Developing Loss Function for Geometry Optimisation of Multi-Module Detectors at EuXFEL

S.Mullai Vaneshwar

National Institute of Technology - Calicut, India

September 7, 2022

Abstract

The calibration of multi-module detectors like the ones used at EuXFEL is not a trivial task. Setting up a pipeline that can automatically optimise the detector geometry would help in reducing the preparation time. Our aim was to devise an algorithm which takes given image data and geometry and returns how objectively good the geometry. This can be used as a loss function for automating the geometry optimisation. This is a report of the work done during the Summer Student program at DESY, Hamburg.

Contents

1 Introduction

The European XFEL (EuXFEL) is the worlds most powerful free electron laser, which is capable of producing a large number of very short, intense, highly coherentX-Ray Pulses. A special feature of EuXFEL is its high pulse repetition frequency and intensity compared to other FELs of similar standing. It is made of a superconducting linear particle accelerator which can accelerate electrons to very high energies. These electrons are sent through undulators which are structures which have dipole magents arranged in a periodic manner. This makes the electrons oscillate and radiate energy in the form of short and intense Flashes of X-Rays. These flashes are then split into beams and sent to different experiments.

DESY is home to class leading free electron lasers, and this in turn brings the need for well adapted detectors to ensure hassle-free data acquisition. To do this there are many detectors developed like AGIPD (Adaptive Gain Integrating Pixel Detector), DSSC (DEPMOS Sensor with Signal Compression), PERCIVAL (Pixelated Energy Resolving CMOS Imager Versatile And Large). They are engineered to have very large dynamic range, high frame rates, single-photon sensitivity with low probability of false positives and (multi)-megapixels. AGIPD is a detector which was made for the EuXFEL as it is beam is different when compared to other XFELs as mentioned before. It is made up of arrays of ASICs (Application Specific Integrated Circuit) which is a sensor bonded to a readout chip, they are resilient to plasma effects that arise due to high beam intensities. And to keep up with the 4.5 MHz repetition rate 352 memory cells are used to momentarily record the image which will be readout by other modules. AGIPD used at SPB/SFX and MID are 1 Mega pixel systems which have four quadrants where each of them are 512 x 512 pixels (where each ASIC is 64 x 64 pixels). An image of the detector is show in Figure (1).This multi modular setup allows for different detector geometry based on the need of the experiment. More details of the detector are available on the official documentation (Ref.[1]).

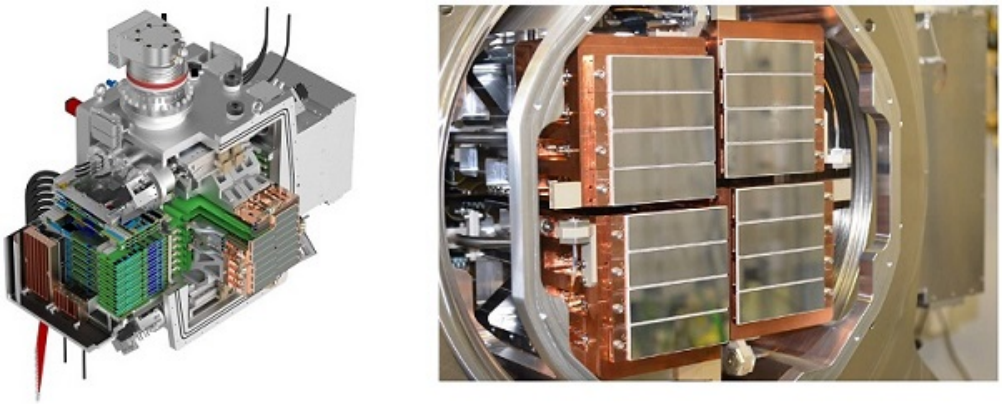

Figure 1: Image of AGIPD, from [1]

There are multiple groups which use the EuXFEL for their experiments, and the multi modular detectors used need to be aligned before every use. Doing this manually is a very laborious and time consuming job. Hence, automating it would help in streamlining the whole process of setting up the experiment. Since every sample will give rise to different results, we use powder diffraction as a standard template. This is attributed to the fact that powder diffraction only gives concentric ring patterns, this can be used in our favour to calibrate the detector. Hence there needs to be a way to quantitatively gauge a given detector geometry so that the process of geometry optimisation can be automated.

2 Powder Diffraction

2.1 Theory

In general crystals give rise to diffraction effects in the X-Ray spectrum. They act as 3-d diffraction gratings in this wavelength region. Now if the crystals were to be made into fine powder, we would now have small crystalliets which have the same orientation within themselves but are randomly oriented with respect to the rest of the crystallites. Since every orientation is equally likely this would give rise to some of them satisfying the Braggs condition and will give rise to cones of diffracted XRays. Hence, when captured on an aligned detector, we expect to get concentric rings. These are called as Scherrer rings and they can be seen in our data (Figure (2)).

2.2 Sample Data

The diffraction pattern we work with was captured by the detector AGIPD which has been discussed in the previous section. Since it is a multi-module detector we have many images that go together rather than one single image. The sample image has been shown in Figure (2), where we can see the similarity between the detector structure (**??**) and the image obtained from it. First we can notice the presence of many rings which are the Bragg Peaks obtained from powder diffraction. There is also some noise present which is visible in yellow color, this is electronic noise which can arise due to the failure of particular detector sub-modules. The gaps between the different modules and the rows are visible as spaces as spaces with zero pixel intensity. Also, one can notice that each ASIC tends to have slightly different intensity levels, this will be dealt with while doing the background subtraction. The borders of each ASIC are also visible hence we use a mask to get rid of any discrepancies. The same mask is also used to mask the edges of the entire image.

3 Image Processing

Our aim is to obtain an well optimised detector geometry. This should ensure that the diffraction pattern (arcs of concentric rings) captured in each module (ASICs) must align with the rest. Now with the image data on hand we need to reduce noise and to process

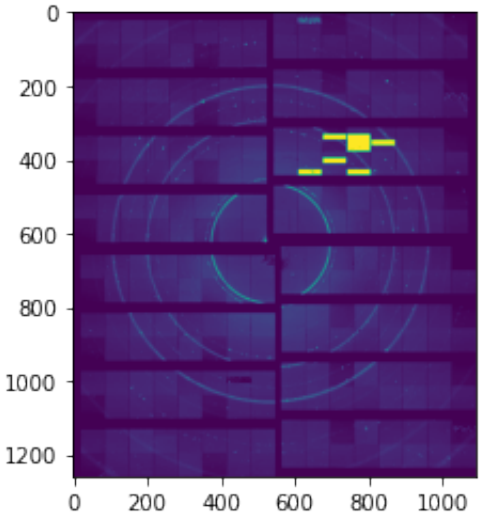

Figure 2: The Diffraction Image from AGIPD

the image to make it more suitable for further analysis. We start by reducing the noise present in the data, next with the help of edge detection we separate the background from the diffraction pattern. We now do a transformation that goes from standard coordinates to polar coordinates, this helps us represent concentric arcs from the image as parallel line segments. Once we have the lines we can do a Hough transform which lets us represent lines as points in parameter space. With this we have simplified the problem to a clustering problem, whose specifics will be discussed in later segments.The following parts go into the details of every image processing step and their implementations.

3.1 Noise Reduction

Multiple schemes of noise reduction were implemented, but we found that a few faired better than the rest. The subtraction was done on two levels one on an ASIC level and another on a slightly bigger level to get rid of the more local effects. A combination of median and mean background subtraction was used. Hence we got 4 different results for the same image, these have been shown in Figure (3). But just looking at this image one cannot say which fairs better, to be able to gauge it we should compare the edge map for different methods (see Figure (5)) and also see which gives rise to better clusters. We have finally used mean subtraction on both a local and an ASIC level, with the local subtraction done over the area of 20 x 20 pixels as this gave the best results. Even though there are predefined methods [1] for doing background subtraction, they weren"t preferred. This is due to the fact that these methods required that the image data was normalised, but this could give rise to problems, in the presence of an outlier it could result in wrongfully scaling the image. Hence these methods were avoided, we also were

[1]We have used methods from `skimage`[2] for most of the image processing

able to see that the normalising also gave rise to artifacts in the edge detection step which are visible in Figure (4) (For this edge map both the background subtractions were done using mean, where the first is the manual and the next uses a predefined method).

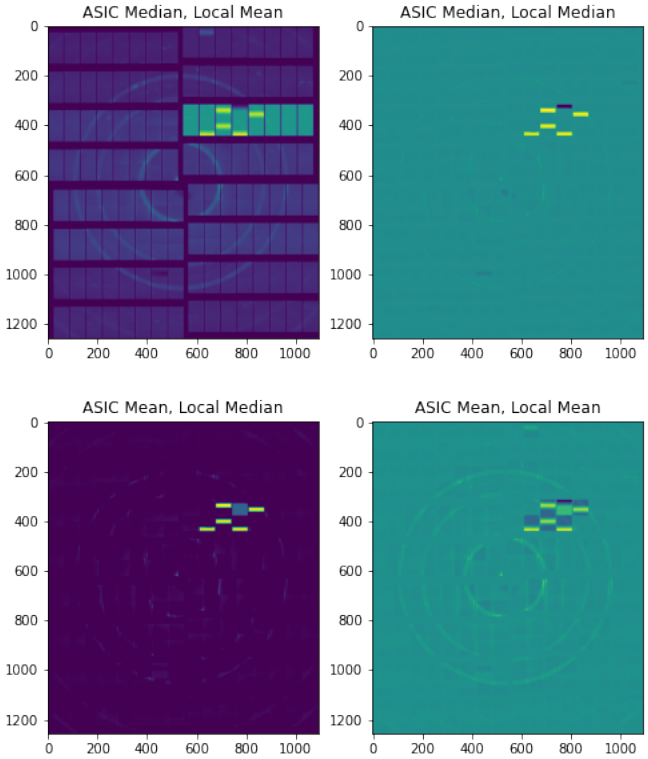

Figure 3: Different Background subtraction schemes

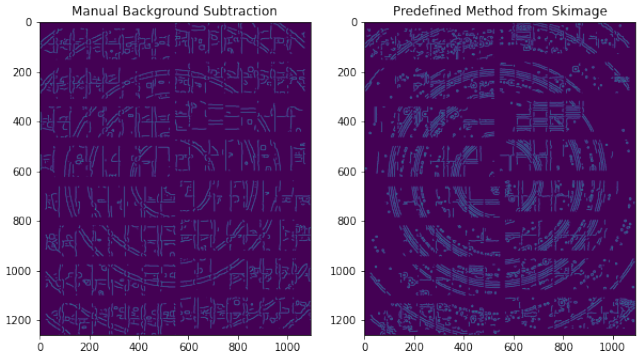

Figure 4: Comparison of background subtraction implementations

3.2 Edge Detection

After reducing the noise we need to find the arcs of the circles, so that they can be transformed in the subsequent steps.
To do this we will be using an edge filter, and our choice is the canny algorithm as it returns 1-pixel thin images. This algorithm first uses Gaussian filter to smoothen the image, then uses the Sobel filters to get the gradients in the image. Then the edges are reduced to 1-pixel width edges by choosing the local maximum from the gradient map. Finally a two level thresholding is done to get the edges. More details on the parameters and the method implemented is available on Ref.[2] and see Ref.[3] for information on the algorithm. By using Canny we can get the edge map, for the image at hand, $\sigma = 8$ for the Gaussian filter worked very well. Even though the high sigma value will smoothen the image a lot and result in the loss of fainter signals it is not of concern. As we only need few circles with good resolution to perform the optimisation. The final edge map from canny is shown in Figure (5). Since the Mean subtraction on both levels offered the best results only data will be considered from this point forward.

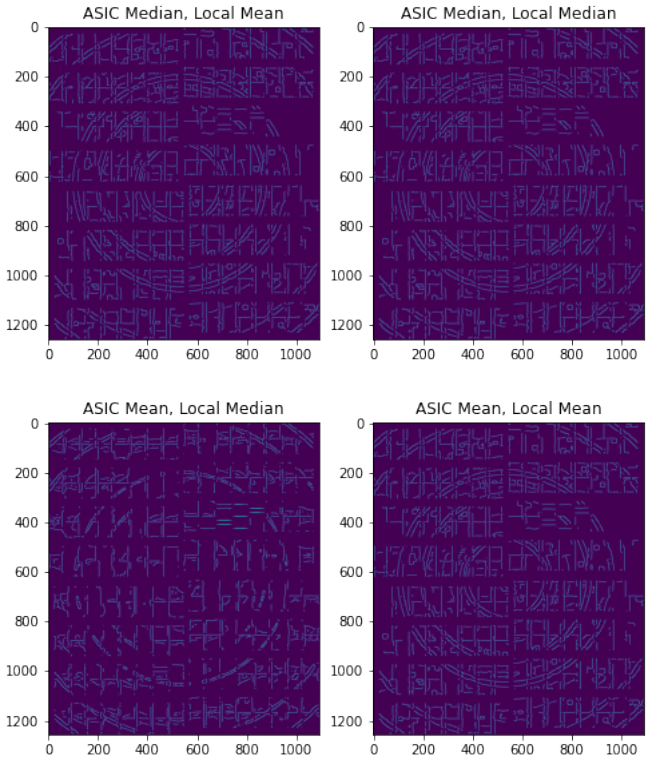

Figure 5: Edge maps obtained after background subtraction

3.3 Coordinate Transformation

The objective of doing a coordinate transform is to be able to represent our data in a form that is more easier to work with or a form that can highlight features that were not present prior to it. In our case we have made a transformation from normal coordinates to polar coordinates. This allows us to represent arcs of circles as line segments. An example of this is shown below, the images before and after the transformation are shown in Figure (6). [2]
The reason for the presence of disjoint line segments instead of a straight line (for a circle) are the gaps between the detector modules and arrays of ASICs, but we also see that these lines are not parallel to each other as one would expect. But rather have some offset. Hence, a well calibrated detector should have little offset. With this we have completed the third step mentioned earlier.

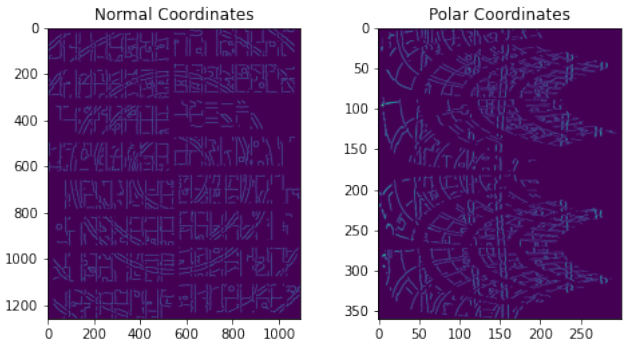

Figure 6: Before and after the coordinate transform

3.4 Hough Transform

After the transformation our image consists of a lot of line segments, now our task is to find these line segments so that their slope and intercept can be calculated. The Hough transform does exactly this, and hence it was used. It works by taking points in the edge map and converts them to sinusoidal curves in the polar hough parameter (r, θ) space. And points which are colinear are apparent from this transformed image as they intersect at one (r, θ) value. For a more detailed description of the Hough Transform refer to [4]. With the edge map in hand we can finally apply the Hough transform to detect the straight lines. For this particular case, we have used the probabilistic Hough transform from `Skimage.features`, which returns the starting and ending points of each line segment detected. With the help of these points, we can calculate the slopes and intercepts of each of the line segment detected.
We only line segments whose slope is close to zero, this thresholding helps us to remove all the artifacts from the edge detection and the leftover noise. Leaving us with a set of horizontal line segments with varied intercepts. By representing each of these line

[2]The `pyFAI` package was used to do this coordinate transformation

segments in the parameter space (slope vs intercept) we can see (Figure (7)) that they are spread over a large range of values. But when we have a well-calibrated detector we would expect to see a very close cluster of points and all of these points lie on the line slope = 0 as they all are horizontal and parallel to each other.

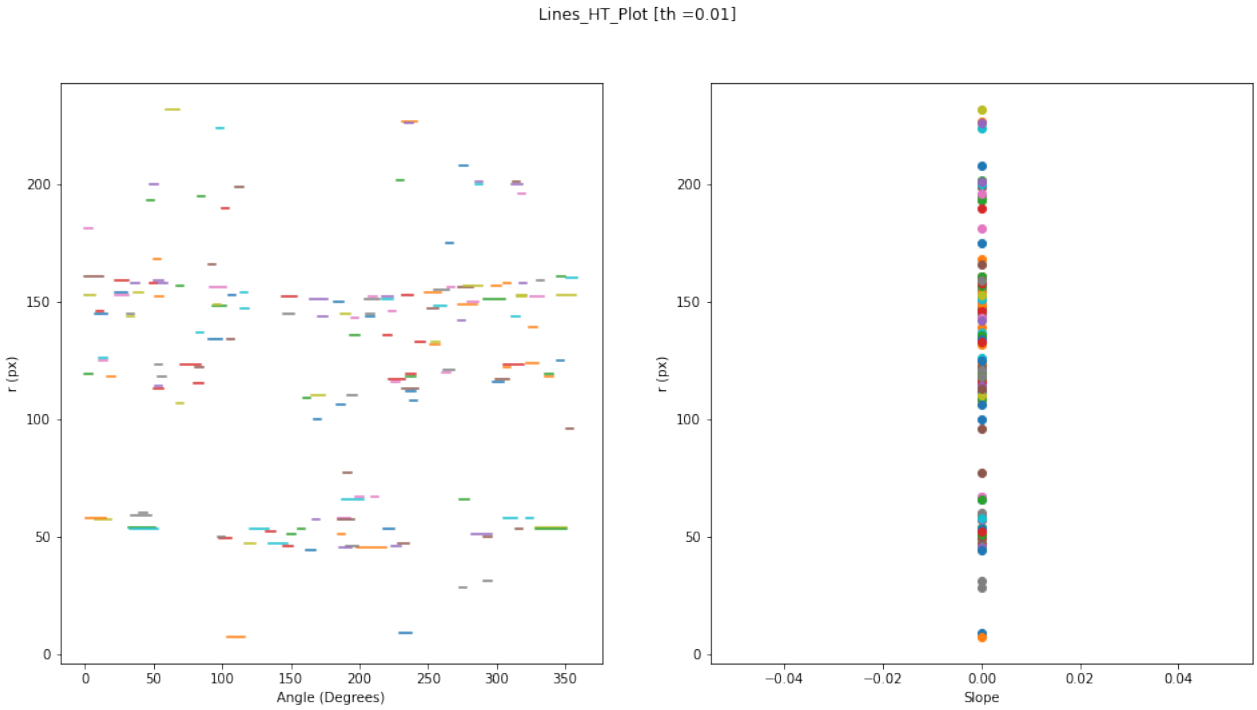

Figure 7: Lines segments detected and their representation in the parameter space (Slope vs Intercept)

4 Clustering

From figure (7) we can see that all the points are lying on the line m = 0, this is after the threshold we have placed on the slope. As discussed before a well-optimized detector geometry would result in having small clusters of points corresponding to circles of different radii. Just by looking at the plot, one can say that the geometry is not optimized. And to be able to change the geometry we need to be able to quantify how good or bad a given configuration is. To do this we will be clustering all the points on the parameter space and we'll define a loss function that says how tight the clusters are.

4.1 Choice of Algorithm

Even after all the noise removal, we have done in the previous sections it isn't enough to get rid of all the outliers. Hence we need to use a clustering method that is resilient to them, hence KMedoids was the clustering algorithm of choice. Not only is this logically

sound, but we were able to see it by comparing it with KMeans. By intentionally increasing the slope threshold we added more outliers to the mix. We noticed that is most cases KMedoids had the least change in terms of the cluster centers when the outliers were added.

Density based clustering methods like DBScan used, but they gave very poor results. This is due to the fact the parameters have to be well tuned for one to get good results. Further studies are necessary to find out the feasibility of DBScan for this use case. Hence algorithms like KMeans and KMedoids were only considered. As the only parameter needed is the number of clusters which can be obtained by doing a Silhouette Score Analysis.

4.2 Silhouette Score Analysis

The Silhouette Score Analysis (SSA) is done to find the optimal number of clusters for a clustering problem, in our case since this is not trivial information we need to find the correct number of clusters for the given image. First clustering is done for different number of clusters, then we calculate the silhouette score for all the points in each case. The silhouette score says how much a given point belongs to the classified cluster. It is defined as $[b - a/max(b - a)]$ where **a** is the mean inter-cluster distance and **b** is the mean nearest-cluster distance for each point.

For the image we have done SSA with cluster number ranging from n = 3 to n = 6. The first plot in Figure (8) has the Average Silhouette Score vs Number of Clusters part, but this alone is not enough to make robust predictions, but we need to pair it with the Proportion of points that have scores greater than or equal to the average silhouette score (second plot in Figure (8)). By taking an average of the two plots, we can incorporate both of them and this will give us the final parameter off which we will choose the number of clusters (third plot in Figure (8)). We can see that for n = 4, we have a clear peak in the third plot, hence this will be our optimum number of clusters. The clusters and the silhouette score of each point has be show in Figure (9). We now have the Cluster centers (c), the points in each cluster (y) and the Number of Clusters (N_c).

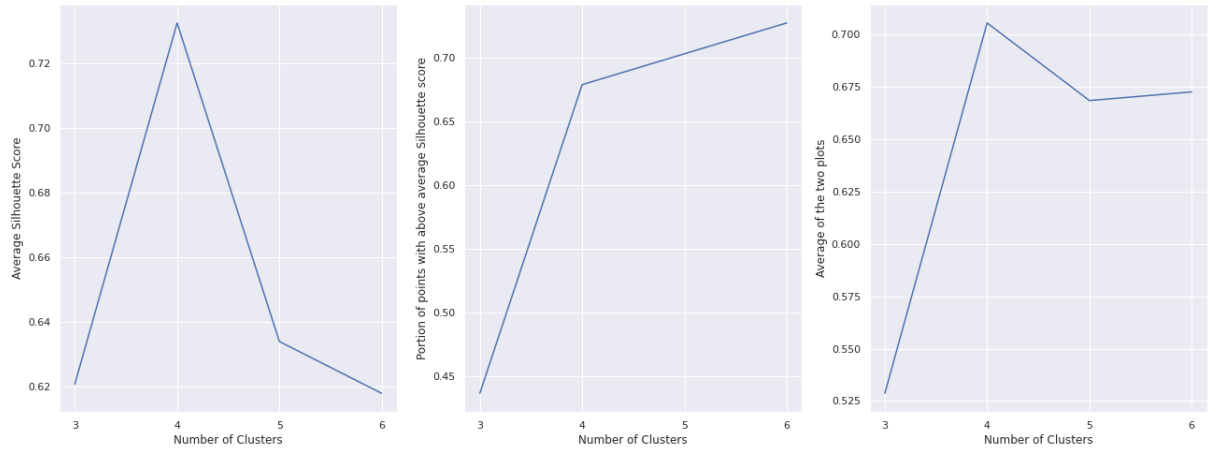

Figure 8: Plots of Average Silhouette Score, Portion of points with above average score and their average of the two plots

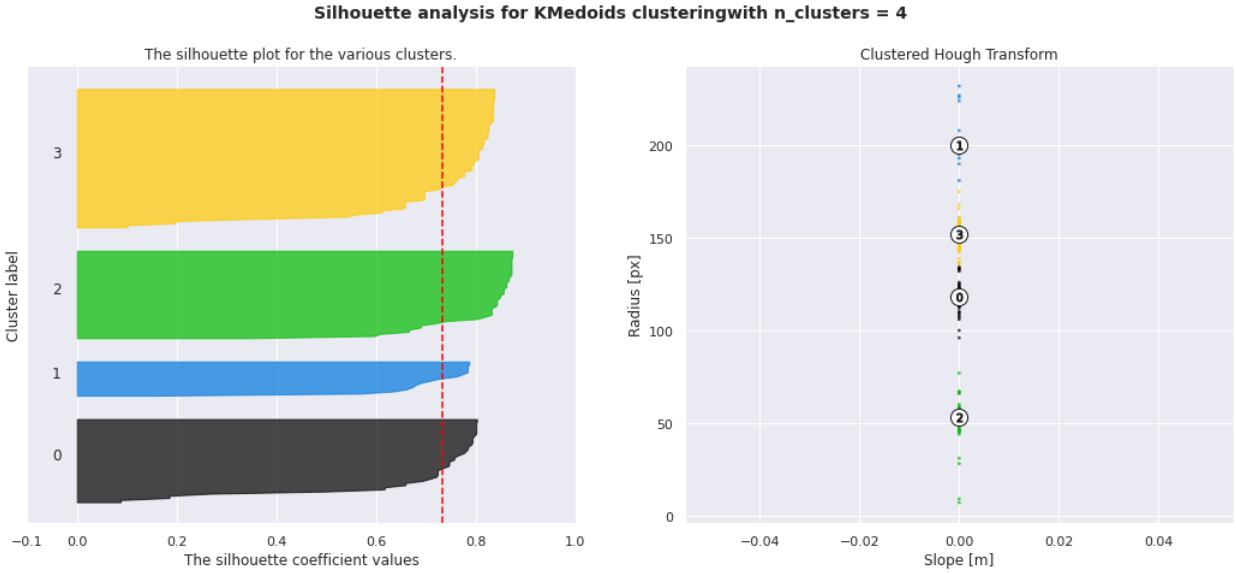

Figure 9: Plot of the silhouette score of each point color coded with respect to it is cluster and the clusters obtained from KMedoids.

4.3 Loss Function

Our motivation was to be able to tell how good or bad a given geometry is, hence everything else has been a precursor to this, the loss function. Hence the loss function must be able to tell if the slopes of the line segments are close to zero as we expect them to be horizontal lines and also how close a given point is to its cluster center in terms

of the Intercept values as well. In this case we define it as the Mean Absolute Error of each point with respect to their cluster center [1]. This method also more resilient to outliers in the data.

$$L(c, y) = \sum_{j=0}^{N_c} \frac{1}{N_j} \sum_{i=0}^{N_j} \mid c_j - y_{i,j} \mid \tag{1}$$

The loss function can be written down as [1], where is N_c Number or clusters, N_j number of points in the j^{th} cluster, c_j is the cluster center for the j^{th} cluster and $y_{i,j}$ represents the i^{th} point in the j^{th} cluster. With the loss function one can use Bayesian Optimisation [3] or any other method for optimising the geometry. For our data the Mean Average Error was found to be 6.706667.

5 Results and Conclusions

The object was to come up with a method to determine how good a given detector geometry is. To do this we have started by doing background subtraction and after testing different schemes the best results were given by using mean subtraction on a local and ASIC level. Next one do edge detection and canny was chosen at it gave thin edges and it also inherently applies a Gaussian filter. And we did a coordinate transformation so that we work with line segments instead of arcs. After this we used the Probabilistic Hough Transform to get the line segments from the edge map. After representing these line segments on the parameter space we used clustering to find points that belong to different circles. Upon a lot of trials we found that KMedoids worked the best with the type of data we had. The only unknown parameter for KMedoids was the number of clusters which was determined by doing a Silhouette Score Analysis. With the classified points and their cluster centers on hand, we only had to define a loss function. Which we chose to be the Mean Absolute Error [1] and this was found to be equal to 6.706667 for the sample data. The next natural step is to use this function for geometry optimisation. But due to time constraints this was not possible to be completed before the end of the summer student program, but it is something I will explore in the near future.

[3] In which case a minus sign must be added as Bayesian Optimisation Maximises a give function

6 Acknowledgement

I would like to thank my supervisors Danilo Ferreira de Lima, Luca Gelisio, Arman Davtyan of the EuXFEL Data Analysis team for their guidance and support without which none of this work would have been possible. I also would like to thank the whole Data Analysis group who welcomed me with open arms, and made me feel like a part of the group. Finally I would like to thank the entire team behind DESY Summer Student program for giving me this wonderful opportunity to learn and interact with people from diverse backgrounds.

References

[1] AGIPD (Adaptive Gain Integrating Pixel Detector.

[2] Skimage Documentation.

[3] Canny Edge Detection.

[4] Hough Transform.