

SISTEMAS EMBEBIDOS

PROYECTO PRIMER BIMESTRE

1. Introducción

Las prácticas del laboratorio pueden ser:

- Prácticas convencionales o de comprobación.
- Prácticas de diseño.
- Prácticas por proyecto (PP)
- Prácticas Integradoras (PI).

Las prácticas convencionales se centran básicamente en comprobar el funcionamiento de circuitos y montajes, por lo que se basan en esquemas repetitivos que permiten al alumno medir sus logros a fin de ir mejorando paulatinamente. Las prácticas de diseño, son aquellas en donde se enfrenta al estudiante con problemas reales de baja y mediana complejidad a resolver electrónicamente, junto con unas especificaciones técnicas a cumplir. Similarmente, las prácticas por proyecto, que involucran conocimientos adquiridos en la asignatura e incluso en otras anteriores de la misma área; y las prácticas Integradoras, que como su nombre lo indica implican integrar conocimientos de diferentes áreas dentro de la carrera. Estas dos últimas modalidades se enfocan hacia el diseño, con la perspectiva de que el conocimiento debe ser coherente con las problemáticas a que se enfrentaran los estudiantes en su rol social de desempeño, incorporando de esta forma una motivación ligada a sus intereses y necesidades en el proceso de aprendizaje.

2. Indicaciones Generales

Al recibir las instrucciones las guías de laboratorio se debe agendar un horario disponible con el técnico docente para el uso de equipos si es necesario. Caso contrario, se lo puede realizar en las horas autónomas de la asignatura. Una vez realizada su práctica, en horario de clases se presenta el funcionamiento de su laboratorio adjuntando su respectivo informe (formato presente en la plataforma).

3. Laboratorio

Ubicación: Laboratorio de Electrónica

PROYECTO

Instrucciones:

El objetivo de este proyecto es adquirir componentes de sensores de forma simulada o emplear un conjunto de datos ya almacenados previamente. Con esto, se busca implementar los criterios de filtros FIR y algoritmos de suavizado de la señal para determinar cuál de ellos es el adecuado. Todo este proceso es mediante la programación dentro del entorno de arduino y comparar el rendimiento por medio de la métrica relación señal-ruido (SNR). Se propone simular el funcionamiento de los sensores por medio de la librería adjunta. Sin embargo, si se tienen inconvenientes, se cuenta con la base de datos [Proyectos Primer Bimestre: Datos de sensores](#). Con ello debe escoger 2 señales. La primera, debe ser de forma obligatoria la señal cardíaca para que pueda trabajar con ambos criterios (convolución y suavizado), revise la teoría de una señal ECG y trabaje con los filtros FIR con las diferentes frecuencias que generarán componentes a usar. Luego, codifique dos algoritmos de suavizado de la señal y compare con la métrica SNR (debe ser programada) cuál es el mejor criterio a ser implementado. Segundo, use otra señal dentro del archivo y solo realice el criterio de suavizado de la señal (nada de filtros FIR) y de igual forma determine al mejor. Se resumen sus actividades de la siguiente forma.

- Instalar la librería de sensores en el software de Arduino. [Video](#) y adquirir datos de dos sensores o usar las componentes adjuntas.
- Revisar los algoritmos de suavizado de la señal del documento que se adjunta en este proyecto y la generación de componentes por ventanas desde el sitio web: <http://t-filter.engineerjs.com/>
- Seleccionar **2 algoritmos** de los presentados en el documento adjunto y codificarlo en el entorno de Arduino.
- Para la señal ECG revisar el filtrado y sus componentes para realizar la convolución
- Codificar la relación señal ruido para estos algoritmos y definir cuál es el mejor con respecto a estos sensores.
- Presentar gráficas de los resultados.
- Presentar un informe con diagramas de flujo y adjuntando su código en los repositorios de GitHub.

Presentación: Martes 15 de diciembre del 2020

4. Contacto

Técnico Docente: Alejandra Pinto.

Fono: 0996392999

correo: ampinto@utn.edu.ec

Docente: Paul Rosero

Fono: 0969432370

correo: pdrosero@utn.edu.ec

Página web: <http://www.paulrosero-montalvo.com>

Review and comparison of smoothing algorithms for one-dimensional data noise reduction

Paweł Kowalski, Robert Smyk
FACULTY OF ELECTRICAL AND CONTROL ENGINEERING
Gdansk University of Technology
ul. Gabriela Narutowicza 11/12, 80-233 Gdańsk, Poland
Email: pawel.kowalski2@pg.edu.pl, robert.smyk@pg.edu.pl

Abstract—The paper considers the choice of parameters of smoothing algorithms for data denoising. The impact of the window size on smoothing accuracy was analyzed. The parameters of denoising filters were selected with respect to the mean-square error between the computed linear regression and the noisy signal. Finally, we have compared mean, median, Savitzky-Golay, Kalman and Gaussian filter algorithms for the data from the digital sensor. The figure of merit was also the algorithm execution time.

Keywords—noise reduction, smoothing, smoothing algorithms, mean filter, median filter, least-square, Savitzky-Golay, Kalman filter, Gaussian filter, kernel smoothing, signal processing, denoising filters,

I. INTRODUCTION

The raw data from sensors requires various procedures to extract the desired information. Because the raw data contains the random noise, it often requires the improvement of the data quality by preprocessing. One of the utilized procedures is the noise reduction. The noise reduction in its simplest form eliminates the extreme values of the signal, giving in effect the reduction of the average magnitude of the signal. Generally, noise reduction algorithms can be classified into the following groups in dependence of the task performed:

Filtering	to estimate the value in the point t uses the data before t , with t
Smoothing	to estimate the value in the point t uses the data before and after t
Prediction	by observing the data before t , the signal value in the point $t+1$ or later can be predict.

We can observe that the choice of one of the above approaches depends on the data availability and the aim of processing. Smoothing may give better results with respect to the denoising, but requires past and future samples of the signal which are not always accessible. Smoothing improves data quality by replacing the noisy irregular signal with the new smoothed signal which probably better describes the measured phenomena. In this work we discuss several smoothing algorithms such as mean filter [1], [2], median filter [3]–[6], Savitzky-Golay [7]–[9], Kalman filter [10]–[12] and Gauss filtering [13], [14]. In Section 2 we shortly review smoothing methods, and in Section 3 we present the results of the individual algorithms and comparison.

II. SMOOTHING METHODS

In this section, we review the chosen known smoothing algorithms. The selection criterion was the small number of calculations that would allow the execution in the real-time while preserving the necessary quality.

A. Mean Filter

The mean filter is one of the simplest smoothing methods. The modified version of this algorithm is discussed in [1], [2]. The output value is calculated by computing the average of all samples from the window. Its formula has the following form:

$$z_j = \frac{\sum_{i=-n}^n x_{j+i}}{2n+1}, \quad (1)$$

where x_i is a sample of the input signal, $2n+1$ is the window length, z_j represents the output value, and j is the current index of the output value. Various forms of averaging are known. One of these is the use of the filter mask consisting of numbers of the same value. For example, when window length is equals to 5 the following mask is used:

$$M_1 = \begin{bmatrix} \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} \end{bmatrix}, \quad (2)$$

In this case, the output value is calculated by convolving of the mask M_1 with the samples in the window. The effectiveness of this filter strictly depends on the window length.

B. Median Filter

The median is a nonlinear filter that smoothes the impulse noise, but without blurring signal changes. It is often used for image denoising because while removing noise the sharpness of edges is preserved. The use of this filter, as well as its modification, have been considered in numerous works [3]–[6]. The traditional median filter works by choosing from the window the midpoint value called the median. Its formula has the following form:

$$z_j = \text{median}(x_{j-n} \dots x_{j-2}, x_{j-1}, x_j, x_{j+1}, x_{j+1} \dots x_{j+n}) \quad (3)$$

In order to find the median, samples from the N -sample window are sorted and the median is determined as:

$$\begin{cases} s_{(N+1)/2} & , \text{ if } N \bmod 2 == 1 \\ \frac{s_{N/2-1} + s_{N/2+1}}{2} & , \text{ if } N \bmod 2 == 0 \end{cases} \quad (4)$$

The formula (4) describes the selection of the median from the sorted list S . If N is odd, the median is the sample $s_{(N+1)/2}$, else the median is the average of two elements $s_{N/2-1}$ and $s_{N/2+1}$.

C. Savitzky-Golay filter

The Savitzky-Golay (SG) smoothing algorithm was firstly presented in [9]. It is a low-pass filter that utilizes the local least-square polynomial approximation. In each step, calculations are performed using the window W_i with the size $2M + 1$:

$$W_i = [x_{-M}, \dots, x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}, \dots, x_{i+M}] \quad (5)$$

For every window $i \in (M; N - M)$, where N is the size of input data, the following polynomial function is used:

$$sg(i) = \sum_{k=0}^n a_k i^k \quad (6)$$

where n is polynomial degree and a is the vector of coefficients. The coefficients are calculated by minimizing the error E . Its formula has the following form:

$$E = \sum_{i=-M}^M [sg(i) - x_i]^2 \quad (7)$$

The details of the procedure of selecting the coefficients is described in [7]–[9]. Using (6), the output signal vector Z is generated. For each window W_i , the $sg(0)$ is added to the output as z_i . The algorithm can be carried out in three steps. In the first step, the segment of the signal samples with a size corresponding to the window W_j length has to be extracted from the input signal. In the second step, the coefficients of (6) are calculated. Then we have $z_j = sg(0)$. Before the next iteration, the window is shifted by one sample forward.

D. Discrete Kalman filter

The discrete Kalman filter algorithm tries to estimate state x of a discrete-time system. It is called an optimal linear filter because incorporates all the information from the process and consequently gives an error that is statistically minimal. The algorithm makes an assumption that process prediction and measurement both are made with a presence of the white gaussian noise.

Typically the algorithm requires two-step calculation procedure: time update step, called a prediction and measurement update step, called the correction. Prediction gives the current state estimate ahead in time and the correction adjusts the projected estimate using the current measurement. In the prediction step the state estimation is

calculated from the following equation:

$$\hat{x}_k^- = A \cdot \hat{x}_{k-1} + B \cdot u_{k-1} \quad (8)$$

where $\hat{x}_k^-$ is the observed state at the timestep k , u_{k-1} is a control input at the timestep $k - 1$. The matrix $A_{n \times n}$ gives a relation between the state at the timestep $k - 1$ to the state at timestep k . The matrix $B_{1 \times n}$ gives the relation of the control input with the state x at the timestep from $k - 1$ to k . In the next stage of the prediction step, the calculation of the error covariance projection has to be performed. Its formula has the following form:

$$P_k^- = A \cdot P_{k-1} \cdot A^T + Q \quad (9)$$

where P_k^- is the error covariance that is *a priori* estimated, P_{k-1} is also error covariance but *a posteriori* estimated and $Q_{n \times n}$ is the process error (noise covariance).

At the output of each particular prediction step, we obtain the consecutive estimates of P_k^- and $\hat{x}_k^-$. At the correction step we have to compute the gain of Kalman filter and use this gain to update the state estimate using current measurement. The gain controls the influence of the measurement on the *a posteriori* state estimate at the timestep k . The Kalman gain is calculated using the following formula:

$$K_k = P_k^- \cdot H^T \cdot (H \cdot P_k^- + R)^{-1} \quad (10)$$

where matrix H relates the measurement with the state x . The matrix R is the measurement inaccuracy (noise covariance). Finally, the formula for the state estimation update with measurements has the following form:

$$\hat{x}_k = \hat{x}_k^- + K_k \cdot (z_k - H \cdot \hat{x}_k^-) \quad (11)$$

where $\hat{x}_k$ is a posteriori state estimate, $\hat{x}_k^-$ is the observed state at the timestep k , K_k the gain that controls the influence of the measurement on $\hat{x}_k$, z_k is the measurement at the timestep k and matrix H gives relationship of the measurement to the state.

At the end of the update procedure the error covariance update is required. Its formula has the following form:

$$P_k = (I - K_k \cdot H) \cdot P_k^- \quad (12)$$

The presented discrete Kalman filter algorithm has to be implemented as a recursive function but in practice it is rather implemented in the iterative form to reduce the calculation time. The main parameter responsible for effectiveness of smoothing is the process noise covariance.

E. The Gaussian filter

The Gaussian filter [13], [14] is based on the normal distribution called the Gaussian or Laplace-Gauss distribution. The probability density of the Gaussian distribution describes

the following formula:

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (13)$$

where σ is the standard deviation, σ^2 the variance, and μ is the average.

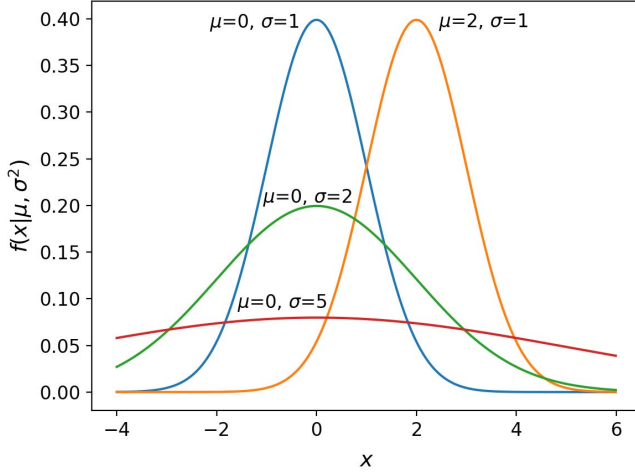


Fig. 1. Impact of parameters σ and μ on Gauss bell shape

Fig. 1 shows the impact of parameters σ and μ on Gauss bell shape. By changing μ , the plot moves left or right, and the change σ changes the width of the bell. In smoothing generally, the middle value of the window is estimated, so μ is 0. Smoothing using Gaussian filter uses the specific mask. The mask is built based on (13) using the distribution samples, for example:

$$G = [0.05, 0.24, 0.4, 0.24, 0.05] . \quad (14)$$

The formula (14) shows the mask for Gaussian smoother with the standard normal deviation $\sigma = 1$, related to five values in the window. In every step the samples from the window are convolved with the sequence representing Gaussian distribution with the mask G , then the window is moved one sample forward. In order to keep the scale of input values, after convolution, the computed value should be divided by sum of mask elements. For (14) result should be divided by 0.98.

F. Kernel Denisty Estimation

Kernel Density Estimation (KDE) [15], [16] is one of the most popular non-parametric methods that enhances some structural features in the data unavailable in parametric approach. The kernel estimator $\hat{f}(x)$ estimates the distribution of probability density f of the signal. Its formula has the following form:

$$\hat{f}(x) = \frac{1}{mh^n} \sum_{i=1}^m K\left(\frac{x-x_i}{h}\right), \quad (15)$$

where h is the bandwidth that controls the smoothness of the

estimate, m the number of samples, n the dimensions of the signal vector X . In this work we use one-dimensional signal, so $n = 1$. K is called the kernel. There are various forms of kernels and the selection of the most appropriate one is typically made on the basis of a calculation of mean integrated squared error $MISE_{opt}(\hat{f})$. The simplest non optimal uniform kernel has the following form:

$$K(u) = \begin{cases} \frac{1}{2}, & \text{if } |u| \leq 1 \\ 0, & \text{if } |u| > 1 \end{cases} . \quad (16)$$

The most popular is Epanechnikov kernel [17]. This kernel is optimal regarding the minimization of MISE. Its formula has the following form:

$$K(u) = \begin{cases} \frac{3}{4}(1-u^2), & \text{if } |u| \leq 1 \\ 0, & \text{if } |u| > 1 \end{cases} . \quad (17)$$

In the implementation of the smoothing procedure the Nadaraya-Watson kernel regression [18] was used, in the following form:

$$\hat{E}(Y|X=x) = \frac{\sum_{i=0}^n K_h(x-x_i)y_i}{\sum_{j=0}^n K_h(x-x_j)} . \quad (18)$$

III. EXPERIMENTAL RESULTS

In order to examine the influence of parameters of individual smoothing algorithms on the quality of the data preprocessing process, a series of experiments were carried out on a representative sample set from the ADC converter. Initially, the input data set has been normalized. In order to check whether data has the normal Gaussian distribution the mean value and standard deviation were calculated.

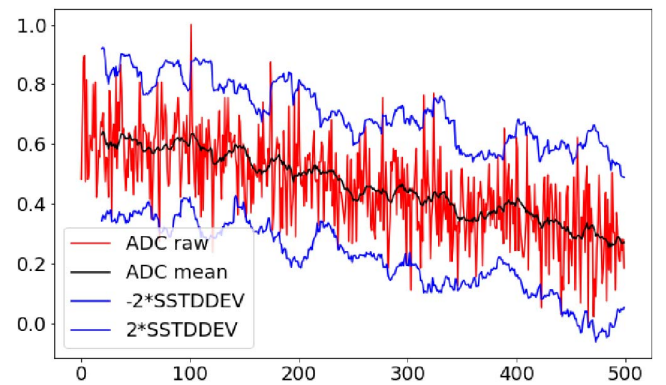


Fig. 2. The input data samples set with mean and standard deviation values

In the data with a normal distribution about 70 percent is within a standard deviation of the mean and 95 percent of values within two standard deviations of the mean as we can see in Fig. 2.

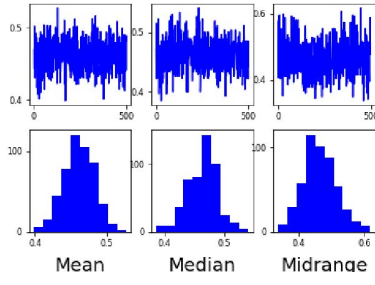


Fig. 3. Statistical tests for input data set

As the scaled data has zero mean (1.13×10^{-16}) and unit variance, we can model the input data using the linear regression based on the least-squares technique and assume that it is the optimal model (Fig. 4).

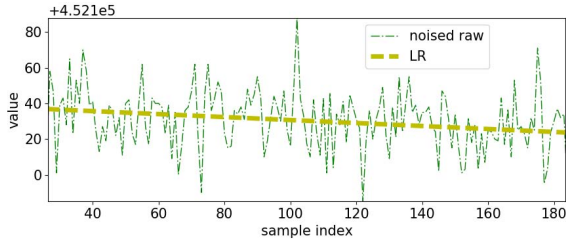


Fig. 4. Illustration of raw data and the result of using the linear regression model

Such a model (Fig. 4) can be further used as a reference to compare the effectiveness of benchmarked smoothing methods (Fig. 5).

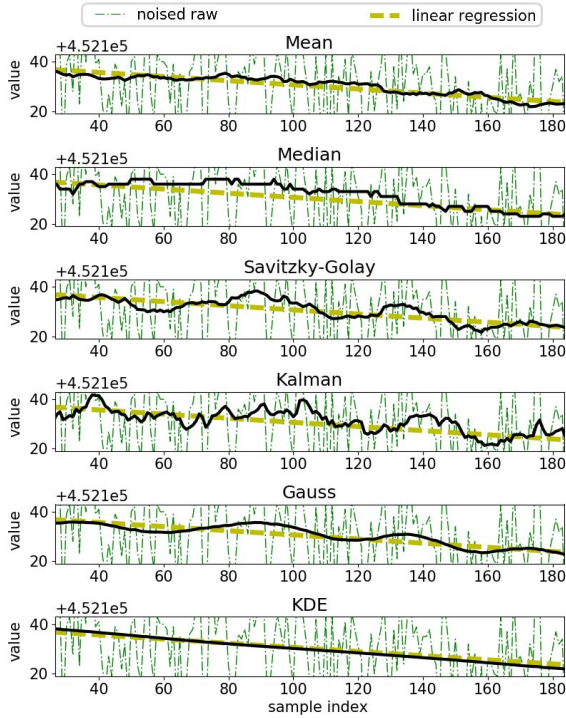


Fig. 5. Illustration of smoothing efficiency for selected algorithms

Finally, experiments were carried out on a set of 210 samples. In order to make a comparison, we also used a very effective smoothing algorithm based on the kernel density estimation (KDE). The nonparametric estimators in comparison to other proposed methods have no fixed structure. The output results depend upon all the data points to obtain an estimate. The mean square error (MSE) level was taken as the principle criterion of the effectiveness of the smoothing process when using the individual algorithms. The second important criterion was the standard deviation. It should be noted that there is a correlation between the mentioned criterions. The main goal was to obtain the MSE at the assumed error level value less than 8 for parameters determined experimentally. The values of the criterion dependent on the windows size are illustrated in Fig. 7 and Fig. 6 and for the window size independent algorithms presented in the Table I. It can be easily seen, that the kernel density estimation algorithm gives the output result that is very close to the ideal model. The results of other methods are very strongly dependent on the size of the window.

TABLE I. STANDARD DEVIATION AND MSE FOR SMOOTHING ALGORITHMS IMPLEMENTED IN PYTHON

Algorithm	Std. dev.	MSE
KDE	2.61	0.64
Kalman	3.22	8.26

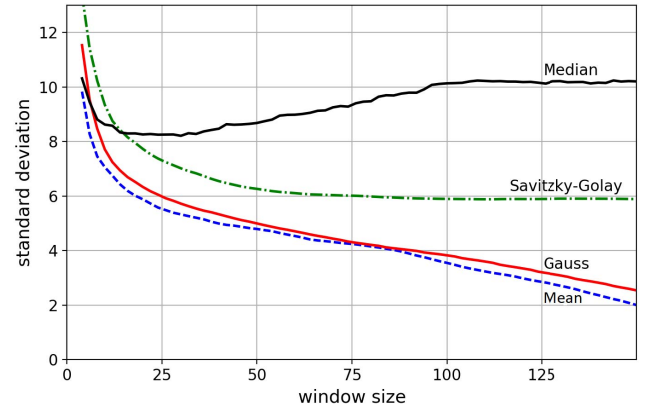


Fig. 6. Standard deviation level for smoothing algorithms

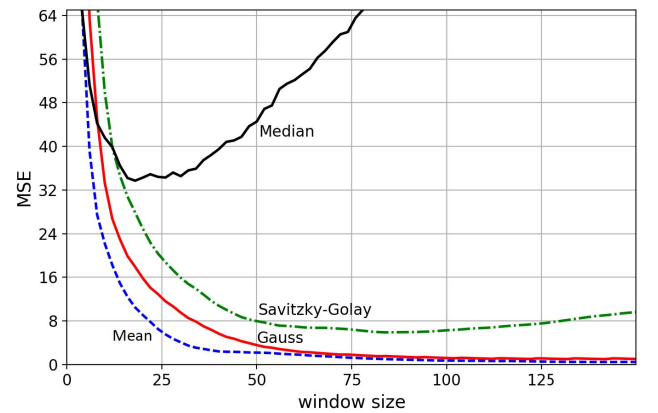


Fig. 7. MSE level for smoothing algorithms

The results of the experiments (Fig. 7) show that for

Savitzky-Golay smoothing the MSE below 8 can be obtained for the window size greater than 50. In comparison to the mean and the Gauss filter this is a slightly worse result, but we should keep in mind that mean and Gauss introduces an interval in the processing of first N samples.

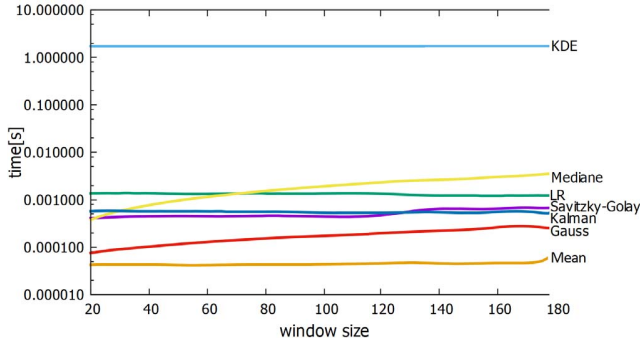


Fig. 8. Execution times for selected algorithms

The execution times of smoothing algorithms for a sample of 180 calls regarding the window size are illustrated in Fig. 8. The algorithms were run on Core i7 class PC in the Python environment with the support of the NumPy and SciPy libraries. As we can see the Savitzky-Golay smoothing and median smoothing for the windows size of 25 has the same time of execution, but when the window size is greater than 25 the Savitzky-Golay is more time-effective than median smoothing. For the processing window size 63, the execution time of the Savitzky-Golay is 3 times shorter than of the median filter (Fig. 8). Additionally, we can see that Savitzky-Golay algorithm execution time is comparable with the Kalman filter. The most time effective is the mean filter, but the algorithm has relatively long initial delay strictly dependent on the length of the window as mentioned earlier. The use of the averaging algorithm makes that the output result is obtained as $N + 1$ sample for the window with the length N . The kernel density estimation smoothing is most time-consuming. The nonparametric methods are known for their computational complexity and thus they are rather not suitable for real-time processing. In practice, nonparametric methods can be applied to data segmented into subsets for which KDE is calculated separately. Because of its high efficiency of smoothing process and the small execution time, it seems that the Gauss smoothing algorithm and the averaging are the best choices for real-time processing. It is also worth to mention that benchmarked methods have the extensive support in the form of ready-to-use functions in many popular digital signal processing libraries.

IV. CONCLUSION

We have presented a review and comparison of a group of smoothing algorithms such as mean, median, Savitzky-Golay, Kalman, and Gaussian filter. We have analysed their behaviour with respect to the quality of approximation, and computational requirements for signals from the temperature sensor. The question was whether a simple algorithm may be able to provide the necessary quality of approximation with the reduced computational requirement. It turned out for this type of the signal that simple smoothers as the mean filter may give satisfactory results with a small number of calculations

that is especially important when the algorithm operates in the real time. We have compared the number of operations for mean, median, Savitzky-Golay, Kalman, and Gaussian filter. It was stated that the best results were obtained for the averaging and Gaussian algorithm.

REFERENCES

- [1] B. Fu, X. Xiong, and G. Sun, "An efficient mean filter algorithm," in *The 2011 IEEE/ICME International Conference on Complex Medical Engineering*, May 2011, pp. 466–470.
- [2] P. Shrivastava and U. P. Singh, "Noise removal using first order neighborhood mean filter," in *2014 Conference on IT in Business, Industry and Government (CSIBIG)*, March 2014, pp. 1–6.
- [3] P. K. Sinha and Q. H. Hong, "An improved median filter," *IEEE Transactions on Medical Imaging*, vol. 9, no. 3, pp. 345–346, Sep 1990.
- [4] N. C. Gallagher, "Median filters: a tutorial," in *1988, IEEE International Symposium on Circuits and Systems*, June 1988, pp. 1737–1744 vol.2.
- [5] Z. Wang and D. Zhang, "Progressive switching median filter for the removal of impulse noise from highly corrupted images," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 1, pp. 78–80, Jan 1999.
- [6] T. Chen, K.-K. Ma, and L.-H. Chen, "Tri-state median filter for image denoising," *IEEE Transactions on Image Processing*, vol. 8, no. 12, pp. 1834–1838, Dec 1999.
- [7] S. R. Krishnan and C. S. Seelamantula, "On the selection of optimum savitzky-golay filters," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 380–391, Jan 2013.
- [8] S. P. Magazine and R. W. Schafer, "What is a savitzky-golay filter?," *IEEE SIGNAL PROCESSING MAGAZINE*, vol. 28, no. 4, pp. 111 – 117.
- [9] A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures," *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, Jul. 1964. [Online]. Available: <http://dx.doi.org/10.1021/ac60214a047>
- [10] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME—Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.
- [11] G. Welch and G. Bishop, "An introduction to the kalman filter," vol. 8, 01 2006.
- [12] S. Pham and A. Dinh, "Using the kalman algorithm to correct data errors of a 24-bit visible spectrometer," *Sensors (14248220)*, vol. 17, no. 12, pp. 1 – 20, 2017.
- [13] C. F. Gauss, *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Hamburgi : I. H. Besser, 1809., 1809.
- [14] C. H. Davis and C. F. Gauss, *Theory of the motion of the heavenly bodies moving about the sun in conic sections a translation of Gauss's "Theoria motus." With an appendix. By Charles Henry Davis*. Boston, Little, Brown and company, 1857.
- [15] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *Ann. Math. Statist.*, vol. 27, no. 3, pp. 832–837, 1956.
- [16] P. Kulczycki, M. Charytanowicz, P. A. Kowalski, and S. Lukasik, "The complete gradient clustering algorithm: properties in practical applications," *Journal of Applied Statistics*, vol. 39, no. 6, pp. 1211–1224, 2012.
- [17] V. A. Epanechnikov, "Non-parametric estimation of a multivariate probability density," *Theory of Probability & Its Applications*, vol. 14, no. 1, pp. 153–158, 1969.
- [18] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.