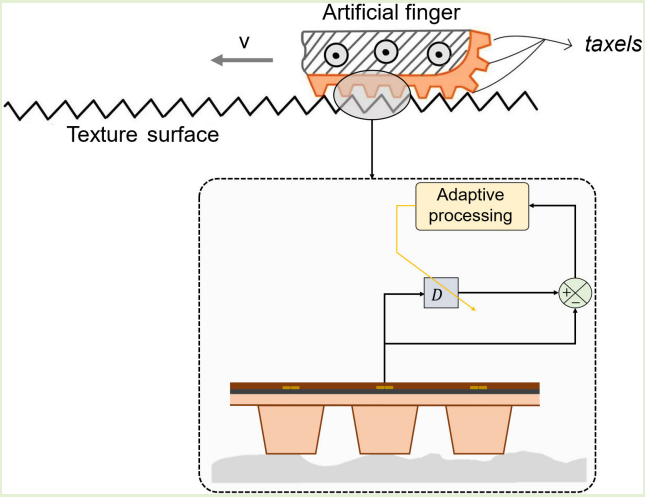
Texture Detection With Feature Extraction on Embedded FPGA

Raúl Lora-Rivera, Óscar Oballe-Peinado, and Fernando Vidal-Verdú

***Abstract*—A feature extraction algorithm for texture detection oriented to its implementation on embedded electronics based on a field-programmable gate array (FPGA) is proposed in this article. Local preprocessing with smart tactile sensors can help to improve dexterity in artificial hands. Simplicity is the goal in order to achieve a hardware-friendly strategy that can be replicated and integrated with other circuitry. This is interesting, considering that tactile sensors are arrays and FPGAs are capable of parallel execution. The proposal was tested with a custom smart tactile sensor mounted on a Cartesian robot to explore different textures. A comparison with a common feature extraction approach based on the fast Fourier transform (FFT) computation was also made. In addition, the whole procedure is implemented on a system on chip (SoC) with the feature extraction on the embedded FPGA and a *k*-means classifier on an advanced RISC machine (ARM) core. The proposed algorithm obtains the spatial frequency components of the tactile signal but not their power. Therefore, some information is lost with respect to that provided by the FFT. Nevertheless, an 89.17% accuracy of the proposed algorithm is obtained versus 91.4% with the FFT when 12 different textures are considered, including complex and fabric textures. There is a noticeable saving in power and hardware resources. In addition, since the size of the feature vector is much smaller, data traffic and memory usage are much lower, and the classifier can be simpler.**



***Index Terms*— Active touch, tactile sensor, texture detection.**

I. Introduction

TACTILE sensors constitute a relevant tool to provide information about the properties of an object [1]. They are especially useful in the case of artificial hands for robotics and prosthetics [1], [2], [3].

The texture is one of the main properties to characterize an object [1]. A first strategy to discriminate textures is based on the processing of the tactile image taken by a high-resolution tactile sensor [4], [5]. This approach can take advantage of resources for feature extraction in image processing [6], and also of their efficient implementation on embedded systems [7], and specifically on field-programmable gate arrays (FPGAs) [8]. Nevertheless, there is still a significant need for computational means, power consumption, and data traffic. Moreover, since there is not any dynamic interaction between the sensor and the explored surface, no information about friction is captured.

Most reported works on texture discrimination are based on active touch, where the sensor and the explored surface move with respect to each other. Active touch allows the detection of features beyond the limitations of the spatial resolution of a tactile sensor. The discrimination of textures can be faced by detecting the microvibrations produced by normal and shear contact forces when different surfaces are explored [9], [10], [11], [12]. This is how the human touch sense works to obtain information about the characteristics of the surface of an object [2].

In artificial systems, the tactile data gathered through active touch are preprocessed to obtain a set of descriptors or feature vectors. These descriptors are typically the input

Manuscript received 28 February 2023; revised 10 April 2023; accepted 17 April 2023. Date of publication 25 April 2023; date of current version 31 May 2023. This work was supported in part by the Spanish Government with a Formación de Profesorado Universitario (FPU) Grant given by the Ministerio de Ciencia, Innovacion y Universidades and in part by the European Regional Development Fund (ERDF) Program Funds under Contract PID2021-125091OB-I00. The associate editor coordinating the review of this article and approving it for publication was Dr. Jurgen Kosel. *(Corresponding author: Raúl Lora-Rivera.)*

Raúl Lora-Rivera is with the Escuela de Ingenierías Industriales, Universidad de Málaga, 29071 Málaga, Spain and also with the Instituto de Investigación Biómedica de Málaga (IBIMA), 29071 Málaga, Spain (e-mail: raul.lora@uma.es).

Óscar Oballe-Peinado and Fernando Vidal-Verdú are with the Escuela de Ingenierías Industriales, Universidad de Málaga, 29071 Málaga, Spain and also with the Instituto Universitario de Investigación en Ingeniería Mecatrónica y Sistemas Ciberfísicos (I3MSC), 29071 Málaga, Spain (e-mail: oballe@uma.es; fvidal@uma.es).

Digital Object Identifier 10.1109/JSEN.2023.3268794

for classification algorithms. Some authors provide the raw tactile dataset from the sensors as input of the classifier [13]. This makes learning more complex and increases the need for computational resources and the risk of overfitting. Therefore, a reduced set of features is commonly obtained from the raw data, such as classical statistical features [14]. Other authors estimate or measure the friction coefficient and use it together with other features that provide information about the signal power, complexity, and frequency content [15], [16]. A set of descriptors robust to changes in the exploratory parameters is proposed in [17]. These parameters provide information about the total power, mean frequency, and bandwidth of the signal, and also about the linear and nonlinear correlations of the signals from different sensors.

Nevertheless, the majority of works that use active touch are based on the computation of the fast Fourier transform (FFT). The Fourier coefficients are then used as input of the classification stage [1], [10], [18], [19], [20], [21]. In this approach, the speed of the relative displacement between the sensor and the surface must be known and constant. The dimension of the feature vector can be reduced using principal component analysis (PCA) to facilitate the learning and classification, especially if the information from many *taxels* (force sensing units) is used [22].

Despite being successful in texture discrimination, the above procedures to obtain a feature vector are complex to implement in embedded systems such as smart sensors in artificial hands. This means high consumption of power and computational resources, as well as data traffic. A different approach sacrifices some performance to obtain benefits in low-cost embedded realizations. This is the case of the algorithm proposed in [23], where the frequency spectrum is divided into bands and the vector of the power in each band is used as a feature. The computations are performed in the time domain and the time and space complexity of the feature extraction algorithm are both linear with the number of bands, which is much lower than the number of coefficients of the FFT.

Another algorithm to measure the surface roughness is reported in [24]. It consists of a simple adaptive bandpass filter whose spatial center frequency is changed through a gradient-descent method until the root mean square of the sensor output reaches a maximum. A single maximum is detected, therefore only a single characteristic frequency of the roughness is provided, and it is the first one that is found by the gradient-descent algorithm, so it can be a local maximum. This article presents an algorithm to identify textures based on that in [24]. The number of sensing elements is reduced to one, being six in [24], which simplifies and makes the result more robust to variability between different *taxels*. Moreover, a sweep procedure is proposed instead of the gradient descent one, which is simpler to implement in the local electronics of embedded systems. This sweep procedure scans the frequency spectrum to find the main characteristic frequencies that identify a given texture. This simple approach is suitable for embedded smart tactile sensors. Moreover, since tactile sensors are arrays, FPGA-based electronics that allow parallel processing is especially appropriate to build these devices [25], [26].

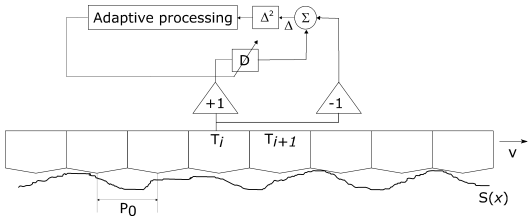


Fig. 1. Illustration of the texture detection with the proposed realization [27].

Some preliminary results of this proposal were presented in the conference paper by Lora-Rivera et al. [27]. In that work, the feature extraction algorithm was implemented on MATLAB,[1] and the results with four basic textures show that the main spatial frequency of the textures could be easily obtained from the first minimum of a certain function. On the other hand, the whole texture identification procedure is implemented in the work of this article, with the feature extraction on an embedded FPGA, and a classifier on an advanced RISC machine (ARM)[1] core. This architecture resembles one where the local electronics can obtain the main tactile features and send them to more powerful processors, reducing latency, and bus traffic in robots. Twelve complex textures are used to test the idea, and the feature extraction is also done with a usual procedure based on the FFT for comparison purposes. The location of all minima is provided, so information about more frequency components is given in a small-size feature vector. The tactile data were obtained with a modified version of the tactile sensor in [25]. The simplicity of the algorithm also allows replicating the circuitry to process the signal from several *taxels* in the tactile array, such as [17] and [22].

The rest of this article is organized as follows. The proposed feature extraction algorithm is presented in Section II. Section III describes the materials and methods. Section IV is devoted to the implementation of the approach on an SoC. Finally, the results and related discussions are shown in Section V, while Section VI summarizes the main conclusions of the work and point to future extensions.

II. Feature Extraction Algorithm

Fig. 1 illustrates the exploration of a texture $S(x)$ through active touch at v linear speed with a tactile sensor. Consider the expression

$$V = \frac{1}{M}\sum_{n=1}^{M}(T_j(vt_n) - T_j(vt_n + P))^2 \tag{1}$$

where M is the number of samples, t_n is the time of the sample n, and P is a variable parameter. Note that V in (1) is proportional to the square of the Euclidean distance between vectors $\mathbf{T_j} = [T_j(vt_1), \ldots, T_j(vt_M)]$ and $\mathbf{T_{j+P}} = [T_j(vt_1 + P), \ldots, T_j(vt_M + P)]$. If the texture $S(x)$ is modeled as a sine wave

$$S(x) = A\sin\left(\frac{2\pi}{\lambda}x\right) \tag{2}$$

[1]Registered trademark.

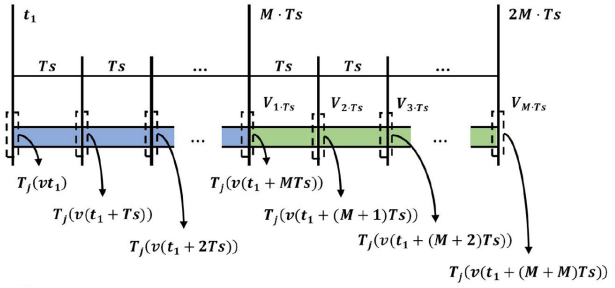


Fig. 2. Timing diagram for the computation of V in (1) for $1 \leq m \leq M$ [27].

where λ is its wavelength. In this case, $\mathbf{T_j}$ and $\mathbf{T_{j+P}}$ are composed of two sets of samples of that sine wave that represent two instances of the wave with a difference in phase of $(2\pi/\lambda)P$: $A\sin((2\pi/\lambda)vt_n)$ and $A\sin((2\pi/\lambda)(vt_n + P))$. Since there is a minimum in V if they are in phase, there is a minimum of $P = \lambda \cdot k$, where k is an integer. For convenience, we can write $P = vD$, where D is a time delay and the minima of V is given by

$$v \cdot D = \lambda \cdot k. \tag{3}$$

Note that the finding of the delays D of these minima allows obtaining a set of spatial wavelengths as features from (3) to characterize a given texture.

To find these minima of (1), a simple procedure consists in computing it for D changing in a certain interval, and localizing the minima with a peak detection algorithm. The highest range and scan resolution in this procedure is achieved when $D = mT_s$, with $1 \leq m \leq M$, and being T_s the sampling period. Since $t_n = nT_s$ and $1 \leq n \leq M$ in (1), the sweep is completed when $m = M$, therefore the sweep time is (see Fig. 2)

$$T_{\text{sweep}} = 2MT_s. \tag{4}$$

Note that the number of samples that have to be acquired to compute (1) for $1 \leq m \leq M$ is $2M$. Note also that this basic approach has a tradeoff between the sweep time in (4) and the minimum spatial frequency or corresponding maximum spatial wavelength $\lambda_{\max}$ to detect given by

$$\lambda_{\max} = MvT_s. \tag{5}$$

Regarding the maximum spatial frequency or minimum wavelength, from the Nyquist–Shannon theorem we have

$$\lambda_{\min} = 2vT_s. \tag{6}$$

III. Materials and Methods

A. Sensor Technology

The tactile data for this work was acquired with the smart tactile sensor in Fig. 3 [25], [27]. It consists of a sheet of sensitive material in contact with an array of electrodes made on a semi-rigid printed circuit board (PCB). This way, an array of force-variable resistance elements was implemented to build a piezoresistive tactile sensor. In this work, the sensitive material is Linqstat[1] MVCF-40012BT50KS/2A, with a sheet resistance of 50 000 Ω/□ and 0.1 mm thickness. To reduce crosstalk between *taxels* in the tactile array, the area of the material associated with every *taxel* is isolated from the rest with laser [28]. Moreover, a 3-D printed flexible cover was added atop of the sensitive material. This cover was made of Filaflex[1], a thermoplastic material with high elasticity [29]. It is shaped to have one cylinder per *taxel* in the tactile array. This cylinder concentrates the force to improve the sensitivity and also reduces the crosstalk between *taxels* when compared with a continuous flexible layer. In addition, regarding the aim of the work of this article, the cylinders interact with the explored surface, which allows detecting variations not only of the normal force but also of the shear force. This is also made in [30], where an array of micro-pyramids is implemented on the cover. The size of the tactile sensor is 40.7 × 15.0 mm and the spatial resolution, and the distance between centers of two adjacent *taxels* is 3.70 mm. It is worth highlighting that these dimensions are large compared with the wavelengths that can be detected with the proposed strategy and active touch.

Regarding the electronics, it is based on an FPGA (Spartan-6[1]) because parallel strategies can be implemented, which is especially interesting for array sensors. Moreover, a direct connection sensor-FPGA proposed by the authors also allows parallel data acquisition. The sampling frequency of the whole tactile array is $F_s = 485$ Hz, so the sampling period is $T_s = 2.06$ ms.

B. Experimental Setup

Fig. 4 shows the experimental setup built to obtain the data and results of this article. The artificial finger with the smart tactile sensor was mounted on a Cartesian robot to acquire data from the texture explorations. The smart finger is connected through a serial peripheral interface (SPI) bus to a AVNET[1] ZedBoard[2] development board. This board is based on the Zynq[2]-7000 SoC XC7Z020-CLG484-1 device [31], which contains an FPGA together with an ARM[1] dual-core Cortex[2] A9 processor. The development board is connected to a personal computer through Ethernet.

C. Texture Samples

In order to cover a wide range of spatial frequencies, the complex textures in Fig. 5 were used. The number of samples to acquire is first established to $M = 1024$, and the sampling frequency is $Fs = 485$ Hz. Taking into account that the exploration speed is $v = 30$ mm/s, the range of exploration in terms of wavelength is from 0.12 to 63.34 mm [see (4)–(6)]. Spatial frequencies outside this range cannot be identified. The 12 textures in Fig. 5 contain their main frequency components inside this range. Details about these textures are provided in Table I. A first set of textures were made of Filaflex[1] (the same flexible material of the tactile sensor cover, see Section III-A) with a 3-D printer (textures #TEX-1, #TEX-4, #TEX-5, and #TEX-6). A second set was made of PLA[1], which is quite rigid [32], also with a 3-D printer (#TEX-2 and #TEX-3). A third set was made with laser engraving on methacrylate (#TEX-7, #TEX-8, and #TEX-9). They were obtained using pulsewidth modulation (PWM),

[2]Trademarked.

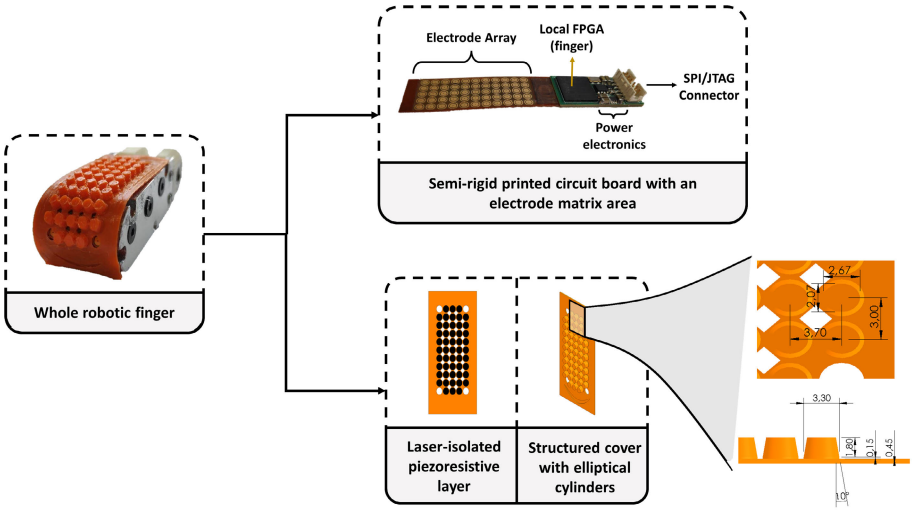


Fig. 3. Parts of the artificial finger. The sensor is composed by a semi-rigid printed board and a structured cover to focus force on the *taxels*. Measures are given in mm.

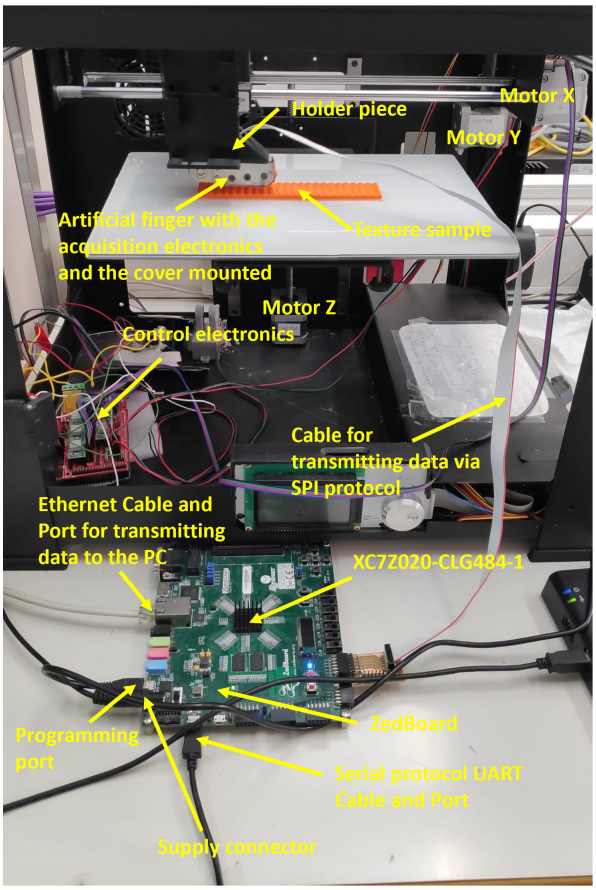


Fig. 4. Experimental setup with the different parts used to explore the textures. The ZedBoard[2] with its different input–output ports and its Zynq-based core is shown.

with a sinusoidal modulator signal of wavelength λ_m and amplitude A_m and a triangular carrier signal of wavelength λ_c and amplitude A_c [33]. The parameters m_a and m_f in Table I are defined as $m_a = (A_m/A_c)$ and $m_f = (\lambda_c/\lambda_m)$. Finally, another set of textures were samples of different fabrics (#TEX-10, #TEX-11, and #TEX-12).

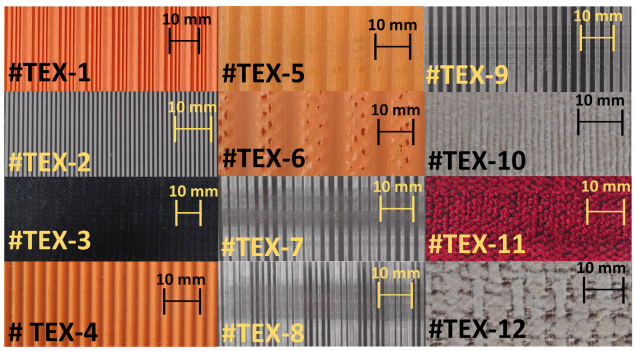


Fig. 5. Top photographs of the 12 textures used in this work.

D. Data Gathering Procedure

Fig. 6 shows the sequence to obtain data from the textures. In particular, Fig. 6(a) shows the initial position of the finger with the tactile sensor, and Fig. 6(b) shows a snapshot of the slide movement across the surface until it reaches the end position in Fig. 6(c). This sequence was repeated at $v = 30$ mm/s 200 times per texture. During this process, the sensor took data in real time, and these data were registered in text files. Every exploration depicted in Fig. 6 provided a 2048-samples vector that is called $frame$ herein, and each sample is a 16-bit length word. Since 12 textures are considered, the total number of frames is 12 textures $\cdot$ 200 frames/texture $=$ 2400, which are 9.4 MB of data. Fig. 7 shows a set of example frames of each texture as obtained after antialiasing filtering.

E. Feature Extraction for Training and Test

As said in Section I, the raw data obtained after the procedure described in Section III-D was processed to reduce the amount of data to be transmitted and stored, and simplify the classifier. Specifically, the vector of minima of (1) can be used as feature vector to identify a given texture. Fig. 8 depicts the result of (1) for the sweep algorithm and the set of textures used in this work. If a polished surface is explored, no minima

TABLE I
MAIN PROPERTIES OF THE 12 TEXTURES EMPLOYED ON THIS WORK

Texture	Material	Modelled by	Defined from spatial wavelength (λ)
#TEX-1	Filaflex	Chirp signal $y = sin(\frac{20\pi x^2}{369} + \frac{x}{3})$, where x is distance, from 0 to 12.3 mm	–
#TEX-2	PLA	Square wave	$\lambda = 1.3$ mm
#TEX-3	PLA	y = $\begin{cases} 1, if(0 \text{ mm} \leq x \leq 1 \text{ mm}, 2 \text{ mm} \leq x \leq 3 \text{ mm}) \\ 0, if(1 \text{ mm} \leq x \leq 2 \text{ mm}, 3 \text{ mm} \leq x \leq 5 \text{ mm}) \end{cases}$	–
#TEX-4	Filaflex	Sinusoidal wave	$\lambda = 3.25$ mm
#TEX-5	Filaflex	Sinusoidal wave	$\lambda = 6.5$ mm
#TEX-6	Filaflex	Sinusoidal wave	$\lambda = 13$ mm
#TEX-7	Methacrylate	Modulator: Sine Wave ($m_f = 7$) Carrier: Triangular Wave ($m_a = 0.4$)	$\lambda_m = 21$ mm $\lambda_c = 3$ mm
#TEX-8	Methacrylate	Modulator: Sum of 2 Sine Waves ($m_{f1} = 3$, $m_{f2} = 9$) Carrier: Triangular Wave ($m_a = 0.4$)	$\lambda_{m1} = 6$ mm $\lambda_{m2} = 18$ mm $\lambda_c = 2$ mm
#TEX-9	Methacrylate	Modulator: Sine Wave ($m_f = 13$) Carrier: Triangular Wave ($m_a = 0.4$)	$\lambda_m = 39$ mm $\lambda_c = 3$ mm
#TEX-10	Fabric	No Specific Spatial Pattern	$\lambda = 1.5$ mm
#TEX-11	Fabric	No Specific Spatial Pattern	$\lambda = 4.5$ mm
#TEX-12	Fabric	No Specific Spatial Pattern	$\lambda = 8$ mm

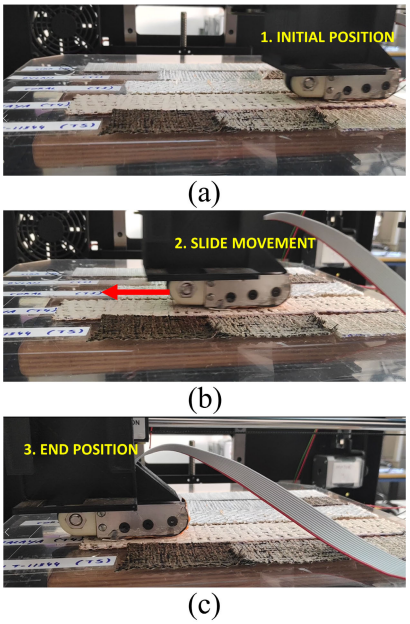


Fig. 6. Exploration sequence. (a) Initial position. (b) Slide movement ($v = 30$ mm/s). (c) End position.

are detected, as shown in Fig. 9 for the smooth surface of a methacrylate plate. It is worth remarking that, since only the location of the extrema is needed (see Section II), the dimension of the feature vector can be drastically reduced to M if the location of minima is signaled with 1's in the corresponding feature vector component. This is illustrated in Fig. 10 for the first texture in Fig. 8. Moreover, as shown in Section V, good classification results are achieved when only a subset of components of this feature vector is sent to the classifier.

On the other hand, since the vector of coefficients of the FFT is commonly used as a feature vector, it has been used in this article to compare the performance with that obtained with the proposed strategy. Fig. 11 shows the FFT output obtained for the set of textures used in this work.

The training of the classifier was realized offline in the work presented in this article, though results from the implemented hardware with the trained classifier are presented. With the purpose of performing training from data with similar restrictions as those found in hardware, raw data were processed with the high-level synthesis C/register-transfer level (HLS C/RTL) simulation tool of the Vivado IDE[1] HLS to obtain the feature vectors from the sweep algorithm. This simulation provides the closest RTL approach to the real system. On the other hand, the FFT was executed in MATLAB[1] with the precompiled MEX functions. The FFT MEX model produces identical and bit-accurate results to those that would be obtained by the hardware FFT intellectual property (IP) core.

The above procedure resulted in files containing the 1024-component feature vectors for every raw data corresponding to an exploration as described in Section III-D. Next, these data had to be split randomly into training (60% of the data) and test (40% of the data) sets. To improve the result of the training, feature vectors whose Euclidean distance to a reference vector deviates more than one sigma from the mean value were removed from the training set (not from the test set).

F. Training Algorithm

An unsupervised general-purpose k-means classifier was chosen to obtain the results of this article. In contrast to other machine-learning techniques such as support vector machines (SVMs) or K-nearest neighbors (KNNs), which would have to store a large amount of data in real-time tasks, the main advantage of the k-means is its simplicity and speed when converging to the different classes. The original choice of centroids (a centroid is a vector that identifies a given class) can be totally random or using the k-means ++ method [34]. In this work, the training was performed as follows. First, the centroids were randomly initialized. Then, these centroids were updated in the learning phase, where the training set was presented. This process was repeated with shuffled data from the training set until the centroids did not change. Finally, the

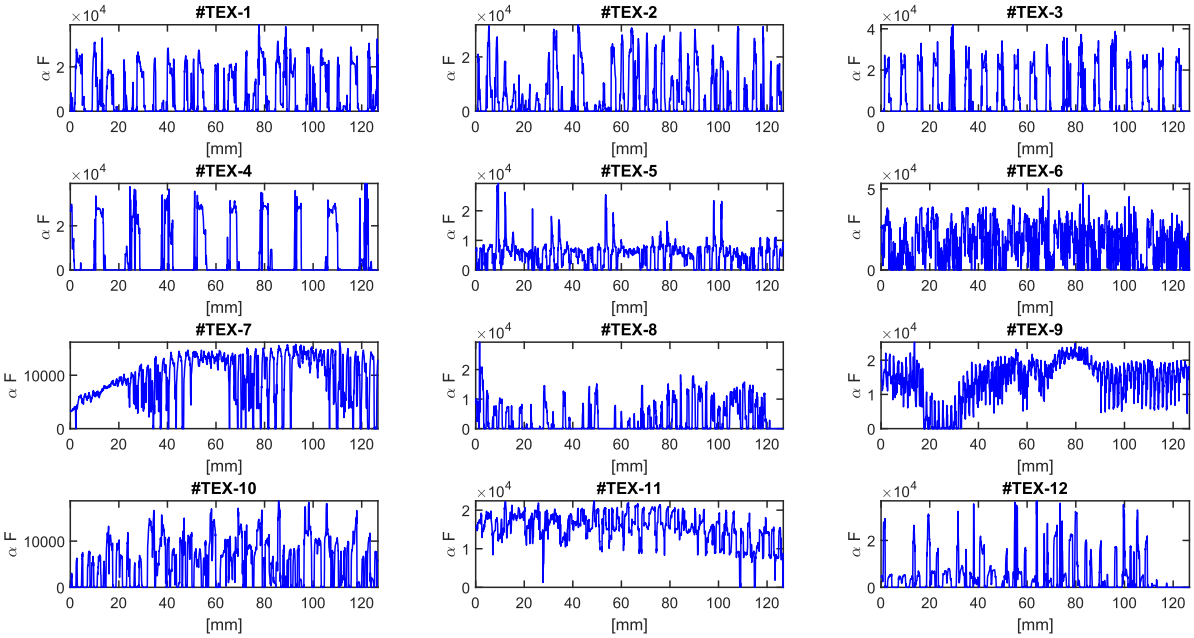


Fig. 7. Tactile sensor digital output taken at $v = 30$ mm/s for the set of textures in Fig. 5.

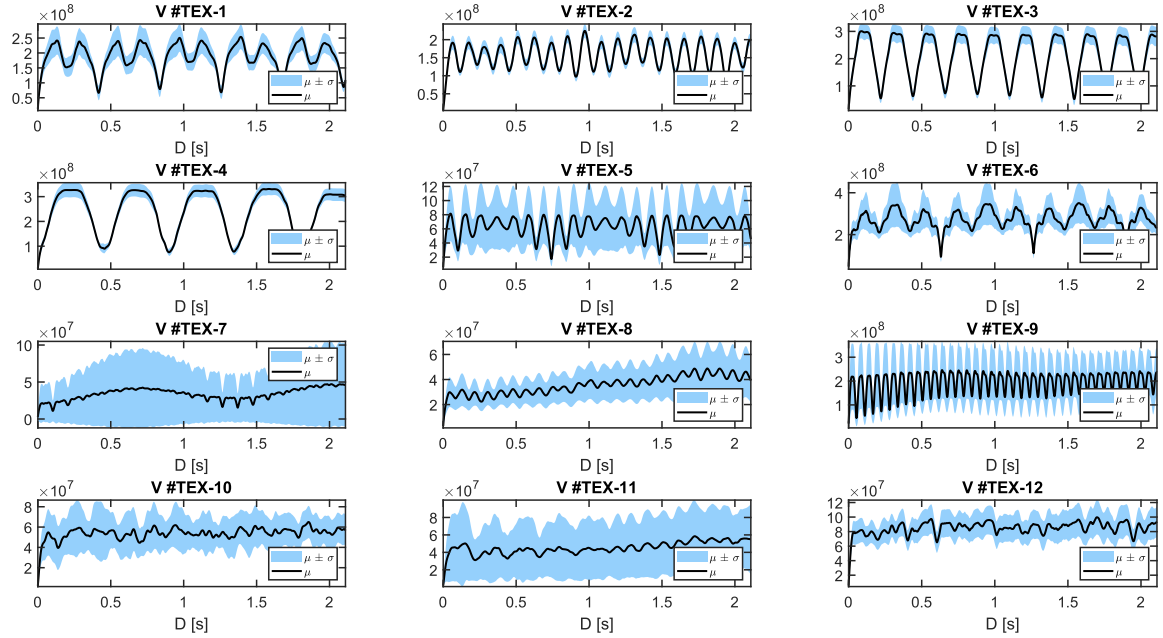


Fig. 8. Output of (1) for $1 \leq m \leq M$ ($D = mT_S$) for the set of textures in Fig. 5.

accuracy of the so-trained classifier was assessed with data from the test set. The whole procedure was repeated 100 times, and centroids that achieved the best accuracy were selected for the final trained classifier [35].

IV. Implementation of Algorithms on SoC

The Vivado Design Suite[2] environment was used to implement the feature extraction algorithm on the FPGA of the SoC. This software allows the integration of hardware-description modules (typically written in VHDL/Verilog) as well as presynthetized cores from IP libraries or the HLS tool from Vivado Design Suite.[2] The system datapath and the scheme for the sweep algorithm implementation are illustrated in Fig. 12. In the first stage, an SPI controller takes the data provided by the tactile sensor. Next, a 16-bit word is sent to the built-in FIR-compiler IPCore module from the Xilinx[1] IP library ("Antialiasing Filter" block in Fig. 12). This module is a 16Fix-frac15-bit coefficient filter that removes the components above the frequency given by the maximum spatial frequency to detect at a given speed. The output from this filter consists of 32-bit words which are then saved into a block RAM module. This module communicates with the custom interface "AXISTREAM Interface" in Fig. 12. The AMBA[1] AXI-Streaming protocol is used for the communication between the rest of FPGA hardware modules in the system [36]. The "HLS-Vivado Sweep Algorithm"

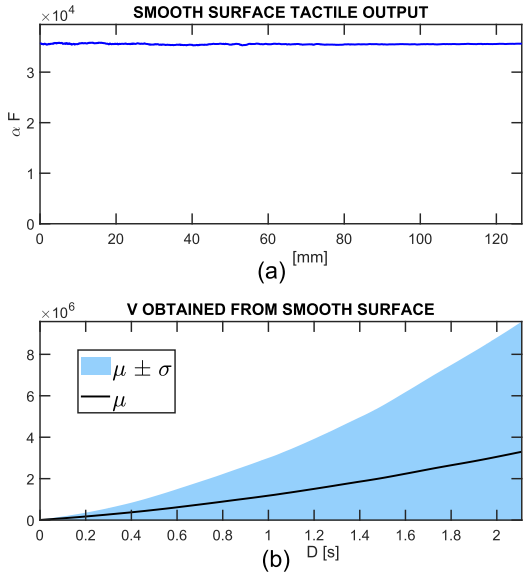


Fig. 9. (a) Tactile sensor digital output taken at $v = 30$ mm/s for the smooth surface of a methacrylate plate. (b) Output of (1) for $1 \leq m \leq M$ ($D = mT_S$) for this smooth surface.

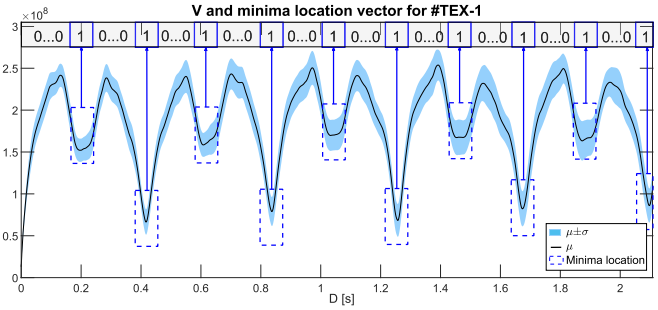


Fig. 10. Output of (1) for $1 \leq m \leq M$ ($D = mT_S$) for the first texture in the set of textures in Fig. 5. In addition, the minima location vector is shown at the top of the figure.

block in Fig. 12 is the sweep algorithm module which computes the feature vectors. The feature vectors obtained from the sweep algorithm are sent to a DDR memory through a first-input–first-output (FIFO) buffer plus a direct memory access (DMA) module. This way, the ARM core can access the data stored in the memory without taking care of the data traffic. The FIFO module seeks frequency decoupling between the processing logic (PL) and processing system (PS) parts. The system development kit (SDK[1]) from Xilinx[1] was used to implement the classifier on the ARM core. For the results of this work, the classifier was trained offline, and the obtained centroids were stored in the memory of the SoC. High throughput communications with the personal computer are achieved with an embedded real-time operating system (RTOS) and a lightweight TCP/IP (lwIP) stack implemented on the ARM core.

In particular, the inset at the bottom of Fig. 12 shows the block diagram of the implementation of (1) for $1 \leq m \leq M$. It has been designed using the HLS tool from Vivado Design Suite.[2] At this stage, the data coming from the previous step are buffered into a $2 \cdot M$ array in the first stage. A pipeline approach with three adders and two multipliers to compute every new term to add in (1) is followed. Every time there is a new input value, the $2 \cdot M$ array is left shifted, so the computation of (1) is made for the last $2 \cdot M$ samples.

The pseudocode in Algorithm 1 shows the System C implementation in HLS of the complete sweep algorithm. In order to achieve the pipeline implementation at RTL, the #HLS PIPELINE Vivado optimization directive is used. Minima of (1) for $1 \leq m \leq M$ are also obtained on the basis of a simple comparison procedure to detect peaks.

The FFT was also implemented as a common extraction feature method to compare with the proposed sweep algorithm. The block diagram is shown in Fig. 13, where the FFT is implemented with the built-in FFT IPCore from the Xilinx[1] IP Library.

V. Results

This section shows results from the active touch exploration of the textures in Fig. 5. Data gathering, feature extraction, and classifier training are performed as explained in Sections III-D–III-F, respectively.

A. Classification Accuracy

The main goal of this article is the reduction of resources needed to identify a given texture with the aim of local implementation on embedded hardware. Once the feature vectors from the FFT (vector of coefficients) and the sweep algorithm (vector of minima) are obtained, the question of further simplifications arises. Specifically, it is observed in Fig. 11 that most power of the signals obtained from exploring the textures is at low frequencies. Therefore, the classification could work if only the first coefficients of the FFT were sent to the classifier. Regarding the sweep algorithm, the information related to a given spatial frequency is not condensed in a coefficient as in the case of FFT, but it is redundant and spread along the whole feature vector. Therefore, a sort of equivalent process of that made for the FFT that still preserves the relevant information does not take V for the first values of D in Fig. 8 but performs a decimation or subsampling in the whole range of D.

Fig. 14 shows the results in classification accuracy when the reduction of the feature vector dimension described above was carried out. The accuracy was that obtained with the procedure explained in Section III-F. Moreover, two curves are shown for each FFT and sweep-based methods. One of them was obtained by taking into account hardware restrictions that are not considered for the other curve. These restrictions consisted in using the MATLAB[1] precompiled MEX functions in the case of the computation of the FFT coefficients, see Section III-E. Concerning the sweep algorithm, the results in Fig. 14 for the curve labeled with "Sweep FPGA test" were obtained with the whole feature extraction and classifier implemented on the SoC (though the training was performed offline and the centroids corresponding to each class or texture were stored on the SoC memory). The accuracy obtained with the FFT coefficients as features for the classifier saturates after the first 64 coefficients. The results in Fig. 14 confirm that most information to identify the textures is contained in

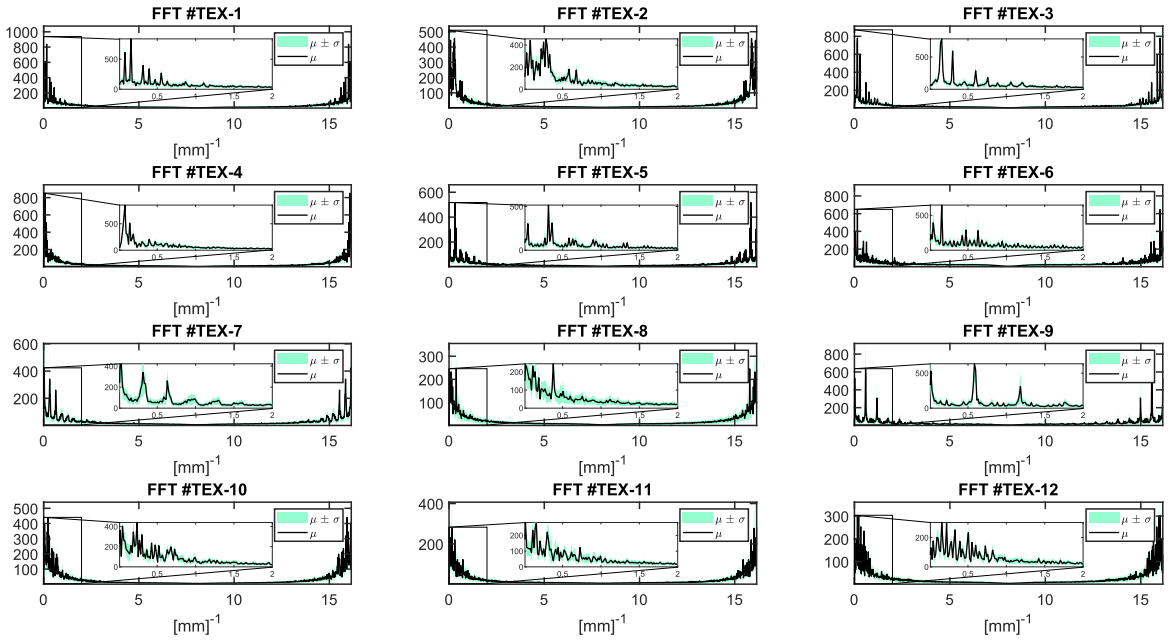


Fig. 11. FFT output obtained for the set of textures in Fig. 5. There is a zoom on the first values of the spectrum.

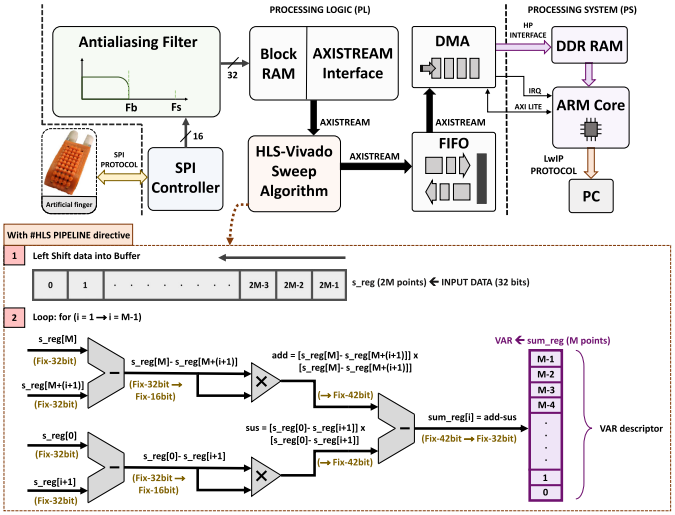


Fig. 12. General diagram of the Zynq-7000 XC7Z020 architecture implementation. At the bottom, there is a zoom on the sweep algorithm part. It only uses three adders and two multipliers.

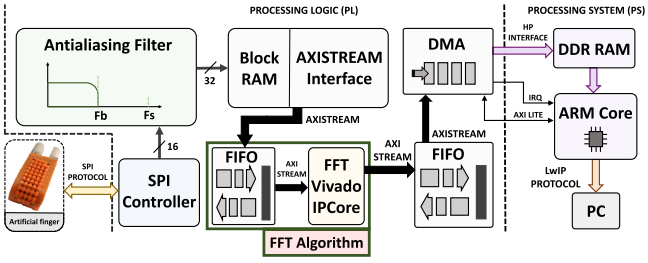


Fig. 13. General diagram of the Zynq-7000 XC7Z020 architecture implementation for the FFT algorithm.

low-frequency components of the signal. It is worth highlighting that the frequency response of the signal depends on the textures but also on the tactile sensor, specifically on the shape and composition of the cover that is in contact with

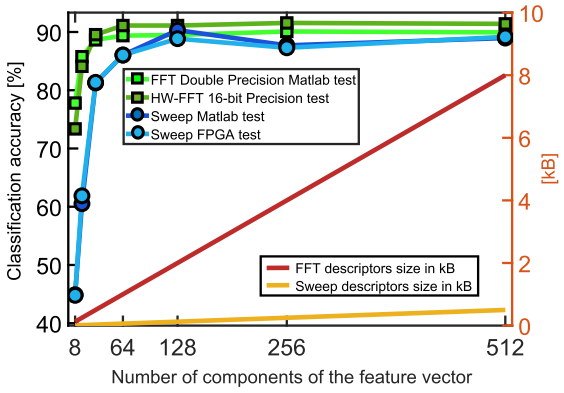


Fig. 14. Accuracy percentages for sweep and FFT algorithms, depending on the number of components of the feature vector sent to the classifier, that is, the decimation or cut applied to sweep and FFT, respectively. Both the sweep and FFT are computed with $M = 1024$.

the textures. Fig. 14 shows that good classification accuracy is also achieved with the sweep algorithm. In addition, the size of the feature vector is also displayed in Fig. 14 depending on the number of components of the feature vector. The size of the feature vector is smaller in the case of the sweep algorithm because only the location of the minima in the feature vector is sent to the classifier (see Section IV).

Confusion matrices in Figs. 15 and 16 show the classification results for feature vectors with 512 components provided by the sweep algorithm and the FFT, respectively. Specifically, the classification accuracy is 89.17% in the case of feature extraction with the sweep algorithm and 91.4% in the case of feature extraction with the FFT.

B. Resource Consumption

The resource consumption and latency of the proposed algorithm depend on the parameter M. For a given

Algorithm 1 Feature Vector Obtaining Procedure for Sweep Algorithm

Input: D
Variables:
1) $ibuf$ (internal buffer to store data samples)
2) add (addition factor to compute V, (1))
3) sus (subtraction factor to compute V, (1))
4) V (output array with the computed V values, (1))
5) $decV$ (vector with decimated values from V array)
6) M (output size from V)
7) n (current sample index)
8) d (decimation factor)

Initiate m as 0
repeat
  1) Obtain new sample p_m from data input:
  $p_m \leftarrow D$

  2) Store p_m and left shift one position internal buffer $ibuf$ of size $2M$:
  $i = 0$;
  repeat
    #Apply→HLS PIPELINE directive
    $ibuf[i] \leftarrow ibuf[i+1]$
    Access to the next index i
  until $i < 2M$
  $ibuf[2M-1] \leftarrow p_m$

  3) Obtaining descriptor procedure:
  $n = 0$;
  repeat
    3a) Obtain V:
    #Apply→HLS PIPELINE directive
    $add \leftarrow (ibuf[M] - ibuf[M+(n+1)]) \times$
      $\times (ibuf[M] - ibuf[M+(n+1)])$
    $sus \leftarrow (ibuf[0] - ibuf[n+1]) \times$
      $\times (ibuf[0] - ibuf[n+1])$
    $V[n] \leftarrow V[n] + add - sus$

    3b) Decimate V by factor d:
    $decimateV(decV[n], V[n], d)$

    3c) Obtain minima locations:
    $computeMinLoc(outDescriptor[n], decV[n])$
    Access to the next index n
  until $n < M$
  $m++$
until $m < M$
Result: A 1-bit array composed by the $outDescriptor$ array of M values.

exploration speed and acquisition sample frequency, this parameter determines the maximum spatial wavelength that can be detected [see (5)]. Results from our experiments are given for $M = 1024$, covering a range of spatial wavelengths from 0.12 to 63.34 mm, see Section III-C. To assess the resource consumption of the proposed approach in comparison

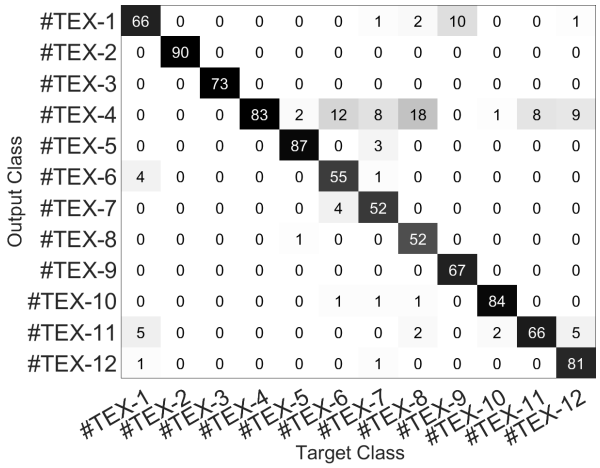


Fig. 15. Confusion matrix with the results from sweep algorithm on the textures from Fig. 5. Input data is with outliers. The test has been performed directly on the Zynq-7020 chip. Accuracy is 89.17%. Speed is 30 mm/s, $M = 1024$, and the decimation factor 2, that is, 512 values are sent to the classifier.

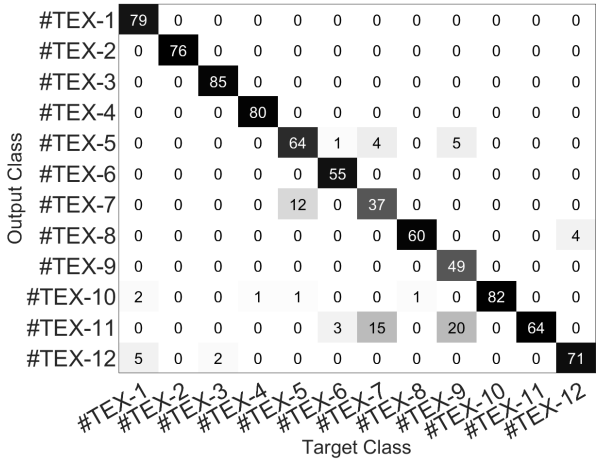


Fig. 16. Confusion matrix with the results from FFT algorithm on the textures from Fig. 5. Input data is with outliers. Accuracy is 91.4%. Speed is 30 mm/s, $M = 1024$, and the spectrum is cut by a factor of 2, that is, half-spectrum is sent to the classifier (512 coefficients).

with that based on the FFT and its dependency on M, the hardware in Figs. 12 and 13 was synthetized and implemented, for different values of M, with Vivado.[1]

The implementation report provides information about the number of hardware resources and time the algorithm needs to perform its computation. Fig. 17 shows this comparison for different hardware resources, power, and time. As inferred from Fig. 17, the sweep algorithm consumes less hardware resources than the FFT. This can be observed in Fig. 17(a)–(f) for the case of logical and memory elements. Note that the consumption of block RAM memory and distributed memory (LUTRAM) of the sweep algorithm in Fig. 17(i) is smaller than for the FFT. The block RAM (BRAM) memory is similar in both cases [see Fig. 17(f)], but the FFT needs much more distributed memory than the sweep algorithm [see Fig. 17(c)].

Regarding the use of digital signal processing (DSP) embedded blocks in Fig. 17(e), note that the hardware for the sweep algorithm requires only two DSPs (see Fig. 12) while the consumption of DSP blocks increases notably with the length

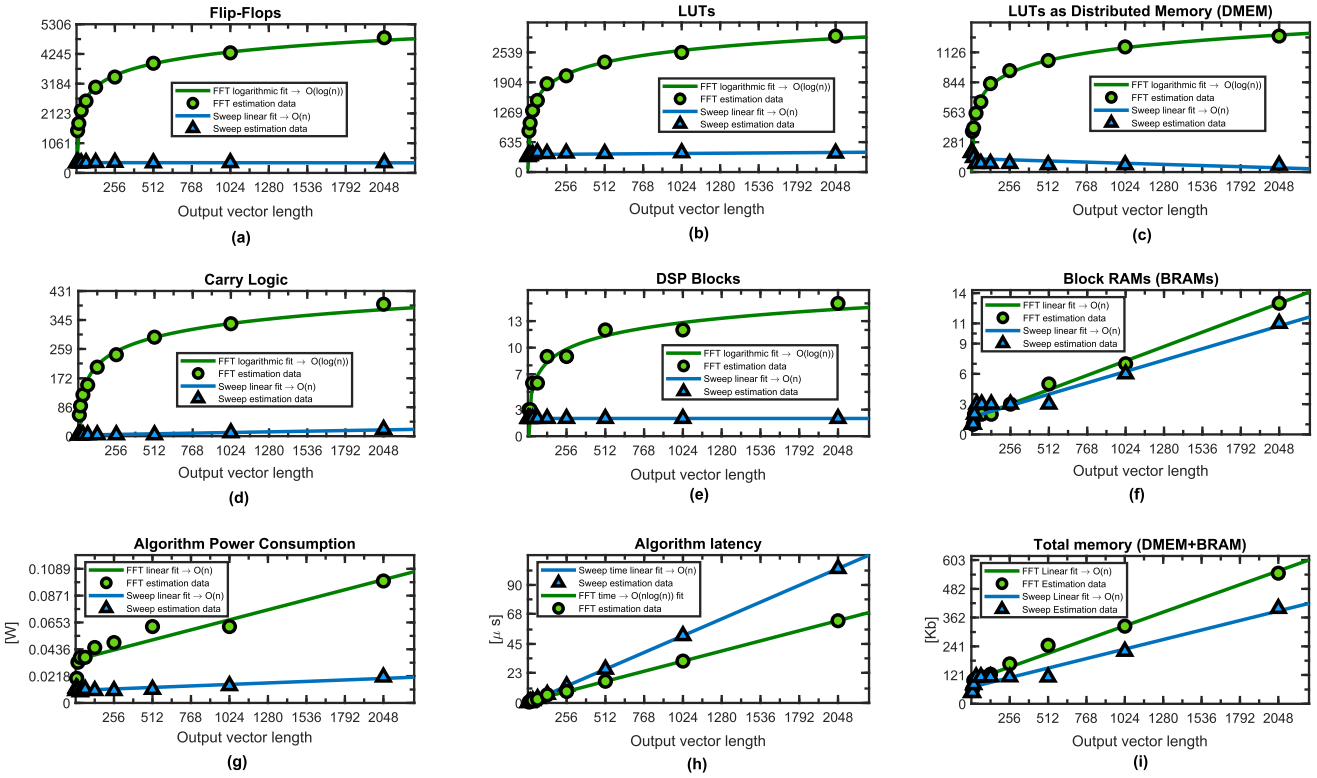


Fig. 17. Hardware resources and time consumption for Fig. 12 (sweep) and Fig. 13 (FFT). The output vector length corresponds to M for Fig. 12 and the number of FFT coefficients in Fig. 13. From left to right and top to bottom (a) flip-flops, (b) look-up-tables (LUTs), (c) DMEM (LUTRAM-based distributed memory), (d) carry logic, (e) DSP embedded blocks, (f) BRAM embedded blocks, (g) power consumption in (W), (h) time to provide feature vectors to ARM in (μs), and (i) sum of BRAM and DMEM in (Kb).

of the output vector and reaches 15 blocks for a length of 2048 components. The power consumption is also lower in the case of the hardware for the sweep algorithm than for the FFT [see Fig. 17(g)].

On the other hand, the latency of the hardware for the sweep algorithm is larger than that for the FFT [see Fig. 17(h)]. It grows as $\mathcal{O}(n)$ for the sweep algorithm and as $\mathcal{O}(n \log n)$ for the FFT. Nevertheless, it is in the order of tens of microseconds, which is a very short time for the application of texture identification, taking into account that the acquisition of $2 \cdot M$ samples of the surface takes 4.22 s (see Section II).

As said in Section I, the work in [23] also proposed an algorithm oriented to embedded systems. The signals from a 3-D accelerometer and a sound microphone were used as inputs for the feature extraction algorithm. This algorithm provides the power of the signal in a set of bands of the frequency spectrum as features to identify the texture. This algorithm does not require buffering the whole input vector, so it consumes low memory. On the other hand, it is more complex than ours. Therefore, the consumption of resources for computation, i.e., multipliers or DSPs, is higher. The same authors present an alternative in [37] that improves the throughput by parallelization at the cost of higher computational resources. In summary, this proposal requires more computational resources but less memory resources than ours.

Nevertheless, the comparison with other proposals is not easy, since it depends on many aspects. First, the performance depends on the sensor technology, because the registered data are produced by the interaction between the sensor and the textures, the contact pressure, and the sort of exploratory actions [22]. Moreover, the size and nature of the set of textures to test the performance also influence the results. However, the final application can be considered in the design of the sensor and the choice of the feature extraction algorithm. Some proposals overcome the human performance in discriminating different textures [16], but it could not be necessary to implement artificial hands unless they are focused on some specialized tasks. Kursun and Patooghy [22] say that very few frequency features suffice to represent the textures, while the work in [23] and this article shows the same since the classification accuracy is high even when the extracted features are a few. Therefore, the feature extraction algorithm could be simplified. Note that our algorithm provides information about the location of the frequency components in the spectrum but not about its power. The simplification of the algorithm can limit its performance, but, as stated in [37], the use of more sensors can compensate for the loss of accuracy with respect to the feature extraction based on FFT. This can be the case with the tactile sensor of this article. Only one channel has been used to show the feasibility of the approach but they could be more (16 channels are used in [22]). In this case, the simplicity of the feature extraction hardware makes it more scalable. The acquisition of the tactile array can then be performed in parallel with the computation of other features such as the moments of the tactile image [25] and the

features for the texture detection. Finally, better classification performance can be achieved by changing the velocities and motion directions [16], [22], which can be done by a robotic system.

VI. Conclusion

This article proposes a simple algorithm for feature extraction of textures oriented to smart tactile systems with embedded FPGA. The raw input data are taken by active touch so the registered signals provide the information from the interaction between the sensor and the textures. The outer cover of the sensor is designed to collect the micro-vibrations produced in this process. The tactile sensor is piezoresistive, and the electronics is based on local FPGA [25]. The proposed algorithm is a simple adaptive filter that scans the frequency spectrum and provides information about the location of the frequency components as features. The resulting feature vector size is small, which reduces the data traffic and simplifies the classifier. Results from the algorithm implemented on the FPGA and a classifier implemented on an ARM processor show the feasibility of the approach. A comparison with a common alternative where the FFT coefficients are the input features for the classifier is presented. The classification accuracy obtained with both methods is quite close, but the hardware resources required to compute the FFT are much higher. The results of this article are obtained when the readings of only one channel are processed. The simplicity of the presented approach is aimed to allow the replication of the hardware and the parallel processing of many channels. This can also improve the classification accuracy and the capacity of the smart sensor. The exploration speed is supposed to be known and constant, like in many other reported works. This is a relative limitation in robotic systems, where this velocity can be controlled and even changed to improve performance. Nevertheless, future work will focus on the evaluation of the limits of the proposal in terms of accuracy and computing resources, when the ranges of spatial frequency and the exploration velocity are changed. The scalability of the approach to process more channels of taxels in contact with the explored surface will also be assessed.

References

[1] S. Luo, J. Bimbo, R. Dahiya, and H. Liu, "Robotic tactile perception of object properties: A review," *Mechatronics*, vol. 48, pp. 54–67, Dec. 2017, doi: 10.1016/j.mechatronics.2017.11.002.

[2] Y. Al-Handarish et al., "A survey of tactile-sensing systems and their applications in biomedical engineering," *Adv. Mater. Sci. Eng.*, vol. 2020, pp. 1–17, Jan. 2020, doi: 10.1155/2020/4047937.

[3] A. Masteller, S. Sankar, H. B. Kim, K. Ding, X. Liu, and A. H. All, "Recent developments in prosthesis sensors, texture recognition, and sensory stimulation for upper limb prostheses," *Ann. Biomed. Eng.*, vol. 49, no. 1, pp. 57–74, Jan. 2021, doi: 10.1007/s10439-020-02678-8.

[4] R. Li and E. H. Adelson, "Sensing and recognizing surface textures using a gelsight sensor," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1241–1247, doi: 10.1109/CVPR.2013.164.

[5] W. Yuan, S. Dong, and E. Adelson, "GelSight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, Nov. 2017, doi: 10.3390/s17122762.

[6] J. M. Gandarias, A. J. Garcia-Cerezo, and J. M. Gomez-de-Gabriel, "CNN-based methods for object recognition with high-resolution tactile sensors," *IEEE Sensors J.*, vol. 19, no. 16, pp. 6872–6882, Apr. 2019, doi: 10.1109/JSEN.2019.2912968.

[7] M. Alameh, Y. Abbass, A. Ibrahim, and M. Valle, "Smart tactile sensing systems based on embedded CNN implementations," *Micromachines*, vol. 11, no. 1, p. 103, Jan. 2020, doi: 10.3390/mi11010103.

[8] D. Liu, G. Zhou, D. Zhang, X. Zhou, and C. Li, "Ground control point automatic extraction for spaceborne georeferencing based on FPGA," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 13, pp. 3350–3366, 2020, doi: 10.1109/JSTARS.2020.2998838.

[9] C. J. Miller, C. A. Zorman, and G. E. Wnek, "An improved tactile sensing device for material characterization via friction-induced vibrations," *Sens. Actuators A, Phys.*, vol. 303, Mar. 2020, Art. no. 111824, doi: 10.1016/j.sna.2019.111824.

[10] S. Ding, Y. Pan, M. Tong, and X. Zhao, "Tactile perception of roughness and hardness to discriminate materials by friction-induced vibration," *Sensors*, vol. 17, no. 12, p. 2748, Nov. 2017, doi: 10.3390/s17122748.

[11] J. Scheibert, S. Leurent, A. Prevost, and G. Debrégeas, "The role of fingerprints in the coding of tactile information probed with a biomimetic sensor," *Science*, vol. 323, no. 5920, pp. 1503–1506, Mar. 2009, doi: 10.1126/science.1166467.

[12] R. Fagiani, F. Massi, E. Chatelet, Y. Berthier, and A. Akay, "Tactile perception by friction induced vibrations," *Tribol. Int.*, vol. 44, no. 10, pp. 1100–1110, Sep. 2011, doi: 10.1016/j.triboint.2011.03.019.

[13] Y. Massalim, Z. Kappassov, and H. Varol, "Deep vibro-tactile perception for simultaneous texture identification, slip detection, and speed estimation," *Sensors*, vol. 20, no. 15, pp. 1–15, 2020, doi: 10.3390/s20154121.

[14] Q. Li, O. Kroemer, Z. Su, F. F. Veiga, M. Kaboli, and H. J. Ritter, "A review of tactile information: Perception and action through touch," *IEEE Trans. Robot.*, vol. 36, no. 6, pp. 1619–1634, Dec. 2020, doi: 10.1109/TRO.2020.3003230.

[15] S. Chen, S. Ge, W. Tang, J. Zhang, and N. Chen, "Tactile perception of fabrics with an artificial finger compared to human sensing," *Textile Res. J.*, vol. 85, no. 20, pp. 2177–2187, Dec. 2015, doi: 10.1177/0040517515586164.

[16] J. A. Fishel and G. E. Loeb, "Bayesian exploration for intelligent identification of textures," *Frontiers Neurorobotics*, vol. 6, pp. 1–20, Jun. 2012, doi: 10.3389/fnbot.2012.00004.

[17] M. Kaboli and G. Cheng, "Robust tactile descriptors for discriminating objects from textural properties via artificial robotic skin," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 985–1003, Aug. 2018, doi: 10.1109/TRO.2018.2830364.

[18] T. E. A. de Oliveira, A. M. Cretu, and E. M. Petriu, "Multimodal bio-inspired tactile sensing module for surface characterization," *Sensors*, vol. 17, no. 6, pp. 1–19, 2017, doi: 10.3390/s17061187.

[19] D. Goger, N. Gorges, and H. Worn, "Tactile sensing for an anthropomorphic robotic hand: Hardware and signal processing," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 895–901, doi: 10.1109/ROBOT.2009.5152650.

[20] A. Drimus, M. B. Petersen, and A. Bilberg, "Object texture recognition by dynamic tactile sensing using active exploration," in *Proc. IEEE Int. Workshop Robot Hum. Interact. Commun.*, Sep. 2012, pp. 277–283, doi: 10.1109/ROMAN.2012.6343766.

[21] N. Jamali and C. Sammut, "Majority voting: Material classification by tactile sensing using surface texture," *IEEE Trans. Robot.*, vol. 27, no. 3, pp. 508–521, Jun. 2011, doi: 10.1109/TRO.2011.2127110.

[22] S.-A. Wang, A. Albini, P. Maiolino, F. Mastrogiovanni, and G. Cannata, "Fabric classification using a finger-shaped tactile sensor via robotic sliding," *Frontiers Neurorobotics*, vol. 16, p. 10, Feb. 2022, doi: 10.3389/fnbot.2022.808222.

[23] O. Kursun and A. Patooghy, "An embedded system for collection and real-time classification of a tactile dataset," *IEEE Access*, vol. 8, pp. 97462–97473, 2020, doi: 10.1109/ACCESS.2020.2996576.

[24] M. Shimojo and M. Ishikawa, "Active touch sensing method using a spatial filtering tactile sensor," *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, May 1993, pp. 948–954, doi: 10.1109/ROBOT.1993.292098.

[25] Ó. Oballe-Peinado et al., "FPGA-based tactile sensor suite electronics for real-time embedded processing," *IEEE Trans. Ind. Electron.*, vol. 64, no. 12, pp. 9657–9665, Dec. 2017, doi: 10.1109/TIE.2017.2714137.

[26] A. Ibrahim, P. Gastaldo, H. Chible, and M. Valle, "Real-time digital signal processing based on FPGAs for electronic skin implementation," *Sensors*, vol. 17, no. 3, p. 558, Mar. 2017, doi: 10.3390/s17030558.

[27] R. Lora-Rivera, A. Guzmán-Manzano, J. Luna-Cortés, O. Oballe-Peinado, and F. Vidal-Verdú, "Texture detection by hardware-friendly low-level preprocessing," *Proc. IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatronics*, Nov. 2020, pp. 922–927, doi: 10.1109/BioRob49111.2020.9224388.

[28] R. Lora-Rivera et al., "Tactile sensor with a structured 3D printed cover and laser-isolated tactels," in *Proc. 9th Cairo Int. Biomed. Eng. Conf. (CIBEC)*, Dec. 2018, pp. 98–101, doi: 10.1109/CIBEC.2018.8641757.

[29] RECREUS. *FILAFLEX 82A ORIGINAL, Technical Data Sheet*. Accessed: Apr. 18, 2023. [Online]. Available: https://drive.google.com/file/d/1aoB6tmYfSAZzNzXSSVivrn7VJ3XXMOEw/view

[30] Y. Cao, T. Li, Y. Gu, H. Luo, S. Wang, and T. Zhang, "Fingerprint-inspired flexible tactile sensor for accurately discerning surface texture," *Small*, vol. 14, no. 16, Apr. 2018, Art. no. 1703902, doi: 10.1002/smll.201703902.

[31] *Zedboard*. Accessed: Apr. 18, 2023. [Online]. Available: http://zedboard.org/product/zedboard

[32] RECREUS. *PLA, Technical Data Sheet*. Accessed: Apr. 18, 2023. [Online]. Available: https://drive.google.com/file/d/1fpLy7KjPz1twhXgqQOb-JzvzXUX5bmF/view

[33] D. W. Hart, *Power Electronics*. New York, NY, USA: McGraw-Hill, 2010.

[34] S. Kumar. *Understanding K-Means, K-Means++ and, K-Medoids Clustering Algorithms*. Accessed: Apr. 18, 2023. [Online]. Available: https://towardsdatascience.com/understanding-k-means-k-means-and-k-medoids-clustering-algorithms-ad9c9fbf47ca

[35] P. Fränti and S. Sieranoja, "How much can k-means be improved by using better initialization and repeats?" *Pattern Recognit.*, vol. 93, pp. 95–112, Sep. 2019, doi: 10.1016/j.patcog.2019.04.014.

[36] ARMdeveloper. (2010). *AMBA 4 AXI4-Stream Protocol*. Accessed: Apr. 18, 2023. [Online]. Available: https://developer.arm.com/documentation/ihi0051/a/

[37] J. Osborne, A. Patooghy, B. Sarsekeyev, and O. Kursun, "Eco-CMB: A hardware-accelerated band-power feature extractor for tactile embedded systems," in *Proc. IEEE Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2021, pp. 198–203, doi: 10.1109/MWSCAS47672.2021.9531685.

Raúl Lora-Rivera was born in Baza, Granada, Spain, in 1993. He received the B.S. degree in electronics, robotics, and mechatronics engineering and the M.S. degree in electronic systems for intelligent environments from the Universidad de Málaga (UMA), Málaga, Spain, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree in mechatronics engineering with the Departamento de Electrónica.

His current research interests include the implementation of field-programmable gate array (FPGA)primitives for tactile sensor applications.

Óscar Oballe-Peinado was born in Málaga, Spain. He received the B.S., M.S., and Ph.D. degrees in computer engineering from the Universidad de Málaga (UMA), Málaga, in 1993, 1999, and 2016, respectively.

Since 2000, he has been with the Departamento de Electrónica, UMA, where he is currently an Associate Professor. His current research interests include ASIC and field-programmable gate array (FPGA) designs, advanced tactile sensors, and tactile signal processing and applications.

Fernando Vidal-Verdú was born in San Fernando, Cádiz, Spain, in 1965. He received the M.Sc. degree in physics from the Universidad de Seville, Seville, Spain, in 1988, and the Ph.D. degree in microelectronics from the Universidad de Málaga (UMA), Málaga, Spain, in 1996.

Since 1991, he has been with the Departamento de Electrónica, UMA, where he is currently a Full Professor. He has authored or coauthored 31 international journal articles, seven chapters of books (edited by IEEE, Kluwer, CRC, Wiley, and Springer), and more than 60 conference papers. His current research interests include smart sensors, haptic and tactile interfaces, and assistive and healthcare applications and technologies.