


Article

Analysis of G-Transformation Modes for Building Neuro-like Parallel–Hierarchical Network Identification of Rail Surface Defects

Vaidas Lukoševičius ^{1,*} , Volodymyr Tverdomed ^{2,*}, Leonid Tymchenko ², Natalia Kokriatska ², Yurii Didenko ², Mariia Demchenko ² and Olena Oliynyk ²

¹ Department of Transport Engineering, Faculty of Mechanical Engineering and Design, Kaunas University of Technology, Studentų Str. 56, 44249 Kaunas, Lithuania

² Kyiv Institute of Railway Transport, State University of Infrastructure and Technology, Kyrylivska Str. 9, 04071 Kyiv, Ukraine; tumchenko_li@gsuite.uit.edu.ua (L.T.); kokriatska_ni@gsuite.uit.edu.ua (N.K.); didenko_yuv@gsuite.uit.edu.ua (Y.D.); sintyurova_me@gsuite.uit.edu.ua (M.D.); oliynuk_oa@gsuite.uit.edu.ua (O.O.)

* Correspondence: vaidas.lukosevicius@ktu.lt (V.L.); tverdomed@gsuite.uit.edu.ua (V.T.)

Abstract: This work presents the construction of a transformation for the identification of surface defects on rails, starting with the selection of elements from the matrix and the creation of different matrices. It further elaborates on the recursive formulation of the transformation and demonstrates that, regardless of the elements' uniqueness, the sum of the transformed matrix remains equal to the sum of the original matrix. This study also addresses the handling of matrices with repeated elements and proves that the G-transformation preserves information, ensuring the integrity of data without any loss or redundancy.

Keywords: transformations; parallel–hierarchical networks; parallelism; signal processing; neural networks; rail surface defects

MSC: 93C57



Academic Editors: Qun Chen, Haibo Chen and Lianbo Deng

Received: 27 February 2025

Revised: 9 March 2025

Accepted: 13 March 2025

Published: 14 March 2025

Citation: Lukoševičius, V.; Tverdomed, V.; Tymchenko, L.; Kokriatska, N.; Didenko, Y.; Demchenko, M.; Oliynyk, O. Analysis of G-Transformation Modes for Building Neuro-like Parallel–Hierarchical Network Identification of Rail Surface Defects. *Mathematics* **2025**, *13*, 966. <https://doi.org/10.3390/math13060966>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Modern signal processing is characterized by increasing interaction between several fields of science and technology: signal analysis, systems theory, and numerical analysis. In each of these areas, traditional solutions, once considered “optimal” within their domain, often require re-evaluation. As shown in numerous studies, the core computational procedures for solving most real-time signal processing tasks reduce to performing a set of matrix operations [1].

Currently, a new trend is emerging in the development of digital data processing techniques, and several specialized multifunctional computational devices are being introduced for military and industrial applications. However, most specialists in narrow fields of signal processing, such as image analysis, still prefer using general-purpose computing tools to develop their own software.

The ease of creating image analysis programs usually results in increased computation times on personal computers. For instance, even simple transformations requiring minimal calculations can take from tens of minutes to several hours of processor time on the best PCs. This creates a demand for new methods and tools featuring innovative algorithms and architectures to build specialized computing devices that require minimal computations [2–4].

the respective input for the next cortical zone. The term “homologous outputs” implies a multiple correlation process of temporal signal coincidence at these inputs.

Neural networks are an ideal mechanism capable of stable operation in uncertain conditions [5]. Neural networks that function based on the principle of dynamic multifunctionality incorporate interactions of convergent–divergent structures in both horizontal and vertical directions, forming a 3D architecture. In such a structure, the complexity of various convergent–divergent process schemes enables variations (genetically governed changes) in the trajectories of horizontal pathways. These pathways may also slightly adjust during the learning process.

To better understand the proposed neural network, let us draw some semantic analogies. Imagine a group of researchers jointly solving a specific scientific problem [6]. Each has its own knowledge base related to the issue. They express their judgments of the problem and arrive at a common conclusion, creating a matrix of judgments representing the first level of discussion.

At each subsequent level of problem solving, the first intermediate result of the discussion undergoes further refinement, forming a matrix of judgments. This refinement is performed whenever all judgments at a given moment in time converge to some approximation. This occurs when, from many divergent judgments, a general judgment is formed that satisfies all researchers. The intermediate results of the discussion are refined outcomes from the previous discussion level. All of them are decorrelated from the other (current) results of the discussion.

The overall result of the discussion is a sequential process of multistage refinement of the problem being solved and consists of individual intermediate judgments [3–5]. Therefore, the parallel–hierarchical process can be defined as the simultaneous analysis of a phenomenon (or object) by identifying hierarchies of increasingly effective representations of it.

An example of the semantic organization of this process is shown in Figure 2, where 1N, 2N, and 3N are the first, second, and third observers identifying a certain visual pattern, and $1_1 - 1_9$, $2_1 - 2_7$, $3_1 - 3_3$, and 4_1 are the results of visual scene identification.

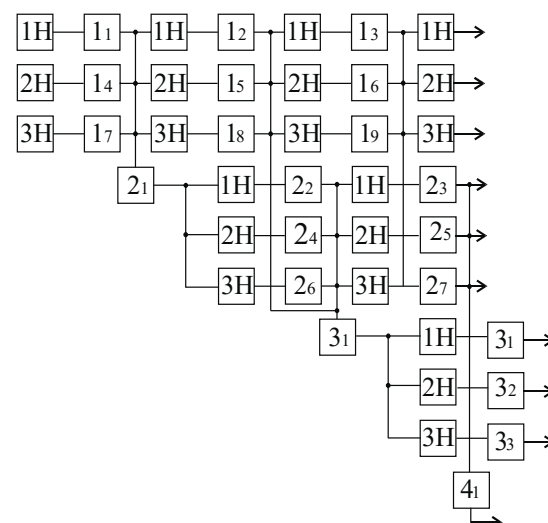


Figure 2. Example of a semantic parallel–hierarchical network.

The specific semantic content of the nodes in the formed network can be, for example, as follows: 1_1 —there is an object on the surface of the rail head; 1_2 —the object is located on the working face of the rail; 1_3 —the surface of the object has depressions; 1_4 —the object is mainly dark in color; 1_5 —there is an influx of metal on the working edge of the rail head;

1₆—the handicap of the object is complicated; 1₇—the object is not similar to a foreign object; 1₈—the surface of the object is different from the surface of the rail; 1₉—there are dark spots near the working edge of the rail head; 2₁—the object on the rail looks like a defect; 2₂—the object is stretched along the rail; 2₃—the length of the object is more than 25 mm; 2₄—there is a micro crack on the working edge of the rail head; 2₅—an unusual shape; 2₆—it seems to be about an object of an elongated shape; 2₇—the depth of irregularities on the object is more than 1 mm; 3₁—indentations are similar to metal exfoliation; 3₂—coloration is similar to a defect; 3₃—the object has metal inhomogeneity; 4₁—the object on the rail refers to the defect scraping of metal on the side working circle of the head due to insufficient contact–fatigue strength of the metal, which has a ripping depth of 1 mm and a ripping length of 25 mm.

In the examined example, it is clear that the proposed semantic AI network (Figure 2) is a semantic organization of a dynamic data structure consisting of nodes representing time-varying objects or concepts, as well as links that indicate the temporal relationship between the nodes.

Unlike known structures of semantic networks, here, it becomes clear how to represent a situation in the network as an exception to the rules. Analogous to the known mechanism of forming and storing information about an object in the form of complex packages called frames, in the proposed structure of the semantic network, information about the object is stored in hierarchically organized frames.

The final information about the object is stored in the terminal nodes of the frames, meaning, for our example, it would be 2₁, 3₁, and 4₁. Perhaps the most serious problem in effectively implementing the idea of multistage processing is the need to detect and use neuro-like mechanisms to analyze sensory information. For this, a comprehensive assessment of achievements in this area is required across various fields of knowledge—applied mathematics, computing technology, and neurobiology. Another equally important issue is the use of neural mechanisms to determine the spatiotemporal correlation of neural coalitions in neural networks that operate in real time. Moreover, each frame is described by its own functional series [7]. For example, for the examined case (Figure 2), frames are formed from the following network nodes:

First frame—1₁ → 1₄ → 1₇ → 2₁;

Second frame—1₂ → 1₅ → 1₈ → 2₂ → 2₄ → 2₆ → 3₁;

Third frame—1₃ → 1₆ → 1₉ → 2₃ → 2₅ → 2₇ → 3₁ → 3₂ → 3₃ → 4₁.

The operation of well-known neural network structures in real time is associated with significant difficulties. Research by many neurobiologists shows that the processes of visual perception and consciousness are distributed across multiple cortical areas. There are no data on any higher zones where all information from previous areas is merged. When considering the structure of artificial neural networks from the perspective of spatial organization, there is a clear contradiction with current views on natural networks [8].

These opportunities can be utilized by using the first level of the network as the input layer, where data are described by the transformation used, and the intermediate elements are used as hidden layers. The resulting (tail) elements form the output layer, resulting in the formation of a multistage 3D structure of the neural network. This network contains multiple input layers, hidden layers, and output layers. The computation result, in this case, is determined by the states of all input layers, which are decorrelated in the spatial–temporal domain from the states of hidden layers [9–12].

The network structure allows it to simulate the operation of a distributed computational PI network, and due to spatial separation over time, it processes information in a deterministic parallel–hierarchical network (Figure 3). The network algorithm is universal and consists of the parallel–hierarchical formation of sets of general and various states of

the spatial–temporal environment (PSE). The generalization of all types of information occurs in the final stage of the transformation.

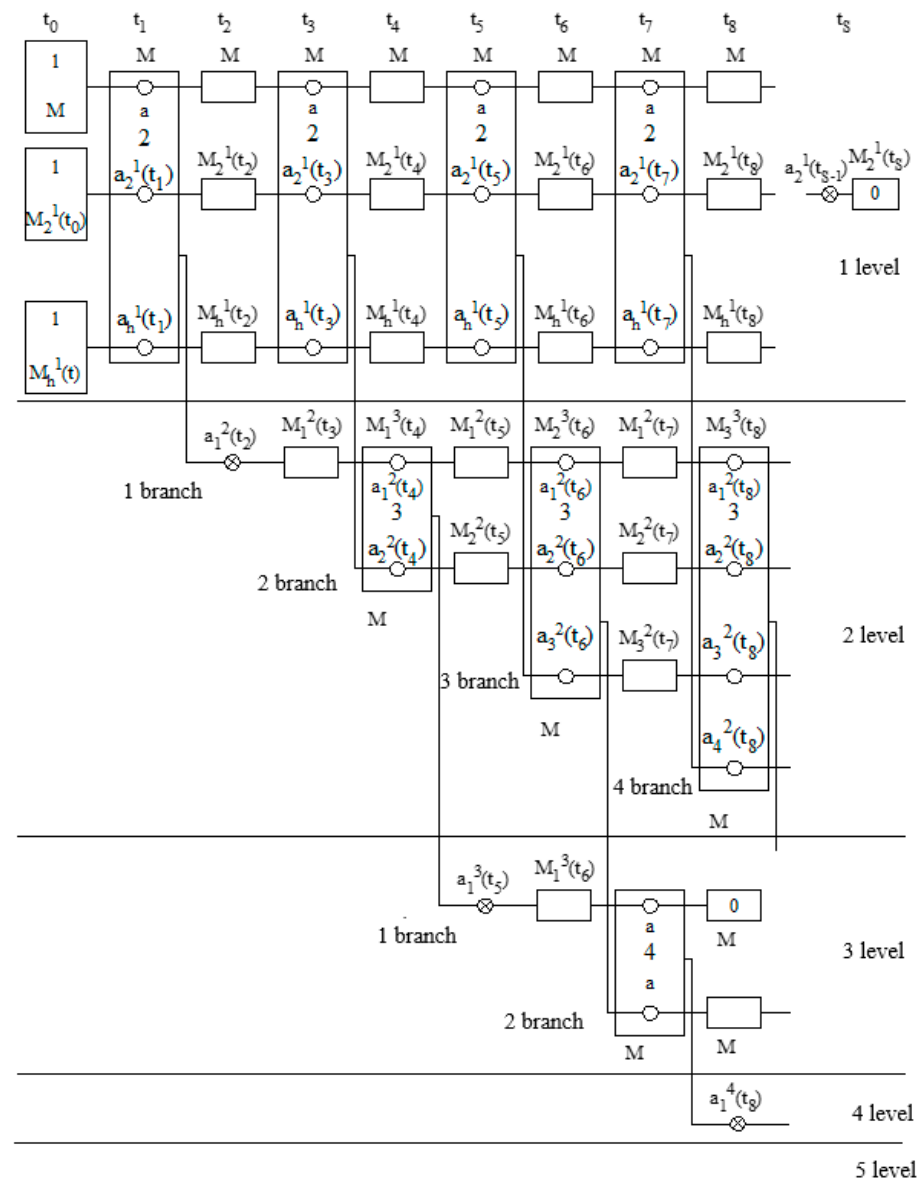


Figure 3. Graph of the computational PI network.

A graphical representation is convenient for illustrating the concepts of the algorithm for operating the computational PI network, where the structure consists of sets and elements. Accordingly, the network graph contains three types of nodes. In Figure 3, the rectangle symbol represents the initial set, the square represents the intermediate set, and the circle represents the element. Directed arcs connect sets and elements—some arcs point from sets to elements, and others point from elements to sets [10].

The arc directed from the set $M_j^i(t_k)$ to the element $a_j^i(t_{k+1})$ defines the criterion for selecting the element, while the arc from the element $a_j^i(t_{k+1})$ to the set $M_j^i(t_{k+2})$ indicates the transformation of the set. The arcs are directed, making this a directed graph, where the null set is denoted by the symbol.

This means that for each element of the graph, certain operations or transformations are defined depending on the selection of the element. This structure allows for the

construction of a hierarchical model where elements have clear directions and interactions based on specified criteria, as shown in Figure 3 [13].

Figure 3 allows us to conclude that there are elements such as $a_2^1(t_{s-1})$, $a_1^2(t_2)$, and $a_1^2(t_5)$. There are elements marked on the graph with the symbol. Each of these elements does not belong to any set, as it is uniquely selected at its level during the current cycle, does not participate in the further processing of arrays, and forms the result of the process. These elements are the terminal elements or tail elements of the computational PI network, and they generate the final output.

3. Experimental Part of G-Transformation for Computational Parallel–Hierarchical Networks

The basic concept and primary tool for constructing computational PI networks are the G-transformations. Using the G-transformation [1], which occurs at each branch of the PI network, the input column matrix is transformed into a band matrix, and an array of column matrices is transformed into an array of band matrices. After several applications of the G-transformation, the input data array is converted into an array of tail elements, which are much easier to work with.

It is necessary to address the question of information preservation during the G-transformation: whether the information is transmitted completely without loss or the emergence of redundant information [14].

Theorem 1. *The sum of the elements of any column matrix is equal to the sum of the elements of the G-transformation of that matrix.*

Case 1. *The matrix consists of elements that are pairwise distinct from one another.*

1. Representation of a column matrix.

Let us consider a column matrix consisting of a finite number (n) of elements:

$$\mu = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} \quad (1)$$

where a_1, a_2, \dots, a_n are elements of the matrix.

2. Renumbering of matrix μ .

Since we do not use a row order for arranging the elements of the matrix μ , the usual notation is a_1 —the first selected element, a_2 —the second, and so on, with a_n being the last. Thus, we obtain a matrix μ , where elements are arranged in a nondecreasing order:

$$\mu = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} \quad (2)$$

where the corresponding elements a_i are arranged in the order of their G-transformation [14–16].

3. Selection of elements of the matrix μ .

We select the first element a_i from matrix μ , then the second element from matrix μ , and so on:

$$G(\mu) = \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} - a_1 = \begin{pmatrix} a_1 - a_1 \\ a_2 - a_1 \\ \dots \\ a_n - a_1 \end{pmatrix} \quad (3)$$

From the obtained matrix, we select the second element $(a_2 - a_1)$ and write the matrix of differences between the elements of the obtained matrix with the element $(a_2 - a_1)$ [15]. At the positions of the elements with a value of 0, we place X:

$$G(\mu) = \begin{pmatrix} 0 \\ a_2 - a_1 \\ \dots \\ a_n - a_1 \end{pmatrix} - (a_2 - a_1) = \begin{pmatrix} X \\ 0 \\ \dots \\ a_n - a_2 \end{pmatrix} \quad (4)$$

We continue this procedure n times. In the final selection, we have the following:

$$G(\mu) = \begin{pmatrix} X \\ X \\ \dots \\ a_n - a_{n-1} \end{pmatrix} - (a_n - a_{n-1}) = \begin{pmatrix} X \\ X \\ \dots \\ 0 \end{pmatrix} \quad (5)$$

It should be noted that the elements of matrix μ are chosen arbitrarily during the construction of the G-transformation.

4. Construction of the row matrix for the G-transformation.

The first element of the G-transformation is formed by multiplying the value of the first selected element a_1 by the number n of nonzero elements. Similarly, the second element is the product of the value of the second selected element $(a_2 - a_1)$ and the number of nonzero elements $(n - 1)$, excluding the element a_1 , which is replaced with 0. The procedure continues for 0 iterations. The penultimate element of the row matrix for the G-transformation is $2 * (a_{n-1} - a_{n-2})$, and the last one is $(a_n - a_{n-1})$ [16–19]. It should be noted that the number of nonzero elements is determined under the condition that matrix μ consists of elements that differ pairwise in magnitude.

The case where matrix μ contains elements of equal magnitude requires a separate proof. Thus, the row matrix of the G-transformation has the following form:

$$G(\mu) = \left(na_1 \quad ((n-1)(a_2 - a_1)) \quad ((n-2)(a_3 - a_2)) \quad \dots \quad (2(a_{n-1} - a_{n-2})) \quad ((a_n - a_{n-1})) \right) \quad (6)$$

Each element of the G-transformation is formed using the recursive formula:

$$g_i = (n - (i - 1))(a_i - a_{i-1}) \quad (7)$$

We show that the sum of the elements of the column matrix μ is equal to the sum of the elements of the row matrix obtained from the G-transformation [18]. The sum of the elements of the column matrix is as follows:

$$\Sigma_{\mu} = \sum_{i=1}^n a_i \quad (8)$$

The sum of the elements of the row matrix from the G-transformation is as follows:

$$\Sigma_{G(\mu)} = na_1 + (n-1)(a_2 - a_1) + (n-2)(a_3 - a_2) + \dots + 2(a_{n-1} - a_{n-2}) + (a_n - a_{n-1}) \quad (9)$$

Expanding and grouping elements a_i with the same indices gives

$$\begin{aligned} \Sigma_{G(\mu)} = & (n - (n-1))a_1 + ((n-1) - (n-2))a_2 + ((n-2) - (n-3))a_3 + \dots + (3-2)a_{n-2} + (2-1)a_{n-1} \\ & + a_n + ((n-2) - (n-3))a_3 + \dots + (3-2)a_{n-2} + (2-1)a_{n-1} + a_n \end{aligned} \quad (10)$$

From this, it is evident that each element a_i appears exactly once. Thus,

$$\Sigma_{G(\mu)} = \sum_{i=1}^n a_i = \Sigma_{\mu} \quad (11)$$

Case 2. The matrix μ contains some identical elements.

1. Representation of the column matrix μ .

Let us consider a column matrix μ consisting of n elements and containing k identical elements, where $1 \leq k \leq n$.

2. Renumbering the elements of matrix μ .

Since the order of elements in matrix μ is not utilized in the construction of the G-transformation, we renumber matrix μ based on the order in which its elements are selected for the G-transformation. In this case, identical elements are all selected together during the first selection of any one of them. Let the first of these identical elements be selected on the j -th step $1 \leq j \leq n$. Then, the k elements $a_j, a_{j+1}, \dots, a_{j+k-1}$ are all identical. Furthermore, the condition $j+k-1 \leq n$ is imposed to ensure valid indexing. Then, matrix μ has the following look:

$$\mu = \begin{pmatrix} a_1 \\ \dots \\ a_j \\ \dots \\ a_{j+k-1} \\ a_{j+k} \\ \dots \\ a_n \end{pmatrix} \quad (12)$$

3. Selection of elements.

We select the first element a_1 and record the matrix of differences for each element with it:

$$G(\mu) = \begin{pmatrix} a_1 \\ \dots \\ a_j \\ \dots \\ a_{j+k-1} \\ a_{j+k} \\ \dots \\ a_n \end{pmatrix} - a_1 = \begin{pmatrix} 0 \\ \dots \\ a_j - a_1 \\ \dots \\ a_{j+k-1} - a_1 \\ a_{j+k} - a_1 \\ \dots \\ a_n - a_1 \end{pmatrix} \quad (13)$$

We continue this procedure for $j - 1$ steps. On the j -th step, we have the following:

$$G(\mu) = \begin{pmatrix} X \\ \dots \\ a_j - a_{j-1} \\ \dots \\ a_{j+k-1} - a_{j-1} \\ a_{j+k} - a_{j-1} \\ \dots \\ a_n - a_{j-1} \end{pmatrix} - (a_j - a_{j-1}) = \begin{pmatrix} X \\ \dots \\ 0 \\ \dots \\ 0 \\ a_{j+k} - a_j \\ \dots \\ a_n - a_j \end{pmatrix} \quad (14)$$

The next step involves selecting an element $a_{j+k} - a_j = a_{j+k} - a_{j+k-1}$:

$$G(\mu) = \begin{pmatrix} X \\ \dots \\ X \\ \dots \\ X \\ a_{j+k} - a_{j+k-1} \\ \dots \\ a_n - a_{j+k-1} \end{pmatrix} - (a_{j+k} - a_{j+k-1}) = \begin{pmatrix} X \\ \dots \\ X \\ \dots \\ X \\ 0 \\ \dots \\ a_n - a_{j+k} \end{pmatrix} \quad (15)$$

The next step is to select the last element:

$$G(\mu) = \begin{pmatrix} X \\ \dots \\ X \\ \dots \\ X \\ X \\ \dots \\ a_n - a_{n-1} \end{pmatrix} - (a_n - a_{n-1}) = \begin{pmatrix} X \\ \dots \\ X \\ \dots \\ X \\ X \\ \dots \\ 0 \end{pmatrix} \quad (16)$$

4. Construction of G-transformation for PI networks.

The first element of the G-transformation is formed by multiplying the first selected element a_i of matrix μ by the number of nonzero elements. Similarly, all elements of the G-transformation are formed up to the j -th element, which is formed by multiplying the element $a_j - a_{j-1}$ of matrix μ located in the j -th place by the number of $n - (j - 1)$ nonzero elements [20]. Along with the selected element, which was in the j -th place, all elements equal to it in magnitude are selected. Therefore, the number of remaining elements decreases not by the 1 element, but by the k equal elements. The next element of the G-transformation is the product of the elements of matrix μ at the $a_{j+k} - a_{j+k-1}$ place by the number $n - (j + k - 1)$ of the remaining elements [21,22]. The last element is the product of $a_n - a_{n-1}$ and the number of $n - (n - 1)$. Thus, the band matrix of the G-transformation looks as follows:

$$G(\mu) = (na_1 \dots (n - (j - 1))(a_j - a_{j-1}) \dots (n - (j + k - 1))(a_{j+k} - a_{j+k-1}) \dots a_n - a_{n-1}) \quad (17)$$

Each element of the G-transformation is formed according to the following recurrence formula:

$$g_i = (n - (i - 1))(a_i - a_{i-1}) \quad (18)$$

which is identical to the recurrence formula for the elements of the G-transformation of a matrix with all pairwise distinct elements [21,23]. The sum of the elements of the column matrix μ is as follows:

$$\Sigma_{\mu} = \sum_{i=1}^n a_i \quad (19)$$

The sum of the elements of the band matrix G-transformation is as follows:

$$\Sigma_{G(\mu)} = \sum_{i=1}^j \sum_{i=j+k}^n (n - (i - 1))(a_i - a_{i-1}) \quad (20)$$

In the G-transformation, the elements are reduced by a quantity of $k - 1$. Let us consider their sum without the reduction (which equals the sum of the elements of the matrix μ) and prove that it is equal to the sum of the elements of the newly formed G-transformation [24]. We split the sum without reduction into three sums:

$$\sum_{i=1}^n (n - (i - 1))(a_i - a_{i-1}) = \sum_{i=1}^j \sum_{i=j+1}^{j+k-1} \sum_{i=j+k}^n (n - (i - 1))(a_i - a_{i-1}) \quad (21)$$

Obviously,

$$\sum_{i=j+1}^{j+k-1} (n - (i - 1))(a_i - a_{i-1}) = 0 \quad (22)$$

since $a_j = a_{j+1} = \dots = a_{j+k-1}$. Given this, we conclude that

$$\sum_{i=1}^n (n - (i - 1))(a_i - a_{i-1}) = \sum_{i=1}^j \sum_{i=j+k}^n (n - (i - 1))(a_i - a_{i-1}) = \sum_{i=1}^n a_i \quad (23)$$

Conclusion. The sum of the elements of the G-transformation formed based on the matrix, which contains some number of identical elements, also equals the sum of the elements of the matrix, just as in the case with all different elements. When the G-transformation is applied to any column matrix, the sum of the elements is preserved.

4. G-Transformations Modeling

Let us consider a square matrix A , with dimensions 4×4 :

$$A = \begin{vmatrix} 2 & 5 & 8 & 4 \\ 21 & 25 & 55 & 6 \\ 89 & 41 & 31 & 10 \\ 35 & 26 & 7 & 3 \end{vmatrix}$$

We split the starting matrix A into smaller ones with dimensions 2×2 .

Let us consider, in turn, the processing of each “window” of the initial matrix A .

$$\text{Input matrix: } A_1 = \begin{vmatrix} 2 & 5 \\ 21 & 25 \end{vmatrix}$$

Iteration	G-Transformation	Shift	Tail Elements	Transposition
1	$\begin{vmatrix} 2 & 3 \\ 21 & 4 \end{vmatrix}$	—	—	$\begin{vmatrix} 2 & 21 \\ 3 & 4 \end{vmatrix}$
2	$\begin{vmatrix} 2 & 19 \\ 3 & 1 \end{vmatrix}$	$\begin{vmatrix} 2 & 19 \\ & 3 & 1 \end{vmatrix}$	$a_{11} = (2)$	$\begin{vmatrix} 19 & 3 \\ & 1 \end{vmatrix}$
3	$\begin{vmatrix} 3 & 16 \\ & 1 \end{vmatrix}$	$\begin{vmatrix} 3 & 16 \\ & & 1 \end{vmatrix}$	$a_{12} = (3)$	$\begin{vmatrix} 16 & 1 \end{vmatrix}$
4	$\begin{vmatrix} 1 & 15 \end{vmatrix}$	—	$a_{13} = (1)$ $a_{14} = (15)$	—

$$\text{Input matrix: } A_2 = \begin{vmatrix} 89 & 41 \\ 35 & 26 \end{vmatrix}$$

Iteration	G-Transformation	Shift	Tail Elements	Transposition
1	$\begin{vmatrix} 41 & 48 \\ 26 & 9 \end{vmatrix}$	—	—	$\begin{vmatrix} 41 & 26 \\ 48 & 9 \end{vmatrix}$
2	$\begin{vmatrix} 26 & 15 \\ 9 & 39 \end{vmatrix}$	$\begin{vmatrix} 26 & 15 \\ & 9 & 39 \end{vmatrix}$	$a_{21} = (26)$	$\begin{vmatrix} 15 & 9 \\ 39 & \end{vmatrix}$
3	$\begin{vmatrix} 9 & 6 \\ 39 & \end{vmatrix}$	$\begin{vmatrix} 9 & 6 \\ & 39 \end{vmatrix}$	$a_{22} = (9)$	$\begin{vmatrix} 6 & 39 \end{vmatrix}$
4	$\begin{vmatrix} 6 & 33 \end{vmatrix}$	—	$a_{23} = (6)$ $a_{24} = (33)$	—

$$\text{Input matrix: } A_3 = \begin{vmatrix} 8 & 4 \\ 55 & 6 \end{vmatrix}$$

Iteration	G-Transformation	Shift	Tail Elements	Transposition
1	$\begin{vmatrix} 4 & 4 \\ 6 & 49 \end{vmatrix}$	—	—	$\begin{vmatrix} 4 & 6 \\ 4 & 49 \end{vmatrix}$
2	$\begin{vmatrix} 4 & 2 \\ 4 & 45 \end{vmatrix}$	$\begin{vmatrix} 4 & 2 \\ & 4 & 45 \end{vmatrix}$	$a_{31} = (4)$	$\begin{vmatrix} 2 & 4 \\ 45 & \end{vmatrix}$
3	$\begin{vmatrix} 2 & 2 \\ & 45 \end{vmatrix}$	$\begin{vmatrix} 2 & 2 \\ & & 45 \end{vmatrix}$	$a_{32} = (2)$	$\begin{vmatrix} 2 & 45 \end{vmatrix}$
4	$\begin{vmatrix} 2 & 43 \end{vmatrix}$	—	$a_{33} = (2)$ $a_{34} = (43)$	—

$$\text{Input matrix: } A_4 = \begin{vmatrix} 31 & 10 \\ 7 & 3 \end{vmatrix}$$

Iteration	G-Transformation	Shift	Tail Elements	Transposition
1	$\begin{vmatrix} 10 & 21 \\ 3 & 4 \end{vmatrix}$	—	—	$\begin{vmatrix} 10 & 3 \\ 21 & 4 \end{vmatrix}$
2	$\begin{vmatrix} 3 & 7 \\ 4 & 17 \end{vmatrix}$	$\begin{vmatrix} 3 & 7 \\ & 4 & 17 \end{vmatrix}$	$a_{41} = (3)$	$\begin{vmatrix} 7 & 4 \\ 17 & \end{vmatrix}$
3	$\begin{vmatrix} 4 & 3 \\ 17 & \end{vmatrix}$	$\begin{vmatrix} 4 & 3 \\ & 17 \end{vmatrix}$	$a_{42} = (4)$	$\begin{vmatrix} 3 & 17 \end{vmatrix}$
4	$\begin{vmatrix} 3 & 14 \end{vmatrix}$	—	$a_{43} = (3)$ $a_{44} = (14)$	—

Let us write the obtained tail elements into a new matrix, A^* :

$$A^* = \begin{vmatrix} 2 & 26 & 4 & 3 \\ 3 & 9 & 2 & 4 \\ 1 & 6 & 2 & 3 \\ 15 & 33 & 43 & 14 \end{vmatrix}$$

We divide the resulting matrix A^* into “windows” according to the mathematical model above. Again, we consider the processing of each “window” of the matrix A^* in turn.

$$\text{Input matrix: } A_1 = \begin{vmatrix} 2 & 26 \\ 3 & 9 \\ 1 & 6 \\ 15 & 33 \end{vmatrix}$$

Iteration	G-Transformation	Shift	Tail Elements	Transposition
1	$\begin{vmatrix} 2 & 24 \\ 3 & 6 \\ 1 & 5 \\ 15 & 18 \end{vmatrix}$	—	—	$\begin{vmatrix} 2 & 3 & 1 & 15 \\ 24 & 6 & 5 & 18 \end{vmatrix}$
2	$\begin{vmatrix} 1 & 1 & 1 & 12 \\ 5 & 1 & 12 & 6 \end{vmatrix}$	$\begin{vmatrix} 1 & 1 & 1 & 12 \\ & 5 & 1 & 12 & 6 \end{vmatrix}$	$a_{11} = (1)$	$\begin{vmatrix} 1 & 5 \\ 1 & 1 \\ 12 & 12 \\ 6 & \end{vmatrix}$
3	$\begin{vmatrix} 1 & 4 \\ 1 & \\ 12 & \\ 6 & \end{vmatrix}$	$\begin{vmatrix} 1 & 4 \\ & 1 & \\ & & 12 & \\ & & & 6 \end{vmatrix}$	$a_{12} = (2)$	$\begin{vmatrix} 4 & 1 \\ 12 & \\ 6 & \end{vmatrix}$
4	$\begin{vmatrix} 1 & 3 \\ 12 & \\ 6 & \end{vmatrix}$	$\begin{vmatrix} 1 & 3 \\ & 12 & \\ & & 6 \end{vmatrix}$	$a_{13} = (1)$	$\begin{vmatrix} 3 & 12 \\ 6 & \end{vmatrix}$
5	$\begin{vmatrix} 3 & 9 \\ 6 & \end{vmatrix}$	$\begin{vmatrix} 3 & 9 \\ & 6 \end{vmatrix}$	$a_{14} = (3)$	$\begin{vmatrix} 9 & 6 \end{vmatrix}$
6	$\begin{vmatrix} 6 & 3 \end{vmatrix}$	—	$a_{15} = (6)$ $a_{16} = (3)$	—

$$\text{Input matrix: } A_2 = \begin{vmatrix} 4 & 3 \\ 2 & 4 \\ 2 & 3 \\ 43 & 14 \end{vmatrix}$$

Iteration	G-Transformation	Shift	Tail Elements	Transposition
1	$\begin{vmatrix} 3 & 1 \\ 2 & 2 \\ 2 & 1 \\ 14 & 29 \end{vmatrix}$	—	—	$\begin{vmatrix} 3 & 2 & 2 & 14 \\ 1 & 2 & 1 & 29 \end{vmatrix}$
2	$\begin{vmatrix} 2 & 1 & 11 \\ 1 & 1 & 27 \end{vmatrix}$	$\begin{vmatrix} 2 & 1 & 11 \\ 1 & 1 & 27 \end{vmatrix}$	$a_{21} = (2)$	$\begin{vmatrix} 1 & 1 \\ 11 & 1 \\ 27 & \end{vmatrix}$
3	$\begin{vmatrix} 1 \\ 1 & 10 \\ 27 \end{vmatrix}$	$\begin{vmatrix} 1 \\ 1 & 10 \\ 27 \end{vmatrix}$	$a_{22} = (1)$	$\begin{vmatrix} 1 \\ 10 & 27 \end{vmatrix}$
4	$\begin{vmatrix} 1 \\ 10 & 17 \end{vmatrix}$	$\begin{vmatrix} 1 \\ 10 & 17 \end{vmatrix}$	$a_{23} = (1)$	$\begin{vmatrix} 10 \\ 17 \end{vmatrix}$
5	$\begin{vmatrix} 10 \\ 17 \end{vmatrix}$	—	$a_{24} = (10)$ $a_{25} = (17)$	—

Let us write the obtained tail elements into a new matrix A^{**} :

$$A^{**} = \begin{vmatrix} 4 & 3 \\ 2 & 4 \\ 2 & 3 \\ 43 & 14 \end{vmatrix}$$

We transform the obtained matrix A^{**} according to the mathematical model given above.

Iteration	G-Transformation	Shift	Tail Elements	Transposition
1	$\begin{vmatrix} 1 & 1 \\ 1 \\ 1 \\ 3 & 7 \\ 6 & 11 \\ 3 \end{vmatrix}$	—	—	$\begin{vmatrix} 1 & 1 & 1 & 3 & 6 & 3 \\ 1 & 7 & 11 & & & \end{vmatrix}$
2	$\begin{vmatrix} 1 & 2 & 3 \\ 1 & 6 & 4 \end{vmatrix}$	$\begin{vmatrix} 1 & 2 & 3 \\ 1 & 6 & 4 \end{vmatrix}$	$a_1 = (1)$	$\begin{vmatrix} 2 & 1 \\ 3 & 6 \\ 4 \end{vmatrix}$
3	$\begin{vmatrix} 1 & 1 \\ 3 & 3 \\ 4 \end{vmatrix}$	$\begin{vmatrix} 1 & 1 \\ 3 & 3 \\ 4 \end{vmatrix}$	$a_{22} = (1)$	$\begin{vmatrix} 1 & 3 \\ 3 & 4 \end{vmatrix}$
4	$\begin{vmatrix} 1 & 2 \\ 3 & 1 \end{vmatrix}$	$\begin{vmatrix} 1 & 2 \\ 3 & 1 \end{vmatrix}$	$a_3 = (1)$	$\begin{vmatrix} 2 & 3 \\ 1 \end{vmatrix}$
5	$\begin{vmatrix} 2 & 1 \\ 1 \end{vmatrix}$	$\begin{vmatrix} 2 & 1 \\ 1 \end{vmatrix}$	$a_4 = (2)$	$\begin{vmatrix} 1 & 1 \end{vmatrix}$

Tail element of PI network: $a_5 = (1)$.

Result: $a = (1, 1, 1, 2, 1)$.

5. Conclusions

Let us compare the PI transform with well-known transforms, such as Fast Fourier Transform (FFT), Hadamard, and Haar, in terms of the number of operations they require. The number of operations for the PI transform is $N(N + 1)$. For comparison, for widely used practical transforms, especially orthogonal transforms, the number of operations is $4N^2 \log_2 N$ for FFT, $N^2 \log_2 N$ for Hadamard, and $4N(N + 1)$ for Haar. The absence of computationally intensive multiplication and division operations indicates that the computational algorithm implementing the PI transform is sufficiently simple. This makes it an efficient method for applications that require a combination of a high degree of parallelism and a compact form of data representation.

The structure of the neuro-like semantic and computational parallel–hierarchical network includes the joint development of the ideas of parallelism, hierarchy, and mixing in the processing of information data streams, which are then propagated in the network’s “horizontal” direction to other elements in the “vertical” direction or, in the general case, to elements of other hierarchical levels along an “arbitrary route”. The transformation applied in each branch of the computational PI network for the data array does not violate its informational content, which enables the seamless application of the transformation within the computational PI networks to organize information processes in artificial intelligence systems. Moreover, this result provides a mathematical foundation supporting the function and organization of neuro-like computational parallel–hierarchical networks when identifying surface defects on rails.

Author Contributions: Conceptualization, L.T., N.K., V.T., Y.D., M.D. and O.O.; methodology, L.T., N.K., V.T., Y.D., M.D. and O.O.; software, L.T., N.K., V.T., Y.D., M.D. and O.O.; validation, L.T., N.K., V.T., Y.D., M.D. and O.O.; formal analysis, L.T., N.K., V.T., Y.D., M.D. and O.O.; investigation, L.T., N.K., V.T., Y.D., M.D. and O.O.; resources, L.T., N.K., V.T., Y.D., M.D. and O.O.; data curation, L.T., N.K., V.T., Y.D., M.D. and O.O.; writing—original draft preparation, L.T., N.K., V.T., Y.D., M.D. and O.O.; writing—review and editing, V.L., L.T., N.K., V.T., Y.D., M.D. and O.O.; visualization, V.L., L.T., N.K., V.T., Y.D., M.D. and O.O.; supervision, V.L., L.T., N.K., V.T., Y.D., M.D. and O.O.; project administration, V.L., L.T., N.K., V.T., Y.D., M.D. and O.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hrytsyk, V.I.N.; Berezhky, O.M. Methods and high-performance parallel systems for real-time image processing and recognition. *Int. J. Comput.* **2014**, *2*, 24–25. [\[CrossRef\]](#)
2. Yufik, Y.M. The Understanding Capacity and Information Dynamics in the Human Brain. *Entropy* **2019**, *21*, 308. [\[CrossRef\]](#) [\[PubMed\]](#)
3. Zwart, H. Dialectical Materialism. In *Continental Philosophy of Technoscience; Philosophy of Engineering and Technology*; Springer: Cham, Switzerland, 2022; p. 38. [\[CrossRef\]](#)
4. Hebb, D.O. *The Organization of Behavior. A Neuropsychological Theory*, 1st ed.; Psychology Press: New York, NY, USA, 2002. [\[CrossRef\]](#)
5. Panaskin, D. Assessment of robustness of neural network models to noise and artifacts in complex operating conditions. *Comput. Integr. Technol. Educ. Sci. Prod.* **2024**, *56*, 226–235. [\[CrossRef\]](#)
6. Hawking, S. Arrow of time in cosmology. *Phys. Rev. D* **1985**, *32*, 2489. [\[CrossRef\]](#) [\[PubMed\]](#)

7. Zeki, S. *A Vision of the Brain*; Blackwell Scientific Publications: Oxford, UK, 1993; p. 366.
8. Hu, Y.; Xie, M. Automatic Segmentation of Brain CT Image Based on Multiplicate Features and Decision Tree. In Proceedings of the International Conference on Communications, Circuits and Systems, Kokura, Japan, 11–13 July 2007; pp. 837–840. [\[CrossRef\]](#)
9. Gorobchenko, O.; Holub, H.; Zaika, D. Theoretical basics of the self-learning system of intelligent locomotive decision support systems. *Arch. Transp.* **2024**, *71*, 169–186. [\[CrossRef\]](#)
10. Hubel, D. *Eye, Brain, and Vision*; W. H. Freeman: New York, NY, USA, 1995.
11. Subbotin, S.; Oleynik, A.A. Evolutionary synthesis of models of complex objects and processes. *Radioelectron. Inform.* **2007**, *2*, 99–104. [\[CrossRef\]](#)
12. Bistouni, F.; Jahanshahi, M. Pars network: A multistage interconnection network with fault-tolerance capability. *J. Parallel Distrib. Comput.* **2015**, *75*, 168–183. [\[CrossRef\]](#)
13. Xin, M.; Wang, Y. Research on image classification model based on deep convolution neural network. *EURASIP J. Image Video Process.* **2019**, *2019*, 40. [\[CrossRef\]](#)
14. Razzak, M.I.; Naz, S.; Zaib, A. Deep Learning for Medical Image Processing: Overview, Challenges and the Future. In *Classification in BioApps. Lecture Notes in Computational Vision and Biomechanics*; Dey, N., Ashour, A., Borra, S., Eds.; Springer: Cham, Switzerland, 2018; Volume 26. [\[CrossRef\]](#)
15. Wang, J.; Zhu, H.; Wang, S.; Zhang, Y. A review of deep learning on medical image analysis. *Mob. Netw. Appl.* **2021**, *26*, 351–380. [\[CrossRef\]](#)
16. Rakushev, M.; Filatov, M. Calculation of the differential–Taylor spectrum of the inverse matrix and the determinant when modeling dynamic systems. *Mod. Inf. Technol. Field Secur. Def.* **2022**, *44*, 5–9. [\[CrossRef\]](#)
17. Butko, T.; Babanin, A.; Gorobchenko, A. Rationale for the type of the membership function of fuzzy parameters of locomotive intelligent control systems. *East. Eur. J. Enterp. Technol.* **2015**, *1*, 4–8. [\[CrossRef\]](#)
18. Tymchenko, L.; Tverdomed, V.; Kokryatska, N.; Petrovsky, N. Development of a method of processing images of laser beam bands with the use of parallel-hierarchical networks. *East. Eur. J. Enterp. Technol.* **2019**, *6*, 21–27. [\[CrossRef\]](#)
19. Wu, X.; Lee, W.-N. Row Transmission for High Volume-Rate Ultrasound Imaging With a Matrix Array. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control* **2024**, *71*, 659–672. [\[CrossRef\]](#) [\[PubMed\]](#)
20. Box, G.; Jenkins, G. *Time Series Analysis: Forecasting and Control*, 3rd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 1994.
21. Holub, H.; Gorobchenko, O.; Muraviov, V.; Hannoshyna, I.; Lushchai, Y.; Dovhopoliuk, L. Research of Methods of Detecting Network Anomalies of Computer System Network States. *Lect. Notes Netw. Syst.* **2023**, *685*, 151–158. [\[CrossRef\]](#)
22. Haykin, S. *Neural Networks*, 2nd ed.; Prentice Hall: Saddle River, NJ, USA, 1999.
23. Sysyn, M.; Kovalchuk, V.; Gerber, U.; Nabochenko, O.; Parneta, B. Laboratory evaluation of railway ballast consolidation by the non-destructive testing. *Commun. Sci. Lett. Univ. Žilina* **2019**, *21*, 81–88. [\[CrossRef\]](#)
24. Timchenko, L.; Kokriatskaia, N.; Shpakovych, V. Modeling of a method of parallel hierarchical transformation for fast recognition of dynamic images. *EURASIP J. Adv. Signal Process.* **2013**, *1*, 87. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.