International Journal of Applied Mathematics, Electronics and Computers 10(2): 049-056, 2022

IJAMEC

e-ISSN: 2147-8228

INTERNATIONAL JOURNAL OF APPLIED MATHEMATICS ELECTRONICS AND COMPUTERS

www.dergipark.org.tr/ijamec

International Open Access

Volume 10
Issue 02

June, 2022

Research Article

A Line Fitting Algorithm: Linear Fitting on Locally Deflection (LFLD)

Mahmut Sami Yasak [a,*], ***Muhammed Said Bilgehan*** [b]

[a]*Deep Learning and Computer Vision Engineer, Alpplas, Istanbul 34500, Turkey*
[b]*Deep Learning and Computer Vision Engineer, Alpplas, Istanbul 34500, Turkey*

ARTICLE INFO

Article history:
Received 1 March 2022
Accepted 1 June 2022

Keywords:
Line Fitting,
Linear Smoothing,
Locally Deflection.

ABSTRACT

The main motivation of the study is to prevent and optimize the deviations in linear connections with complex calculations related to the previous and next steps. This purpose is used for more stable detection and therefore segmentation of object edge/corner regions in Quality Control Systems with Image Processing and Artificial Intelligence algorithms produced by authors within Alpplas Industrial Investments Inc. The dataset used in this area was originally obtained as a result of the edge approaches of the plastic panels manufactured by Alpplas Inc., extracted from the images taken from the AlpVision Quality Control Machine patented with this research. The data consists entirely of the pixel values of the edge points. Dispersed numeric data sets have quite changeable values, create high complexity and require the computation of formidable correlation. In this study, dispersed numeric data optimized by fitting to linearity. The LFLD (Linear Fitting on Locally Deflection) algorithm developed to solve the problem of linear fitting. Dispersed numeric data can be regulated and could be rendered linearly which is curved line smoothing, or line fitting by desired tolerance values. The LFLD algorithm organizes the data by creating a regular linear line (fitting) from the complex data according to the desired tolerance values.

1. Introduction

The LFLD algorithm is briefly used for the optimization of one-dimensional numerical data by making local controls within the tolerance information. In the first stage of LFLD, out-of-tolerance values are detected by checking locally depending on previous and next values. Values detected step-by-step are labeled as extreme values, and sub-arrays (AKA out-of-tolerance or extreme arrays) are created. All values between the start and end points of these sub-arrays are re-calculated in the optimization function and replaced in the main array.

The data that LFLD aims to optimize are noisy or scattered numerical data, and signals can be described as planar or 2D data. The noisy data might be available in sensor measurements, communication, transferring, compression, etc. data. To optimize the data, parts of noisy data need to be separated.

Some of the solutions applied to optimize the noisy parts of the data are line smoothing or linear fitting. Also, locally weighted regression [4] method is used to smooth

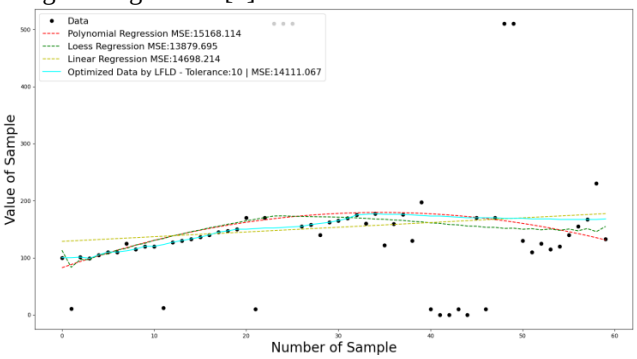

Figure 1. Comparison of Polynomial, Loess, Linear Regression, and LFLD Algorithm with 10 Tolerances on the Sample Dataset

variables are among the methods used to solve the fitting problems. That study is also categorized as an approach to regression analysis by local fitting with nonparametric regression [1]. Locally independent variables [7] [8] are

* Corresponding author. E-mail address: *mahmut.yasak@alpplas.com*
DOI: 10.18100/ijamec.1080843

computations of dynamic analogy types such as time series. This method is commonly used as Locally Estimated Scatterplot Smoothing (LOESS) [3] in a distinct name. Furthermore, this method was developed and also known as locally weighted [16] polynomial regression and abbreviated as LOWESS [4]. Another study in this field is the extraction of linear modeling [9] for a scalar response with dependent and independent variables [5], called linear regression [10] [11] [12] [13] [14] which is widely used for regression analysis. The linear model has estimated the parameters from data in linear regression [6]. There is another method that provides the smoothing of the data by using polynomial regression [2]. Polynomial regression is a class of regression analysis by extracting the relation between dependent and independent variables.

There is an approach study that developed to solve the linear fitting problem by Parasad [17]. That study using the extracted image pixel values such as our proposed solution. These pixels have dispersed conditions and they are trying to solve the fitting in order to linearize these pixel datas. This fitting occurs when an analytical expression of the maximum deviation of the pixels from the digital line can be derived. They are using a control parameter in the first line of the independent data.

In this study, the primary motivation is to apply local optimization and smoothen the parts of the scattered line data that are not within the desired tolerance values. The data extracted from the edge approaches of the plastic panels manufactured by Alpplas Inc., extracted from the images taken from the AlpVision Quality Control Machine patented with this research. LFLD is tested on one-dimensional object edge points extracted with a Canny Edge [17] Detector of plastic panels. Canny Edge Detector output can contain incorrect edges because of environmental parameters such as lighting system, specks of dust, or scratches. Blurring algorithms may fail to clear the image from scratches, or it may fail because of the lighting system. The LFLD is produced to solve these kinds of situations. The stated line output of LFLD is the optimum line to correlate the dependent variables and eliminate the independent variables throughout the way-line. The tolerance parameter is changeable with desired values (with the condition of staying in the data range) that provide the calibrated direction of the line. With the given optimum tolerance for any specified solution, the LFLD can re-produce given scattered line data as more linear (out-of-tolerance values are re-calculated as within tolerance) line data.

These signified related works such as linear regression, polynomial regression, locally weighted polynomial regression, and LFLD algorithms compared with line-fitting performance in Figure 1 and Figure 5. Here, the used dataset is (explained in the proposed methods section) shown as dispersed and noisy and having more dependent variables.

2. Proposed Method

2.1. Line Fitting on Locally Deflection Algorithm

LFLD is an algorithm that takes one-dimensional numeric data as input and applies optimization to data. LFLD mainly focuses on line fitting to dispersed data with noisy as desired tolerance values which are used to determine the direction of the line. In this meaning, the data may be an edge of an object in images, a history of route data with missing locations of a vehicle, or noisy voice data. In the aim of focus, the study tested on one-dimensional object edge points extracted with a Canny Edge Detector shown in Figure 2.

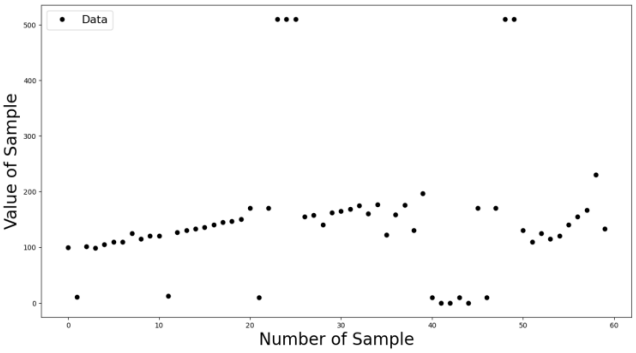

Figure 2. Object Edge Points Extracted with Canny Edge Detector One-Dimensional Data for Testing Purpose

On test data, LFLD starts with the detection of out-of-tolerance values by checking locally. In the step meant as "Checking Locally" is LFLD walks on one-dimensional data and saves the value index before the first out-of-tolerance value (the value within the last tolerance - alpha) in the memory, and after this event, it continues to walk with knowing it is at the out-of-tolerance area (blind area) until it encounters the value within the last tolerance value index (beta) and applies optimizing process between the range of alpha (last out-of-tolerance value index before blind area) and beta (first tolerance value index after blind area). The optimization process is calculated with the formula given in the following equations.

$$\mu = \frac{\beta - \alpha}{\varepsilon + 1} \tag{1}$$

$$\Delta^i = \sum_{i=1}^{\varepsilon} (\beta + i * \mu) \tag{2}$$

$$\Delta = \left[\Delta^1, \Delta^2, \Delta^3, \ldots, \Delta^i, \ldots, \Delta^\varepsilon\right] \tag{3}$$

In equation 1, where μ is a new value that is the result of the Linear Fitting Process which calculation shown in equation 1 and algorithm, β is Average Difference at subsequence of the one-dimension array that is encountered out-of-tolerance. In equation 1 where α is the start value at the out-of-tolerance sequence and ε is the number of samples in the out-of-tolerance sequence shown in algorithm 1 of Figure 9. On the other hand, equation 3 is to show the array of all re-calculated subsequences of the one-dimension array that is encountered out-of-

tolerance where it starts from one to ε. After the calculation of μ, with algorithm 3 in Figure 9, the result will be used in Δ^i the calculation which shows in algorithm 2 of Figure 9. After calculation of all Δ^i's are where i is from first element index of out-of-tolerance by sequence to last element index of out-of-tolerance sequence, result array will be syncs to the main array with between specified array index ranges.

Table 1. Error metrics of the line fitting algorithms

Methods Name	Data	MSE
LOESS	Numeric Data	13879.695
Linear Regression	Numeric Data	14698.214
Polynomial Regression	Numeric Data	15168.114
LFLD	Numeric Data	**14111.067**

In Table 1, As a result of the application of the related works and the LFLD algorithm are shown a performance metric as Mean Squared Error (MSE). The MSE error metric is calculated by the following equation.

$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i')^2}{n} \quad (4)$$

In equation 4, where $y_i$is actual values, y_i' is the predicted or calculated values, i is data to forecasting or calculation, n refers to the size of data. The calculated and error metrics were determined above equation as MSE=14111.067 for LFLD. The reason that the MSE scores of all related works are too high is that the used sample data regression values range is too high. Also, the data regression range is shown in Figure 2 and Figure 4. Thus, all MSE results of related works and LFLD are too high by virtue of the high distance between all regression points in the sample data.

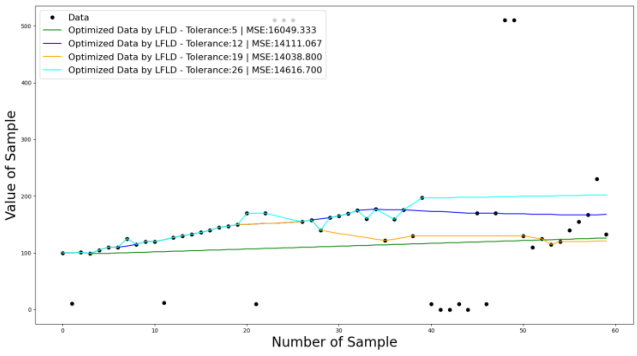

Figure 3. The Line Fitting Performance by Different Tolerances of the LFLD Algorithm

In Figure 3 and Figure 6, there are four desired tolerance values such as 5, 12, 19, and 26 to perform to determine the direction of the line. The direction is collapsed by the tolerable noise value or independent value as a specified parameter. The used sample tolerance parameters show how the degree of contained view and how it occurs.

Also, our proposed method applied another dataset [18] shown in Figure 4 that has signal and noise variables as dispersed conditions and linear relationship between independent variables and dependent variables.

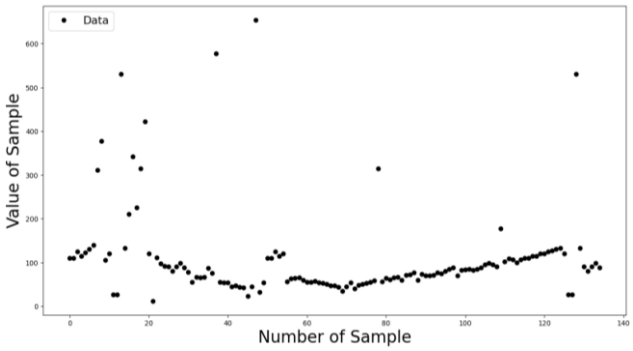

Figure 4. Second Sample Dataset

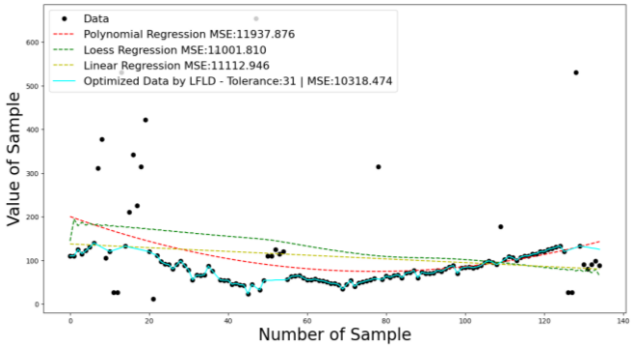

Figure 5. Comparison of Polynomial, Loess, Linear Regression, and LFLD Algorithm with 31 Tolerances on the Second Sample Dataset

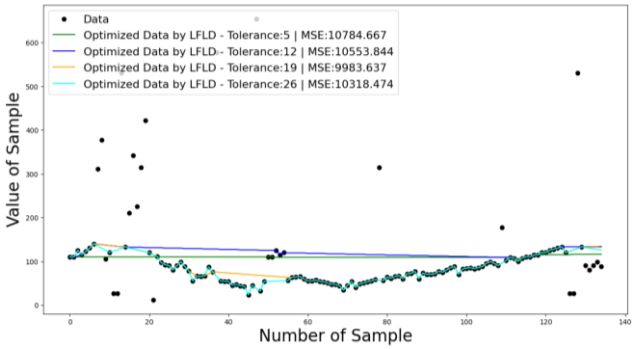

Figure 6. The Line Fitting Performance by Different Tolerances of the LFLD Algorithm

2.2. Algorithm Schematic

In Equations 1, 2, and 3 are completed explanations in Figures 4 and 5. Also, algorithm schematics are detailed and represented pseudocode in Figure 7.

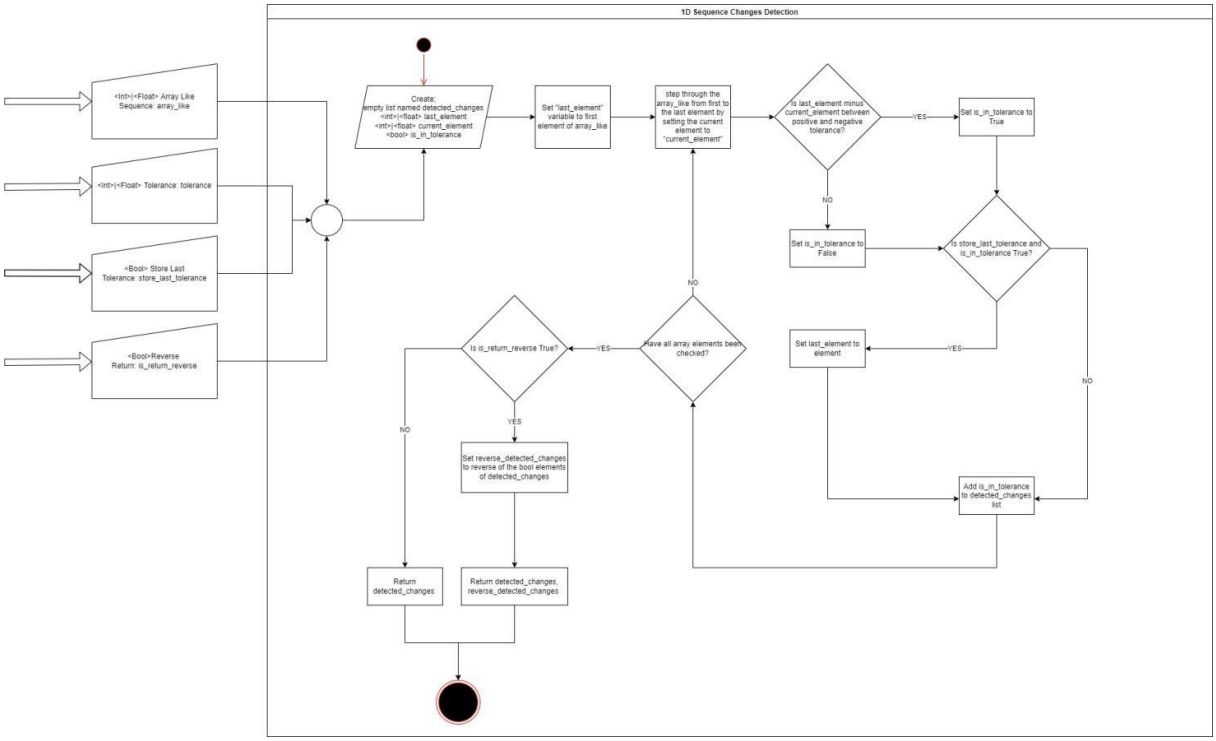

Figure 7. Extraction of coefficient with repetitive values at algorithm-1D sequence changes detection

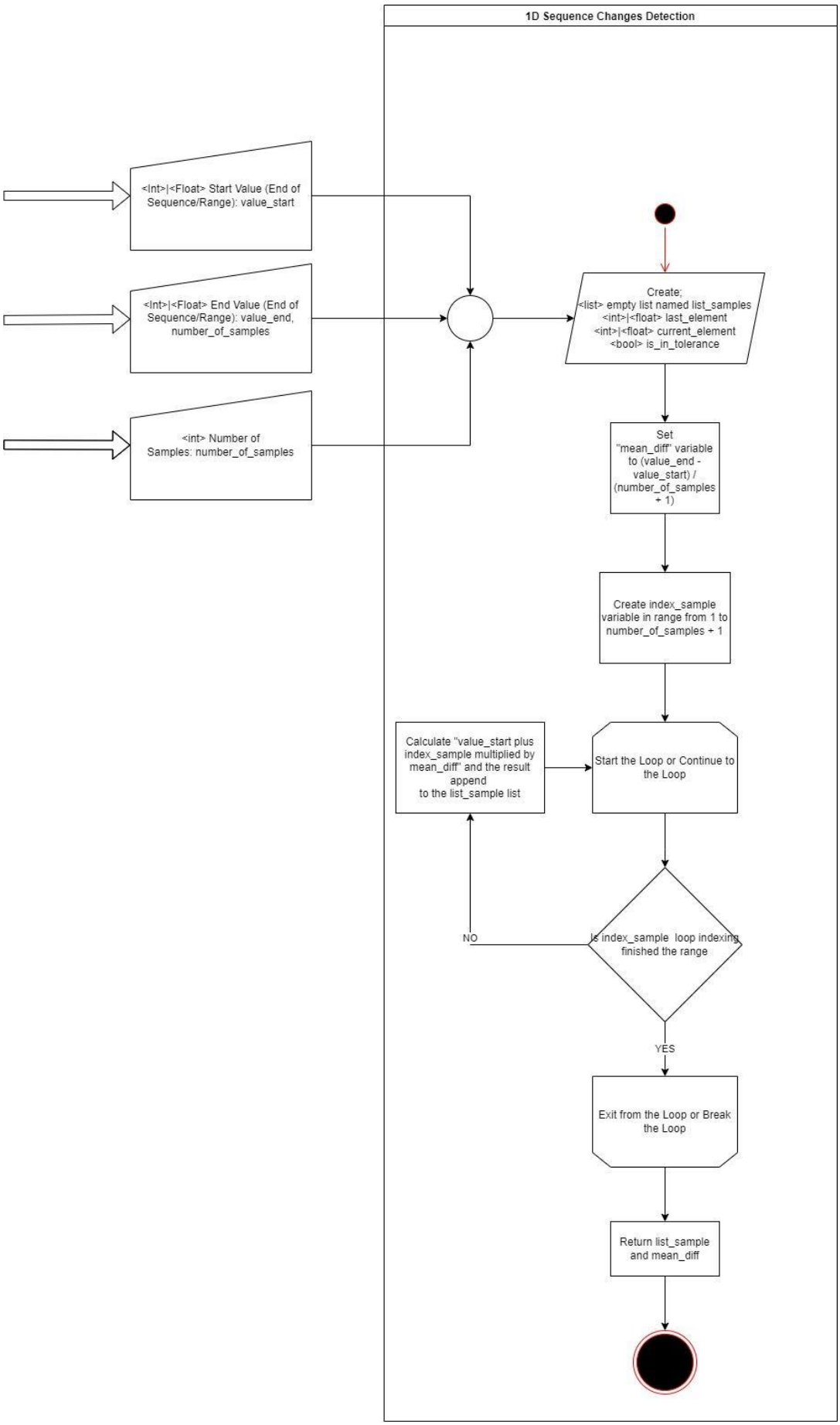

Figure 8. Extraction of coefficient with repetitive values at sequence on algorithm-optimized sampling in range

Algorithm 1 Linear Fitting on Locally Deflection

Input: One Dimensional Numeric Array, Tolerance in *Initialisation* :
array_like ← *Input* 1
tolerance ← *Input* 2
local_buffered_optimized ← *Array*
detected_change_points ← **1D Sequence Change Point Detection**(*array_like*, *tolerance*)
array_like_length ← *size of array variable array_like*
last_mean_diff ← 0
mean_diff ← 0
mean_diff_number ← 0
sequence_start_index ← 0
sequence_end_index ← 0
for *index from zero* to *array_like_length* **do**
if *detected_change_points*[*index*] *is* **true** **then**
for *sub_indexfromrangesequence_start_index* + 1*toarray_like_length* **do**
if *detected_change_points*[*sub_index*] *is* **false or** *sub_index* + 1 *equal to array_like_length* **then**
sequence_end_index = *sub_index*
index = *sequence_end_index*
break
end if
if *sequence_end_index* + 1 *is equal to array_like_length* **then**
local_buffered_optimized, *last_mean_diff* ← Optimized Sampling in Range(array_like[sequence_start_index - 1], array_like[sequence_start_index - 1] + ((array_like_length - sequence_start_index) * (mean_diff / mean_diff_number)), array_like_length - sequence_start_index) array_like[from sequence_start_index until end of array_like] ← local_buffered_optimized
else
local_buffered_optimized, *last_mean_diff* ← Optimized Sampling in Range(array_like[sequence_start_index - 1], array_like[sequence_end_index], array_like_length - sequence_start_index) array_like[from sequence_start_index to sequence_end_index] ← local_buffered_optimized
end if
end for
mean_diff ← mean_diff + last_mean_diff mean_diff_number ← mean_diff_number + 1
end if
if *sequence_end_index is equal to array_like_length* **then**
break
end if
end for
return *array_like* =0

Algorithm 2 (Sub-Algorithm) 1D Sequence Change Point Detection

Input: One Dimensional Numeric Array, Tolerance in
Output: One Dimensional Binary Array out *Initialisation* :
array_like ← *Input* 1
tolerance ← *Input* 2
element ← 0
sequence_changes ← *array*
sequence_start ← 0
sequence_end ← 0
last_element ← *array_like*[0]
is_in_tolerance ← **false**
for *element* in *array_like* **do**
is_in_tolerance ← $(-tolerance) < (last\_element - element) < tolerance$
sequence_changes ← *sequence_changes* + *not is_in_tolerance*
if *is_in_tolerance* **then**
last_element ← *element*
end if
end for
return *sequence_changes* =0

Algorithm 3 (Sub-Algorithm) Optimized Sampling in Range

Input: Value Start, Value End, Number of Samples in
Output: One Dimensional Binary Array out *Initialisation* :
value_start ← *Input* 1
value_end ← *Input* 2
number_of_samples ← *Input* 3
list_sample ← *array*
mean_diff ← $(value\_end - value\_start)$ / $(number\_of\_samples + 1)$
for *index* in $range(1, number\_of\_samples + 1)$ **do**
$list\_sample \leftarrow list\_sample + (value\_start + index * mean\_diff)$
end for
return *list_sample*, *mean_diff* =0

Figure 9. The main LFLD algorithm is explained in Algorithm 1, Also sub-algorithms in Algorithm 2 and Algorithm 3 call the main algorithm to optimize and solve the locally deflection

3. Performance Hardware

Intel i7-10750H 2.60GHz CPU, 16GB 2133MHz RAM, and NVidia GTX 1650Ti with Max-Q Design 4GB GPU RAM hardware was used to test the algorithm. The processing times of the whole applied algorithms, which are mentioned in the tables, were obtained with the specified hardware and the programming phase of the study was developed in Python 3.7 Programming Language.

Table 2. Hardware information

Hardware	Process Time (ms)
CPU	Intel i7-10750H CPU @ 2.60GHz
RAM	16 GB 2667 MHz
Memory	SSD 512 GB
GPU	NVIDIA GeForce GTX 1650Ti 4GB

4. Conclusions

The Line Fitting Locally Deflection algorithm has been applied as a solution for the dispersed noisy data to solve the linear smoothing problem. Also, the proposed tolerance feature can be able to determine the direction of the line in every step into the dataset.

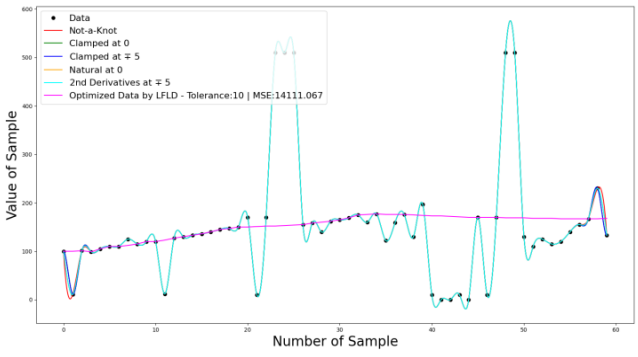

Figure 10. LFLD Compare Alternatives Scipy Library Algorithms

In this meaning, the dataset can be an edge of an object in images, a history of route data with missing locations of a vehicle, or noisy voice data. In the aim of focus, the study tested on one-dimensional object edge points extracted with a Canny Edge Detector shown in Figure 2. Also, both LFLD and scipy library algorithms calculations are shown in Figure 10. As shown in the figure, LFLD Algorithm with tolerance 10 has the minimum slip that fits to the purpose, which is to get the optimum edge points of an object. For future works, LFLD may contain machine learning algorithms to detect optimum tolerance for well-known purposes such as edge detection, segmentation errors correction [15], etc. Also, LFLD has the problem that if the starting point value of the master data is incorrect, the whole process will be calculated based on the improper starting value. Thus the algorithm should trust the start value of the dataset.

In Table 3, the LFLD algorithm is compared with process time performance with related loess, linear, and polynomial algorithms and it has also been added to the comparison by using some interpolation algorithms such as Not-a-Knot, Clamped, and 2nd Derivative in the scipy library, which is an open library publication.

Table 3. The process time metric of applied algorithms

Algorithm	Process Time (ms)
Not-a-Knot	0.00100
Clamped at 0	0.00100
Clamped at 5	0.00100
Natural at 0	0.00100
2nd Derivatives at 5	0.00100
Polynomial Regression	0.00798
Loess Regression	0.04288
Linear Regression	0.00199
LFLD – Tolerance: 26	0.00200
LFLD – Tolerance: 19	0.00200
LFLD – Tolerance: 12	0.00200
LFLD – Tolerance: 5	0.00100
LFLD – Tolerance: 10	0.00200

References

[1] Ethem Alpaydın. “Adaptive Com-putation and Machine Learning”. *Introduction to Machine Learning*, MIT Press, Cambridge, MA, 3 edition, 2014, pp. 79.

[2] Y. W. Chang et al., “Training and testing low-degree polynomial data mappings via linear svm”, *J. Mach. Learn.* Res., 11, pp. 1471–1490, 2010.

[3] W. S. Cleveland, “Robust locally weighted regression and smooth-ing scatterplots”, *Journal of the American Statistical Association*, 74(368), pp. 829–836, 1979.

[4] W. S. Cleveland and S. J. Devlin, “Locally weighted regression: An approach to regression analysis by local fitting”, *Journal of the American Statistical Association*, 83(403), pp. 596–610, 1988.

[5] D. Freedman, “*Statistical Models: Theory and Practice*”, Cambridge University Press, August 2005.

[6] H. L. Seal, “Studies in the history of probability and statistics. xv: The historical development of the gauss linear model”, *Biometrika*, 54(1/2), pp. 1–24, 1967.

[7] Caruana, Rich and Virginia R. de Sa. “Benefitting from the Variables that Variable Selection Discards.” *J. Mach. Learn. Res.*, 3, pp. 1245-1264, 2003.

[8] V. N. Vapnik, “Conditions for Consistency of Empirical Risk Minimization Principle”. *Statistical Learning Theory*, Wiley Interscience Publication, 1998, pp. 82.

[9] Chatfield, Chris, “Non-linear and non-stationary time series analysis”. *M.B. Priestley Academic Press*, London, 1989, pp. 237.

[10] H. J. Seltman, “Simple Linear Regression”. *Experimental Design and Analysis*, Carnegie Mellon University, 2013, pp.227.

[11] Y. Dodge, “Simple Linear Regression”. *The Concise Encyclopedia of Statistics*, Springer New York, 2008, pp.491-497.

[12] D. M. Lane, “Introduction to Linear Regression”. *Introduction to Statistics*, David Lane Rich University, 2008, pp.462.

[13] K. H. Zou, K. Tuncali, and S. G. Silverman, “Correlation and simple linear regression”, *Radiology*, 227(3), pp. 617–622, 2003.

[14] N. Altman, M. Krzywinski, “Simple linear regression”, *Nat Methods*, 12(11), pp. 999-1000, 2015.

[15] I. Sereda et al., “Segmentation by Neural Networks: Errors and Correction”, *Computing Research Repository (CoRR)*, 2018.

[16] W. S. Cleveland and S. J. Devlin, “Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting”, *Journal of the American Statistical Association*, 83(403), pp. 596-610, 1988.

[17] J. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis andMachine Intelligence*, PAMI-8 (6), pp. 679-698, 1986.

[18] Kaggle, Linear Regression: Signal and Noise, USA [Online]. Available:https://www.kaggle.com/competitions/inclass-signal-and-noise/data, Accessed on: May. 23, 2022.