Parallel Algorithms for Interpolation with Bezier Curves and B-Splines for Medical Data Recovery

Lesia Mochurad [a], Yulianna Mochurad [b]

[a] *Lviv Polytechnic National University, 12 Bandera street, Lviv, 79013, Ukraine*
[b] *Danylo Halytsky Lviv National Medical University, 69 Pekarska street, Lviv, 79010, Ukraine*

Abstract
Clinical data are increasingly being used to generate new medical knowledge to improve diagnostic accuracy, better personalize therapeutic regimens, improve clinical outcomes, and more efficiently use healthcare resources. However, clinical data often become available only at irregular intervals, contingent upon patients and data types, with missing records or unknown indicators. Consequently, missing data often constitute a primary impediment to optimal knowledge extraction from clinical data. This study analyzes proposed parallel interpolation algorithms using Bézier curves and B-splines for medical data restoration. Parallelization is achieved through CUDA technology. The research reveals that the considered algorithms do not compromise the accuracy of subsequent forecasting; in fact, they marginally enhance it. Acceleration indices are obtained, indicating an acceleration factor of 11 for B-splines and – 97 for Bézier curves. The latter allows for swift data restoration, exerting minimal impact on the overall forecasting time – a crucial aspect in cardiovascular disease scenarios. Further exploration of this research can involve variations and selection of an optimal degree of interpolation curves, forecasting methodologies, and the size of the studied dataset.

Keywords 1
Medical diagnosis, interpolation, Bézier curves, B-splines, CUDA parallel computing technology.

1. Introduction

Medical diagnostics are advancing each year, reaching new heights of precision through emerging technologies [1-3]. However, the challenge of medical data loss during collection or transmission persists, leading to difficulties or even rendering subsequent analysis infeasible. This underscores the need for exploring various approaches to restore lost medical data for further processing and utilization. One such approach is interpolation, which facilitates the recovery of incomplete data based on known values, thereby generating more accurate forecasts.

Particularly, interpolation finds application in various domains including Geodesy and Cartography (for constructing maps and Earth surface models using geodetic data) [4]; Physics (for approximating and constructing smooth functions describing experimental data) [5-8]; Computer Graphics (for building smooth curves and surfaces utilized in animation and 3D modeling) [9]; Finance (for market prediction and data analysis) [10, 11]; and Medicine (interpolation is often used for analyzing and processing medical images) [12].

Different types of interpolation methods exist: linear interpolation, nearest-neighbor method, cubic spline interpolation method, Lanczos interpolation, shape-preserving method, thin plate spline method, biharmonic interpolation method, Bézier curves, and B-splines. In [13], the author conducts research on and compares various interpolation methods. Unlike this study, our research is conducted on a medical dataset, whereas the examined article deals with image interpolation.

IDDM'2023: 6th International Conference on Informatics & Data-Driven Medicine, November 17 - 19, 2023, Bratislava, Slovakia
EMAIL: lesia.i.mochurad@lpnu.ua (L. Mochurad); yuliannamochurad@gmail.ua (Yu. Mochurad)
ORCID: 0000-0002-4957-1512 (L. Mochurad); 0009-0008-4486-7139 (Yu. Mochurad)

Article [14] describes the Data Analytics Challenge on Missing data Imputation (DACMI), which introduced a joint clinical dataset with known outcomes to assess and develop advanced methods for filling missing data in clinical time series. The dataset involved thirteen commonly measured blood laboratory indicators. Methods such as Recurrent Neural Network (RNN) combined with Multilayer Perceptron (MLP), weighted averaging from K-Nearest Neighbors (KNN), linear interpolation, piecewise-linear interpolation, nonlinear extension of Multivariate Imputation by Chained Equations (MICE), among others, were employed to evaluate the effectiveness of missing data imputation. Notably, KNN-based methods outperformed 3D-MICE, highlighting limitations in relying solely on a restricted search for similar patients when imputing missing values.

In [15], a new method for data imputation using artificial neural networks is proposed. Despite its high accuracy, it demands significant time for training, particularly in the analysis of high-dimensional datasets.

Based on the analysis of available literature, the use of B-splines and Bézier curves in medical data interpolation is relatively rare. This indicates the necessity for further in-depth investigation and performance comparison of these methods in the context of medical data. Additionally, it's essential for the data restoration phase to be swift and minimally affect the overall analysis time in more complex tasks involving medium and large-sized data. Thus, the study of parallelization processes to optimize interpolation methods becomes pertinent.

Article [16] proposes a parallel version of the cross-interpolation algorithm for calculating high-dimensional integrals, particularly those arising in the Ising model in quantum physics. This algorithm differs from traditional methods like Monte Carlo and quasi-Monte Carlo, using computation results on individual dimensions instead of random points. The authors demonstrate the algorithm's high efficiency, especially with strongly superlinear convergence. Due to its flexibility and potential for parallel execution across processes, the algorithm is deemed beneficial for high-dimensional integration tasks in various fields including statistics, probability, and quantum physics.

In [17], the authors explore the application of B-spline interpolation in the medical field. This method enables the reconstruction of three-dimensional objects derived from medical images and is used for image-guided surgery (IGS) navigation and medical image registration tasks. Given the complexity and data-intensive nature of IGS tasks, the authors propose utilizing graphical processing units (GPUs) for parallelization. A novel B-spline interpolation method on GPUs is introduced, minimizing data transfer between memory and computational cores during input mesh loading. Trilinear interpolation is also employed to reduce computational complexity and enhance method accuracy. The authors integrate the method into a liver deformation compensation workflow, demonstrating a 6.5-fold interpolation acceleration on their dataset using GPU parallelization. The goal of our work is to develop, analyze, and compare parallel interpolation algorithms using Bézier curves and B-splines for the restoration of medical data. To achieve this objective, we will conduct a study of interpolation algorithms, perform experiments to compare the effectiveness of Bézier curves and B-splines in medical data restoration, and explore the possibility of optimizing these methods through parallelization to enhance computational speed and result accuracy.

The research task can be outlined as follows: Let us consider an input dataset, denoted as D –a matrix comprising n rows and s columns. This dataset contains missing values. The aim is to determine the values of the missing elements within the matrix D using Bézier curve and B-spline interpolation. For each missing value to be restored, an interpolation function is employed, based on known values of neighboring points.

It is of paramount importance to rapidly restore missing data, as the resulting dataset D' will subsequently be utilized for the analysis and prediction of heart attacks.

2. Methodic and materials

2.1. Dataset description

For this study, the 'Heart Failure Prediction Dataset' [18] was utilized, consisting of 918 rows and 12 columns. Cardiovascular diseases (CVD) are the number one cause of death worldwide, claiming approximately 17.9 million lives each year, representing 31% of all global deaths. Four out of five CVD-related deaths are caused by heart attacks and strokes, with one-third of these deaths occurring prematurely in individuals under 70 years of age. Heart failure is a prevalent event resulting from cardiovascular diseases, and this dataset comprises 11 features that can be employed to predict possible heart conditions.

Features:

- Age: patient's age [years];
- Sex: patient's gender [M: male, F: female];
- ChestPainType: type of chest pain [TA: typical angina, ATA: atypical angina, NAP: non-anginal pain, ASY: asymptomatic];
- BloodPressure: resting blood pressure [mm Hg];
- Cholesterol: serum cholesterol [mg/dl];
- FastingBS: fasting blood sugar level [1: if FastingBS > 120 mg/dl, 0: otherwise];
- RestingECG: resting electrocardiogram results [Normal: normal, ST: ST-T wave abnormality (T wave inversion and/or ST elevation or depression > 0.05 mV), LVH: possible or definite left ventricular hypertrophy by Estes' criteria];
- MaxHR: maximum heart rate achieved [numerical value from 60 to 202];
- ExerciseAngina: exercise-induced angina [Y: Yes, N: No];
- Oldpeak: oldpeak = ST [numerical value measured in depression];
- ST_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping];
- HeartDisease: output class [1: heart disease, 0: normal].

This dataset does not contain missing data, so we create missing values as follows:

```
Replacing random values with NaN
for col in cols:
mask = np.random.rand(len(origin)) < 0.2
origin.loc[mask, col] = np.nan
```

As a result, we will have two datasets, one loaded without missing values, and the other one generated by us for interpolation purposes. Having a dataset without missing values will help us analyze the reliability and advantages of using the proposed interpolation algorithms more effectively.

2.2 B-Splines

B-Splines are integral components of many algorithms used in computer graphics, signal processing, image processing, and various other fields of science and technology [19]. B-Splines can be constructed using a recursive formula, dependent on their degree and the number of control points. Let's consider a vector of control points $\boldsymbol{t_i}$, $\boldsymbol{i = \overline{0,\ m}}$, and a function $\boldsymbol{y = f(x)}$ that we wish to interpolate. The process of B-Spline interpolation involves constructing B-Spline basis functions $\boldsymbol{B_{i,k}(x)}$, where $\boldsymbol{k}$ is the degree, satisfying the conditions:

- each basis function $B_{i,k}(x)$ is defined over the interval $[t_i,\ t_{i+k+1}]$;
- $\boldsymbol{B_{i,k}(x)}$ is a polynomial of degree k within each interval $[\boldsymbol{t_i},\ \boldsymbol{t_{i+k+1}}]$;
- each point $\boldsymbol{x}$ lies within at most $\boldsymbol{k+1}$ neighboring basis functions.

B-Spline basis functions can be computed recursively using formulas (1)-(2), which define $\boldsymbol{k}$ basis functions in terms of degree $\boldsymbol{k-1}$ basis functions:

$$B_{i,0}(x) = \begin{cases} 1, & t_i \le x \le t_{i+1} \\ 0, & else.' \end{cases} \tag{1}$$

$$B_{i,k}(x) = \frac{x - x_i}{t_{i+k} - t_i} B_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} B_{i+1,k-1}(x). \tag{2}$$

Further, we will formulate a general algorithm for finding missing values in the dataset:

1. Input: $\boldsymbol{D}$ (dataset with missing values).
2. For each of the $\boldsymbol{s}$ columns in the dataset:
   - Find those n rows (index $\boldsymbol{i}$) for which there are missing values in the column.
   - For each of the $\boldsymbol{n}$ rows with missing values:
     - Find the nearest preceding row with a value (index $\boldsymbol{i-1}$) and the nearest succeeding row with a value (index $\boldsymbol{i+1}$).

    - Record the values from these rows as $\boldsymbol{x\_1}$ and $\boldsymbol{x\_2}$, respectively.
    - Record the corresponding y values, $\boldsymbol{y\_1}$ and $\boldsymbol{y\_2}$.
    - Calculate B-Spline coefficients.
    - Compute the interpolated value using B-Spline coefficients and $\boldsymbol{x}$ values.
    - Replace the missing value in the dataset with the interpolated value.
- Output the updated dataset $\boldsymbol{D'}$ with filled missing values.

Description of the parallelization algorithm using CUDA [20] and Python:

1. Set up and configure the CUDA-compatible environment on the respective computing device (GPU).
2. Load the dataset into the device's memory (GPU).
3. Create a CUDA kernel for B-Spline computation.
4. Divide the dataset into blocks for parallel processing.
5. Launch the CUDA kernel concurrently for each data block, computing B-Spline for each missing value.
6. Store the interpolated values on the device (GPU).
7. Load the updated values from the device (GPU) back to the host memory (CPU).
8. Replace missing values in the dataset with updated values.
9. Output the updated dataset with filled missing values.

In this pseudocode, functions from the PyCUDA library are used for interaction with CUDA. The parallelization process involves computing B-Spline for each missing value on the GPU. Each missing value is processed individually in its own computation thread.

The main function 'interpolate_missing_values' performs the following steps:

1. Identifies the indices of rows with missing values in the dataset.
2. Creates and copies data to the device (GPU).
3. Defines block sizes and grids for launching the CUDA kernel.
4. Launches the CUDA kernel 'compute_spline' in parallel for each data block.
5. Loads interpolated values from the device back to the host (CPU).
6. Replaces missing values in the dataset with the updated values.
7. Returns the filled dataset.

Evaluation of the computational complexity of each algorithm stage:

1. Identification of rows with missing values: This stage requires iterating through all rows in the dataset, thus having a complexity of $\boldsymbol{O(n)}$.
2. Finding nearest preceding and succeeding rows with values: This stage also requires iterating through all rows in the dataset, resulting in a complexity of $\boldsymbol{O(n)}$.
3. Calculation of B-Spline coefficients: This stage doesn't require iterations over all rows, but for each missing value, coefficients need to be calculated. Considering this, the complexity of this stage can be estimated as $\boldsymbol{O(1)}$.
4. Computation of interpolated values: This stage also has a complexity of $\boldsymbol{O(1)}$ since it involves basic mathematical operations on B-Spline coefficients and x values.
5. Replacement of missing values with interpolated values: This stage also has a complexity of $\boldsymbol{O(n)}$, as it requires updating corresponding values in the dataset.

Therefore, the overall computational complexity of the algorithm for one iteration of B-Spline interpolation for one dataset column is estimated to be $\boldsymbol{O(n)}$.

For the entire dataset containing n rows, the total computational complexity of the algorithm will depend on the number of missing values in the columns where B-Spline interpolation is performed. If we denote p as the count of missing values in the dataset, the total computational complexity of the algorithm for the entire dataset will be $\boldsymbol{O(n \cdot p)}$. This is due to the fact that each missing value requires performing an iteration of B-Spline interpolation, resulting in a complexity of $\boldsymbol{O(n)}$.

2.3 Bezier curve

A Bézier curve of degree k is defined by the formula:

$$R(t) = \sum_{i=0}^{k} B_i^k\,(t) P_i,\ t \in [0, 1], \tag{3}$$

where $B_i^k(t) = C_k^i t^i (1-t)^{k-i}$ – Bézier curve basis functions, also known as Bernstein polynomials, $C_k^i = \frac{k!}{i!(k-i)!}$, $P_i = \binom{x_i}{y_i}$.

Some of the properties of Bernstein polynomials significantly affect the behavior of Bézier curves. Let's mention the main ones: Bernstein polynomials take non-negative values; the sum of Bernstein polynomials yields one, satisfying condition (2); Bernstein polynomials do not depend on the vertices of the array P but only on the number of points in it.

$$\sum_{i=0}^{k} B_i^k(t) = 1 \tag{4}$$

In scalar form, equation (3) is written as follows:

$$x(t) = \sum_{i=0}^{k} C_k^i\, t^i (1-t)^{k-i} x_i;\ y(t) = \sum_{i=0}^{k} C_k^i\, t^i (1-t)^{k-i} y_i \tag{5}$$

The general algorithm for finding missing values in the dataset based on Bézier curves and its parallelization using CUDA is analogous to the use of B-splines. As a result, the computational complexity estimation of the proposed algorithm is as follows:

1. Reading the dataset has a complexity of $O(n \cdot s)$, where n is the number of rows, and s is the number of columns.
2. Iterating over each column to find missing values has a complexity of $O(s)$.
3. For each missing value, searching for neighboring rows, computing intermediate points, and replacing the missing value have a complexity of $O(1)$.
4. The overall computational complexity of the algorithm for a single CUDA kernel is $O(s)$.
5. The overall computational complexity of the algorithm with parallelization depends on the number of CUDA kernels. If we have p CUDA kernels, the complexity becomes $O(\frac{s}{p} + p)$.

Therefore, the total computational complexity of the algorithm for the entire dataset with parallelization using p processes or CUDA kernels is $O(n \cdot s + \frac{s}{p} + p)$.

3. Numerous experiments

For the analysis and comparison of interpolation results in the study, the following metrics were utilized:

- Mean Absolute Error (MAE): This metric measures the average absolute difference between interpolated values and original values. A lower MAE indicates better performance.
- Coefficient of Determination (R-squared, R^2): This metric measures the proportion of the variance in the original values that can be explained by the interpolated model. R^2 values range from 0 to 1, where a value of 1 indicates a perfect fit. A higher R^2 value indicates a better fit.

These metrics provide insights into the quality and accuracy of the interpolation methods used. A lower MAE and a higher R^2 value are indicative of more accurate interpolation and better model fit.

Table 1

The values of the metrics for the utilized interpolation methods are as follows

metrics	B-Spline interpolation	Bezier curve interpolation
MAE	0.19019	0.09976
R^2	0.8664	0.9259

As evident from Table 1, Bezier curve interpolation yields better results than B-spline interpolation. Based on these values, one can conclude that the Bezier curve interpolation model can explain approximately 92.6% of the variation between interpolated and original values. This suggests that the model fits the data quite well and performs interpolation effectively.

Since the dataset considered in the study is created for predicting heart attacks, it is important to conduct post-interpolation forecasting to ensure that interpolation doesn't compromise prediction accuracy.

For the purpose of comparison and visualization of prediction results, the following metrics are used:

- Accuracy: The ratio of correctly classified samples to the total number of samples in the test set. This metric indicates how accurately the model predicts classes.
- Precision: The ratio of correctly classified positive samples to the total number of predicted positive samples. This metric measures how well the model identifies positive classes.
- Recall: The ratio of correctly classified positive samples to the total number of actual positive samples in the test set. This metric shows how well the model detects positive classes.
- F1-score: The harmonic mean of precision and recall. This metric is used to assess model accuracy in cases of class imbalance.
- ROC AUC: A metric that measures the effectiveness of the model at different binarization thresholds. The ROC curve illustrates the relationship between the true positive rate and false positive rate. A higher AUC indicates a better model.

Without loss of generality, logistic regression is used for prediction. It is relatively simple to implement and performs well for binary classification tasks.

Table 2

Metrics Values Before and After Interpolation Prediction

Metrics	Without Interpolation	Based on Interpolation	
		Using B-spline Interpolation	Using Bezier Curves
Accuracy	0.8432	0.8745	0.8691
Precision	0.8571	0.8712	0.8664
Recall	0.8532	0.8695	0.8641
F1-score	0.8539	0.8699	0.8646
AUC-ROC	0.9257	0.9392	0.9283

Based on the results from Table 2, the following conclusions can be drawn:

1. Prediction without prior interpolation has the lowest values for all metrics, indicating that a model not accounting for missing values is not accurate enough.
2. Prediction using Bezier curves yields slightly lower AUC-ROC values than the model with B-spline interpolation. Bezier curves are interpolation curves that use control points to model curve dependencies in the data.

Therefore, interpolation of missing values can enhance the quality of prediction models, and it's preferable to use more complex interpolation methods like B-splines and Bezier curves to achieve better results.

For a more objective assessment, additional research should be conducted on different datasets using alternative prediction methods. However, it's important to note that the use of the proposed algorithms does not worsen the accuracy of predicted data; in some cases, it even improves it.

To determine acceleration factors, the following formula will be utilized:

$$S = T\_seq / T\_par, \tag{6}$$

where ***T_seq*** is the execution time of the sequential version of the algorithm, and ***T_par*** is the execution time of the parallel version.

Table 3

Execution Time of Interpolation with and without Parallelization and Corresponding Acceleration Metrics of the Proposed Algorithm

Metrics	CPU		GPU	
	B-spline itnerpolation	Bezier Curve Interpolation	B-spline itnerpolation	Bezier Curve Interpolation
Time,seconds	0.76152	7.1316	0.028451	0.073027
	S		11.45265	97.65703

Based on the results presented in Table 3, the obtained findings indicate significant acceleration of computations using CUDA compared to the conventional sequential algorithm execution. The utilization of parallel computations on GPU enables the simultaneous utilization of multiple cores, resulting in accelerated calculations.

The achieved outcomes reveal substantial potential for parallelization using CUDA to enhance the computational efficiency of B-spline and Bezier curve interpolations. Leveraging GPUs can markedly amplify computation speed and improve interpolation accuracy. These results underscore the significance of further investigations into the efficiency of parallel computation algorithms and their practical applicability, especially within the realm of medical data processing [21-23].

4. Conclusions and future research

A majority of medical datasets suffer from missing information, which can lead to significant errors in data processing and forecasting analyses. This challenge is particularly relevant for cardiology systems aiming to predict cardiovascular diseases. These datasets encompass a plethora of physiological indicators derived from periodic examinations and studies, including arterial pressure, serum cholesterol, blood glucose levels, and more. The adequacy of interpreting such data in the presence of missing values remains questionable.

This work explored one approach to addressing the problem of missing data recovery, specifically investigating interpolation methods based on B-splines and Bezier curves. Ensuring that the preprocessing of data does not compromise the accuracy of subsequent predictions is of paramount importance. This assertion is convincingly supported by a series of experiments employing the proposed algorithms, where not only is the accuracy maintained but sometimes even improved. Another requirement when dealing with missing data is to maximize the acceleration of this process, so that the data preprocessing does not unduly impact the overall forecasting time. The proposed solution in this study, employing parallel computation technology CUDA, has significantly accelerated the process of data imputation.

B-spline interpolation presents a viable option for medical data, as it ensures smoother approximation and reduced data distortion. B-splines yield more accurate results, especially when substantial gaps exist between neighboring data points. However, it should be noted that B-splines might lead to excessive smoothness or overfitting, which can impact the precision of reproduction.

Bezier curve interpolation demonstrated superior results. It provides flexibility and yields smooth outcomes. Nevertheless, employing Bezier curves might require greater computational power and time compared to the previous method.

For further research in this domain, it is recommended to conduct additional investigations and validations to determine the optimal interpolation method for specific medical data, accounting for the peculiarities of the particular application.

5. Acknowledgements

The authors would like to thank the Armed Forces of Ukraine for providing security to perform this work. This work has become possible only because of the resilience and courage of the Ukrainian Army.

6. References

[1] L. Mochurad, R, Panto. A Parallel Algorithm for the Detection of Eye Disease, In: Hu, Z., Wang, Y., He, M. (eds) Advances in Intelligent Systems, Computer Science and Digital Economics IV. CSDEIS (2022). Lecture Notes on Data Engineering and Communications Technologies, vol 158. Springer, Cham. doi:10.1007/978-3-031-24475-9_10.

[2] L. Mochurad, Ya.Hladun, Modeling of Psychomotor Reactions of a Person Based on Modification of the Tapping Test, International Journal of Computing, (2021), 20(2), 190-200. doi: 10.47839/ijc.20.2.2166.

[3] Ya. Tolstyak, M. Havryliuk, An Assessment of the Transplant's Survival Level for Recipients after Kidney Transplantations using Cox Proportional-Hazards Model. Proceedings of the 5th

International Conference on Informatics & Data-Driven Medicine, Lyon, France, November 18 - 20, CEUR-WS.org, (2022), 260-265.

[4] X. Zhu, K.C. Thompson, T.J. Martínez, Geodesic interpolation for reaction pathways, J. Chem. Phys. 2019 Apr 28;150(16):164103. doi: 10.1063/1.5090303.

[5] L. Zhou, Sh. Ren, G. Meng, Zh. Ma, Node-based smoothed radial point interpolation method for electromagnetic-thermal coupled analysis, Applied Mathematical Modelling, Volume 78 (2020) 841-862. doi: 10.1016/j.apm.2019.09.047.

[6] S. Ljaskovska, Y. Martyn, I. Malets, O. Velyka.Optimization of parameters of technological processes means of the flexsim simulation program. Proceedings of the 2020 IEEE 3rd International Conference on Data Stream Mining and Processing, DSMP 2020, (2020): 391–397, 9204029.

[7] S. Liaskovska, Y. Martyn. Investigation of Anomalous Situations in the Machine-Building Industry Using Phase Trajectories Method. Lecture Notes in Networks and Systemsthis link is disabled, 463, (2022): 49–59.

[8] Y. Martyn, S. Liaskovska, M. Gregus, O. Velyka. Optimization of Technological's Processes Industry 4.0 Parameters for Details Manufacturing via Stamping: Rules of Queuing Systems. Procedia Computer Science, 191, (2021): 290–295.

[9] G. Legnani, I. Fassi, A. Tasora, D. Fusai, A practical algorithm for smooth interpolation between different angular positions, Mechanism and Machine Theory, Volume 162 (2021):104341. doi:10.1016/j.mechmachtheory.2021.104341.

[10] S. Fujimoto, T. Mizuno, A. Ishikawa, Interpolation of non-random missing values in financial statements' big data using CatBoost, Journal of Computational Social Science, 5 (2022):1281–1301. doi: 10.1007/s42001-022-00165-9.

[11] M. Havryliuk, I. Dumyn, O. Vovk, Extraction of Structural Elements of the Text Using Pragmatic Features for the Nomenclature of Cases Verification. In: Hu, Z., Wang, Y., He, M. (eds) Advances in Intelligent Systems, Computer Science and Digital Economics. Lecture Notes on Data Engineering and Communications Technologies, 158(2023):703–711. doi: 10.1007/978-3-031-24475-9_57.

[12] Yuan Luo, Evaluating the state of the art in missing data imputation for clinical data, Briefings in Bioinformatics, Volume 23, Issue 1, (2022), bbab489. doi: 10.1093/bib/bbab489.

[13] Shreyas Fadnavis. Image Interpolation Techniques in Digital Image Processing: An Overview, International Journal of Engineering Research and Applications, (2014), 4(10):2248-962270.

[14] Yuan Luo, Evaluating the state of the art in missing data imputation for clinical data, Briefings in Bioinformatics, Volume 23, Issue 1, January 2022, bbab489. doi:10.1093/bib/bbab489.

[15] I. Izonin, R. Tkachenko, V. Verhun, Kh. Zub, An approach towards missing data management using improved GRNN-SGTM ensemble method, Engineering Science and Technology, an International Journal, (2021), 24(3):749-759. doi:10.1016/j.jestch.2020.10.005.

[16] S. Dolgov, D. Savostyanov, Parallel cross interpolation for high-precision calculation of high-dimensional integrals, Computer Physics Communications, Volume 246, 2020, 106869. doi:10.1016/j.cpc.2019.106869.

[17] O. Zachariadis, A. Teatini, . et al., Accelerating B-spline interpolation on GPUs: Application to medical image registration, Computer Methods and Programs in Biomedicine, Volume 193, (2020), 105431. doi:10.1016/j.cmpb.2020.105431.

[18] Heart Failure Prediction Dataset. Available online: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction (accessed on 13August 2023).

[19] J. Wang, J. Kosinka, A. Telea, Spline-Based Dense Medial Descriptors for Lossy Image Compression, J. Imaging, (2021), 7, 153. doi:10.3390/jimaging7080153.

[20] R.S. Dehal, C. Munjal, A.A. Ansari and A.S. Kushwaha, GPU Computing Revolution: CUDA, 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), Greater Noida, India, (2018), 197-201. doi: 10.1109/ICACCCN.2018.8748495.

[21] S. Dash, S.K. Shakyawar, M. Sharma et al. Big data in healthcare: management, analysis and future prospects, J. Big Data 6, 54 (2019). doi:10.1186/s40537-019-0217-0.

[22] R PraveenKumar, Medical Data Processing and Prediction of Future Health Condition Using Sensors Data Mining Techniques and R Programming, International Journal of Scientific Research and Engineering Development, (2020), 3(4), 9 p. Available at SSRN: https://ssrn.com/abstract=3686347.

[23] Ya. Tolstyak, V. Chopyak, M. Havryliuk, An investigation of the primary immunosuppressive therapy's influence on kidney transplant survival at one month after transplantation, Transplant Immunology, (2023). doi:10.1016/j.trim.2023.101832.