sensors

Article

Hardware Implementation for Triaxial Contact-Force Estimation from Stress Tactile Sensor Arrays: An Efficient Design Approach

María-Luisa Pinto-Salamanca [1,2,*], Wilson-Javier Pérez-Holguín [1] and José A. Hidalgo-López [2]

[1] Programa de Doctorado en Ingeniería-Énfasis en Ingeniería Electrónica, Grupo GIRA, Universidad Pedagógica y Tecnológica de Colombia UPTC, Sogamoso 152211, Colombia; wilson.perez@uptc.edu.co
[2] Programa de Doctorado en Ingeniería Mecatrónica, Departamento de Electrónica, Universidad de Málaga, 29071 Malaga, Spain; jahidalgo@uma.es
* Correspondence: marialuisa.pinto@uptc.edu.co

Abstract: This paper presents a contribution to the state of the art in the design of tactile sensing algorithms that take advantage of the characteristics of generalized sparse matrix-vector multiplication to reduce the area, power consumption, and data storage required for real-time hardware implementation. This work also addresses the challenge of implementing the hardware to execute multiaxial contact-force estimation algorithms from a normal stress tactile sensor array on a field-programmable gate-array development platform, employing a high-level description approach. This paper describes the hardware implementation of the proposed sparse algorithm and that of an algorithm previously reported in the literature, comparing the results of both hardware implementations with the software results already validated. The calculation of force vectors on the proposed hardware required an average time of 58.68 ms, with an estimation error of 12.6% for normal forces and 7.7% for tangential forces on a 10 × 10 taxel tactile sensor array. Some advantages of the developed hardware are that it does not require additional memory elements, achieves a 4× reduction in processing elements compared to a non-sparse implementation, and meets the requirements of being generalizable, scalable, and efficient, allowing an expansion of the applications of normal stress sensors in low-power tactile systems.

Keywords: tactile sensing; triaxial contact forces estimation; hardware implementation; sparse matrix-vector multiplication; FPGA

Citation: Pinto-Salamanca, M.-L.; Pérez-Holguín, W.-J.; Hidalgo-López, J.A. Hardware Implementation for Triaxial Contact-Force Estimation from Stress Tactile Sensor Arrays: An Efficient Design Approach. *Sensors* **2024**, *24*, 7829. https://doi.org/10.3390/s24237829

Academic Editor: Min Li

Received: 29 October 2024
Revised: 30 November 2024
Accepted: 5 December 2024
Published: 7 December 2024

1. Introduction

In artificial tactile sensing systems, the measurement of contact forces plays a crucial role in the description of the contact phenomenon [1,2] seeking to resemble human dexterity for object manipulation [3]. Force estimation provides direct information about the contact area and facilitates the reconstruction of other tactile properties such as roughness [4,5], and hardness [6]. Moreover, triaxial forces are critical for slip and grip sensing in control loops for robotic manipulation [3,4,7–12], wearable robots [13], human-computer interaction [2], prosthetic hands and wearable devices [14–16].

The scientific community has already provided solutions for reconstructing normal, shear, and multiaxial contact forces. Some authors [12,17–19] have estimated forces using integrated biomimetic-design tactile sensors for robotic grippers using vision-based transducers. Alternatively, the piezoelectric effect [20,21], piezoresistive-based tactile sensor arrays [2,14,15,22], capacitive sensors [13,16,23–27], magnetic transducers [28,29], optical fibers [30,31], and Hall effect transducers [32] can be used. However, contact-force sensing remains a major research challenge due to issues such as the complexity and multiplicity of mathematical force estimation models, the different transduction technologies employed, the commercial availability of sensors, the area size and contact medium, and the demand for real-time processing [2,33].

Despite recent advances in multiaxial contact-force reconstruction, the community continues to work towards achieving tactile sensing systems that meet the criteria of generalization, scalability, and efficiency described in [34]. A challenge to generalization lies in the fact that force estimation methods depend on the information provided by the tactile sensor and the contact medium used. Scalability is limited by the transduction technology and sensor size used with contact areas typically small (less than 164 mm $\times$ 164 mm) [25,35]. In terms of efficiency, force reconstruction time must be predictable and done in real time [2,13,15,23,25]. All of this involves systems with high processing requirements that need very short response times. The approach to these issues is based on the development of specialized software that runs on high-performance PC stations, which limits their application in cases where high portability is required, such as in electronic skin applied in biomedical or robotic systems.

Regarding sensor types used in tactile sensing applications, pressure-sensor arrays offer advantages such as high resolution, high sensitivity, low noise, simple electronics, physical flexibility, large surface area, wide commercial availability and price, dynamic response, low thickness, large-scale coverage, and high bandwidth [36–40]. In spite of their advantages, the use of pressure-sensor arrays in multiaxial force reconstruction has been quite limited, as they only provide information on normal stresses and require complex mathematical models to process the data they generate. Their applications have been limited to obtaining normal forces [37], and through subsequent processing stages, other authors have estimated shear [39] and triaxial forces [41], but none of those methods have been validated in hardware using efficient performance metrics. In [42], an optimized option for the hardware implementation of triaxial force estimation in real-time tactile decoding tasks is provided. However, since such implementation has only been validated in software, it is necessary to investigate the implications of the properties of its mathematical model on hardware design, memory consumption, and data-processing efficiency.

An alternative to address the high complexity, real-time execution, and large data volume requirements of tactile decoding systems is to leverage the benefits of highly parallel embedded systems such as field-programmable gate arrays (FPGAs), graphics processing units (GPUs), and others. These systems provide high concurrency, lower power consumption, and high computational acceleration, which, if properly exploited through a highly optimized design, can greatly exceed the performance of software-based (PC) solutions [43]. Embedded systems have been used to implement model- and data-driven solutions in tactile sensing applications like texture estimation [44], texture classification [45], tactile data decoding [46,47], slip detection [48], and force estimation [41]. However, hardware implementations in this context are scarce because the hardware design process remains slow and complex [43], is highly dependent on the sensor characteristics, transduction technology, and contact type, and the mathematical model is not always easy to modify as it involves significant changes in the hardware deployment. One way to overcome these problems is to use high-level design methodologies, which have been validated in other fields but rarely used to develop hardware for tactile sensing systems [49].

This work contributes to the theoretical development of algorithms for multiaxial contact-force reconstruction from normal stress tactile sensor arrays by proposing using sparse matrices to reduce the computational requirements for its implementation in hardware in terms of area, power consumption, and data storage. It also contributes to the practical implementation of tactile sensing algorithms in hardware by comparing the results of [42] with those obtained for a new model presented herein. Although it has only been validated for two single static contacts on flat surfaces, we believe this work will contribute to expanding the use of stress sensor arrays for multiaxial force estimation in large-scale wearable electronic skin systems.

Among the most relevant aspects of the hardware implementation carried out here are that it is efficient (in terms of computation times and area occupation), generalizable (independent of the transduction sensor technology and the contact phenomenon), scalable (independent of the size of the contact sensing area) and described at a high level (using high-level tools such as Matlab® MathWorks™, AMD Vitis™ HLS, and Vivado™ Design Suite) which allows a reduction of the design effort and quick validation of the functionality of the system.

The rest of the paper is organized as follows: Section 2 focuses on recalling the main features of the reconstruction forces algorithm addressed, Section 3 describes the strategy of integrating sparse matrices into the forces algorithm, Section 4 presents the hardware design flow and the efficiency criteria considered, Section 5 discusses the results obtained, and Section 6 concludes the paper.

2. Triaxial Forces Reconstruction Algorithm from Normal Stress Tactile Sensor Arrays

In the field of tactile property decoding, the TFRA algorithm [42] stands out for allowing the reconstruction of triaxial forces in large contact areas from tactile sensor arrays. Such an algorithm inputs a vector with scalar values of normal stresses and produces as output three vectors with the triaxial force magnitudes (one for each axis), which correspond to a solution to the problem of estimating multiaxial forces from a discrete normal stress distribution as shown in Figure 1. In addition, it is characterized by being generalizable, efficient, and scalable, which makes its implementation feasible for real-time tactile decoding systems.

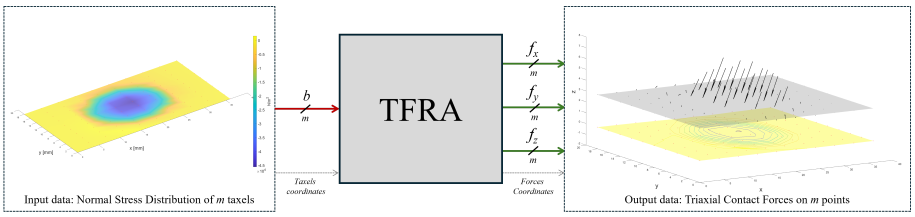
Input data: Normal Stress Distribution of m taxels
b
m
Taxels coordinates
TFRA
f_x
m
f_y
m
f_z
m
Forces Coordinates
Output data: Triaxial Contact Forces on m points

Figure 1. Triaxial forces reconstruction algorithm (TFRA).

The TFRA algorithm is based on the inverse problem of the classical Boussinesq equation, which determines the stress fields in a homogeneous and linearly elastic half-space under a concentrated load applied at a specific point on its upper surface [50]. The relationship between stress fields and load forces, established by the Boussinesq equation, is very significant for contact modeling on flat surfaces using tactile sensing arrays [41] since if the contact load is concentrated into an array of f point forces, then a b stress discrete distribution can be calculated through a linear vector equation as:

$$b = Cf \tag{1}$$

where C is a rectangular matrix defined by the distances between the coordinates of the m-stress points and those of the n-force vectors, $b = [b_x, b_y, b_z]^T$ is a vector of size $[3m \times 1]$ with b_x and b_y represents the tangential stress, while b_z is the normal stress, and $f = [f_x, f_y, f_z]^T$ is a vector of size $[3n \times 1]$ of f_x, f_y, and f_z components. Under the premise of Equation (1), if $m = n$, C is a square full-rank matrix, and the direct application of this equation generates m-values of the spatial stress distribution from m-triaxial force vectors; see Figure 2.

For the same case, the inverse application of Equation (1), expressed as:

$$f = C^{-1}b \tag{2}$$

This allows the discovery of m-force vectors from the stress values at m-points, and models the contact forces on a tactile sensor array of m-discrete sensing units (taxels), with $m = u \times v$, where u is the number of rows and v the number of columns of the array; see Figure 3. For this problem, the application of Equation (1) allows us to find the normal stress distribution at the m-points located at a depth h of the surface (sensor thickness), defined as:

$$b_z = \begin{bmatrix} C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{3}$$

with C_{31}, C_{32}, and C_{33} constant submatrices of size $[m \times m]$ defined from the sensor geometry characteristics, as:

$$\begin{bmatrix} C_{31} & C_{32} & C_{33} \end{bmatrix}^T = \begin{bmatrix} C_0 \cdot \hat{r}(ji)_x \\ C_0 \cdot \hat{r}(ji)_y \\ C_0 \cdot h \end{bmatrix} \tag{4}$$

where $C_0 = (3h^2)/(2\pi(\hat{r}(ji)^2 + h^2)^{5/2})$, $\hat{r}(ji)$ is the projection on the xy-plane of the $r(ji)$ distance vectors between the coordinates $(\hat{r}(ji)_x, \hat{r}(ji)_y)$ of the i-th taxel $(i = 1, \ldots, m)$ and the j-th force vector $(j = 1, \ldots, m)$ [51], see Figure 3b.

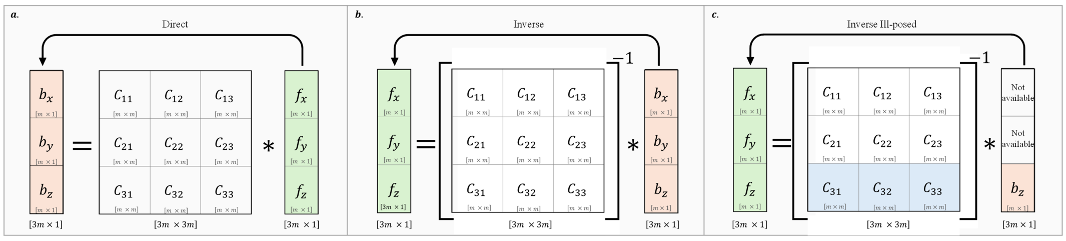

Figure 2. Relationship between m-stress values and m-triaxial forces for an m-taxel tactile sensor array under the Businessq Equation: (**a**) Direct problem, (**b**) Inverse problem, and (**c**) Ill-posed inverse problem from normal stress data.

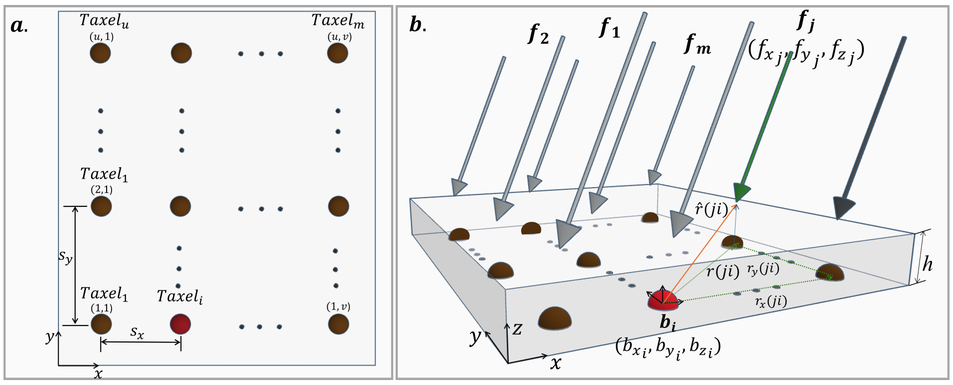

Figure 3. Normal stress on a tactile sensor array of m taxels: (**a**) Sensor top view, and (**b**) Interaction between the i-th taxel (dot in red) and the j-th force vector (green arrow) on the sensor surface for m-stress values and m-force vectors.

For the case of a tactile sensor that only provides discrete normal stress data b_z for the m-taxels, the inverse problem becomes ill-posed because the number of unknowns in Equation (2) is three times larger than the sensor outputs, requiring an additional mathematical description. In such a case, the Moore–Penrose pseudo-inverse matrix provides a solution of the form $f = A^{-1} \cdot b_z$ [52]:

$$f = \begin{bmatrix} f_x & f_y & f_z \end{bmatrix}^T = A_{+}b_z + (I - A_{+}A)w \tag{5}$$

where A is a rectangular matrix of size $[m \times 3m]$, defined as:

$$A = \begin{bmatrix} C_{31} & C_{32} & C_{33} \end{bmatrix} \tag{6}$$

$A_{\dagger}$ is the pseudo-inverse matrix of A of size $[3m \times m]$, such that:

$$A_{\dagger} = A^T\left(AA^T\right)^{-1} = [A_{\dagger 1}A_{\dagger 2}A_{\dagger}3]^T \tag{7}$$

in which $A_{\dagger 1}$, $A_{\dagger 2}$, and $A_{\dagger 3}$ are constant submatrices of size $[m \times m]$, and w is an optimal vector of size $[m \times 1]$, for which

$$w = \begin{bmatrix} \mu_x|w_z| \\ \mu_y|w_z| \\ w_z \end{bmatrix} \tag{8}$$

where μ_x and μ_y are two continuous scalars in the range $[-1, 1]$ defined into an optimization process, and w_z a vector made up of the negative values of the tactile sensor, as follows:

$$w_z = \begin{cases} C_{33}^{-1} \cdot b_z & ;\left(C_{33}^{-1} \cdot b_z\right)(i) < 0 \\ 0 & ;\text{other case} \end{cases} \tag{9}$$

By replacing Equations (6)–(9) in Equation (5), the $TFRA$ allows the discovery of an optimal solution for this problem by calculating the components of a f triaxial forces as follows:

$$\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} = \begin{bmatrix} g_{10} + \mu_x g_{11} + \mu_y g_{12} \\ g_{20} + \mu_x g_{21} + \mu_y g_{22} \\ g_{30} + \mu_x g_{31} + \mu_y g_{32} \end{bmatrix} \tag{10}$$

where g_{pq} vectors of size $[m \times 1]$ are computed as shown in Equations (11)–(13) for $p = [1:3]$, $q = [0:2]$, such that:

$$\begin{bmatrix} g_{10} & g_{11} & g_{12} \end{bmatrix}^T = \begin{bmatrix} A_{\dagger 1}b_z + D_{10}w_z \\ \mid w_z \mid + D_{11} \mid w_z \mid \\ D_{12} \mid w_z \mid \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} g_{20} & g_{21} & g_{22} \end{bmatrix}^T = \begin{bmatrix} A_{\dagger 2}b_z + D_{20}w_z \\ D_{21} \mid w_z \mid \\ \mid w_z \mid + D_{22} \mid w_z \mid \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} g_{30} & g_{31} & g_{32} \end{bmatrix}^T = \begin{bmatrix} A_{\dagger 2}b_z + w_z + D_{30}w_z \\ D_{31} \mid w_z \mid \\ D_{32} \mid w_z \mid \end{bmatrix} \tag{13}$$

D_{pq} are nine matrices of size $[m \times m]$ calculated by an offline precomputation carried out by matrix multiplications between the submatrices of $A_{\dagger}$ and C, as follows:

$$\begin{bmatrix} D_{10} & D_{11} & D_{12} \\ D_{20} & D_{21} & D_{22} \\ D_{30} & D_{31} & D_{32} \end{bmatrix} = \begin{bmatrix} -A_{\dagger 1}C_{33} & -A_{\dagger 1}C_{31} & -A_{\dagger 2}C_{32} \\ -A_{\dagger 2}C_{33} & -A_{\dagger 2}C_{31} & -A_{\dagger 2}C_{32} \\ -A_{\dagger 3}C_{33} & -A_{\dagger 3}C_{31} & -A_{\dagger 3}C_{32} \end{bmatrix} \tag{14}$$

To implement Equation (10), TFRA requires the execution of 16 matrix-vector multiplications, denoted as $M_0 - M_{15}$, and the iterative calculation of the optima of μ_x and μ_y. The first 13 operations ($M_0 - M_{12}$) correlate the sensor geometry and the normal stress distribution measured by the tactile sensor array so that:

$$M_0 = C_{33}^{-1}b_z \tag{15}$$

$$\begin{bmatrix} M_1 & M_2 & M_3 \\ M_4 & M_5 & M_6 \\ M_7 & M_8 & M_9 \\ M_{10} & M_{11} & M_{12} \end{bmatrix} = \begin{bmatrix} D_{10}w_z & D_{11} \mid w_z \mid & D_{12} \mid w_z \mid \\ D_{20}w_z & D_{21} \mid w_z \mid & D_{22} \mid w_z \mid \\ D_{30}w_z & D_{31} \mid w_z \mid & D_{32} \mid w_z \mid \\ A_{\dagger 1}b_z & A_{\dagger 2}b_z & A_{\dagger 3}b_z \end{bmatrix} \tag{16}$$

The last three operations ($M_{13} - M_{15}$) produce b_z^z and b_z^{xy} vectors to evaluate the fulfillment of the optimal conditions:

$$\begin{bmatrix} M_{13} & M_{14} & M_{15} \end{bmatrix} = \begin{bmatrix} C_{33} f_z^k & C_{31} f_x^k & C_{32} f_y^k \end{bmatrix} \tag{17}$$

where f_x^k, f_y^k, and f_z^k are force components estimated for the k-th iteration of the algorithm, while the vectors b_z^z of size $[m \times 1]$, and b_z^{xy} of size $[2m \times 1]$ are defined as:

$$b_z^z = C_{33} f_z^k \tag{18}$$

$$b_z^{xy} = \begin{bmatrix} C_{31} & C_{32} \end{bmatrix} \begin{bmatrix} f_x^k \\ f_y^k \end{bmatrix} \tag{19}$$

As mentioned in [42], the TFRA optimization process allows us to ensure that: (i) normal forces (f_z^k) will be correctly reconstructed if they generate only a compression effect at the sensor base when they are considered independently, and (ii) tangential forces (f_x^k, f_y^k) will be correctly reconstructed if they generate a traction distribution similar to that measured by the sensor. Formally, each condition is evaluated as follows:

Condition 1:

$$B^z \equiv \sum_{i=1}^{m} b_z^z(i) = 0; b_z^z(i) > 0 \tag{20}$$

Condition 2:

$$\begin{gathered} B^{xy} \equiv \left| \sum_{i=1}^{m} b_z^{xy}(i) - \sum_{i=1}^{m} b_z(i) \right| = 0; \\ b_z^{xy}(i) > 0, b_z(i) > 0 \end{gathered} \tag{21}$$

To evaluate the optima μ_x or μ_y in Equation (10), TFRA selects only one of these, μ_x or μ_y, as an independent variable and evaluates the other (the unselected one) from the angle between the tangential forces (ϕ) and the G_{pq} resultants of the vectors g_{pq}. Therefore, the dependent variable selected can be computed as:

$$\mu_x = \frac{G_{20} - G_{10} \tan\phi + \mu_y (G_{22} - G_{12} \tan\phi)}{G_{11} \tan\phi - G_{21}} \tag{22}$$

or

$$\mu_y = \frac{G_{10} \tan\phi - G_{20} + \mu_x (G_{11} \tan\phi - G_{21})}{G_{22} - G_{12} \tan\phi} \tag{23}$$

where $G_{pq} = \sum_{i=1}^{m} g_{pq}(i)$, and ϕ is calculated by identifying the compression centroids, with coordinates (c_x, c_y), and the tension centroids, with coordinates (t_x, t_y), of the stress distribution, as:

$$\phi = \begin{cases} \tan^{-1}\left(\frac{c_y - t_y}{c_x - t_x}\right) & ; t_x < c_x \\ \tan^{-1}\left(\frac{c_y - t_y}{c_x - t_x}\right) + 180^\circ & ; t_x \geq c_x \end{cases} \tag{24}$$

Following the aforementioned specifications, TFRA requires six stages (see Figure 4) to evaluate Equation (10) for:

i. [For one time only (offline stage)] Calculate and save in memory the set of matrices ($A_{\dagger 1}$, $A_{\dagger 2}$, $A_{\dagger 3}$, C_{33}^{-1}, D_{pq}, C_{31}, C_{32}, and C_{33}), the vectors describing the taxel coordinates (p_x, p_y), and the sensor resolution (s_x, s_y),
ii. Read and store in memory the data for sensor b_z as an $m \times 1$ size vector,
iii. Calculate the contact centroids and angle ϕ of tangential forces,
iv. Calculate the w_z vector,
v. Calculate the g_{pq} coefficient vectors,
vi. Find the optima μ_x and μ_y, and compute the forces triaxial reconstruction.

Equations (15)–(17) are computed in stages iv–vi, requiring 16 matrix-vector multiplications with a computational complexity of $O(m^2)$ for m-taxels in a tactile sensor array. When in a software implementation, these algebraic operations can be easily performed. In the case of a dedicated hardware implementation, design decisions must include processing requirements, memory size, and multiple memory accesses in advance. Thus, the hardware design process to implement the TFRA involves $16m$ accumulates, $16m^2$ data store operations, $16m^2$ memory accesses to a constant array, and $16m$ memory accesses to vector data.

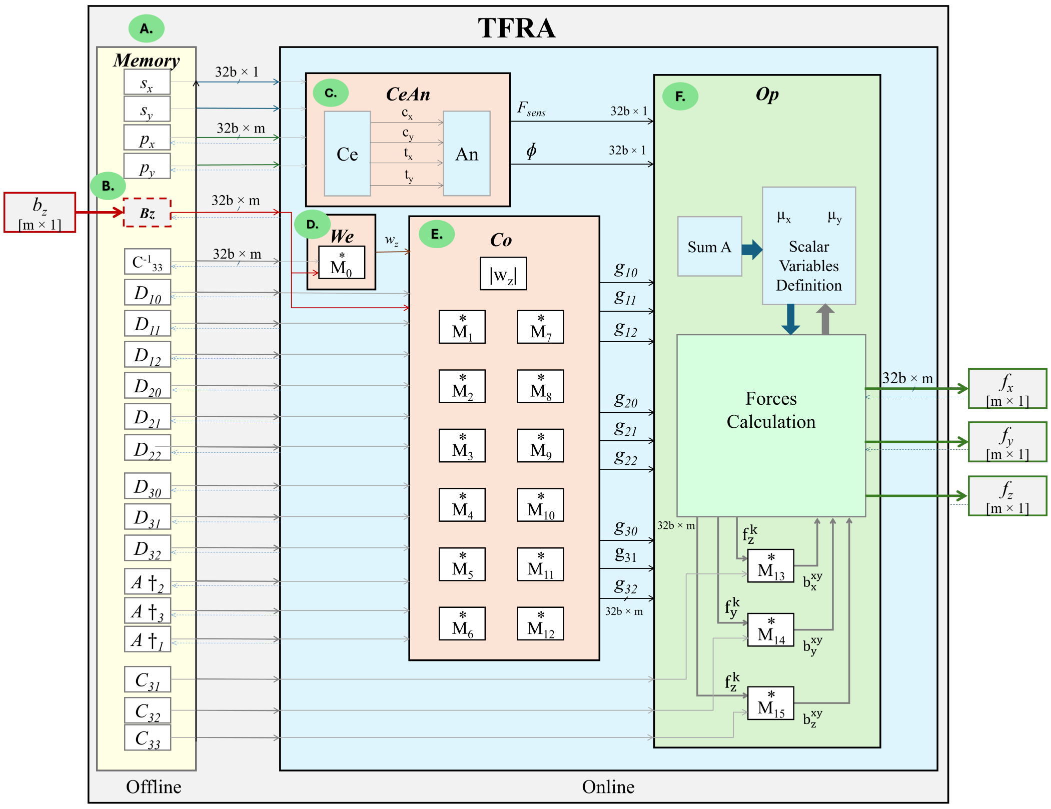

Figure 4. Blocks for the TFRA implementation in hardware: (**A**) ***Memory*** stores the set of matrices in memory, (**B**) ***Bz*** reads the data from the b_z sensor in memory, (**C**) ***CeAn*** computes the contact centroids and tangential force angle, (**D**) ***We*** calculates the initial solution weight vector w_z, (**E**) ***Co*** calculates the g_{pq} coefficients, and (**F**) ***Op*** finds the optimal values of the algorithm. Note that the symbol * represents a functional block in hardware that implements a matrix-vector multiplication.

3. SpTFRA Algorithm for Contact Forces Reconstruction

This section introduces a new algorithm, named SpTFRA (Sparse Triaxial Forces Reconstruction Algorithm), that allows the improvement of the performance of the TFRA algorithm by reducing the hardware resource consumption and the number of operations required for its calculation through the use of sparse matrices. The software implementation of the TFRA algorithm showed that in constant matrices $A_{\dagger 1}$, $A_{\dagger 2}$, $A_{\dagger 3}$, C_{33}^{-1}, and D_{pq} (computed by Equations (5), (7) and (14)), their components with the highest values are concentrated in coordinates close to their diagonals (see Figure 5) and exhibit a data distribution similar to sparse matrices (mostly zeros), a behavior demonstrated in different contact cases. This can be attributed to the fact that the components of these matrices are defined by the relative distances between the coordinates of the i-th taxel and the j-th estimated force, meaning that the force vectors are mainly influenced by nearby taxels in the xy-plane.

Matrix–vector multiplication is a basic operation required in many physical systems, so its optimization is of interest in several scientific domains. The generalized sparse-matrix dense-vector multiplication (SpMv) is defined as $y = Bx$, where B is a sparse matrix and x is a dense vector. This multiplication can be efficiently compressed using data compression and encoding strategies that store only the non-zero (Nnz) values of the matrix and its coordinates.

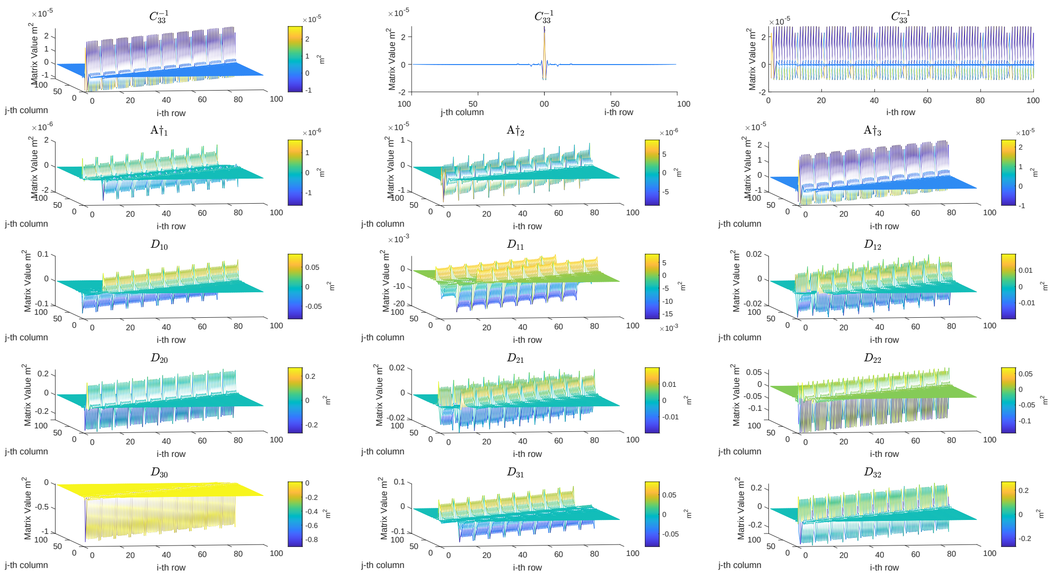

Figure 5. Behavior of magnitudes of the TFRA matrices (size 100 × 100) for a tactile sensor of 10 × 10 taxels. The first three graphs show the components of the matrix C_{33}^{-1} for different rows or columns.

One of the challenges to the hardware implementation of matrix operations relates to the on-chip and off-chip memory access [53] and the design of processing blocks that exactly suit the distribution of zeros in sparse matrices. Some algorithms to optimize SpMv on hardware have already been studied for large-scale matrix dimension problems in high-performance computing for physical or biological model simulations [54,55], data analytics [56], large-scale graphics processing [57], and artificial intelligence [58,59]. These algorithms provide solutions on GPU, FPGA, or heterogeneous architectures to process matrices as large as 28,338 × 28,338, and response times less than 5 ms [60].

In SpTFRA, it is assumed that all values of sparse-like matrices close to zero are effectively zero. This may slightly increase the error, but it reduces the resources required for data storage and the number of operations to be performed. In this way, the implementation of the SpTFRA algorithm requires the use of SpMv operations for the calculation of the matrix–vector multiplications $M_0 - M_{12}$ defined by Equations (15) and (16) in which the set of matrices ($A_{\dagger 1}$, $A_{\dagger 2}$, $A_{\dagger 3}$, C_{33}^{-1}, and D_{pq}) is offline replaced with a set of approximate sparse matrices. Multiplications M_{13}, M_{14}, and M_{15} were already included in the Op block of the original TFRA as sparse operations, so no modifications are needed for the SpTFRA implementation.

Although SpTFRA uses the same six stages described in Section 2, there are three main differences between both the TFRA and SpTFRA algorithms: (i) calculating and storing the sparse matrices in the memory block, (ii) executing M_0 as SpMv in the We block, and (iii) executing $M_1 - M_{12}$ as SpMv operations in the Co block. All other functional blocks are common for both algorithms (see Figure 6).

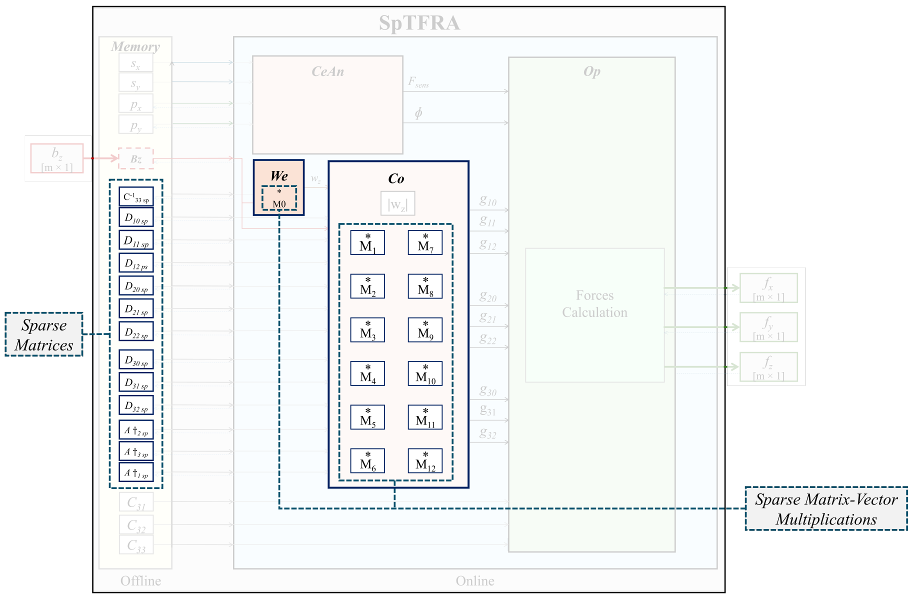

Figure 6. Hardware implementation of the SpTFRA algorithm. This figure highlights the blocks that are modified with respect to the original TFRA, which change from normal matrix operations to sparse matrix operations.

To implement SpTFRA, the proposed approach requires applying three filters that make zero some components of a matrix if its magnitude is less than a previously determined threshold. This allows the obtaining of an approximate sparse matrix B_{sp} (i.e., $A_{\dagger 1sp}$, $A_{\dagger 2sp}$, $A_{\dagger 3sp}$, C^{-1}_{33sp}, and D_{pq-sp}) from a dense matrix B ($A_{\dagger 1}$, $A_{\dagger 2}$, $A_{\dagger 3}$, C^{-1}_{33}, and D_{pq}). The threshold value is set under three specific criteria, named Sparse Filters, which are depicted in Figure 7, and defined as follows:

- SpTFRA$-$F1: This filter admits only the non-zero values $B(i,j)$ closest to the diagonal in a dense matrix B, by evaluating the function:

$$B_{sp1}(i,j) = \begin{Bmatrix} B(i,j) & if|i-j| \leq L \\ 0 & other-case \end{Bmatrix} \tag{25}$$

  where L is a value that establishes the expanded diagonal matrix made up of the components of B that satisfy $|i-j| \leq L$, with L changing at each experiment by 10 in a range from 10 to 90 (according to the estimation error calculated by the SpTFRA). Figure 8 shows an example of applying this filter on the C^{-1}_{33} matrix, where non-zero values are in blue.
- SpTFRA$-$F2: This filter accepts the components greater or equal to a percentage p of the maximum value for each row $p * max(B(i,:))$, by evaluating the function,

$$B_{sp2}(i,j) = \begin{Bmatrix} B(i,j) & if|B_{ij}| \geq (1/100p) * max(B(i,:)) \\ 0 & other-case \end{Bmatrix} \tag{26}$$

  where p changes by 10 in a range from 10 to 90 at each experiment. Figure 9 shows the results of applying this filter to the C^{-1}_{33} matrix.
- SpTFRA$-$F3: This filter selects between Filters 1 and 2, depending on which of these generates the sparse matrix Bsp with the minimum value of Nnz and the minimum error in the estimation of the contact forces, according to this function:

$$B_{sp}(i,j) = \begin{Bmatrix} B_{sp1}(i,j) & e_{\mu}(B_{sp1})Nnz(B_{sp1}) \leq e_{\mu}(B_{sp2})Nnz(B_{sp2}) \\ B_{sp2}(i,j) & other-case \end{Bmatrix} \tag{27}$$

where e_{μ} is the error estimate of the friction coefficient in a force reconstruction and comes defined as follows:

$$e_{\mu} = 100\frac{\mu_{ref} - \mu}{1.0}\% \tag{28}$$

with μ calculated as:

$$\mu = \frac{\sqrt{F_x^2 + F_y^2}}{|F_z|} \tag{29}$$

in which $F_x = \sum_{i=1}^{m} f_x(i)$ and $F_y = \sum_{i=1}^{m} f_y(i)$ are the estimated resultant tangential forces, and $F_z = \sum_{i=1}^{m} f_z(i)$ is the reconstructed resultant normal force.

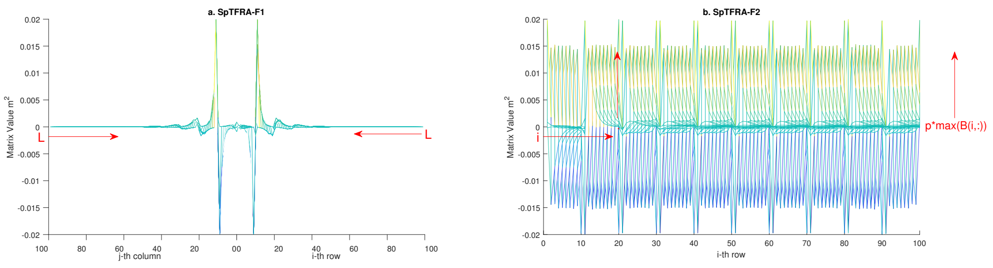

Figure 7. SpTFRA filters for matrix B applied to the: (**a**) SpTFRA−F1, selects the non-zero values closest to the diagonal; (**b**) SpTFRA−F2, selects the non-zero values greater or equal to a percentage p of the maximum value for each row.

After filtering the matrices of the TFRA, the modified compressed sparse row (MCSR) format [60] is used to store the compressed matrix B_{sp} in memory and calculate the SpMv on three vectors, which include the Nnz values of the matrix B_{sp}, the column corresponding to each Nnz in B_{sp}, and the number of multiplication accumulations (MAC) for each row. Due to the MCSR representation, the matrix–vector multiplication algorithm does not require adding or multiplying by zero, which means saving arithmetic operations proportional to the Nnz of each matrix and reducing memory consumption when storing SpTFRA matrices.

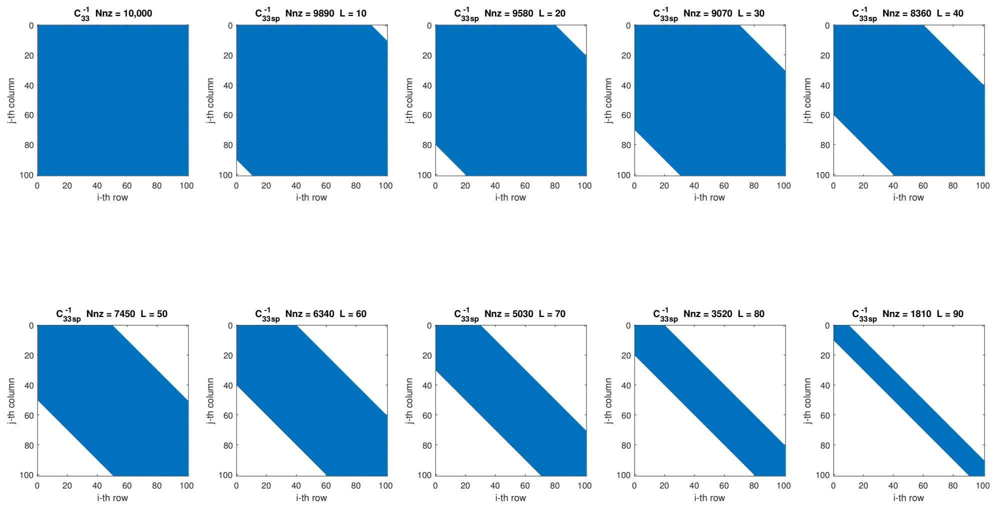

Figure 8. Application of the SpTFRA−F1 on the C_{33}^{-1} matrix.

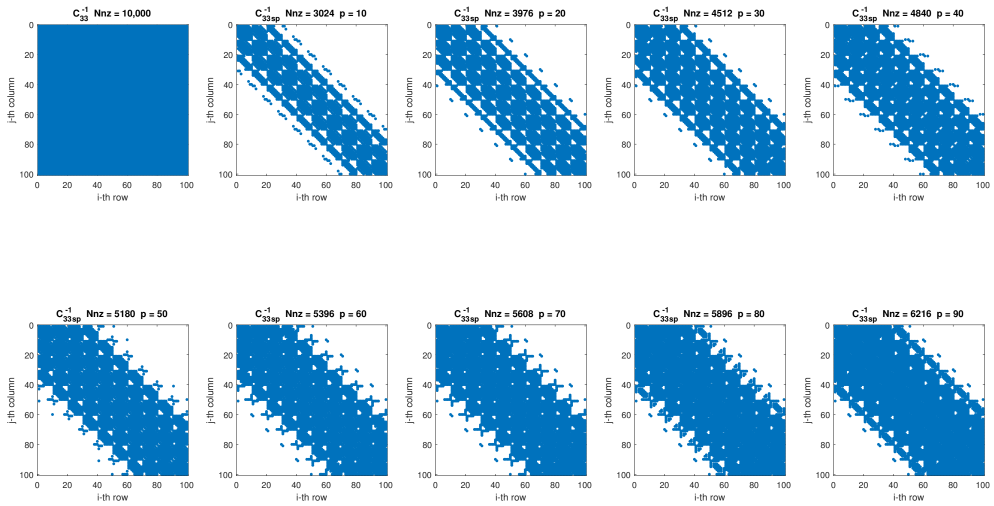

Figure 9. SpTFRA−F2 application on C_{33}^{-1} matrix.

4. Hardware Design for the TFRA and SpTFRA Implementations

The design of the hardware architecture for the TFRA and SpTFRA are based on the four subsystems ($CeAn$, We, Co, and Op) shown in Figure 4 and correspond to the six stages of the force reconstruction algorithm described in Section 2. The development of the architecture was carried out by evaluating (in software) the functionality of the proposed algorithms. Then, it was implemented in hardware (on a development platform for FPGA) following the classic integrated circuit design flow presented in [61].

To evaluate the functionality and efficiency of the TFRA and SpTFRA hardware implementations, we established two criteria:

Algorithm functionality: TFRA and SpTFRA should be generalizable (i.e., they should depend only on the contact event but not on the transduction technology), scalable (i.e., operate over different contact area sizes), and efficient (i.e., have a predictable runtime) as established in [34]. For the experimental verification of this criterion, we calculate the forces estimation error and the response time, evaluating by simulation, two cases of simple contacts (Hertzian and non-Hertzian) using two pressure-sensor arrays of different resolutions:

- Sensor 1. It is a 10 × 10 taxel array with a $(s_x, s_y) = (4\text{ mm}, 2\text{ mm})$ resolution arranged as a rectangular prism of dimensions 40 mm × 20 mm × 3 mm, on which a single static Hertzian contact was modeled, equivalent to a distribution of elliptical normal stresses measured at the base of the tactile sensor and centered on the coordinates $(x_0, y_0) = (18\text{ mm}, 9\text{ mm})$.
- Sensor 2. It is a 10 × 10 taxel array with a $(s_x, s_y) = (4\text{ mm}, 4\text{ mm})$ resolution arranged as a rectangular prism of dimensions 40 mm × 40 mm × 3 mm, on which a single static non-Hertzian contact was modeled measured at the base of the tactile sensor and centered in the coordinates $(x_0, y_0) = (18\text{ mm}, 18\text{ mm})$.

The TFRA and SpTFRA software implementations were carried out in Matlab® MathWorks™ 2021. To evaluate the estimation error and response time, we compared the Matlab results with those obtained for a finite element analysis (FEA) model implemented in COMSOL Multiphysics® 6.0.

Hardware Efficiency: A hardware efficient implementation of TFRA and SpTFRA should perform triaxial force reconstruction with the lowest error estimation, power consumption, hardware resource usage and memory requirements, the highest throughput, and the shortest response time. That is why, in the design verification process, we consider the corresponding metrics for each aspect and select the best alternative.

The entire hardware design process was carried out on AMD Vitis™ HLS 2022 and Vivado™ Design Suite 2022 running on a laptop based on an Intel Dual-Core i7-4600U CPU at 2.1 GHz and 8 GB of RAM. The SpTFRA hardware implementation was evaluated on a

Zynq UltraScale+ MPSoC ZCU102 FPGA. Both cases used a 32-bit floating-point format and a system clock frequency of 125 MHz.

High-Level Desing Approach of the TFRA and SpTFRA

High-level synthesis (HLS) is a modern and efficient automated design process for digital systems that focuses on mapping behavioral/algorithmic specifications (in C/C++ or SystemC) on register-transfer level (RTL) structures. This technique speeds up the verification procedures at early stage designs, supporting hardware designers to improve functional features while tuning up optimization targets [62], considerably reducing the design effort and supporting the exploration of test and reliability analyses [49,63].

This work exploits the advantages of an HLS tool to implement the TFRA and SpTFRA algorithms in hardware. Before that, a software-level description ($SW_{SF(L|p)}$) was used to better understand the behavior of such algorithms and predict their characteristics at the hardware level ($HW_{SF(L|p)}$) when using the sensors S ($S : 1-2$), the filters F ($F : 1-3$) and performing a sweep on their variables L ($L : 10 : 10 : 90$) or p ($p : 10 : 10 : 90$) for each filter.

The following nomenclature was used for the TFRA software (SW_1, SW_2), and hardware (HW_1, HW_2) implementations, depending on the type of sensor ($S : 1-2$). Similarly, for the SpTFRA, the software and hardware implementations were named $SW_{SF(L|p)}$ and $HW_{SF(L|p)}$, respectively, depending on the type of sensor ($S : 1-2$) and filter used ($F : 1-2$). For example, $SW_{1F1(L10:L90)}$ represents Sensor 1 with Filter 1 and a sweep of L from 10 to 90. Meanwhile, $SW_{1F2(p10:p90)}$ represents Sensor 1 with Filter 2, with a sweep of p from 10 to 90. Finally, the SpTFRA implementations for the case of Filter 3 were named SW_{1F3}, SW_{2F3} and HW_{1F3}, HW_{2F3}, according to the type of sensor used.

The testbench was also implemented in HLS, allowing for reports on hardware synthesis, latency, resources, and power consumption. From test bench results, it was possible to reduce from 42 hardware implementations to only four (two for each sensor).

5. Results

5.1. Sparse Filters Response

Figure 10 shows the friction coefficient μ obtained in the TFRA and SpTFRA software implementations for a reference value $\mu_{ref} = 0.5$ and variations in the angle of the orientation of the tangential forces in the range of ϕ = [0–360°] for the Hertzian (Sensor 1) and non-Hertzian (Sensor 2) single contacts modeled. μ was computed at each ϕ value from the triaxial forces reconstructed by applying Equation (29).

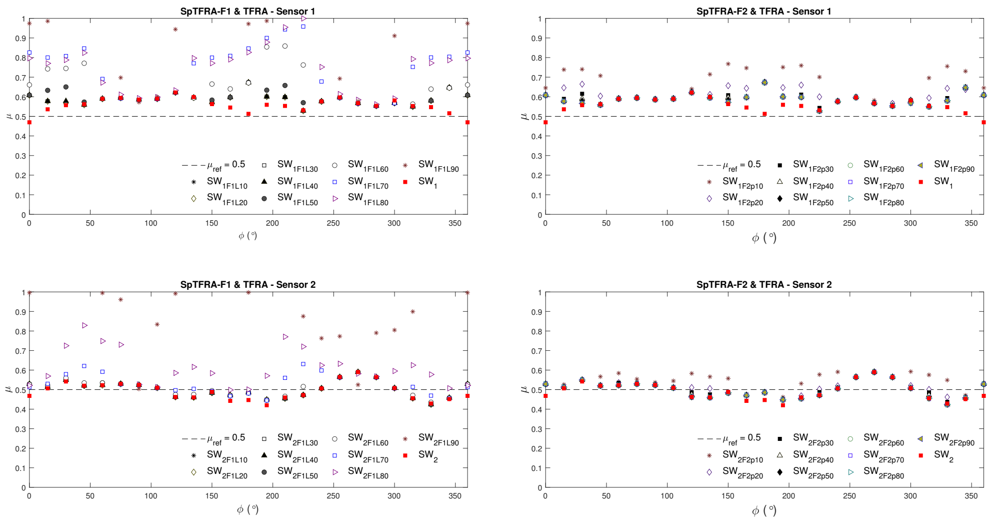

Figure 10. Estimated friction coefficient by applying the SpTFRA−F1 and SpTFRA−F2 filters.

Figure 11 summarizes the behavior of all the filters proposed in Section 3 for each sparse matrix in both algorithms. The values presented were calculated from Equations (27)–(29), in which the number of Nnz is correlated with the estimation error of μ obtained for each filter. In this figure, the best cases are given for the cells with the lowest values, as this corresponds to the minimum error and Nnz. This leads to sparser matrices that require both less memory and fewer processing elements. Similarly, the white-on-orange text cells in Figure 11 represent the best cases for the SpTFRA−F3 filter implementation.

Table 1 describes the filters selected at the hardware design stage, their Nnz values, and the estimation of the average friction coefficient due to the variation of ϕ. The best reconstruction cases were obtained for $L = 40$ and $L = 50$ at SpTFRA−F1 and $p = 30$ and $p = 50$ at SpTFRA−F2 for both sensors.

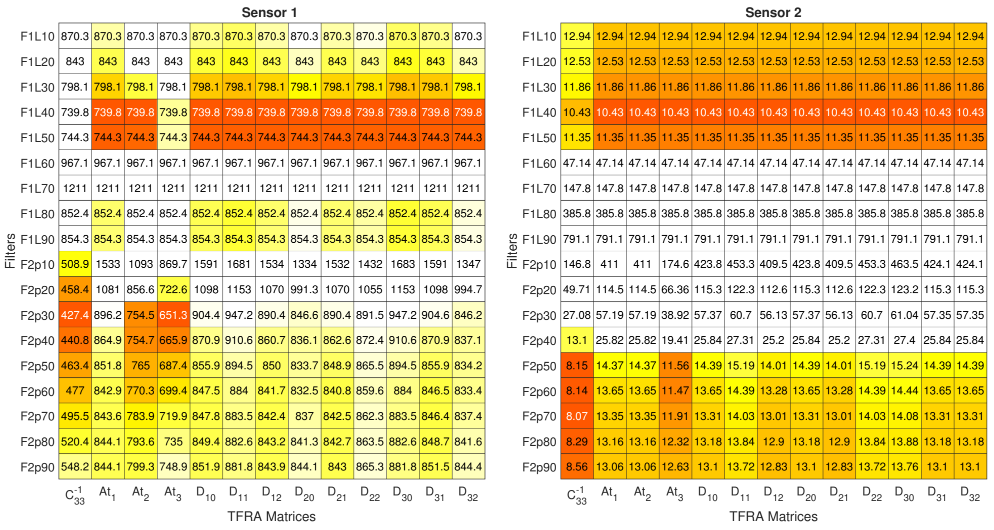

Figure 11. Comparative response of the filters applied to the SpTFRA model evaluating $e_{\mu}(B_{sp})Nnz(B_{sp}))$ for each matrix. Note that the orange cells with white text represent the best cases for SpTFRA−F3.

Table 1. Nnz values per matrix generated at the best reconstruction cases for both sensors.

Sensor	Filter	C_{33}^{-1}	A_{t1}	A_{t2}	A_{t3}	D_{10}	D_{11}	D_{12}	D_{20}	D_{21}	D_{22}	D_{30}	D_{31}	D_{32}	$\overline{\mu}$	HW Name
1	None	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	0.56	HW$_1$
	F1 L = 40	8360	8360	8360	8360	8360	8360	8360	8360	8360	8360	8360	8360	8360	0.59	HW$_{1F1L40}$
	F1 L = 50	7450	7450	7450	7450	7450	7450	7450	7450	7450	7450	7450	7450	7450	0.60	HW$_{1F1L50}$
	F2 p = 30	4512	9462	7966	6876	9548	10,000	9400	8938	9400	9412	10,000	9550	8934	0.59	HW$_{1F2p30}$
	F2 p = 50	5180	9522	8552	7684	9568	10,000	9502	9320	9490	9676	10,000	9568	9326	0.59	HW$_{1F2p50}$
	F3	4512	8360	8360	6876	8360	8360	8360	8360	8360	8360	8360	8360	8360	0.49	HW$_{1F3}$
2	None	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	0.49	HW$_2$
	F1 L = 40	8360	8360	8360	8360	8360	8360	8360	8360	8360	8360	8360	8360	8360	0.50	HW$_{2F1L40}$
	F1 L = 50	7450	7450	7450	7450	7450	7450	7450	7450	7450	7450	7450	7450	7450	0.50	HW$_{2F1L50}$
	F2 p = 30	4036	9294	9294	5388	9360	9932	9146	9360	9146	9932	10,000	9362	9362	0.51	HW$_{2F2p30}$
	F2 p = 50	5348	9430	9430	7588	9440	9968	9196	9440	9196	9968	10,000	9442	9442	0.50	HW$_{2F2p50}$
	F3	5732	8360	8360	8360	8360	8360	7395	8360	8360	8360	8360	8360	8360	0.50	HW$_{2F3}$

5.2. Hardware Resource Consumption

From the information presented in Table 1, the use of digital signal processors (DSP), block random access memory (BRAM), look-up tables (LUT), and Flip-Flops (FF) elements required by each hardware implementation was compared. The results of this comparison are shown in Figure 12. As can be seen, the SpTFRA requires considerably fewer DSP blocks than the original TFRA because it does not need to operate MAC with zero values. Regarding BRAM consumption, the SpTFRA approach does not offer a significant advantage because of the MCSR format and the oversized memory allocation performed by EDA tools (in this case, VITIS HLS™ and Vivado™). Nevertheless, all BRAM requirements for the SpTFRA are for fully integrated memory, reducing memory access times and avoiding needing external hardware. The FFs and LUTs consumption behaves similarly for all the analyzed SpTFRA configurations; however, in all cases, these values are lower than those required by the basic TFRA.

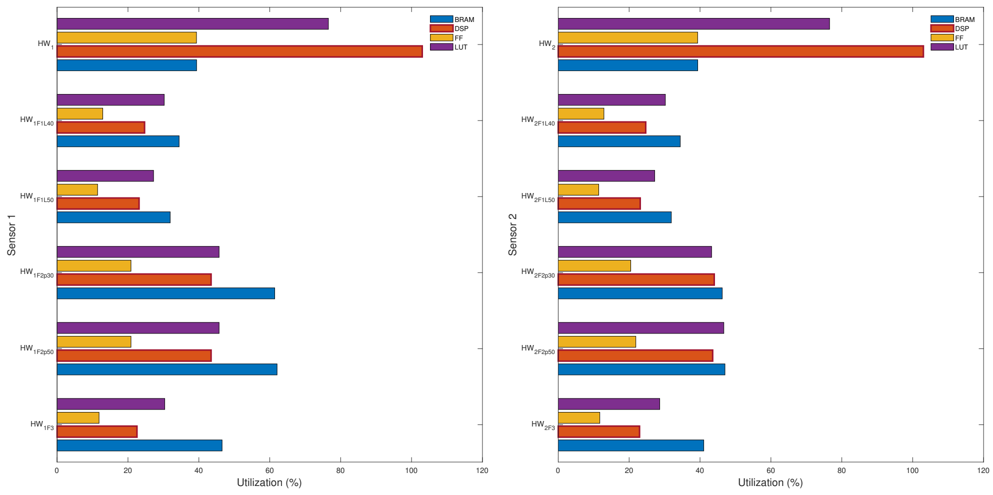

Figure 12. Hardware resource consumption for the TFRA (HW_1, HW_2) and SpTFRA for the two sensors analyzed.

Table 2 presents the main hardware characteristics obtained for the pre-synthesis stage of the SpTFRA implementations for the lowest resource consumption cases. The TFRA hardware implementations HW_1 and HW_2 are shown for reference only, as these are not physically implementable because they exceed 100% of the resources available on the FPGA platform used. TFRA and SpTFRA implementations execute steps i to v (see Figure 4) only once, so the latency and response time of the $CeAn$, We, and Co blocks become predictable.

$CeAn$ is a common block for all hardware implementations and has a latency of 3481 clock cycles for a system clock frequency of 125 MHz. Blocks responsible for processing sparse matrices, such as coefficients calculations Co and weights vector calculation We, show differences in their latency figures due to the variation in the number of operations required for each matrix. Op is also a common block for all proposed hardware implementations and has a latency of 320,527 clock cycles per iteration. However, the total time execution of Op block depends on the number of iterations performed to evaluate the TFRA optimization functions (Equations (20) and (21)).

Table 2. Characteristics of the evaluated TFRA and SpTFRA hardware implementations.

Hardware	Resources Utilization				On-Chip	Latency			Throughput
	BRAM	DSP	FF	LUT	Power [W]	CeAn	We-Co	Op	[MBps]
HW_1	718	2596	215,602	209,814	3992.9 *		859,480		64.67
HW_{1F1L40}	628	622	70,662	82,842	0.871 **	3481	683,252	320,527	65.89
HW_{1F3}	849	568	64,917	83,277	0.894 **		740,387		59.96
HW_2	718	2596	215,602	209,814	3992.9 *		859,480		64.67
HW_{2F2p50}	858	1098	120,014	128,040	0.953 **	3481	804,205	320,527	62.72
HW_{2F3}	749	578	64,287	78,489	0.983 **		844,606		55.32

* Physically not implementable because this case exceeds the hardware platform capability. ** All data in this table were obtained for a junction temperature of around 26 °C.

5.3. Design Verification

Figures 13 and 14 represent the results of the functional validation of the proposed hardware implementations evaluated through behavioral simulation. As observed, the obtained values are close to those expected for the resultant forces and the variables μ and ϕ. These variables verified that the SpTFRA model meets the generalization and scalability criteria when applied to two tactile sensors with different resolutions, two single contact models, and different tangential force orientations.

The values presented in Table 3 were found when evaluating the system response. These include the maximum values of the estimation error, the error distribution, and the maximum response time for the TFRA and SpTFRA implementations studied. The maximum relative error in the estimation of the tangential resultant forces is 7.70%, and of the normal resultant forces is 12.57%. The maximum relative errors in the estimation of μ and ϕ were 14.67% and 1.93%, respectively. These values are close to those obtained by software implementations for the TFRA.

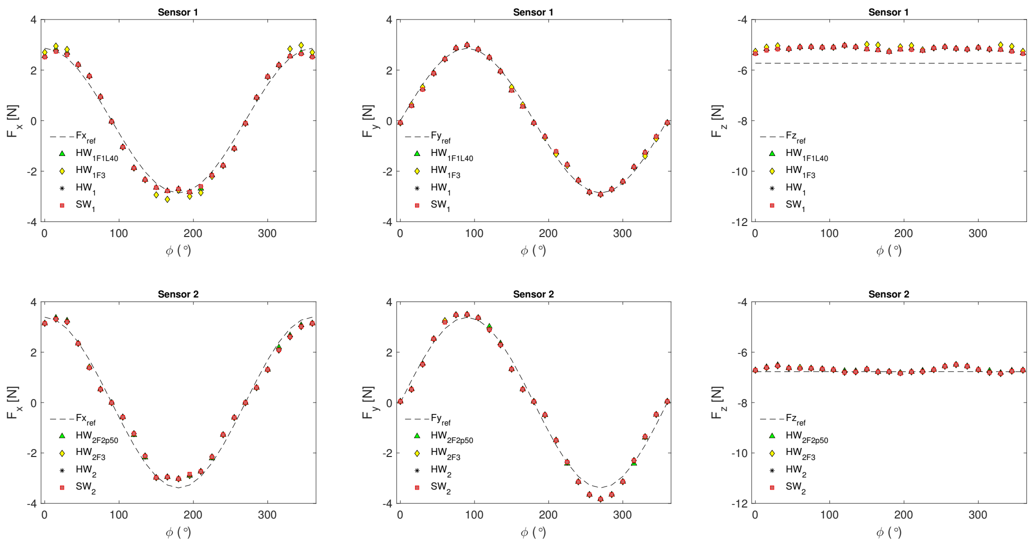

Figure 13. Resultant forces obtained in the TFRA and SpTFRA hardware implementations.

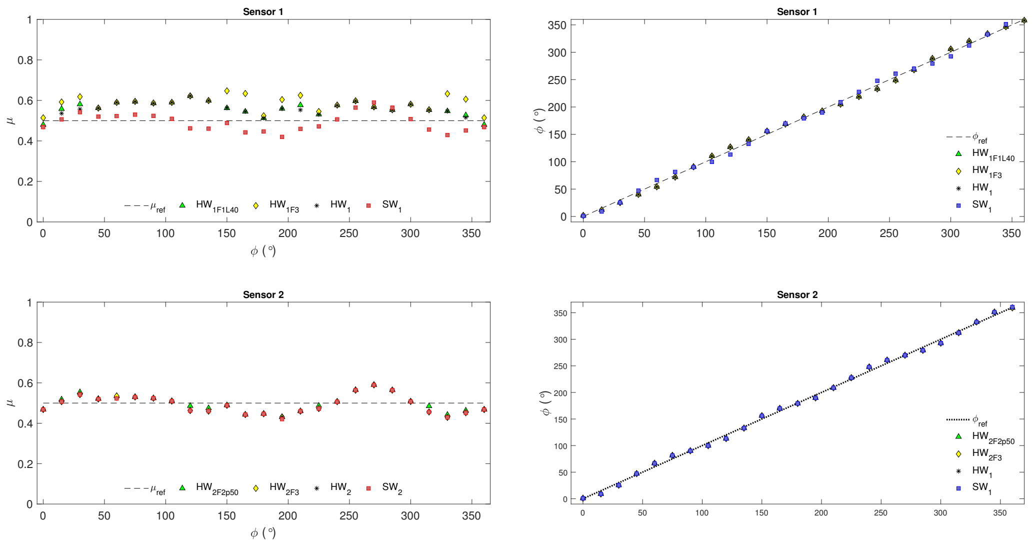

Figure 14. The friction coefficient and tangential force orientation results for SpTFRA hardware implementations.

The obtained results verified that force reconstruction performs better in square tactile sensor arrays (Sensor 2) than rectangular ones (Sensor 1). Performed tests included normal stress input data up to 50,000 N/m^2 to reconstruct forces with magnitudes of the resultant forces on each axis around 6 N. However, the input data can be changed without affecting the hardware implementation, allowing the reconstruction of contact forces at different scales. In this work, force magnitudes were estimated in contact areas up to 40 $\times$ 40 mm^2. Other authors estimate triaxial forces of 1 N on contact areas of 12.5 $\times$ 12.5 mm^2 [2] and 10 N in 14.2 $\times$ 14.2 mm^2 [10] in applications of robotic manipulation and implemented in PC workstations. This demonstrates the capabilities of TFRA and SpTFRA to process triaxial forces with the portability, low power, and high processing power of a high-performance FPGA.

Figure 15 shows the response times obtained by simulating the behavior of the TFRA and SpTFRA implementations on an FPGA operating at a clock frequency of 125 MHz. The force estimation for SpTFRA implementations was performed in an average time of

58.68 ms, with a worst case of 78.88 ms. These results are in the same ranges as other works that report operation in real time, such as in [16] (33 ms), [23] (44 ms), and in [2] (300 ms). However, it should be noted that all those response times were obtained using software implementations running on PC workstations.

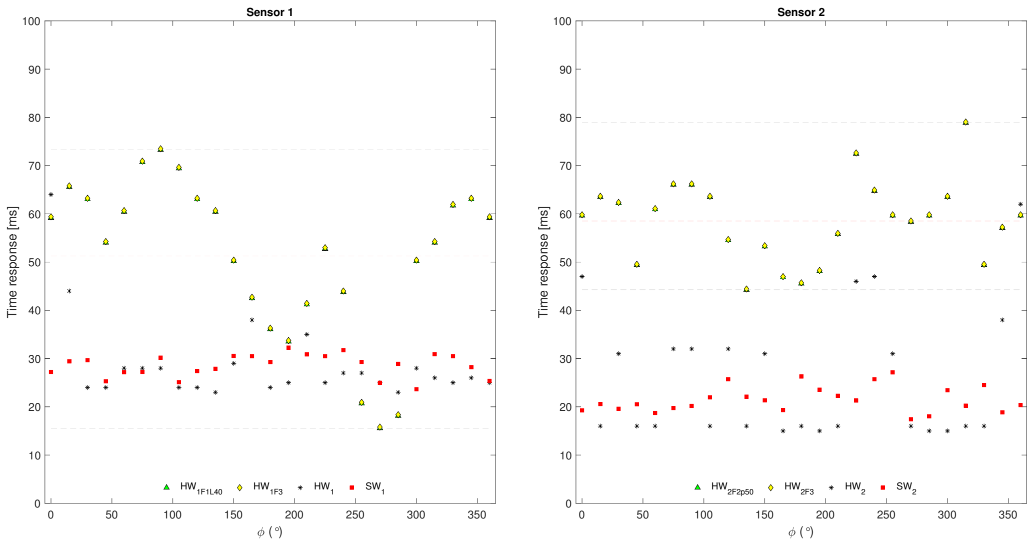

Figure 15. Time response for SpTFRA-HW10 and SpTFRA-HW13.

Table 3. Error estimation and response time.

Hardware	**Maximum Relative Error [%]**					**Standard Error [%]**					**Av. Resp. Time**
	e_{F_x}	e_{F_y}	e_{F_z}	e_μ	e_ϕ	SE_{F_x}	SE_{F_y}	SE_{F_z}	SE_μ	SE_ϕ	$\overline{t_r}$
SW_1	5.94	3.44	10.93	9.41	1.64	2.04	1.13	1.35	3.64	0.50	28.56
HW_1	5.95	3.45	11.87	9.42	1.93	2.03	1.13	1.35	2.75	0.50	28.76
HW_{1F1L40}	7.45	4.79	10.83	12.15	1.93	1.93	1.07	1.27	2.77	0.50	51.27
HW_{1F3}	7.70	4.17	12.57	14.67	1.93	1.81	0.85	1.30	3.59	0.50	51.49
SW_2	7.84	5.44	2.69	4.46	3.16	1.73	1.84	1.02	2.82	0.69	21.53
HW_2	6.12	5.90	3.66	8.97	2.20	1.69	1.83	1.02	2.21	0.68	25.96
HW_{2F2p50}	5.51	5.92	3.37	9.01	2.20	1.60	1.88	1.05	2.23	0.68	58.52
HW_{2F3}	6.12	5.91	3.67	8.99	2.20	1.68	1.84	1.04	2.19	0.68	58.68

6. Conclusions

This work demonstrated an efficient hardware implementation for the reconstruction of triaxial contact forces in a distribution of discrete normal stresses obtained from pressure-sensor arrays based on different transduction technologies, including piezoresistive, resistive, and capacitive, as well as the potential of high-performance embedded platforms, such as FPGAs, to process contact forces in tactile sensing systems.

The presented hardware design approach leverages the generalized matrix–vector multiplication operation to optimize hardware resource consumption by replacing dense matrices with approximate sparse matrices, which is achieved using filters specifically designed for this task.

The use of high-level design tools reduces the design effort by allowing different software and hardware implementation options to be evaluated simultaneously, therefore reducing the data storage requirements, the number of processing elements, and the energy consumption of the developed system.

Future works could include exploring alternatives for data access, addressing, and reuse for TFRA and SpTFRA implementations that take advantage of the on-chip memory availability of FPGAs and other embedded systems to overcome the bottleneck caused by poor memory access times. In addition, since the total response time of the system depends on the number of iterations of the optimization algorithm (stage vi of the TFRA algorithm), it becomes very sensitive to delays in the *Op* block, so efforts could be made to achieve higher task concurrency and explore other minimization techniques in the optimization algorithm.

Author Contributions: Conceptualization, M.-L.P.-S., W.-J.P.-H. and J.A.H.-L.; methodology, M.-L.P.-S., W.-J.P.-H. and J.A.H.-L.; software, M.-L.P.-S., W.-J.P.-H. and J.A.H.-L.; formal analysis, M.-L.P.-S., W.-J.P.-H. and J.A.H.-L.; investigation, M.-L.P.-S., W.-J.P.-H. and J.A.H.-L.; resources, W.-J.P.-H. and J.A.H.-L.; writing—original draft preparation, M.-L.P.-S., W.-J.P.-H. and J.A.H.-L.; writing—review and editing, M.-L.P.-S., W.-J.P.-H. and J.A.H.-L.; supervision, W.-J.P.-H. and J.A.H.-L.; funding acquisition, W.-J.P.-H. and J.A.H.-L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Spanish Government under Grant PID2021-125091OB-I00 and Universidad Pedagógica y Tecnológica de Colombia [VIE DIN Convocatoria No 08-2021 SGI 3160].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this paper can be requested from the corresponding author upon request.

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

References

1. Ciotti, S.; Sun, T.; Battaglia, E.; Bicchi, A.; Liu, H.; Bianchi, M. Soft tactile sensing: Retrieving force, torque and contact point information from deformable surfaces. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 4290–4296. [CrossRef]
2. Han, C.; Cao, Z.; Hu, Y.; Zhang, Z.; Li, C.; Wang, Z.L.; Wu, Z. Flexible Tactile Sensors for 3D Force Detection. *Nano Lett.* **2024**, *24*, 5277–5283. [CrossRef] [PubMed]
3. Kitouni, D.; Chelly, E.; Khoramshahi, M.; Perdereau, V. Fingertip Contact Force Direction Control using Tactile Feedback. *arXiv* **2024**, arXiv:2406.11545.
4. Houhou, Y.; Singh, R.P.; Limon, R.C. Learning to Classify Surface Roughness Using Tactile Force Sensors. In Proceedings of the 2024 IEEE/SICE International Symposium on System Integration (SII), Ha Long, Vietnam, 8–11 January 2024; pp. 1–7. [CrossRef]
5. Roberts, R.D.; Loomes, A.R.; Allen, H.A.; Di Luca, M.; Wing, A.M. Contact forces in roughness discrimination. *Sci. Rep.* **2020**, *10*, 5108. [CrossRef] [PubMed]
6. Higashi, K.; Okamoto, S.; Yamada, Y.; Nagano, H.; Konyo, M. Hardness perception through tapping: Peak and impulse of the reaction force reflect the subjective hardness. In *Haptics: Science, Technology, and Applications: 11th International Conference, EuroHaptics 2018, Pisa, Italy, June 13–16. 2018, Proceedings, Part I 11*; Springer International Publishing: Berlin/Heidelberg, Germany, 2018. [CrossRef]
7. Eguíluz, A.G.; Rañó, I.; Coleman, S.A.; McGinnity, T.M. Reliable object handover through tactile force sensing and effort control in the Shadow Robot hand. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 372–377. [CrossRef]
8. Zhang, T.; Cong, Y.; Li, X.; Peng, Y. Robot Tactile Sensing: Vision Based Tactile Sensor for Force Perception. In Proceedings of the 2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Tianjin, China, 19–23 July 2019; pp. 1360–1365. [CrossRef]
9. Masoumian, A.; Montazer, M.C.; Valls, D.P.; Kazemi, P.; Rashwan, H.A. Using the Feedback of Dynamic Active-Pixel Vision Sensor (Davis) to Prevent Slip in Real Time. In Proceedings of the 2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE), Barcelona, Spain, 12–15 February 2020; pp. 63–67. [CrossRef]
10. Yu, J.; Yao, S.; Li, X.; Ghaffar, A.; Yao, Z. Design of a Three-dimensional Tactile Sensing Array for Incipient Slip Detection in Robotic Dexterous Manipulation. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 9514214. [CrossRef]
11. Niederhauser, L.; Prabhakar, A.; Reber, D.; Billard, A. A Predictive Model for Tactile Force Estimation Using Audio-Tactile Data. *IEEE Robot. Autom. Lett.* **2024**, *9*, 1596–1603. [CrossRef]
12. Lin, C.; Zhang, H.; Xu, J.; Wu, L.; Xu, H. 9DTact: A Compact Vision-Based Tactile Sensor for Accurate 3D Shape Reconstruction and Generalizable 6D Force Estimation. *IEEE Robot. Autom. Lett.* **2024**, *9*, 923–930. [CrossRef]
13. Jeong, H.; Choi, K.; Park, S.J.; Park, C.H.; Choi, H.R.; Kim, U. Rugged and Compact Three-Axis Force/Torque Sensor for Wearable Robots. *Sensors* **2021**, *21*, 2770. [CrossRef]
14. Gong, Y.; Cheng, X.; Wu, Z.; Liu, Y.; Yu, P.; Hu, X. A Flexible Tactile Sensor Array for Dynamic Triaxial Force Measurement Based on Aligned Piezoresistive Nanofibers. *IEEE Sens. J.* **2021**, *21*, 21989–21998. [CrossRef]
15. Zhang, Y.; Zeng, J.; Wang, Y.; Jiang, G. Flexible Three-Dimensional Force Tactile Sensor Based on Velostat Piezoresistive Films. *Micromachines* **2024**, *15*, 486. [CrossRef]
16. Xu, D.; Hong, W.; Hu, B.; Zhang, T.; Chen, D.; Yan, Z.; Yao, X.; Zhang, X.; Zhao, Y.; Sun, T.; et al. River valley-inspired, high-sensitivity, and rapid-response capacitive three-dimensional force tactile sensor based on U-shaped groove structure. *Smart Mater. Struct.* **2024**, *33*, 35006. [CrossRef]

17. Li, H.; Nam, S.; Lu, Z.; Yang, C.; Psomopoulou, E.; Lepora, N.F. BioTacTip: A Soft Biomimetic Optical Tactile Sensor for Efficient 3D Contact Localization and 3D Force Estimation. *IEEE Robot. Autom. Lett.* **2024**, *9*, 5314–5321. [CrossRef]
18. Yuan, W.; Dong, S.; Adelson, E.H. GelSight: High-Resolution Robot Tactile Sensors for Estimating Geometry and Force. *Sensors* **2017**, *17*, 2762. [CrossRef] [PubMed]
19. Lambeta, M.; Chou, P.W.; Tian, S.; Yang, B.; Maloon, B.; Most, V.R.; Stroud, D.; Santos, R.; Byagowi, A.; Kammerer, G.; et al. DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor with Application to In-Hand Manipulation. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3838–3845. [CrossRef]
20. Wang, Z.; Zhang, D.; Yang, L.; An, J. Three-dimensional force detection using PVDF and room temperature-vulcanized silicone rubber layers. *Meas. Sci. Technol.* **2023**, *34*, 45111. [CrossRef]
21. Yu, P.; Liu, W.; Gu, C.; Cheng, X.; Fu, X. Flexible piezoelectric tactile sensor array for dynamic three-axis force measurement. *Sensors* **2016**, *16*, 819. [CrossRef]
22. Song, Y.; Zhang, Y.; Wang, F.; Jiang, X.; Sun, N. Three-dimensional Force Detection and Decoupling of a Flexible Tactile Sensor Array based on Porous Composite Piezoresistive Materials. *Appl. Math. Nonlinear Sci.* **2024**, *9*, 1–23. [CrossRef]
23. Yu, H.; Guo, H.; Wang, J.; Zhao, T.; Zou, W.; Zhou, P.; Xu, Z.; Zhang, Y.; Zheng, J.; Zhong, Y.; et al. Skin-Inspired Capacitive Flexible Tactile Sensor with an Asymmetric Structure for Detecting Directional Shear Forces. *Adv. Sci.* **2024**, *11*, 2305883. [CrossRef]
24. Lee, D.H.; Kim, U.; Jung, H.; Choi, H.R. A Capacitive-Type Novel Six-Axis Force/Torque Sensor for Robotic Applications. *IEEE Sens. J.* **2016**, *16*, 2290–2299. [CrossRef]
25. Park, S.; Hwang, D. Three-Axis Flat and Lightweight Force/Torque Sensor for Enhancing Kinesthetic Sensing Capability of Robotic Hand. *IEEE Trans. Ind. Electron.* **2024**, *71*, 12707–12717. [CrossRef]
26. Liang, G.; Wang, Y.; Mei, D.; Xi, K.; Chen, Z. Flexible Capacitive Tactile Sensor Array With Truncated Pyramids as Dielectric Layer for Three-Axis Force Measurement. *J. Microelectromech. Syst.* **2015**, *24*, 1510–1519. [CrossRef]
27. Sun, Y.; Liu, F.; Yuan, Z.; Huang, W.; Wang, B. A Novel Three-Axial Force Tactile Sensor Based on the Fringing Effect of Electric Field. *IEEE Trans. Magn.* **2019**, *55*, 7500305. [CrossRef]
28. Zhang, B.; Wang, B.; Li, Y.; Jin, S. Magnetostrictive tactile sensor of detecting friction and normal force for object recognition. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420932327. [CrossRef]
29. Hellebrekers, T.; Chang, N.; Chin, K.; Ford, M.; Kroemer, O.; Majidi, C. Soft Magnetic Tactile Skin for Continuous Force and Location Estimation using Neural Networks. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3892–3898. [CrossRef]
30. Ogli, O.G.B.; Matyunin, S.A. Experimental Study of the Characteristics of Macro-Bending Fiber-Optic Sensors of Tactile Force for Anthropomorphic Robot Grippers. In Proceedings of the 2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon), Vladivostok, Russia, 1–4 October 2019; pp. 1–6. [CrossRef]
31. Baimukashev, D.; Kappassov, Z.; Varol, H.A. Shear, Torsion and Pressure Tactile Sensor via Plastic Optofiber Guided Imaging. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2618–2625. [CrossRef]
32. Mohammadi, A.; Xu, Y.; Tan, Y.; Choong, P.; Oetomo, D. Magnetic-based soft tactile sensors with deformable continuous force transfer medium for resolving contact locations in robotic grasping and manipulation. *Sensors* **2019**, *19*, 4925. [CrossRef]
33. Kim, H.; Choi, S.; Chung, W.K. Contact Force Decomposition Using Contact Pressure Distribution. *IEEE Robot. Autom. Lett.* **2017**, *2*, 290–297. [CrossRef]
34. Wasko, W.; Albini, A.; Maiolino, P.; Mastrogiovanni, F.; Cannata, G. Contact Modelling and Tactile Data Processing for Robot Skins. *Sensors* **2019**, *19*, 814. [CrossRef] [PubMed]
35. Xiong, L.; Guo, Y.; Jiang, G.; Zhou, X.; Jiang, L.; Liu, H. Six-Dimensional Force/Torque Sensor Based on Fiber Bragg Gratings With Low Coupling. *IEEE Trans. Ind. Electron.* **2021**, *68*, 4079–4089. [CrossRef]
36. Almassri, A.M.; Wan Hasan, W.Z.; Ahmad, S.A.; Ishak, A.J.; Ghazali, A.M.; Talib, D.N.; Wada, C. Pressure Sensor: State of the Art, Design, and Application for Robotic Hand. *J. Sens.* **2015**, *2015*, 846487. [CrossRef]
37. Zheng, W.; Liu, K.; Guo, D.; Yang, W.; Zhu, J.; Liu, H. A Large-area Tactile Sensor for Distributed Force Sensing Using Highly Sensitive Piezoresistive Sponge. In Proceedings of the 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 13–17 May 2024; pp. 1063–1069. [CrossRef]
38. Huang, T.Y.; Tseng, S.H.; Lu, M.S.C. Design and Characterization of a CMOS Capacitive Sensor Array for Fast Normal Stress Analysis. *IEEE Sens. Lett.* **2022**, *6*, 2500604. [CrossRef]
39. Kane, B.J.; Cutkosky, M.R.; Kovacs, G.T.A. A traction stress sensor array for use in high-resolution robotic tactile imaging. *J. Microelectromech. Syst.* **2000**, *9*, 425–434. [CrossRef]
40. Sun, X.; Zhang, M. A pragmatic data processing system for large resistive sensor arrays. *Rev. Sci. Instrum.* **2024**, *95*, 85108. [CrossRef] [PubMed]
41. Seminara, L.; Capurro, M.; Valle, M. Tactile data processing method for the reconstruction of contact force distributions. *Mechatronics* **2015**, *27*, 28–37. [CrossRef]
42. Pinto-Salamanca, M.L.; Castellanos-Ramos, J.; Pérez-Holguín, W.J.; Hidalgo-López, J.A. An estimation of triaxial forces from normal stress tactile sensor arrays. *Mechatronics* **2023**, *96*, 103070. [CrossRef]
43. Podlubne, A.; Mey, J.; Andreou, A.; Pertuz, S.; Aßmann, U.; Göhringer, D. Model-Based Generation of Hardware/Software Architectures With Hybrid Schedulers for Robotics Systems. *IEEE Trans. Comput.* **2024**, *73*, 1640–1654. [CrossRef]

44. Lora-Rivera, R.; Oballe-Peinado, Ó.; Vidal-Verdú, F. Texture Detection With Feature Extraction on Embedded FPGA. *IEEE Sens. J.* **2023**, *23*, 12093–12104. [CrossRef]
45. Al Haj Ali, H.; Abbass, Y.; Gianoglio, C.; Ibrahim, A.; Oh, C.; Valle, M. Neuromorphic Tactile Sensing System for Textural Features Classification. *IEEE Sens. J.* **2024**, *24*, 17193–17207. [CrossRef]
46. Magno, M.; Ibrahim, A.; Pullini, A.; Valle, M.; Benini, L. An Energy Efficient E-Skin Embedded System for Real-Time Tactile Data Decoding. *J. Low Power Electron.* **2018**, *14*, 101–109. [CrossRef]
47. Alameh, M.; Abbass, Y.; Ibrahim, A.; Valle, M. Smart tactile sensing systems based on embedded CNN implementations. *Micromachines* **2020**, *11*, 103. [CrossRef]
48. Mendoza-Peñaloza, J.; Muñoz, D.M. Hardware Implementation of a Sliding Detection Algorithm for Robotic Hands Using Force Sensors. In Proceedings of the 2023 36th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI), Rio de Janeiro, Brazil, 28 August–1 September 2023; pp. 1–6. [CrossRef]
49. Pinto-Salamanca, M.L.; Condia, J.E.R.; Hidalgo-López, J.A.; Pérez-Holguín, W.J. Analyzing the Reliability of Stream Sparse Matrix-Vector Multiplication Accelerators: A High-Level Approach. In Proceedings of the 2024 IEEE 25th Latin American Test Symposium (LATS), Maceio, Brazil, 9–12 April 2024; pp. 1–4. [CrossRef]
50. Podio-Guidugli, P.; Favata, A. *Elasticity for Geotechnicians: A Modern Exposition of Kelvin, Boussinesq, Flamant, Cerruti, Melan, and Mindlin Problems*, 2014th ed.; Springer Nature: Berlin/Heidelberg, Germany, 2013; Volume 204.
51. Johnson, K.L. *Contact Mechanics*; Cambridge University Press: Cambridge, UK, 1985. [CrossRef]
52. Albert, A. *Regression and the Moore-Penrose Pseudoinvers*; Academic Press: Cambridge, MA, USA, 1972; Volume 94, pp. 1–180.
53. Li, S.; Liu, D.; Liu, W.L. Efficient FPGA-Based Sparse Matrix–Vector Multiplication With Data Reuse-Aware Compression. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2023**, *42*, 4606–4617. [CrossRef]
54. Kusakabe, R.; Fujita, K.; Ichimura, T.; Hori, M.; Lalith, M. GPU-Accelerated Sparse Matrix Vector Product based on Element-by-Element Method for Unstructured FEM using OpenACC. In Proceedings of the 2022 Workshop on Accelerator Programming Using Directives (WACCPD), Dallas, TX, USA, 13–18 November 2022; pp. 52–61. [CrossRef]
55. Tran, H.D.; Fernando, M.; Saurabh, K.; Ganapathysubramanian, B.; Kirby, R.M.; Sundar, H. A scalable adaptive-matrix SPMV for heterogeneous architectures. In Proceedings of the 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Lyon, France, 30 May–3 June 2022; pp. 13–24. [CrossRef]
56. Sadi, F.; Fileggi, L.; Franchetti, F. Algorithm and hardware co-optimized solution for large SpMV problems. In Proceedings of the 2017 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 12–14 September 2017; pp. 1–7. [CrossRef]
57. Serrano, M.J. Efficient implementation of sparse matrix-sparse vector multiplication for large scale graph analytics. In Proceedings of the 2019 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 24–26 September 2019; pp. 1–7. [CrossRef]
58. Nunez-Yanez, J.; Hosseinabady, M. Sparse and dense matrix multiplication hardware for heterogeneous multi-precision neural networks. *Array* **2021**, *12*, 100101. [CrossRef]
59. Lehnert, A.; Holzinger, P.; Pfenning, S.; Müller, R.; Reichenbach, M. Most Resource Efficient Matrix Vector Multiplication on FPGAs. *IEEE Access* **2023**, *11*, 3881–3898. [CrossRef]
60. Hosseinabady, M.; Nunez-Yanez, J.L. A Streaming Dataflow Engine for Sparse Matrix-Vector Multiplication Using High-Level Synthesis. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *39*, 1272–1285. [CrossRef]
61. Kahng, A.B.; Lienig, J.; Markov, I.L.; Hu, J. *VLSI Physical Design: From Graph Partitioning to Timing Closure*; Springer: Dordrecht, The Netherlands, 2011.
62. Skalicky, S.; Wood, C.; Łukowiak, M.; Ryan, M. High level synthesis: Where are we? A case study on matrix multiplication. In Proceedings of the 2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig), Cancun, Mexico, 9–11 December 2013; pp. 1–7.
63. Fibich, C.; Horauer, M.; Obermaisser, R. Reliability-Enhanced High-Level Synthesis using Memory Profiling and Fault Injection. In Proceedings of the IEEE 28th International Symposium on Industrial Electronics (ISIE), Vancouver, BC, Canada, 12–14 June 2019; pp. 1363–1370.