for too large clusters for split is applied
            if ( (sizeX > (1.3*0.037)) || (sizeY > (1.3*0.037)))
                % cluster too big, so we use kmeans clustering to
                % subdivide it
                Data2D = [cl(:,1), cl(:,2)];
                Data3D = [cl(:,1), cl(:,2), cl(:,3)];
                [idx,C] = kmeans(Data2D,2);

                % add the first new subcluster to our new data list
```

```
                new_index = find(idx == 1);
                nloc = [nloc; cl(new_index,:)];
                nlabels = [nlabels, ones(1, numel(new_index))*NewNumClusters];
                NewNumClusters = NewNumClusters + 1;
                % and add the second new subcluster
                new_index = find(idx == 2);
                nloc = [nloc; cl(new_index,:)];
                nlabels = [nlabels, ones(1, numel(new_index))*NewNumClusters];
                NewNumClusters = NewNumClusters + 1;
            else
                % the cluster checks out (i.e. sis small), so we simply
                % concatinate it with the output data
                nloc = [nloc; cl];
                nlabels = [nlabels, ones(1,numel(index))*NewNumClusters];
                NewNumClusters = NewNumClusters + 1;
            end

        end
        % clip first elements
        nloc = nloc(2:end,:);
        nlabels = nlabels(2:end);

        %write the data back into the class variables
        obj.loc = nloc;
        obj.labels = nlabels;
        obj.numClusters = NewNumClusters;
        % update the point cloud object
        obj.ptCloud = pointCloud(obj.loc);

        % 2nd step of the iteration: vertical fusion of the clusters
        obj.calcClusterCenter();
        obj.fuseClusters();
    end

    end
```

Clustering: Iterative Approach - Passive

```
    % Process data using clustering techniques
    function board_state = ClusterProcessing(obj, min_distance, iteration_count, ...
        slice_ratio, min_distance_increase)

    % min_distance = 0.003; %0.0075;
    final_data = [obj.X, obj.Y, obj.Z];
    cluster_center_list = [];

    for iteration = 1:iteration_count
        sizeZ = size(final_data(:,3));
        newZ = zeros(sizeZ(1), 1);

        new_pc = pointCloud([final_data(:,1), final_data(:,2), newZ]);
        %new_pc = pointCloud([final_data(:,1), final_data(:,2), final_data(:,3)]);
```

```
        obj.PlotPointCloud(new_pc, iteration);
        [labels, numClusters] = pcsegdist(new_pc, min_distance);
        %                 if numClusters ~= 0 && iteration == 5
        %                     obj.ColorClusters(new_pc, labels, numClusters, 2);
        %                 end

        cluster_center_list = zeros(numClusters, 3);
        new_data = [0, 0, 0];

        for i = 1:numClusters
            % Find data belonging to cluster
            index  = find(labels == i);
            newX = final_data(index, 1);
            newY = final_data(index, 2);
            newZ = final_data(index, 3);

            data = [newX, newY, newZ];

            [~,idx] = sort(data(:,3)); % sort just by the height column
            data = data(idx,:);   % sort the whole matrix using the sort indices

            d_size = size(data);

            % Keep Portion of data
            data = data(1:round(d_size(1)/slice_ratio), :);
            new_data = [new_data; data];

            % Find center of mass of piece and insert into array
            cluster_center = mean(data);
            cluster_center_list(i,:) = cluster_center;

            % % Print Cluster Centers on a plot
            % figure(3)
            % hold on
            % plot3(cluster_center_list(i,1),
            % cluster_center_list(i,2),...
           %  cluster_center_list(i,3), 'O','LineWidth',3);
            % hold off

        end
        new_data(1,:) = [];
        final_data = new_data;
        min_distance = min_distance + min_distance_increase;
    end

    %Map centers to grid and interpolate which squares a piece is on

    board_state = "";

    % Outer Loop: Rows. Inner loop: Columns.
    for column = -0.148:0.037:0.111
        for row = -0.148:0.037:0.111

            [tmpX, tmpY, tmpZ] = obj.ClipPointCloud(cluster_center_list(:,1), cluster_cente
```

```
                cluster_center_list(:,3), [row+0.037, row],...
                [column+0.037, column],...
                [0.4, 0.3]);

            % Get number of data points
            d_size = size(tmpX);

            % Append to string if a piece is there or not
            if d_size(1) > 0
                board_state = board_state + "1";
            else
                board_state = board_state + "0";
            end
        end
        board_state = board_state + "/";
    end
    % DEBUG: Display final string
    %disp(board_state);
    end
```