

UNIVERSIDAD DE MÁLAGA

Investigación en primitivas de pre-procesado en electrónica local basada en FPGA de un sensor táctil

Raúl Lora Rivera

Universidad de Málaga

Escuela de Ingenierías Industriales

Departamento de Electrónica

Tesis doctoral presentada para la obtención del grado de
PhilosophiæDoctor (PhD) en Ingeniería Mecatrónica

Directores:

Fernando Vidal Verdú

Óscar Oballe Peinado

Septiembre de 2023

UNIVERSIDAD DE MÁLAGA



AUTOR: Raúl Lora Rivera

https://orcid.org/0000-0002-8746-3117

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional:
https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode
Cualquier parte de esta obra se puede reproducir sin autorización
pero con el reconocimiento y atribución de los autores.
No se puede hacer uso comercial de la obra y no se puede alterar, transformar o hacer obras derivadas.

Esta Tesis Doctoral está depositada en el Repositorio Institucional de la Universidad de Málaga (RIUMA): riuma.uma.es

UNIVERSIDAD DE MÁLAGA

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D. RAÚL LORA RIVERA
Estudiante del programa de doctorado INGENIERÍA MECATRÓNICA de la Universidad de Málaga, autor/a de la tesis, presentada para la obtención del título de Doctor por la Universidad de Málaga, titulada: INVESTIGACIÓN EN PRIMITIVAS DE PRE-PROCESADO EN ELECTRÓNICA LOCAL BASADA EN FPGA DE UN SENSOR TÁCTIL.

Realizada bajo la tutorización de D. FERNANDO VIDAL VERDÚ y dirección de D. FERNANDO VIDAL VERDÚ Y ÓSCAR OBALLE PEINADO,

DECLARO QUE:

La tesis presentada es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, conforme al ordenamiento jurídico vigente (Real Decreto Legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), modificado por la Ley 2/2019, de 1 de marzo.

Igualmente asumo, ante a la Universidad de Málaga y ante cualquier otra instancia, la responsabilidad que pudiera derivarse en caso de plagio de contenidos en la tesis presentada, conforme al ordenamiento jurídico vigente.

En Málaga, a 7 de JULIO de 2023

Fdo.: RAÚL LORA RIVERA Doctorando/a	Fdo.: FERNANDO VIDAL VERDÚ Tutor/a
Fdo.: FERNANDO VIDAL VERDÚ Y ÓSCAR OBALLE PEINADO Director/es de tesis	

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

AUTORIZACIÓN DE LECTURA DE TESIS DOCTORAL

D. FERNANDO VIDAL VERDÚ y D. ÓSCAR OBALLE PEINADO, tutor y directores de la tesis titulada INVESTIGACIÓN EN PRIMITIVAS DE PRE-PROCESADO EN ELECTRÓNICA LOCAL BASADA EN FPGA DE UN SENSOR TÁCTIL, realizada por el estudiante del Programa de Doctorado en Ingeniería Mecatrónica de la Universidad de Málaga, D. RAÚL LORA RIVERA, autorizan la lectura de dicha tesis doctoral, y confirman que la misma viene avalada por las siguientes publicaciones en revistas científicas indexadas en el JCR:

- **R. Lora-Rivera**, Ó. Oballe-Peinado and F. Vidal-Verdú, "Texture Detection With Feature Extraction on Embedded FPGA," in *IEEE Sensors Journal*, vol. 23, no. 11, pp. 12093-12104, 1 June1, 2023, doi: 10.1109/JSEN.2023.3268794. (Primer cuartil en 2022)
- **R. Lora-Rivera**, Ó. Oballe-Peinado, and F. Vidal-Verdú, "Proposal and Implementation of a Procedure for Compliance Recognition of Objects with Smart Tactile Sensors," *Sensors*, vol. 23, no. 8, p. 4120, Apr. 2023, doi: 10.3390/s23084120. (Segundo cuartil, primer tercil, en 2022)

Asimismo, la tesis viene avalada por varias contribuciones en congresos internacionales:

- **R. Lora-Rivera** *et al.*, "Tactile Sensor with a Structured 3D Printed Cover and Laser-Isolated Tactels," *2018 9th Cairo International Biomedical Engineering Conference (CIBEC)*, Cairo, Egypt, 2018, pp. 98-101, doi: 10.1109/CIBEC.2018.8641757.
- **R. Lora-Rivera** *et al.*, "Object stiffness recognition with descriptors given by an FPGA-based tactile sensor," *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*, Delft, Netherlands, 2020, pp. 561-566, doi: 10.1109/ISIE45063.2020.9152575.
- **R. Lora-Rivera**, A. d. Guzmán-Manzano, J. A. Luna-Cortés, Ó. Oballe-Peinado and F. Vidal-Verdú, "Texture Detection by Hardware-Friendly Low-Level Preprocessing," *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, New York, NY, USA, 2020, pp. 922-927, doi: 10.1109/BioRob49111.2020.9224388.
- **R. Lora-Rivera,** Ó. Oballe-Peinado and F. Vidal-Verdú, "Texture Detection with Tactile Sensors based on the Goertzel Algorithm," in *IWANN International Work Conference on Artificial Neural Networks*, Azores (Portugal), 2023, pp. 7-8, ISBN: 978-84-19214-58-4.

En Málaga, a 7 de JULIO de 2023.

Fdo.- FERNANDO VIDAL VERDÚ
Director y tutor de tesis

Fdo.- ÓSCAR OBALLE PEINADO
Director de tesis

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Esta tesis doctoral ha sido llevada a cabo con una ayuda del programa de Formación de Profesorado Universitario concedida por el Ministerio de Ciencia, Innovación y Universidades (referencia FPU18/03106). También cabe mencionar la ayuda recibida a través de los proyectos TEC2015-67642-R y PID2021-125091OB-I00, financiados por el Gobierno de España y por el Fondo Europeo de Desarrollo Regional (FEDER).

UNIVERSIDAD DE MÁLAGA



UNIVERSIDAD
DE MÁLAGA

Investigación en primitivas de pre-procesado en electrónica local basada en FPGA de un sensor táctil

Raúl Lora Rivera

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Para vosotros, papá y mamá

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Agradecimientos

Recorrer el camino de la tesis doctoral ha supuesto uno de los mayores retos que he enfrentado. Para superarlo hay que llevar a cabo un ejercicio continuo de constancia, método y organización, además de seguir dándote oportunidades cuando las cosas no salen bien a la primera. El hecho de lograrlo, también es en gran parte, gracias a las personas que me rodean día a día. De forma honesta, en esta sección quiero agradecer el ánimo y apoyo recibido por todas ellas a lo largo de los años, con especial relevancia en aquellas que me han influenciado en mayor medida para la consecución del doctorado.

Siento una total admiración y gratitud hacia mis directores de tesis Fernando Vidal Verdú y Óscar Oballe Peinado. A lo largo de estos años, he tratado de seguir todas las indicaciones que me han proporcionado, llevando a cabo todas las tareas en base a su consejo y experiencia. Gracias por corregirme cuando no estaba en la dirección correcta, por atenderme siempre aunque tuvierais mil tareas, gracias por ayudarme a afrontar adecuadamente la publicación de los *papers* que tanto dolor de cabeza nos dieron, gracias por ayudarme en los congresos, en mis horas de docencia y en cualquier trámite necesario. Gracias también por entenderme en el peor de los momentos anímicos que he afrontado (casi tirando la toalla), y por animarme a superarlo. Me da la sensación de haber vivido mil batallas con vosotros. Muchas gracias a ambos por ofrecerme la oportunidad de trabajar con el grupo EIS, y otorgarme la confianza necesaria para desarrollar la tesis.

Quiero destacar también toda la ayuda recibida por parte de otros miembros del grupo. Gracias a Julián Castellanos Ramos, por ayudarme con LabVIEW®, con los preparativos de mi viaje al congreso CIBEC y por tantas cosas más, en las que siempre se puede contar con él. Gracias a Andrés Trujillo León y José Antonio Sánchez Durán, siempre dispuestos a echar una mano, y a los que tengo un gran respeto por todo su trabajo y dedicación. Gracias también a todos los demás miembros del departamento, algunos de ellos como Rafa Navas, Raquel Fernández, Jorge Romero, Alberto Daza o Carlos

UNIVERSIDAD DE MÁLAGA

González, profesores míos, que de alguna manera, han contribuido a lo largo del tiempo a mi afán por la ingeniería electrónica. También me gustaría destacar que, aunque no han estado en el departamento hasta el final de mi tesis, sí es verdad que hay compañeros que han pasado por el laboratorio, y que han ayudado en el desarrollo de parte del proyecto. En este sentido, me gustaría agradecer el trabajo de Ángela Fernández Ruiz, Paula Ruiz Barroso, Álvaro Galán Cuenca y José Manuel Cuesta Ramírez.

Por otra parte, me gustaría mencionar la aventura que conlleva vivir en una ciudad como Málaga compartiendo piso durante tantos años. Las personas con las que vives te influyen de una u otra forma, pero especialmente, quiero agradecer a Álex y José Luis, cuya convivencia ha transcendido más allá de un simple contrato de arrendamiento, por haber sabido escucharme y animarme después de cada jornada de trabajo.

En mi círculo más cercano, quiero agradecer profundamente a mi pareja Carol Peiró Alemany, por brindarme apoyo día y noche, tanto en los momentos buenos como en los malos, y también en los muy muy malos, y siempre que lo he necesitado, sin excepción. Gracias de corazón, te quiero. También me gustaría reconocer que habría sido imposible finalizar la tesis sin el apoyo constante de mis amigos Daniel Pereira Martín, Abdel Hadij El Haouti, Arturo de Guzmán Manzano, José Antonio Luna Cortés y Rocío Egea Guerrero. Esos cafés de la mañana, los almuerzos viendo *Family Guy* o los ratos de pádel y cenas después de todo el día programando, son inolvidables. Por otra parte, si alguien ha estado ahí siempre que lo he necesitado, es mi amigo y compañero inseparable desde que entré a la universidad, Francisco Naranjo Bernal. Él, junto a Esther, Dani y Ana, han conseguido muchas veces que deje de pensar por un ratito en el trabajo. No puedo dejar de agradecer a también mi compañero de carrera, de piso, de empresa, de senderismo (avanzado) y amigo, Ángel Maldonado, y a Jony y Silvia, por todos los momentos que siempre disfrutamos juntos y que me permiten despejarme de la universidad. Y, de igual forma, a Iván Villacreces y a Daniel Fernández, aunque se encuentren un poquito más lejos. Por último, quiero dar las gracias a Martínez, Antonio, Manu, Alberto, Chema, Cristi (Rizos) y Analoy por estar a mi lado durante tanto tiempo, aunque se sucedan los años y cada uno tenga su propio camino. Muchísimas gracias a todos vosotros por vuestro ánimo y apoyo infinito. Es increíble teneros como amigos.

Finalmente, quiero agradecer y recordar a toda mi familia, y en especial a mis padres Ana y José, mi hermano José Miguel, mis abuelos Paquita y Miguel, mi tía abuela Ofelia y mi tío Michi, por su apoyo incondicional, ya no sólo en el doctorado, sino a lo largo de toda mi trayectoria académica y también en todos los aspectos que rodean mi vida.

Raúl Lora Rivera
Málaga, septiembre de 2023

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Lista de acrónimos

FPGA	*Field-programmable gate array*
FFT	*Fast Fourier Transform*
PCA	*Principal component analysis*
MEMS	*Microelectromechanical Systems*
INE	*Instituto Nacional de Estadística*
I2C	*Inter-Integrated Circuit*
CAN	*Controller Area Network*
LAN/WLAN	*Local Area Network/Wireless Local Area Network*
RS232	*Recommended Standard 232*
USB	*Universal Serial Bus*
Conversión A/D	*Conversión Analógico/Digital*
IEEE	*Institute of Electrical and Electronics Engineers*
RAS/EMBS	*Robotics and Automation Society/Engineering in Medicine and Biology Society*
SoC	*System on Chip*

UNIVERSIDAD DE MÁLAGA

JCR	*Journal Citation Reports*
ISIE	*International Symposium on Industrial Electronics*
MDPI	*Multidisciplinary Digital Publishing Institute*
EIS	*Electrónica para Instrumentación y Sistemas*
TFG	*Trabajo Fin de Grado*
TFM	*Trabajo Fin de Máster*
FPU	*Formación de Profesorado Universitario*
I3MSC	*Instituto Universitario de Investigación en Ingeniería Mecatrónica y Sistemas Ciberfísicos*
IBIMA	*Instituto de Investigación Biomédica de Málaga*
FEDER	*Fondo Europeo de Desarrollo Regional*
VHDL	*VHSIC Hardware Description Language*
SDK	*System Development Kit*
HLS	*High-Level Synthesis*
PVDF	*Polifluoruro de Vinilideno*
PDMS	*Polidimetilsiloxano*
PVC	*Cloruro de Polivinilo*
PTFE	*Politetrafluoroetileno*
PCB	*Printed Circuit Board*
PROM	*Programmable Read-Only Memory*
SPI	*Serial Peripheral Interface*

UNIVERSIDAD DE MÁLAGA

JTAG	*Joint Test Action Group*
MOSI	*Master Output Slave Input*
MISO	*Master Input Slave Output*
SCK	*Serial Clock*
SS	*Slave Select*
SD	*Secure Digital*
CIBEC	*Cairo International Biomedical Engineering Conference*
HZ	*High Impedance*
α**F**	*Proporcional a la fuerza aplicada*
P.D.	*Puerto de descarga*
P.M.	*Puerto de medida*
CO_2	*Dióxido de carbono*
RAMPS	*RepRap Arduino Mega Pololu Shield*
IDE	*Integrated Development Environment*
NI	*National Instruments™*
DAQ	*Data acquisition*
ARM	*Advanced RISC Machines*
lwIP	*lightweight IP*
GPP	*General Purpose Processor*
ASIC	*Application-Specific Integrated Circuit*
GPU	*Graphics Processing Unit*

UNIVERSIDAD DE MÁLAGA

PLC	*Programmable logic cells*
AXI	*Advanced eXtensible Interface*
AMBA	*Advanced Microcontroller Bus Architecture*
PL	*Programmable Logic*
PS	*Processing System*
IP	*Intellectual Property*
VHSIC	*Very High-Speed Integrated Circuit*
CLB	*Configurable Logic Block*
I/O	*Input/Output*
LC	*Logic cell*
LUT	*Look-Up Table*
SRL	*Shift register LUT*
FF	*Flip-Flop*
BRAM	*Block RAM*
DSP	*Digital signal processor*
HDL	*Hardware Description Language*
ISE	*Integrated Software Environment*
ILA	*Integrated Logic Analyzer*
GUI	*Graphical User Interface*
RTOS	*Real-Time Operating System*
UART	*Universal Asynchronous Receiver/Transmitter*

UNIVERSIDAD DE MÁLAGA



TCP/IP	*Transmission Control Protocol/Internet Protocol*
DFT	*Discrete Fourier Transform*
IWANN	*International Work Conference on Artificial Neural Networks*
FIR	*Finite Impulse Response*
LTI	*Linear Time-Invariant*
PWM	*Pulse-width modulation*
IIR	*Infinite Impulse Response*
SNR	*Signal-to-Noise Ratio*
PLA	*Ácido poliláctico*
RTL	*Register Transfer Level*
MEX	*MATLAB Executable*
SVM	*Support Vector Machine*
KNN	*K-Nearest Neighbors*
VC	*Vectores de características*
DDR	*Double Data Rate*
FIFO	*First-Input First-Output*
DMA	*Direct Memory Access*
RAM	*Random Access Memory*
BUFG	*Global buffers*
ISR	*Interrupt Service Routine*

UNIVERSIDAD DE MÁLAGA

ROC	*Receiver Operating Characteristic*
VP	*Verdaderos positivos*
FP	*Falsos positivos*
VN	*Verdaderos negativos*
FN	*Falsos negativos*
TPR	*True positive rate*
FPR	*False positive rate*
FPCA	*Functional principal component analysis*
CNN	*Convolutional neural network*
DTW	*Dynamic time warping*
IQR	*Interquartile Range*
TPU	*Poliuretano termoplástico*
PCx	*Componente principal x*
SRAM	*Static Random Access Memory*
LPDDR	*Low Power Double Data Rate*
OTG	*On-The-Go*
SDIO	*Secure Digital Input/Output*
GPIO	*General Purpose Input/Output*
AES	*Advanced Encryption Standard*
SHA	*Secure Hash Algorithm*
ACP	*Advanced eXtensible Interface*

UNIVERSIDAD DE MÁLAGA

GMACS	*Giga Multiply-Accumulate Operations per Second*
CO	*Monóxido de carbono*

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Índice general

Agradecimientos XIII

Lista de acrónimos XVII

1. Introducción 1

1.1. Contexto y antecedentes 2

1.1.1. Sensores táctiles en manos robóticas 3

1.1.2. Dificultades en la implantación de sensores táctiles 4

1.1.3. Potenciales soluciones 5

1.2. Punto de partida 6

1.3. Objetivos de la tesis 7

1.4. Contribución de la tesis 7

1.4.1. Otras actividades dentro del grupo de investigación 9

1.4.2. Proyectos de investigación 9

1.5. Organización del documento 9

2. Materiales y métodos 13

2.1. Introducción 13

2.2. Transductor básico 14

2.3. Sensores táctiles utilizados 16

2.3.1. Sensor de dedo artificial 16

2.3.2. Sensor de palma artificial 32

2.4. Desarrollo del *setup* experimental 33

2.4.1. Electrónica de control de los motores 34

2.4.2. Software para el control de motores 36

2.4.3. Modalidad para texturas 38

2.4.4. Modalidad para rigidez 39

UNIVERSIDAD DE MÁLAGA

2.5. Herramientas para la implementación en SoC 40
2.5.1. Placa de desarrollo ZedBoard™ y Zynq-7020® 40
2.5.2. Entorno de desarrollo Vivado IDE ® 43
2.5.3. Entorno de desarrollo SDK 44
2.5.4. Utilización de FreeRTOS® + lwIP Stack 45
2.6. Resumen 45

3. Caracterización y clasificación de texturas **47**
3.1. Introducción 48
3.2. Algoritmos de extracción de características 50
3.2.1. Algoritmo de barrido o *sweep* 50
3.2.2. Resultados de simulación de barrido iniciales 54
3.2.3. Algoritmo de gradiente descendente 54
3.2.4. Resultados de aplicación del barrido para un conjunto sencillo de texturas 56
3.2.5. Algoritmo de Goertzel 58
3.3. Diseño de un filtro *antialiasing* 61
3.3.1. Especificaciones del filtro *antialiasing* 61
3.3.2. Obtención de los coeficientes del filtro con la herramienta FDATool® de Matlab® 62
3.4. Caracterización de texturas 64
3.4.1. Simulación con diseños ideales de distintas texturas 65
3.4.2. Simulación con una modulación en frecuencia progresiva 69
3.4.3. Selección del conjunto de texturas para experimentos 72
3.4.4. Exploración de texturas 75
3.5. Extracción de características 76
3.5.1. Algoritmo de barrido o *sweep* 77
3.5.2. *Fast Fourier Transform* (FFT) 81
3.5.3. Búsqueda de los índices del espectro para el algoritmo de Goertzel 83
3.6. Algoritmo de entrenamiento y clasificación *k-means* 86
3.6.1. Eliminación de *outliers* en el conjunto de entrenamiento 88
3.6.2. Algoritmo *k-means* 90
3.6.3. Visualización de la separación entre clases 91
3.7. Resultados en Matlab® 94
3.7.1. Resultados para el algoritmo de barrido o *sweep* 94
3.7.2. Resultados para *Fast Fourier Transform* (FFT) 98
3.7.3. Resultados para el algoritmo de Goertzel 99
3.8. Implementación de algoritmos en SoC 100

UNIVERSIDAD DE MÁLAGA

3.8.1. Módulos hardware en la parte PL del diseño 101
3.8.2. Implementación del barrido. Comparación en términos de consumo de recursos, potencia y latencia con la FFT 105
3.8.3. Implementación del algoritmo de Goertzel en SoC 114
3.8.4. Consumo real de recursos hardware y potencia en la parte PL . 117
3.8.5. Implementación de la parte PS del diseño 119
3.9. Resultados de clasificación 121
3.9.1. Comparativa entre barrido y FFT 121
3.9.2. Evaluación del algoritmo de barrido a través de las matrices de confusión 122
3.9.3. Análisis de la sensibilidad y especificidad para cada textura en un espacio *Receiver Operating Characteristic* (ROC) 126
3.9.4. Conclusiones en base a los resultados de clasificación del barrido 128
3.9.5. Resultados del algoritmo de Goertzel 129
3.9.6. Discusión general 130
3.10. Resumen 131

4. Reconocimiento y clasificación de objetos por rigidez 135
4.1. Introducción 136
4.2. Características propuestas para la clasificación 137
4.3. Exploración de objetos 142
4.3.1. *Setup* experimental 142
4.3.2. Objetos a explorar 142
4.3.3. Procedimiento para obtener los datos 144
4.3.4. Algoritmo de entrenamiento 146
4.4. Implementación en SoC 147
4.4.1. VHDL Synchronization Modules 148
4.4.2. VHDL Features Computing Module 150
4.4.3. HLS Principal Component Analysis (PCA) Computing Module 153
4.5. Resultados y discusión 156
4.5.1. Resultados obtenidos sin PCA 156
4.5.2. Resultados obtenidos con PCA 163
4.6. Conclusiones 168
4.7. Resumen 170

5. Conclusiones y líneas futuras 173
5.1. Conclusiones 173
5.2. Líneas futuras de trabajo 176

UNIVERSIDAD DE MÁLAGA

A. Figuras complementarias **177**

B. Tablas complementarias **189**

C. Consideraciones adicionales para el motor NEMA 17 y *driver* Pololu A4988 **193**
C.1. Resolución de los motores NEMA 17 y frecuencia de resonancia 193
C.2. Electrónica interna del *driver* Pololu A4988 y técnica de *microstepping* 195

D. Discretización de la lámina piezoresistiva mediante la impresora láser **199**
D.1. Descripción del *setup* utilizado 199
D.2. Procedimiento de discretización por láser de la lámina piezoresistiva . . 205

E. Diagramas de la arquitectura en Vivado® y configuración en SDK **213**
E.1. Diagramas de bloques para la detección de texturas 213
E.2. Diagramas de bloques para la distinción de la rigidez de objetos 216
E.3. Configuración SDK 221

F. Simulación en Vivado® **223**
F.1. Ejemplo de simulación para el barrido 223
F.2. Ejemplo de simulación para la detección de rigidez de objetos 227

Índice de figuras **231**

Índice de tablas **249**

Bibliografía **253**

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

1

Introducción

Esta tesis doctoral se centra en el desarrollo de distintos algoritmos de pre-procesado para la extracción de características que permitan reconocer propiedades a partir de la información táctil obtenida a través de la manipulación de objetos. Estos algoritmos se implementan en la electrónica local de sensores táctiles inteligentes que se encuentran en los dedos y palma de una mano robótica. La electrónica de estos sensores está basada en dispositivos lógicos programables, como *field-programmable gate arrays* (FPGAs). En concreto, en esta tesis se aborda la clasificación de un conjunto de texturas con diferentes frecuencias espaciales, y el reconocimiento de un conjunto de clases de objetos de diferente rigidez. En el primer caso, se propone la implementación en FPGA de un algoritmo de barrido que permite conocer la localización espectral de los armónicos principales de la textura explorada. La posición de estas frecuencias espaciales son los vectores de características que, posteriormente se envían a un clasificador que se ejecuta en tiempo real en un procesador. Este algoritmo ha demostrado ser muy eficiente en términos de consumo de recursos lógicos y de memoria, comparado con otros algoritmos de especificación en frecuencia como la transformada rápida de Fourier o *Fast Fourier Transform* (FFT). Por otro lado, para el reconocimiento de la rigidez de objetos, se propone la implementación hardware de cuatro características sencillas, extraídas en tiempo real a partir de las series temporales de los momentos de la imagen táctil. Estas características proporcionan información acerca de la rigidez del objeto cuando éste se deforma. Para diferentes escenarios, esta implementación se compara con la implementación de estas características junto a la técnica *principal component analysis* (PCA)

UNIVERSIDAD DE MÁLAGA

para reducir la dimensión del vector de características. Los resultados sostienen que la implementación sin PCA es ventajosa en términos de consumo de recursos y latencia.

En este capítulo de introducción, se presenta inicialmente el contexto de investigación y antecedentes (sección 1.1). A continuación, se explica cuál es el origen o punto de partida de esta tesis (sección 1.2). Después, se enumeran los objetivos generales y específicos a abordar, y las aportaciones de la tesis (sección 1.3 y 1.4, respectivamente). Finalmente, se presenta la estructura del documento (sección 1.5).

1.1. Contexto y antecedentes

En la actualidad, el reconocimiento de propiedades de objetos por parte de robots equipados con sensores inteligentes en los dedos y palma de manos artificiales, sigue siendo uno de los principales desafíos en el campo de la robótica. En particular, la detección de texturas y rigidez permite a los robots discernir entre diferentes superficies y materiales. Esta información táctil es crucial para realizar tareas de manipulación precisa, reconocer objetos con variaciones sutiles en su textura o rigidez, y adaptar el control y la fuerza aplicada durante la interacción con el entorno. Para ello, es fundamental que los robots manipuladores posean la habilidad necesaria para llevar a cabo estas tareas de manera eficiente. En este contexto, los sensores táctiles se han convertido en herramientas fundamentales como dispositivos de entrada al sistema [1] [2] [3].

Con el avance de la tecnología, se ha dotado a los sensores táctiles de inteligencia, lo que ha hecho evolucionar su ámbito de aplicación. Inicialmente, los robots manipuladores se utilizaban principalmente en entornos industriales para tareas de automatización y cadenas de montaje. Sin embargo, en la actualidad, estos autómatas, junto con los sensores táctiles, se han vuelto más versátiles y se utilizan en aplicaciones más especializadas. Por ejemplo, se emplean en la medición y detección de propiedades en diversos materiales, lo cual resulta útil en campos como la ingeniería de materiales [4]. Además, los sensores táctiles desempeñan un papel fundamental en la colaboración entre humanos y robots, ya que permiten una interacción segura y eficiente en entornos compartidos. Asimismo, se utilizan en la asistencia a personas mayores, proporcionando una retroalimentación táctil precisa y adaptada a sus necesidades [5]. Esta evolución hacia sensores táctiles inteligentes ha ampliado las posibilidades y ha abierto nuevas fronteras en el campo de la robótica.

Otra aplicación directa de los sensores táctiles se encuentra en el campo de la ingeniería biomédica. Un ejemplo a destacar es el desarrollo de prótesis cada vez más avanzadas [6] [7], debido a que el número de personas que sufren algún tipo de amputación

UNIVERSIDAD DE MÁLAGA

de extremidades es cada vez más significativo en todo el mundo. Sólo en España, es de 1.88 por cada 1000 habitantes[1]. En Estados Unidos hay 1.6 millones de personas con esta dolencia, estimándose a más del doble para el año 2050 [9]. Esto motiva el estudio y utilización de mecanismos táctil-sensorizados que permitan a estas personas mejorar su calidad de vida [10] [11], figura 1.1.

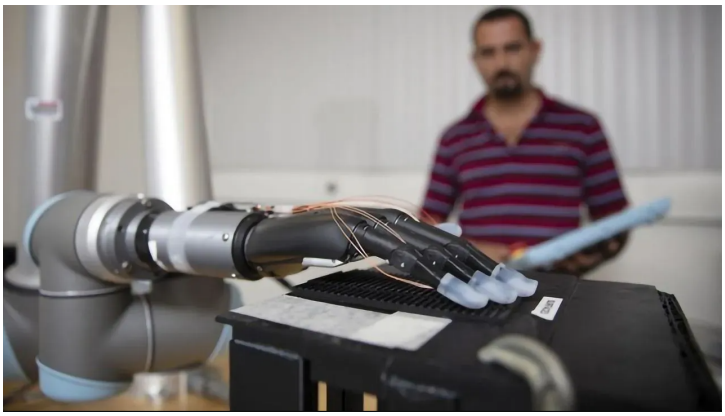


Figura 1.1. El desarrollo de sensores táctiles en manos robóticas se puede utilizar para aplicaciones en prostética [12].

1.1.1. Sensores táctiles en manos robóticas

Uno de los principales campos de estudio en la investigación de sensores táctiles es la medida de la presión y la fuerza [13]. Por ello, muchas investigaciones incorporan estos dispositivos en los dedos de las manos robóticas con el objetivo de medir fuerzas de forma multidimensional [14] [15] [16] [17] [18] [19]. Los sensores multimodales se pueden integrar en manos robóticas. Los autores en [20] utilizan una mano robótica recubierta por una red de sensores capaz de diferenciar fuerzas normales y tangenciales en tiempo real. Por otro lado, en [21] utilizan una mano robótica con falanges elásticas, que integra múltiples sensores y circuitos electrónicos sin cableado (*wireless*). Aunque este tipo de sensores puede detectar la dirección de la fuerza, no son tan precisos a la hora de detectar la posición de contacto. De forma gradual, la mayoría de investigaciones han evolucionado hacia *arrays* de sensores táctiles. En [22] se diseña un sensor bio-inspirado con un *array* táctil basado en dispositivos MEMS[2]. Una gran parte de manos robóticas están equipadas con sensores multi-*array*. Actualmente, muchos de los sensores comerciales son de este tipo. Entre los más representativos se encuentra el sensor multimodal BioTac, diseñado por Syntouch® [23]. Este sensor puede proporcionar información de

[1] Información más reciente proporcionada por el Instituto Nacional de Estadística (INE), según [8].

[2] Los dispositivos microelectromecánicos o abreviadamente MEMS son una tecnología de bajo costo y alto rendimiento, que permite integrar sensores táctiles en la microescala.

UNIVERSIDAD DE MÁLAGA

la fuerza de contacto midiendo microvibraciones y variaciones de temperatura inducidas por el contacto con objetos del entorno. De esta forma, los sensores incorporados en manos robóticas se pueden utilizar para medición de temperatura, humedad o presión en diferentes entornos, almacenamiento de energía para sistemas táctiles autosostenibles, y comunicación sin cables a partir de la integración de módulos de comunicación que permiten el procesamiento y la transmisión de datos en tiempo real.

Los últimos avances en sensores táctiles flexibles han ampliado su ámbito de aplicación, constituyendo herramientas muy prometedoras en las interfaces humano-máquina y de control. Gracias a estos sistemas, las personas pueden interactuar con robots o máquinas a través de teclados, paneles y guantes inteligentes. Además, el uso de sensores táctiles también está en auge en aplicaciones de realidad aumentada, permitiendo la manipulación precisa de objetos frágiles, la investigación de propiedades de nuevos materiales y la detección de pequeñas fisuras o relieves en superficies [5]. Otra área de aplicación en desarrollo es la cirugía robótica mínimamente invasiva, en la que los sensores táctiles permiten una mayor precisión y control en la manipulación de los instrumentos quirúrgicos [24].

1.1.2. Dificultades en la implantación de sensores táctiles

La adopción y aplicación efectiva de los sensores táctiles sigue siendo un desafío debido a la dificultad en la adquisición y procesamiento de grandes cantidades de datos provenientes de múltiples fuentes [25] [26] [27]. La destreza en la manipulación de robots aún está muy lejos de igualar a las de las manos humanas, y la retroalimentación táctil sigue siendo difícil de resolver. La detección táctil implica la gestión de una gran cantidad de datos, y los buses de comunicación en el robot deben ser capaces de transmitir la información proporcionada por estos y otros sensores, así como los comandos para el control de los actuadores [25]. En el caso de la comunicación en robots humanoides, se prefieren los buses serie para minimizar el espacio y reducir las interferencias mecánicas y eléctricas. Aunque se ha utilizado el bus serie I2C en algunos trabajos, su implantación puede conllevar un tiempo de muestreo largo del sistema de sensores y no garantiza una latencia limitada [28] [29]. Para asegurar una latencia limitada en la información crítica en tiempo real, un enfoque común es implementar redes basadas en bus CAN en los niveles sensoriales, y redes basadas en LAN/WLAN Ethernet, RS232 o USB en los niveles de control y organización donde no se requiere un tiempo real estricto [30] [31] [32].

UNIVERSIDAD DE MÁLAGA

La frecuencia de muestreo específica del sistema también puede variar dependiendo del tipo de actividad, que puede ser acción para la percepción (por ejemplo, reconocimiento de objetos) o percepción para la acción (por ejemplo, control de agarre) [25]. Los requisitos son más altos para tareas que implican acciones de percepción, necesitando una resolución espacial de hasta 0.8 mm en la punta de los dedos de una mano robótica y un tiempo de respuesta en torno a pocos milisegundos. Además, para lograr un control de retroalimentación suave y estable, la frecuencia de muestreo debe ser más de diez veces la frecuencia de resonancia del sistema controlado, lo que implica un período de muestreo de 2-100 ms [33]. También se sabe que la información registrada por el dedo humano sobre el estado inicial de fricción del contacto (forma de la superficie y dirección de las fuerzas) comienza a reflejarse en la acción ejercida por el dedo en aproximadamente 70 ms [3] [34] [35]. En resumen, se requiere una transmisión y procesamiento eficiente de grandes cantidades de datos en un breve período de tiempo para lograr una detección táctil efectiva y un control de retroalimentación preciso.

1.1.3. Potenciales soluciones

El pre-procesado de los datos cerca del sensor, antes de transmitir la información a niveles superiores del sistema, puede reducir significativamente el tráfico de información entre los sensores y los controladores, lo que permite aumentar la complejidad del sistema incrementando el número de sensores que adquieren información. Esta arquitectura distribuida libera a la unidad central de procesamiento, típicamente basada en un microprocesador o FPGA, para que pueda centrarse en la realización de tareas inteligentes de alto nivel. La información esencial debe extraerse y enviarse al procesador central en el menor tiempo posible para cumplir con los requisitos dinámicos de tiempo real del sistema.

La discriminación de texturas a través de su frecuencia espacial y el reconocimiento de rigidez son propiedades esenciales en aplicaciones con sensores táctiles. El preprocesamiento de la información táctil de estos sensores en la electrónica local del sensor reduce la cantidad de datos que se envían al procesador central. Para lograr esto, es primordial el desarrollo de algoritmos de pre-procesado embebidos en la electrónica del sensor. Estos algoritmos permiten la extracción de características que ayudan a la detección de texturas y rigidez de objetos. Por lo tanto, la investigación en el desarrollo de sensores táctiles inteligentes, que permitan un pre-procesado de las imágenes táctiles, se hace necesaria para cumplir con las exigencias de manipulación en tiempo real.

UNIVERSIDAD DE MÁLAGA

1.2. Punto de partida

En el trabajo relativo a esta tesis se parte del sensor táctil de dedo artificial diseñado y fabricado en [36]. Este dedo contiene una electrónica local basada en FPGA. Una de las partes del dedo consiste en un *array* de electrodos. Para mejorar la respuesta del sensor se discretizó una lámina piezoresistiva que, junto al diseño y fabricación 3D de un recubrimiento con superficie estructurada, se montó encima de los electrodos. Además, se utiliza la palma artificial también desarrollada en [36]. Estos sensores se pueden colocar en una mano de Barrett [37], ver figura 1.2. La peculiaridad de esta propuesta frente a otras es que la conversión A/D se hace por conexión directa sensor-FPGA, sin convertidores específicos. Por tanto, la adquisición y el procesamiento se puede llevar a cabo de forma paralela, lo que permite la implementación local de primitivas de pre-procesado complejas, para obtener resultados parciales que se transfieran a unidades de procesamiento de alto nivel. Esta aproximación es única en el escenario del hardware para sensores táctiles [25].

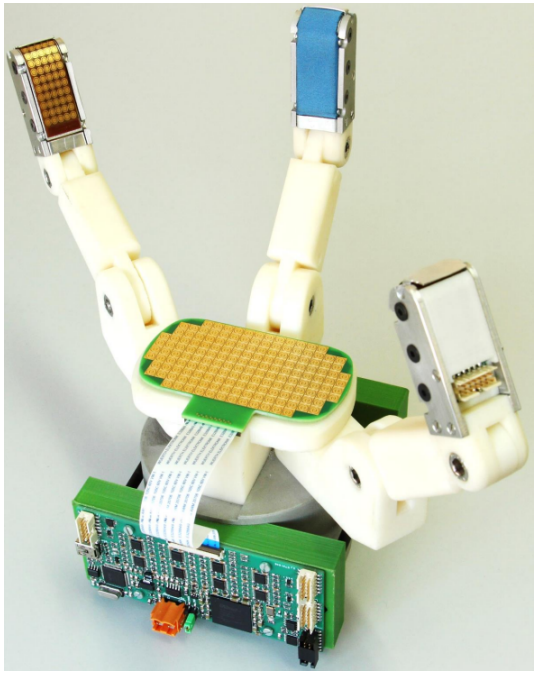


Figura 1.2. Prototipo de mano de Barrett [38], con los sensores correspondientes a los dedos y palma artificiales de los que se parte en esta tesis y la PCB con la electrónica de adquisición y comunicación para estos sensores [36].

A partir de la información táctil recogida por los sensores de dedo y palma, se pueden implementar algoritmos de pre-procesado que permitan encontrar descriptores o características propias de la superficie de la textura u objeto a explorar. Estos algoritmos estarán embebidos en la electrónica local, próxima al sensor táctil. Posteriormente, se usa un algoritmo de clasificación, que normalmente incluye un aprendizaje previo.

UNIVERSIDAD DE MÁLAGA

1.3. Objetivos de la tesis

El objetivo general que se plantea es la implementación de primitivas de pre-procesado de la información táctil en la electrónica local basada en FPGA para el reconocimiento de texturas y rigidez de objetos. En la figura 1.3 se presentan esquemas conceptuales de la aplicación de estos algoritmos.

Para el desarrollo de esta tesis, se han llevado a cabo otros objetivos más específicos:

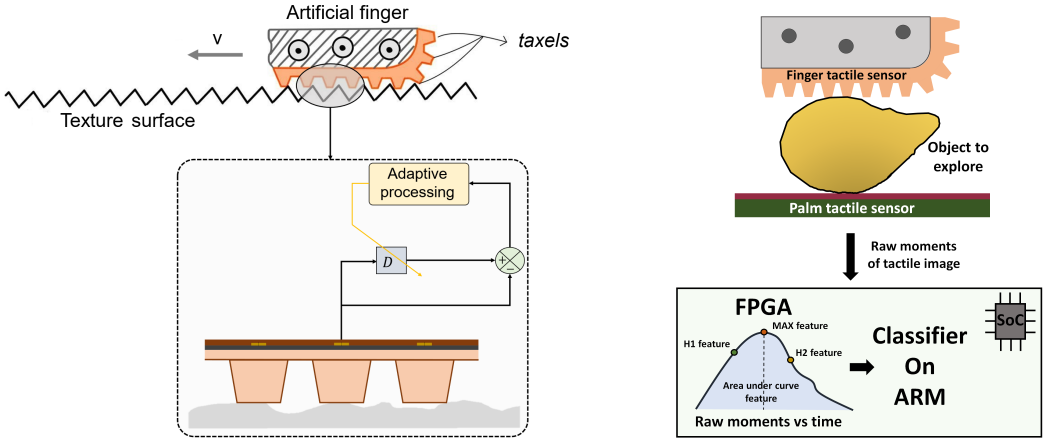
1. Diseño de la cubierta o *cover* del sensor táctil del dedo artificial y palma. En concreto, para el dedo, desarrollo de una cubierta con la estructuración de su superficie y discretización de la lámina piezoresistiva, con la triple finalidad de concentrar la fuerza en los tácteles (o unidades de sensado de fuerza) de la matriz de electrodos, aumentar por tanto la sensibilidad del sensor, y reducir la diafonía o *crosstalk*.
2. Propuesta e implementación de algoritmos de pre-procesado en la electrónica local al dedo, basada en FPGA, que permitan la extracción de características para identificación de texturas en tiempo real.
3. Comparación de las propuestas de algoritmos con la aplicación de una FFT embebida, en términos de consumo de recursos lógicos y de memoria, consumo de potencia y latencia.
4. Propuesta e implementación de algoritmos de pre-procesado en FPGA, para el reconocimiento de la rigidez a partir de la deformación de diferentes objetos explorados en tiempo real. Análisis del consumo de esta propuesta en FPGA.

1.4. Contribución de la tesis

La presente tesis tiene como aportación más relevante la propuesta e implementación de algoritmos de pre-procesado originales en la electrónica local de un sensor táctil inteligente. Desde una perspectiva científica, se busca explorar el potencial del tacto activo para la extracción de características en la detección de texturas y rigidez de objetos.

En concreto, se propone un algoritmo de barrido que permite la identificación de texturas en tiempo real y se comparan sus resultados con los obtenidos a través de una FFT. Esta técnica se ha desarrollado a partir del algoritmo adaptativo presentado en [41]. Inicialmente, se propone una primera versión del algoritmo en Matlab®. Los resultados de este primer trabajo se presentaron en el congreso *IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics*, (BioRob 2020) [42]. El posterior

UNIVERSIDAD DE MÁLAGA



(a) (b)

Figura 1.3. (a) Aplicación del algoritmo de pre-procesado para identificación de texturas de diferente resolución espacial [39]. (b) Experimentos para el reconocimiento de la rigidez de objetos, mediante su palpación experimental [40].

desarrollo e implementación en un dispositivo *System on Chip* (SoC) de este algoritmo, permitió la caracterización e identificación de hasta 12 texturas de diferentes frecuencias espaciales. Estos resultados están avalados por una publicación en la revista *IEEE Sensors Journal* [39], en la que el autor de la tesis es primer autor. Se trata de una revista JCR de alto índice de impacto, situada en primer cuartil en 2022. Por otro lado, se propone la implementación de un algoritmo adicional de detección en frecuencia, llamado "algoritmo de Goertzel", y se comparan sus resultados con la FFT y el algoritmo de barrido. Resultados preliminares de este trabajo se presentaron en el congreso *International Work Conference on Artificial Neural Networks* (IWANN 2023) [111]. El análisis y resultados de esta parte de la tesis se presentan en el capítulo 3.

Por otra parte, en esta tesis también se implementa un procedimiento completo, que permite reconocer un conjunto amplio de objetos de diferentes rigideces. La extracción de características se hace a partir del cálculo de los momentos de la imagen táctil, ya implementado en la electrónica del sensor. Resultados preliminares de esta parte se presentaron en el congreso *IEEE International Symposium on Industrial Electronics* (ISIE 2020), [43]. Este trabajo se amplió para poder clasificar hasta 42 clases diferentes de objetos, a partir de una propuesta de implementación de nuevos descriptores. Estos resultados están avalados por una publicación en la revista *MDPI Sensors* [40], en la que el autor de la tesis es primer autor. Se trata de una revista JCR de alto índice de impacto, situada en segundo cuartil (primer tercil) en 2022. El análisis y resultados de esta parte se presenta en el capítulo 4.

UNIVERSIDAD DE MÁLAGA

1.4.1. Otras actividades dentro del grupo de investigación

Asimismo, durante el desarrollo de esta tesis, y dentro del grupo de investigación, "Electrónica para Instrumentación y Sistemas" (EIS), el autor de la misma ha participado en otras tareas de investigación en relación con el desarrollo de un bastón instrumentado [44]. También ha participado en el seguimiento de dos TFG y tres TFM. Además, al estar la tesis vinculada a una beca de "Formación de Profesorado Universitario" (FPU), el doctorando ha dedicado parte del tiempo de formación a impartir las prácticas de la asignatura "Electrónica Digital" del Grado en Ingeniería de Tecnologías Industriales.

Finalmente, y en este contexto, mencionar que el grupo de investigación EIS ha estado desde hace muchos años ligado a iniciativas relacionadas con la mecatrónica y la robótica en la Universidad de Málaga, como Cursos de Doctorado, Máster de Ingeniería Mecatrónica, Escuelas de Verano, Congresos, etc., teniendo un contacto fluido con grupos dedicados a robótica. Recientemente, el grupo se ha incorporado al Instituto Universitario de Investigación en Ingeniería Mecatrónica y Sistemas Ciberfísicos (I3MSC), y pertenece al Instituto de Investigación Biomédica de Málaga (IBIMA), acreditado por el Instituto de Salud Carlos III, base para posibles transferencias de conocimiento.

1.4.2. Proyectos de investigación

Esta tesis doctoral se ha llevado a cabo en el marco de desarrollo de dos proyectos financiados por el Gobierno de España y por el Fondo Europeo de Desarrollo Regional (FEDER):

- Proyecto del Plan Nacional: "Sensores táctiles en sistemas mecatrónicos para salud", con referencia TEC2015-67642-R, con inicio en enero de 2016 y finalizado en diciembre de 2020. El autor de esta tesis formó parte del equipo del proyecto.
- Proyecto de Generación de Conocimiento: "Instrumentación inteligente y aplicación en salud", con referencia PID2021-125091OB-I00, con inicio en enero de 2022 y finalización en diciembre de 2024. El autor de esta tesis forma parte del equipo del proyecto.

1.5. Organización del documento

Esta tesis doctoral se ha estructurado en un total de cinco capítulos:

- **Capítulo 1**. Es el presente capítulo, que contiene una introducción al contexto de este trabajo de tesis, así como sus objetivos y aportaciones.

UNIVERSIDAD DE MÁLAGA

- **Capítulo 2**. Explica la tecnología empleada en los sensores relativos al dedo artificial y la palma, así como el *setup* experimental utilizado para los experimentos de esta tesis. En primer lugar, se introducen los diferentes tipos de sensores utilizados en robótica. A continuación, se profundiza en la estructura del dedo y el mecanizado de la superficie del material que lo cubre. También se explica por qué es importante discretizar la lámina piezoresistiva que está en contacto con los electrodos. Finalmente, se describen las herramientas utilizadas para la implementación de los algoritmos de pre-procesado.

- **Capítulo 3**. Se centra en la caracterización de diferentes texturas y su clasificación. Se describe la propuesta de algoritmo basada en un barrido para identificar las frecuencias fundamentales de la textura explorada. Este algoritmo es novedoso respecto a las propuestas de trabajos previos. Se compara el rendimiento de este algoritmo respecto a la implementación de una FFT, en términos de porcentaje de clasificación y consumo de recursos hardware y latencia. En este capítulo además, se presentan resultados de implementación de la aplicación de un algoritmo en frecuencia, denominado "algoritmo de Goertzel", comparándolo también al rendimiento de la FFT y el algoritmo de barrido.

- **Capítulo 4**. Se propone un procedimiento para la clasificación de un conjunto de objetos de distinta rigidez, a partir de la implementación hardware de cuatro características sencillas, extraídas de las secuencias temporales de los momentos de la imagen táctil. Se realiza una comparación de la implementación de estas características por separado y junto a la técnica de PCA, teniendo en cuenta los resultados de clasificación y el consumo de recursos hardware y latencia.

- **Capítulo 5**. Presenta las conclusiones obtenidas de este trabajo de tesis y propone posibles líneas de investigación futuras que se derivan del mismo.

Adicionalmente, se incluyen seis apéndices:

- **Apéndice A**. Apéndice de figuras, que incluye figuras adicionales, que complementan a los capítulos de la tesis.

- **Apéndice B**. Apéndice de tablas, que incluye tablas adicionales, que complementan a los capítulos de la tesis.

- **Apéndice C**. Incluye consideraciones adicionales sobre los motores NEMA 17 y el *driver* Pololu utilizados en el *setup* experimental.

- **Apéndice D**. Explica cómo se hace la discretización de la lámina de piezoresistivo mediante la máquina de grabado y corte por láser.

UNIVERSIDAD DE MÁLAGA

- **Apéndice E**. Incluye los diagramas de Vivado® que se han diseñado para la arquitectura con los algoritmos de extracción de características.
- **Apéndice F**. Incluye simulaciones realizadas con la herramienta XSIM que proporciona Vivado®.

Finalmente, en el repositorio público en *https://github.com/raulLkRb/thesis.git* se pueden encontrar códigos fuente utilizados a lo largo de la tesis; en VHDL, Matlab®, SDK y HLS de Vivado®, así como algunos *scripts* (archivos .tcl) de simulación y enlaces a *datasets* utilizados.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

2

Materiales y métodos

Los sensores táctiles y el *setup* experimental utilizados para llevar a cabo los experimentos de esta tesis, se describen a lo largo de este capítulo. La sección 2.1 proporciona una introducción al tema. En la sección 2.2 se explican los diferentes tipos de sensores que se utilizan en aplicaciones robóticas. A continuación, sigue la sección 2.3, que describe los sensores utilizados en los experimentos de esta tesis. En esta parte, se profundiza en el diseño de la capa de material piezoresistivo y el *cover* estructurado incorporados al diseño del dedo robótico original. En la sección 2.4 se describen las diferentes partes de las que consta el *setup* experimental. Esto incluye la electrónica de control utilizada para el movimiento y el control de las trayectorias de los ejes de la impresora 3D adaptada para el *setup*, así como el sistema de adquisición de datos. En particular, se detallan las principales diferencias en el *setup* experimental para los experimentos de texturas y los de rigidez de forma independiente. En la sección 2.5 se presentan las herramientas para el diseño e implementación del sistema en el dispositivo SoC considerado. Finalmente, en la sección 2.6 se presentan las principales conclusiones de este capítulo.

2.1. Introducción

La exploración de texturas u objetos con distintas propiedades, se realiza mediante robots o sistemas autónomos con diferentes grados de libertad en sus movimientos. Estos robots

UNIVERSIDAD DE MÁLAGA

utilizan pinzas o *grippers* en su efector final, que consisten en falanges o dedos equipados con distintos sensores que permiten la manipulación de objetos a través del palpado y la realimentación de fuerzas en el sistema. A menudo se fabrican cubiertas o *covers* basados en materiales sólidos continuos, que distribuyen la presión ejercida en un punto a lo largo de todo el área de sensado [41]. Sin embargo, esto puede provocar el fenómeno del *crosstalk* o lecturas fantasmas entre tácteles del mismo sensor. Para evitar este problema, se pueden diseñar *covers* con elastómeros que contengan algún tipo de estructura sobre su superficie, lo que permite concentrar la fuerza sobre tácteles de forma independiente y mejorar la sensibilidad del sensor, así como reducir el *crosstalk* entre tácteles [45].

Además, los elastómeros estructurados facilitan la deformación elástica y su posterior recuperación mecánica de energía, lo que mejora la respuesta dinámica del sensor [1] y reduce la histéresis [46]. Para la exploración de diferentes superficies, los tácteles de los sensores deben ser capaces de detectar microvibraciones durante el desplazamiento, lo que permite clasificar correctamente los distintos tipos de material explorados [27]. Para detectar estas vibraciones, se pueden utilizar acelerómetros [47] o materiales piezoeléctricos [48] [32], aunque el uso de sensores piezoresistivos es más económico. El uso de *covers* estructurados también es ventajoso para amplificar mecánicamente estas vibraciones. Se han propuesto soluciones en forma de crestas artificiales [45] [47] [48], cilindros o conos [32], e incluso estructuras micropiramidales [49].

En este trabajo de tesis, se colocó sobre la matriz de electrodos del dedo artificial propuesto en [36], una capa de material piezoresistivo con tácteles aislados mediante una máquina de corte por láser, y sobre ésta, un *cover* estructurado que permite la detección de microvibraciones y concentrar las fuerzas sobre las áreas de sensado de los tácteles de la matriz. Una vez se tiene el sensor completo, se añade a un robot cartesiano, adaptado a partir de una impresora 3D. Con el uso de este sensor, y controlando las trayectorias de los ejes de movimiento de este robot, se pueden explorar diferentes texturas y superficies con patrones espaciales definidos. Además, el uso combinado de este sensor con la palma robótica, también propuesta en [36], permite la exploración de objetos con diferentes geometrías y propiedades mecánicas.

2.2. Transductor básico

En sistemas robóticos, se utilizan transductores consistentes en sensores de alta resolución y sensibilidad para obtener información tanto del movimiento del robot como de su entorno. Los sensores intrínsecos (o propioceptivos) proporcionan datos sobre la cinética y dinámica del movimiento, como el torque y las fuerzas en los tendones de

UNIVERSIDAD DE MÁLAGA

robots [50] [51] [52]. Por otro lado, los sensores extrínsecos (o exteroceptivos) detectan cambios en el entorno, como fuerza, presión y temperatura [53] [3]. Por tanto, estos últimos son adecuados para su uso en aplicaciones táctiles. Según su tecnología y principio de funcionamiento, se clasifican en piezoresistivos, piezoeléctricos, capacitivos, triboeléctricos u ópticos [54].

Los sensores capacitivos miden cambios en la capacidad de un condensador para detectar estímulos mecánicos. Son comunes en pantallas inteligentes y también se utilizan en aplicaciones de robótica quirúrgica. Por su parte, los sensores piezoeléctricos generan potenciales eléctricos bajo presión y son ideales para detectar eventos dinámicos a altas frecuencias, pero no son adecuados para estímulos estáticos [55]. Algunos materiales piezoeléctricos, como el fluoruro de polivinilideno (PVDF) y sus co-polímeros, se utilizan en la construcción de estos sensores. Así, en [32], un sensor táctil basado en PVDF se integra en una mano antropomórfica.

Por otro lado, los sensores triboeléctricos convierten energía mecánica en eléctrica. Algunos materiales orgánicos como PDMS, PVC, PTFE, el grafeno o el aluminio son ideales para estos sensores debido a su comportamiento de transporte ambipolar[1] [56]. Estos sensores se utilizan en dispositivos de almacenamiento de energía, en dispositivos con teclado táctil o en interfaces humano-máquina. En cuanto a los sensores ópticos, estos detectan cambios en la iluminación y son adecuados para aplicaciones que requieren alta resolución espacial, como la detección de rugosidad en superficies y elasticidad en objetos [57].

Por último, los sensores piezoresistivos tienen un menor coste y se comercializaron antes que el resto de sensores disponibles. Estos sensores permiten detectar cambios en la resistencia eléctrica del material al cambiar la fuerza o presión sobre ellos. Gracias a su excelente conductividad eléctrica, facilidad en la implementación de circuitos de entrada/salida y flexibilidad en la nanoescala, estos sensores son comúnmente usados en aplicaciones industriales. Los sensores táctiles empleados en esta tesis incorporan láminas de material piezoresistivo, que se describirán en la próxima sección.

[1]El comportamiento de transporte ambipolar en semiconductores resulta del desplazamiento en el nivel de Fermi, de partículas de carga eléctrica opuesta que se encuentran bajo la acción de un campo eléctrico.

UNIVERSIDAD DE MÁLAGA

2.3. Sensores táctiles utilizados

En esta sección se presentan los dos sensores utilizados en los experimentos de este trabajo. Por un lado, tanto para los experimentos de clasificación de texturas como de rigidez, se ha utilizado el sensor en el dedo artificial. Por otra parte, para el *setup* específico en los experimentos de palpación de objetos, también se hace uso del sensor en la palma robótica.

2.3.1. Sensor de dedo artificial

En la figura 2.1 se muestran las diferentes partes del sensor correspondiente al dedo robótico artificial utilizado en los experimentos de esta tesis doctoral.

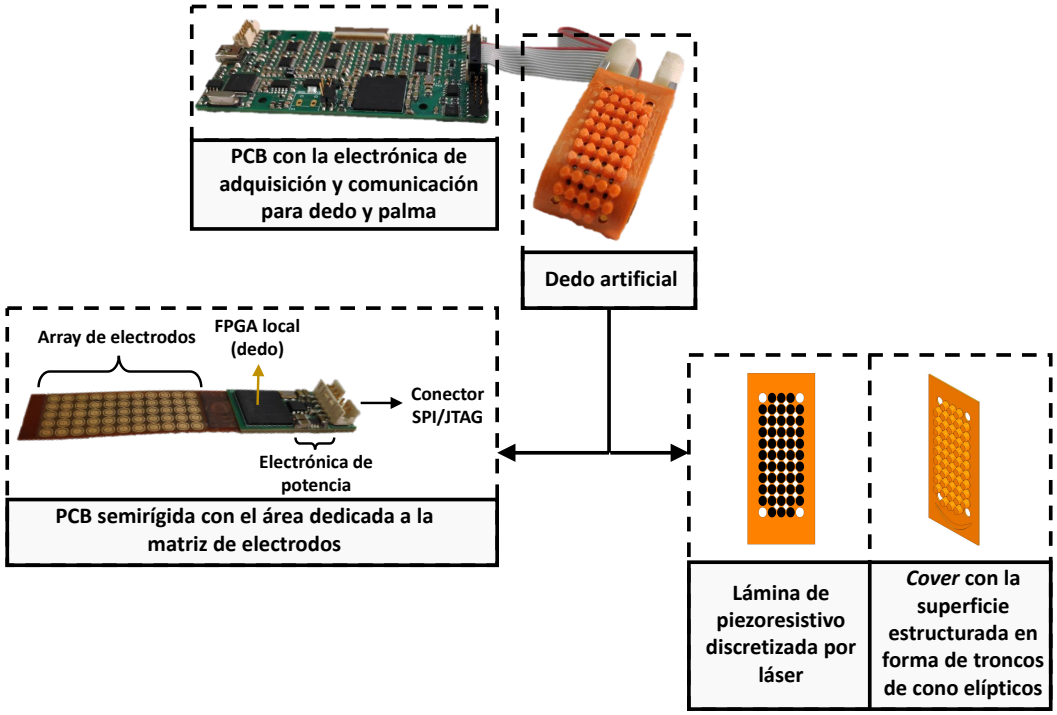


Figura 2.1. Sensor correspondiente al dedo artificial, consistente en una placa de circuito impreso o *Printed-Circuit Board* (PCB) semirígida con el *cover* estructurado que ayuda a concentrar la fuerza en los diferentes tácteles de la matriz de electrodos. Por la parte de atrás del *cover* se monta una lámina piezoresistiva discretizada. El sensor se puede conectar mediante el protocolo bus serie *Serial Peripheral Interface* (SPI) a una placa basada en FPGA con la electrónica de adquisición y comunicación propuesta en [36]. Las figuras no están a escala.

El sensor del dedo parte del diseño proporcionado en [36]. Este sensor táctil contiene una parte electrónica consistente en una PCB semirígida, integrada en una estructura mecánica. La electrónica del dedo está basada en una FPGA de la familia Spartan-6® de Xilinx®. La placa a su vez incorpora un circuito de alimentación, una PROM para la configuración de la FPGA, circuito de reloj y un conector para comunicación externa

UNIVERSIDAD DE MÁLAGA

por SPI[2]/JTAG. La parte semirígida consiste en una matriz de 55 electrodos dispuesta en 5 filas y 11 columnas. La matriz táctil de electrodos tiene una dimensión de (15×40.7) mm y la resolución espacial del sensor, o mínima distancia entre dos tácteles, es de 3.7 mm.

Diseño y fabricación del *cover* estructurado

Un *cover* consiste en una capa externa añadida al material piezoresistivo para mejorar la lectura en los electrodos del sensor. Normalmente, el material de esta capa suele ser flexible y deformable, lo que le permite tener elasticidad cuando se aplican fuerzas sobre él.

El diseño del *cover* se ha llevado a cabo con el programa de edición 3D SolidWorks®. Por otra parte, se ha fabricado con una impresora 3D[3] y el material Filaflex®. Este material se caracteriza por su gran elasticidad ya que se trata de un elastómero termoplástico. Su densidad es 1.2 g/cm^3 y su temperatura de fusión está en el rango de 210 a 230 grados centígrados [58].

Para ilustrar el comportamiento de un único táctel ante estímulos laterales de la fuerza, se realizó una simulación con el software Comsol®, como se muestra en la figura 2.2. En particular, la figura 2.2(a) muestra las dimensiones utilizadas en la simulación, mientras que la figura 2.2(b) muestra la distribución de presión a lo largo del contacto entre la capa sensible de piezoresistivo para diferentes fuerzas laterales (Fx), con dimensiones fijas para el sustrato continuo en la parte inferior ($Hb = 0.3$ mm) y la altura del cilindro encima del electrodo ($H = 1.2$ mm). Como se puede observar, la distribución de presión es menos uniforme a medida que la fuerza lateral aumenta, lo que puede causar variaciones significativas en la salida del sensor debido a la relación no lineal entre la presión de contacto y la resistencia de contacto [37].

Las figuras 2.2(c) y 2.2(d) muestran la influencia de la altura del cilindro y el ancho del sustrato inferior. En general, cuanto mayor sea la altura del cilindro H, mayor será la

[2]El protocolo SPI es una interfaz de comunicación síncrona que permite la transferencia de datos entre dispositivos mediante cuatro líneas de comunicación: *Master Output Slave Input* (MOSI), *Master Input Slave Output* (MISO), *Serial Clock* (SCK) y *Slave Select* (SS). La transferencia de datos se realiza mediante la sincronización entre el reloj maestro y el dispositivo esclavo, que envía o recibe datos a través de las líneas de datos de entrada y salida. El protocolo es ampliamente utilizado en aplicaciones de sensores, memorias, pantallas y otros dispositivos periféricos en sistemas embebidos.

[3]A partir del software de código abierto Ultimaker Cura, se pueden transformar los archivos generados por SolidWorks® a archivos con extensión .gcode, con instrucciones específicas para la impresora. Estos archivos se pueden almacenar en una tarjeta SD, que se inserta en una ranura de entrada de la impresora 3D antes de comenzar el proceso de impresión.

UNIVERSIDAD DE MÁLAGA

disparidad en la distribución de presión. Por otro lado, la influencia de la altura del sustrato, Hb, en la presión de contacto es pequeña para las dimensiones de nuestro táctel, ya que $Hb \ll H$. Además, gracias a una reducida área de contacto con el objeto explorado, se logra reducir la adherencia, lo que mejora la respuesta a estímulos tangenciales, como se ha observado para formas cónicas [32] y piramidales [59].

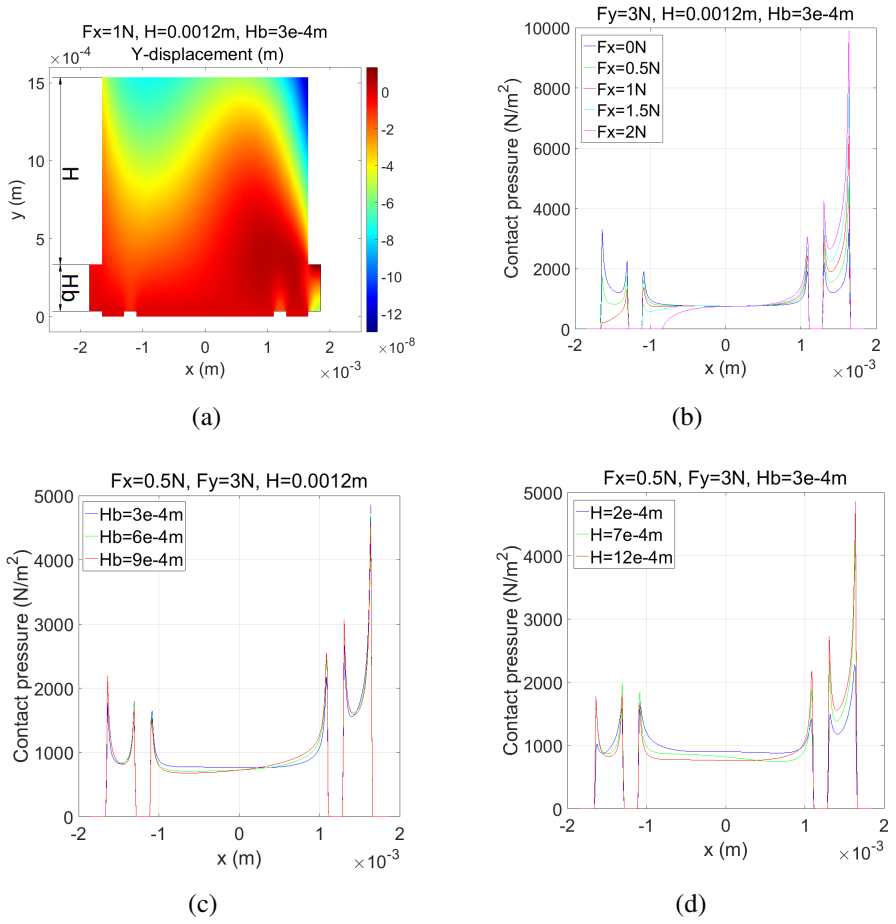


Figura 2.2. Simulaciones de elementos finitos en Comsol®: (a) dimensiones del *cover* estructurado, (b) distribución de la presión a lo largo del área de contacto para diferentes fuerzas tangenciales, (c) diferentes alturas para la estructura en cada táctel del sensor, y (d) anchura del sustrato. Estos resultados se presentaron en el congreso CIBEC 2018 [60].

En la figura 2.3 se muestra una imagen del diseño final considerado para el *cover* estructurado. Los salientes de la superficie se diseñan en forma de tronco de cono para ser sensibles tanto a fuerzas normales como tangenciales [41].

El *cover* de la figura 2.3 se escogió como el diseño principal para los experimentos de texturas y rigidez de objetos. No obstante, se valoraron otras alternativas antes de esta elección, las cuales se muestran en la figura 2.4.

Algunas diferencias del diseño utilizado respecto a los de la figura 2.4 son las siguientes:

UNIVERSIDAD DE MÁLAGA

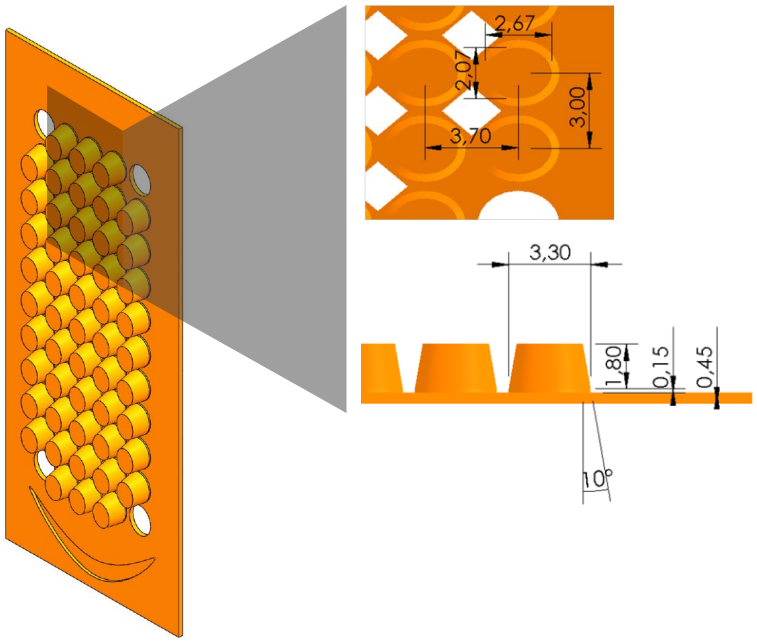


Figura 2.3. Diseño del *cover* estructurado para el dedo. Las cotas de distancia están dadas en mm.

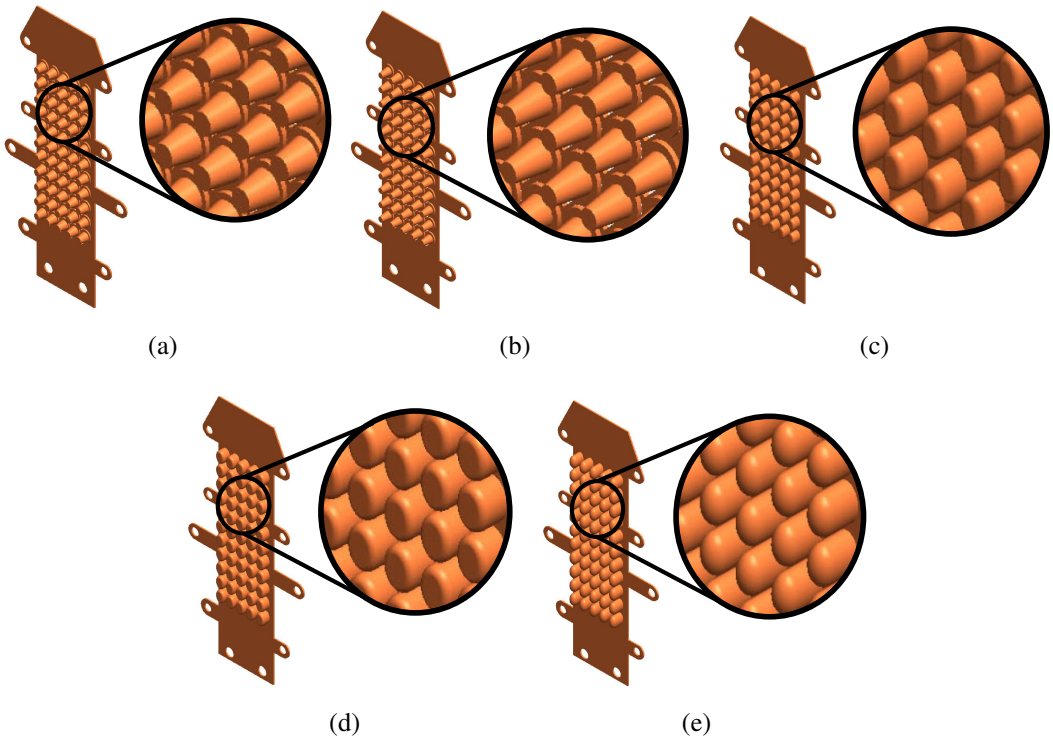


Figura 2.4. Perspectiva isométrica de otros diseños para el *cover* con *zoom* del detalle de la superficie estructurada. Los diseños (a) y (b) están basados en saliente en forma de tronco de cono más esbeltos que el de la figura 2.3. Los diseños (c) y (d) son cilíndricos con los extremos redondeados. El diseño (e) tiene un acabado semiesférico.

- En los diseños de la figura 2.4, se incluyeron solapas laterales para atornillar la estructura mecánica del dedo. Posteriormente, se eliminaron porque causaban un

UNIVERSIDAD DE MÁLAGA

abultamiento en la parte interna en contacto con los electrodos.

- Los diseños (a) y (b) de la figura 2.4 tienen salientes en forma de tronco de cono. Sin embargo, a diferencia del diseño principal en la figura 2.3, los salientes originales eran más esbeltos (elipse con diámetros más pequeños y mayor altura). Se cambió a un diseño de menor altura y más achatado, ya que el primero no concentraba la fuerza adecuadamente en el área de sensado de la matriz de electrodos.
- La base del *cover* inicial era más estrecha, pero se descubrió que los troncos de cono se desplazaban ligeramente durante las exploraciones, ya que no eran lo suficientemente resistentes a las fuerzas laterales aplicadas.

El objetivo principal de estructurar la superficie del *cover* es doble. Por un lado, se consigue concentrar las fuerzas sobre las áreas correspondientes a los electrodos de la matriz táctil, mejorando así la respuesta del sensor. Por otro lado, la estructuración de la superficie del *cover* permite detectar vibraciones, y por tanto, amplificar los estímulos producidos por fuerzas tangenciales. El diseño presentado en la figura 2.3 cumple con ambos requisitos.

Funcionamiento de la electrónica del sensor del dedo

La figura 2.5 muestra una imagen de la electrónica local al dedo, que incluye una PCB basada en una FPGA de la familia Spartan-6® de Xilinx™ y una matriz de electrodos que se utiliza para el sensado.

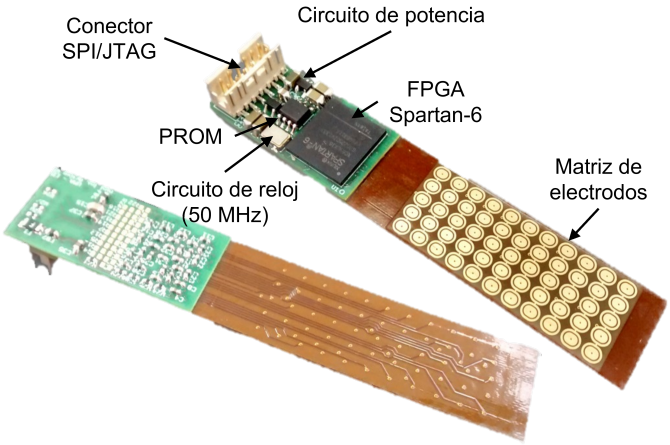


Figura 2.5. Detalle de la electrónica local al dedo propuesta en [37].

Específicamente, el circuito equivalente a la electrónica interna se muestra en la figura 2.6. Se trata de una propuesta de conexión directa entre el sensor y la FPGA [36]. La matriz de tácteles tiene 5 filas y 11 columnas. Todos los tácteles de cada columna comparten un electrodo común. La FPGA incluye una serie de pines para las filas $P_{1,c}$ a $P_{5,c}$

UNIVERSIDAD DE MÁLAGA

(donde $c = \{1, 2, 3, ...11\}$) y para las columnas PV_1 a PV_{11}. Para las filas, estos pines se pueden configurar como salida (valor lógico '0') o como entrada (alta impedancia 'HZ'). Para las columnas, se pueden configurar como salida (valores lógicos '0' o '1') o como entrada (alta impedancia 'HZ'). En esta tesis además, se ha trabajado con dos filas adicionales de resistencias de calibración P_{c1} y P_{c2}.

Concretamente, para una fila y columna de la matriz en la arquitectura de la figura 2.6, el dispositivo FPGA excita un circuito RC para obtener una señal que se puede medir mediante un temporizador o *timer* digital. El principio de funcionamiento para este circuito consta de dos fases, de carga y descarga del condensador C. De forma progresiva, se selecciona cada fila, colocando el valor lógico '0' en todos los pines correspondientes a la fila (f) $P_{f,c}$, y se leen en paralelo todas las columnas para dicha fila, de PV_1 a PV_{11}. Si se aplica una presión sobre algún táctel mientras se selecciona alguna de las filas de la matriz de electrodos, entonces aparece una resistencia R_x y el condensador se descarga por dicha resistencia. Midiendo el tiempo de descarga del condensador hasta alcanzar un umbral dado por V_{TL}, se puede obtener una curva cuya medida, en número de ciclos de reloj, es aproximadamente proporcional a la fuerza aplicada (αF). Los valores nominales de tensiones dados por Xilinx® son $V_o = 0$ V, $V_{TL} = 1.3$ V y $V_{cc} = 3.3$ V. El tiempo de descarga del condensador T_x, responde a la ecuación (2.1).

$$T_x = \tau \cdot Ln \left[\frac{V_{cc} - V_o}{V_{TL} - V_o} \right] \tag{2.1}$$

donde la constante de tiempo es $\tau = R_x \cdot C$.

Linealización de la respuesta del sensor

El material piezoresistivo utilizado, en contacto con los electrodos, disminuye su resistencia cuando se aplica una presión determinada [61]. Algunos materiales piezoresistivos presentan resistencias elevadas ante la aplicación de estímulos de fuerza pequeños. Esto tiene un efecto en la respuesta del condensador, ya que la descarga de éste puede ser más lenta si se miden resistencias más grandes, tardándose un tiempo muy largo en alcanzar el valor umbral V_{TL}, lo que repercutiría en la frecuencia de muestreo del sistema.

Para solucionar esto, es necesario incluir un *timeout*, que indique el número máximo de ciclos de descarga del condensador que se pueden medir. El valor del *timeout* siempre se espera antes de iniciar un nuevo ciclo de carga/descarga del condensador. Por tanto, está ligado a la frecuencia de muestreo del sistema. En [36] se comprobó que con un *timeout* de 14 bits (16384 ciclos de descarga del condensador) era suficiente.

UNIVERSIDAD DE MÁLAGA

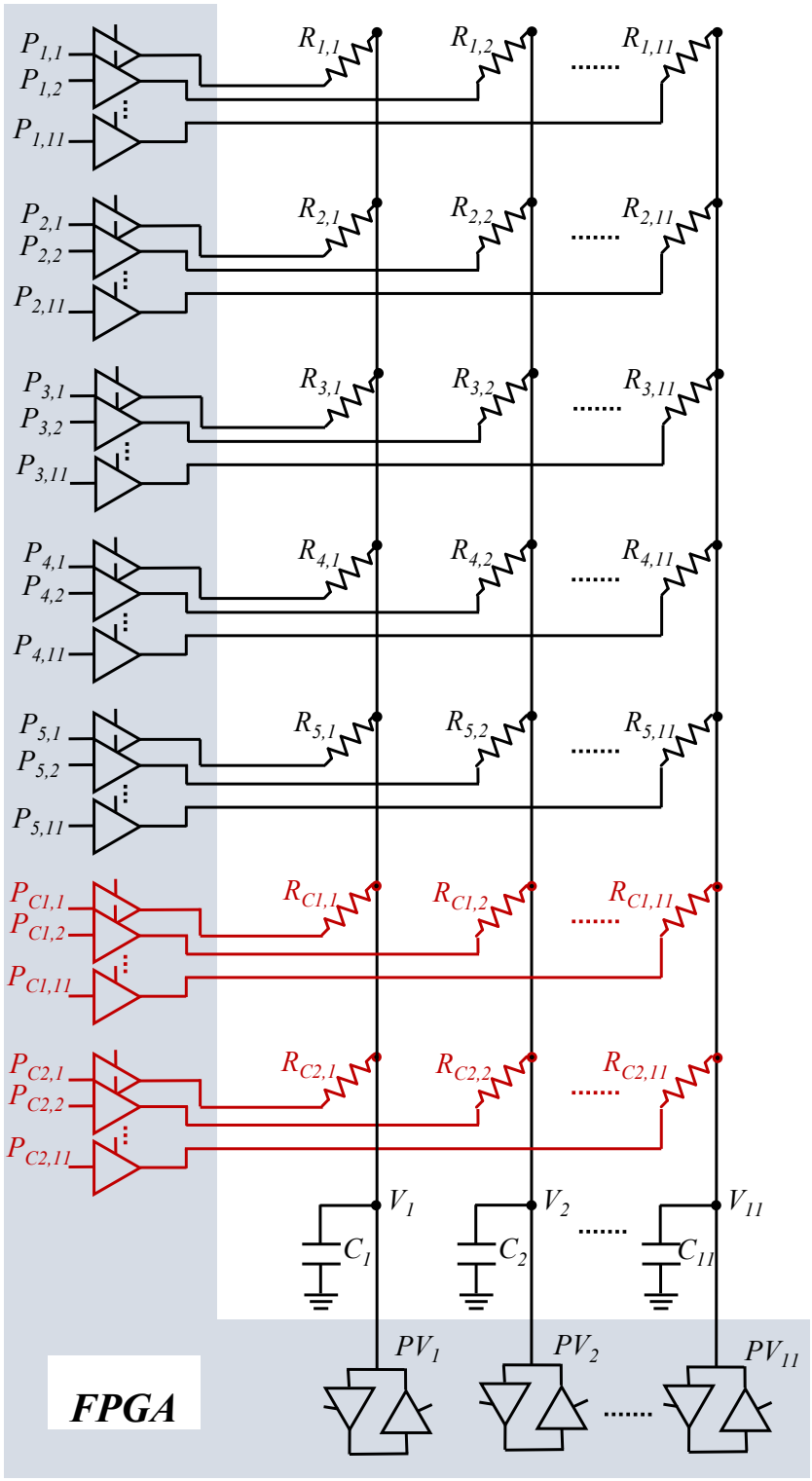


Figura 2.6. Propuesta de conexión directa entre sensor y FPGA tomada de la tesis [36]. La matriz del sensor tiene 5 filas y 11 columnas. Se incluyen dos filas de resistencias de calibración.

También se puede aumentar la sensibilidad del sensor frente a pequeños estímulos de fuerzas normales y tangenciales, aunque con el coste de perderla en rangos superiores de la fuerza. Esto es de interés para el desarrollo de los experimentos en esta tesis. El

UNIVERSIDAD DE MÁLAGA

valor de la resistencia a medir R_x se puede deducir a partir de la expresión dada en (2.1), siempre que se conozcan los valores de los parámetros V_{cc}, V_o, V_{TL} y C. Sin embargo, este método no es muy preciso debido a la variabilidad de estos valores con el tiempo y la temperatura. Para evitar este problema, se incluyen las dos filas de calibración de la figura 2.6. Estas filas tienen el doble propósito de linealizar la respuesta del sensor en un rango de fuerzas pequeño, y de estimar el valor de R_x a través de la técnica de calibración en 1 ó 2 puntos [36]. En la figura 2.7 se muestra cómo cambia la descarga del condensador al incluir resistencias de calibración. Al descargar por el paralelo de R_x y el equivalente de una o dos resistencia en paralelo R_c, la resistencia equivalente $R'_x = \frac{R_x \cdot R_c}{R_x + R_c}$ es más pequeña que R_x, y por tanto, se alcanzará V_{TL} antes.

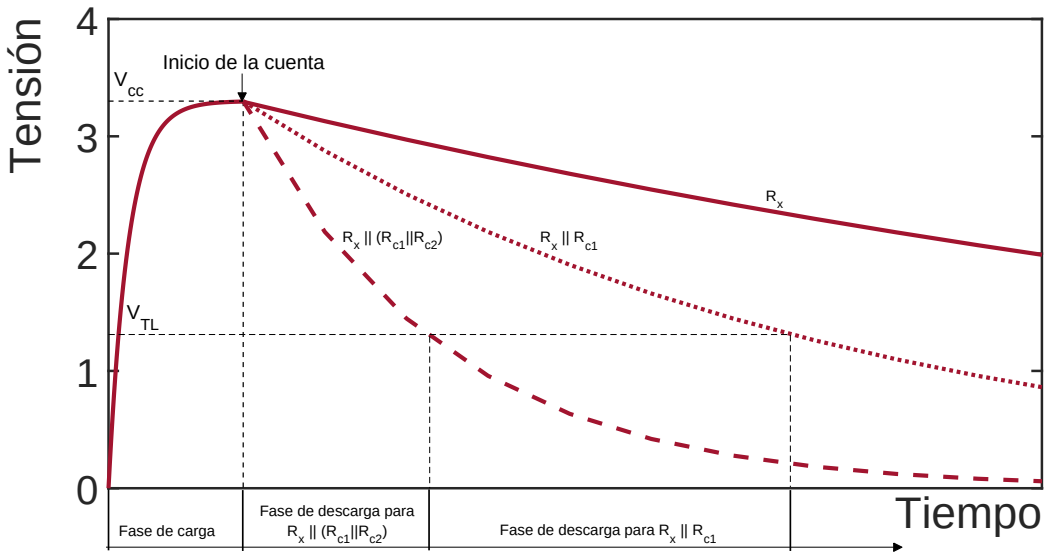


Figura 2.7. Forma de la señal para la carga/descarga del condensador con o sin resistencias en paralelo.

La figura 2.8 muestra el circuito equivalente para la carga y descarga del condensador cuando se utilizan resistencias de calibración R_c en paralelo con R_x. En este esquema, la FPGA consta de dos puertos, P.D. (puerto de descarga) y P.M. (puerto de medida).

Inicialmente, cuando no se selecciona fila ni columna, el condensador se carga con la tensión de alimentación V_{cc}, a través de la resistencia de salida del puerto de medida r_M. En la figura 2.8(a) se muestra la fase de carga del condensador. El puerto de medida está al valor lógico '1', es decir, a V_{cc}, y el puerto de descarga está en alta impedancia, 'HZ', es decir, abierto o sin conexión.

Al seleccionar una fila y columna, el condensador se descarga a través del paralelo de la resistencia de medida y la de calibración $R_x||R_c$. La figura 2.8(b) muestra este escenario junto a su circuito equivalente. El puerto de descarga ahora está configurado como salida al valor lógico '0'. De esta forma, se establece un camino a tierra, de forma

UNIVERSIDAD DE MÁLAGA

que el *buffer* de salida drena la corriente por el equivalente del paralelo entre R_x y R_c. El puerto de medida (P.M.) está configurado como entrada (alta impedancia 'HZ').

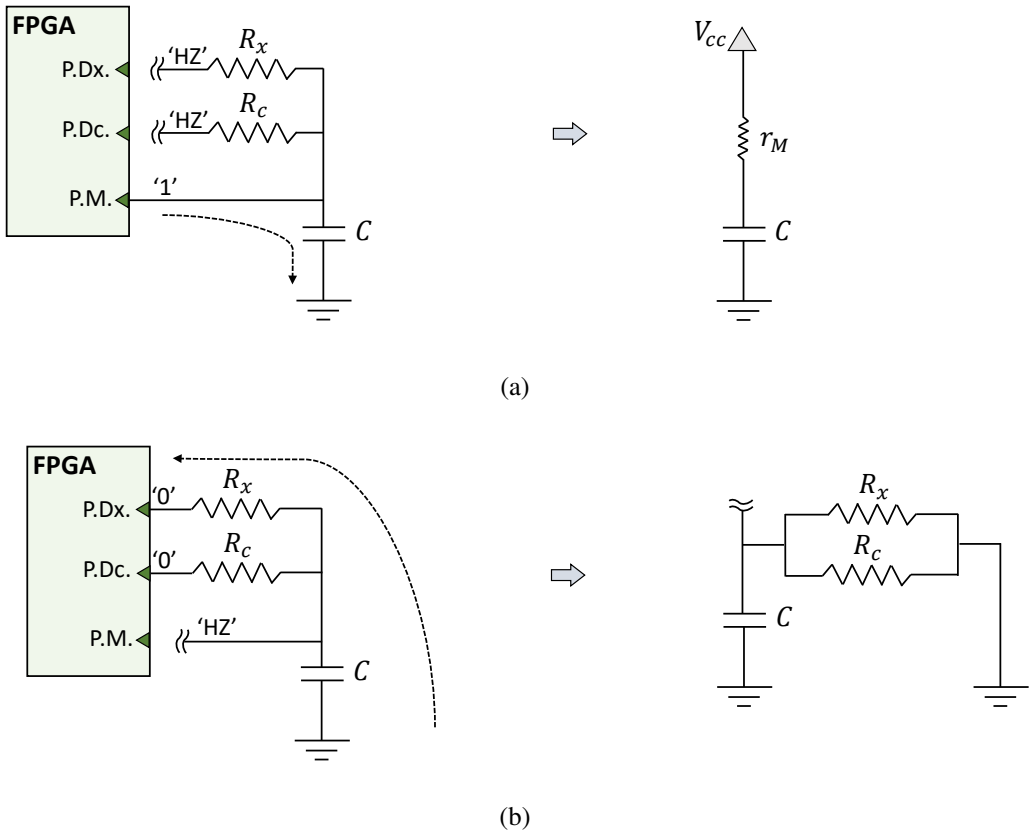
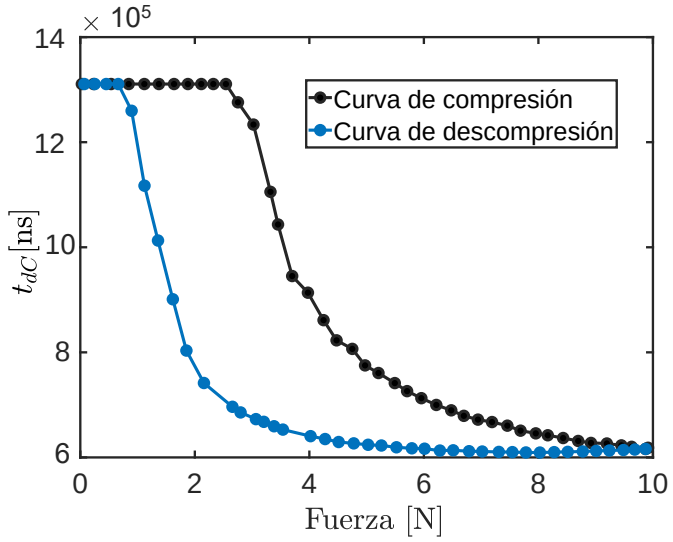


Figura 2.8. (a) Carga y, (b) descarga del condensador utilizando resistencias de calibración R_c en paralelo con R_x.

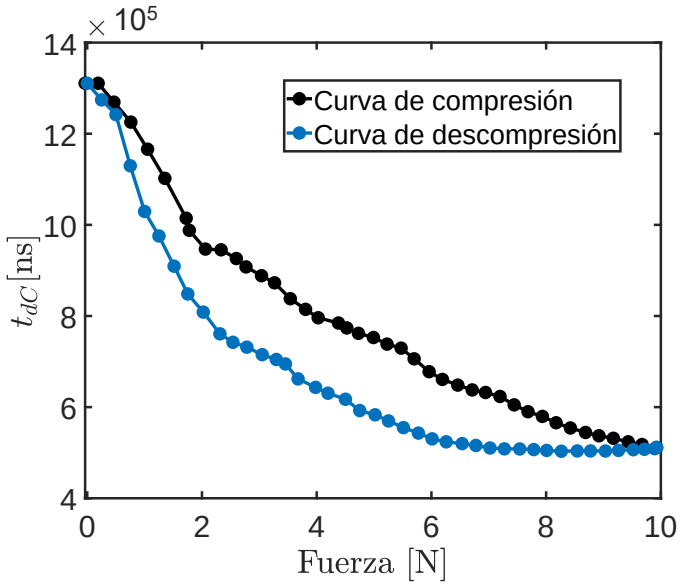
En esta tesis se utiliza Linqstat® MVCF-40012BT50KS/2A como capa de material piezoresistivo sensible. Este material presenta una resistencia de $50000\ \Omega$/sq y un ancho de 0.1 mm [62]. El valor del condensador en la figura 2.6 es de $C = 47$ nF, y los valores comerciales seleccionados para las resistencias de las filas de calibración son $R_{c1} = 6.8\ k\Omega$ y $R_{c2} = 4.7\ k\Omega$. En concreto, para los experimentos realizados, se hace el paralelo con las resistencias correspondientes a la primera fila de calibración ($R_{c1} = 6.8\ k\Omega$). En la figura 2.9, se muestran las curvas de caracterización obtenidas para un táctel, cuando se utiliza un elástomero de Filaflex® continuo y sin resistencias de calibración, y cuando se utiliza el *cover* estructurado y la primera fila de calibración en paralelo, respectivamente.

Como se ve en la figura 2.9, cuando se utiliza un elastómero continuo sin resistencias de linealización, la respuesta del sensor presenta una elevada resistencia frente a bajos valores de la fuerza, por lo que la FPGA no puede leer dichos valores. Este problema, se soluciona colocando la resistencia de linealización y utilizando un *cover* estructurado.

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura 2.9. (a) Curva obtenida sin resistencia de linealización para un elástomero continuo hecho de Filaflex®. (b) Curva obtenida con una resistencia de linealización de 6.8 $k\Omega$ para el *cover* estructurado. El tiempo de descarga del condensador t_{dC} está expresado en ns, teniendo en cuenta que la frecuencia de reloj en la FPGA Spartan-6® es de 50 MHz (20 ns).

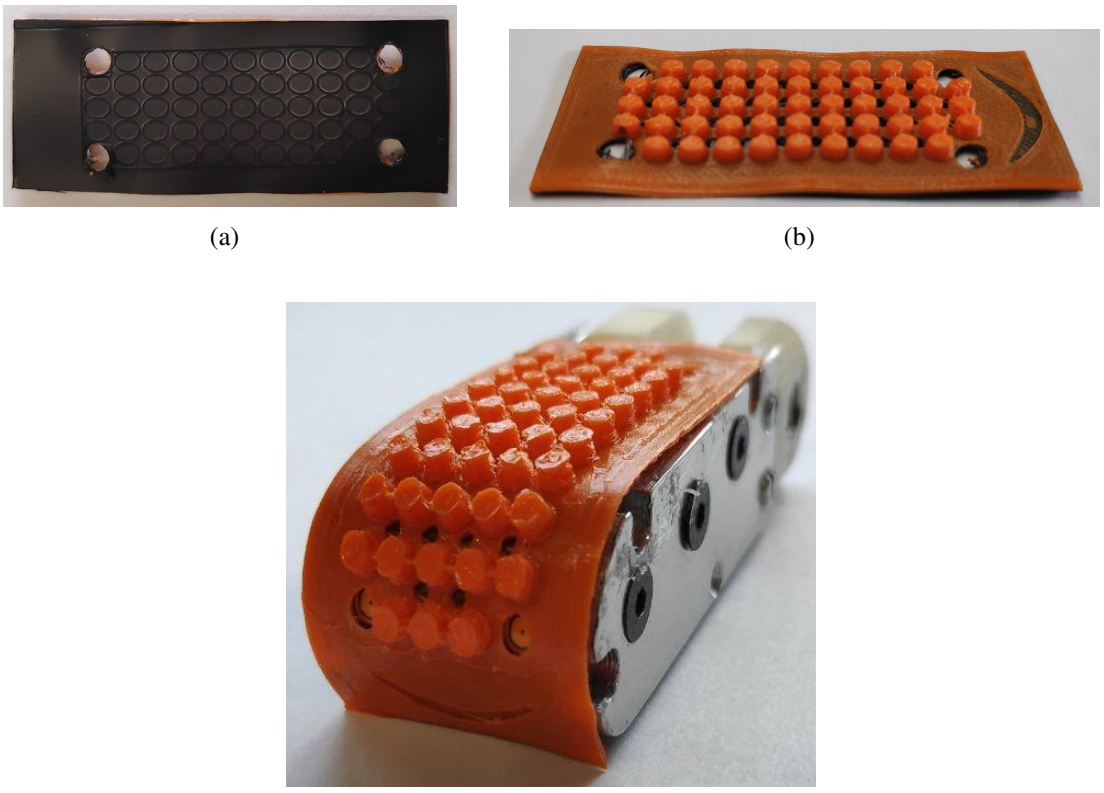
De esta forma, conseguimos concentrar la fuerza en las áreas de los electrodos y linealizar la respuesta del sensor en un rango pequeño de fuerzas. Los resultados de la figura 2.9 se han obtenido a través de una aplicación en el software LabVIEW® de National Instruments™, que guarda estos datos en ficheros de texto y un calibrador triaxial (en el apéndice A de figuras se puede ver una imagen del calibrador), realizando barridos de 0 N a 10 N, con incrementos de 0.25 N, y un tiempo de espera entre medidas de 3 s.

UNIVERSIDAD DE MÁLAGA

Capa de material piezoresistivo discretizada

El diseño del sensor presenta un punto clave que se encuentra en el grabado de la capa de material piezoresistivo. Debido a la continuidad de la lámina, surgen errores de diafonía o lecturas fantasma entre tácteles. Este fenómeno, conocido como *crosstalk* [37], puede evitarse mediante el aislamiento eléctrico de los tácteles en la matriz del sensor a través de la discretización de la lámina de material piezoresistivo colocada encima de los electrodos.

Para el proceso de grabado y aislamiento por tácteles de este material, se emplea una máquina de corte por láser de CO_2 de 40 W de potencia, que requiere una calibración inicial y un ajuste fino para establecer todos los parámetros de configuración del láser. En el apéndice D, se detallan los pasos a seguir para realizar este procedimiento. En la figura 2.10, se presenta un detalle con el diseño de la capa de material piezoresistivo discretizada, el *cover* estructurado y el dedo completo. Los electrodos tienen forma elíptica para cubrir toda el área posible, dada las restricciones geométricas del diseño.



(a) (b)

(c)

Figura 2.10. (a) Imagen de la capa sensible de piezoresistivo discretizada mediante impresora láser. (b) *Cover* estructurado. (c) Montaje completo del dedo.

UNIVERSIDAD DE MÁLAGA

Pruebas de *crosstalk* (o diafonía)

Como se ha comentado en el apartado 2.3.1, la disposición electrónica del dedo puede generar *crosstalk*. Para reducir este fenómeno, es necesario aislar eléctricamente los electrodos en la lámina sensible piezoresistiva y utilizar un *cover* estructurado. Además, esta estructuración permite aumentar la sensibilidad ante la aplicación de fuerzas normales en la superficie del sensor. Para comprobar este efecto se realizan algunas pruebas ante estímulos estáticos con la lámina piezoresistiva discretizada, aplicando fuerza progresivamente, de forma incremental primero y después decremental, con el calibrador triaxial.

En la figura 2.11, se muestran algunos pasos de esta secuencia sobre un táctel individual para un *cover* no estructurado, hecho de una resina de silicona, caracterizado por un módulo de Young de 323.52 MPa y un ancho de 0.72 mm. Por otra parte, en la figura 2.12, se muestra la secuencia para el *cover* estructurado de la figura 2.3. Como se puede observar, en este último caso, al estructurar la superficie, la respuesta del sensor es mejor que en el caso sin estructurar.

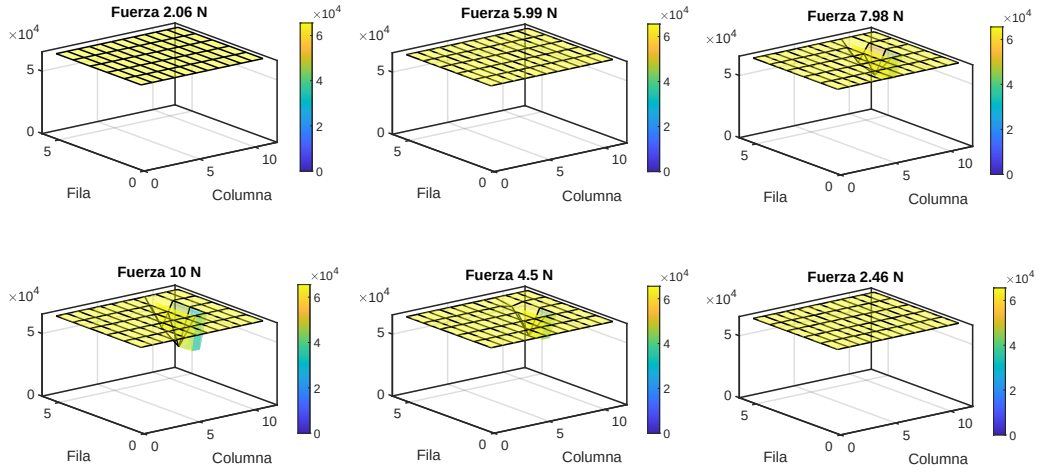


Figura 2.11. Secuencia de aplicación progresiva de fuerza para un *cover* continuo no estructurado, en algunos instantes de tiempo. En este test el rango de aplicación va de 0 N a 10 N. La primera fila se corresponde con la aplicación de una fuerza creciente, y la segunda con una fuerza decreciente.

En la figura 2.13 se muestran los resultados de la respuesta de los tácteles del sensor cuando se aplica una fuerza en una parte de la matriz usando el calibrador triaxial. En concreto, la figura 2.13(a) presenta la salida del sensor con el *cover* no estructurado, mientras que la figura 2.13(b) muestra la respuesta obtenida con el *cover* estructurado. Como se puede apreciar, en el segundo caso el efecto del *crosstalk* es imperceptible, lo que indica que la estructuración de la superficie del *cover* ha sido efectiva para reducir este fenómeno.

UNIVERSIDAD DE MÁLAGA

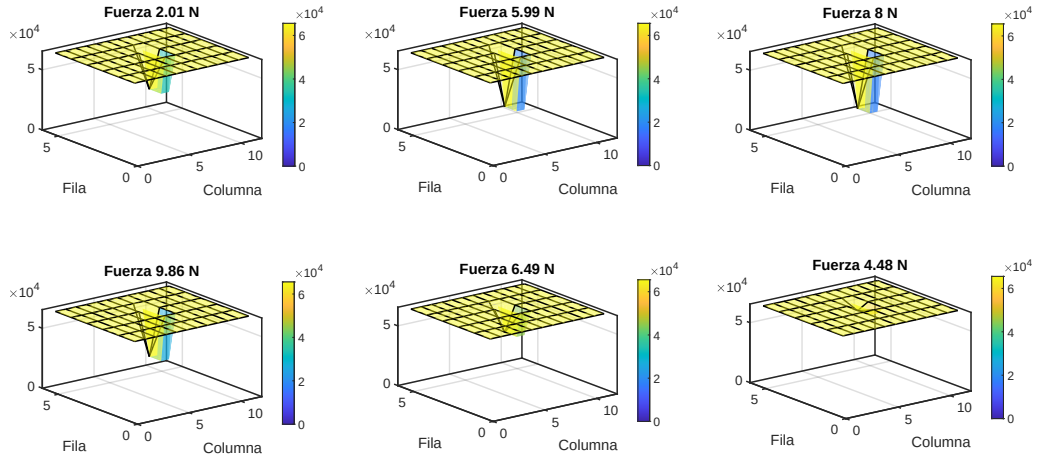
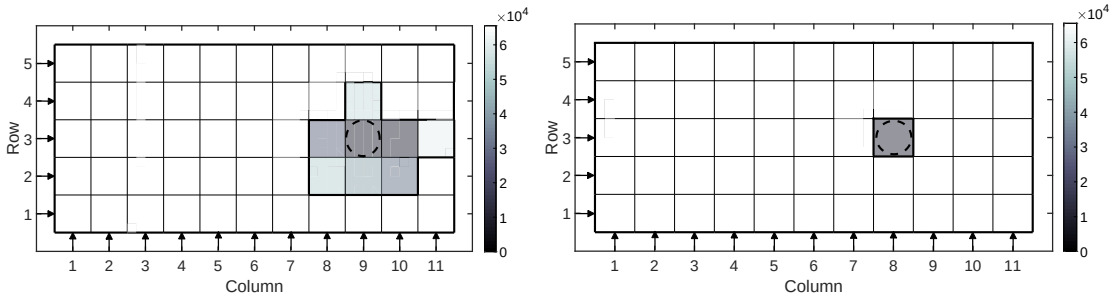


Figura 2.12. Secuencia de aplicación progresiva de fuerza para un *cover* estructurado, en algunos instantes de tiempo. En este test el rango de aplicación va de 0 N a 10 N. La primera fila se corresponde con la aplicación de una fuerza creciente, y la segunda con una fuerza decreciente.



(a) (b)

Figura 2.13. Salida de la matriz del sensor (a) con el *cover* continuo aplicando una fuerza con el calibrador triaxial, sobre un táctel, de 27.94 N, y (b) con el *cover* estructurado con una fuerza de 27.87 N.

Adicionalmente, se llevaron a cabo otros dos experimentos sencillos de *crosstalk*, utilizando un *cover* estructurado y discretizando o no la lámina sensible piezoresistiva. En ambos experimentos se presionaron dos zonas de la matriz táctil con los extremos respectivos de una pieza delgada con forma de puente, tal como se muestra en la figura 2.14.

Por un lado, en la figura 2.15 se presentan los resultados obtenidos al aplicar una secuencia progresiva de fuerza sobre la lámina piezoresistiva continua. Por otro lado, en la figura 2.16 se muestra la misma secuencia para la lámina piezoresistiva discretizada, obtenida mediante la máquina de corte por láser. Finalmente, en la figura 2.17 se pueden ver los resultados cuando se aplica una fuerza normal estática, sobre el puente, de 23.75 N con el calibrador triaxial. Se puede ver que el *crosstalk* se reduce siempre que se utiliza el *cover* estructurado.

UNIVERSIDAD DE MÁLAGA

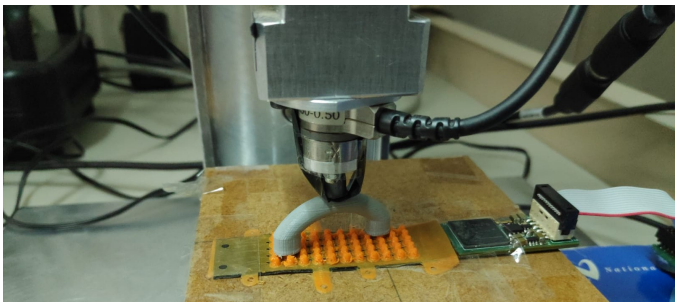


Figura 2.14. Aplicación de fuerzas sobre pieza en forma de puente con el calibrador triaxial.

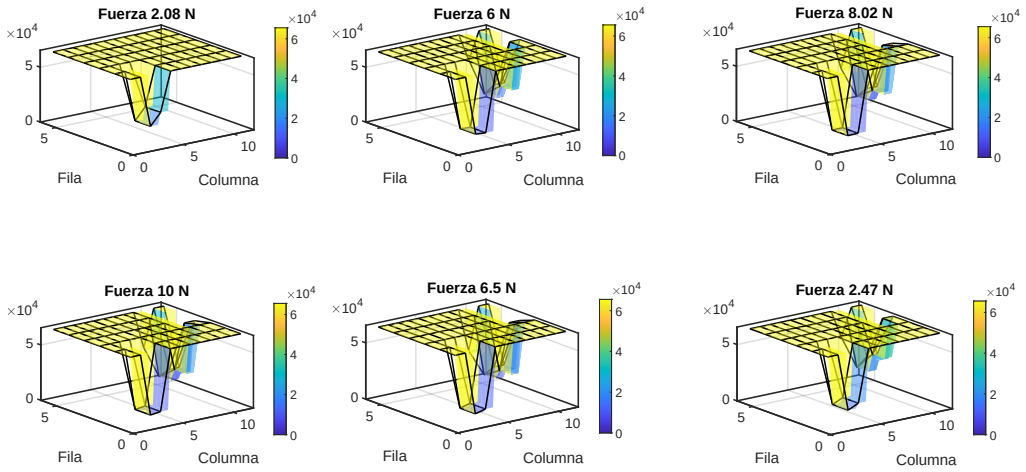


Figura 2.15. Secuencia de aplicación de fuerza progresiva de 0 N a 10 N en la pieza puente de la figura 2.14. Se utiliza una lámina piezoresistiva continua. La primera fila muestra fuerza creciente y la segunda, fuerza decreciente.

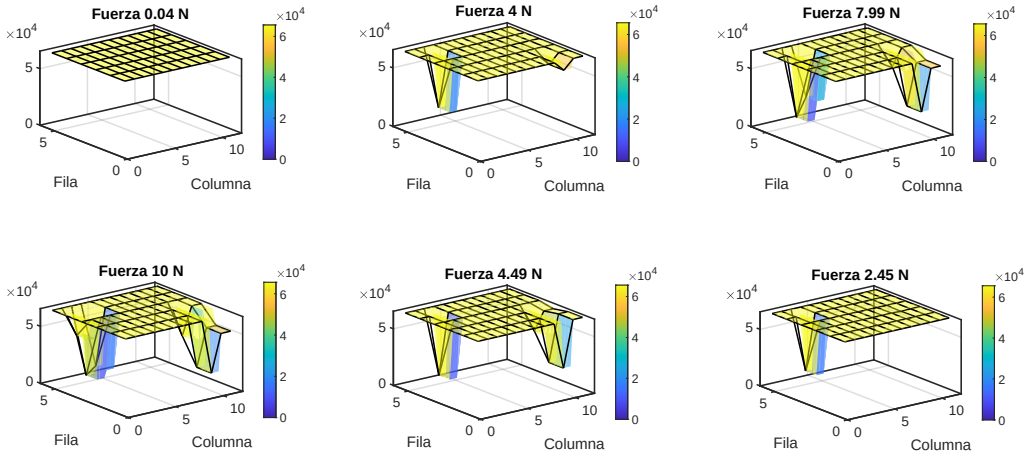


Figura 2.16. Secuencia de aplicación de fuerza progresiva de 0 N a 10 N en la pieza puente de la figura 2.14. Se emplea una lámina piezoresistiva discretizada por láser. La primera fila muestra fuerza creciente y la segunda, fuerza decreciente.

UNIVERSIDAD DE MÁLAGA

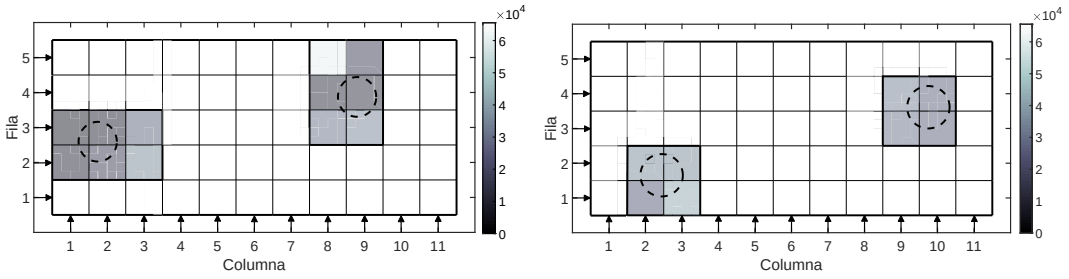


Figura 2.17. Salida de la matriz del sensor al realizar un experimento con la pieza-puente de la figura 2.14; (a) con la capa de piezoresistivo continuo aplicando una fuerza con el calibrador triaxial de 23.75 N, y (b) con el piezoresistivo discretizado por láser con una fuerza de 23.66 N.

A continuación se repiten los mismos experimentos de *crosstalk* para la pieza en forma de puente, pero cambiando ésta, por una pieza delgada en forma de prisma, ver figura 2.18. Al igual que con la pieza puente, el efecto de diafonía se reduce cuando se pasa de un *cover* continuo, figura 2.19, a un *cover* estructurado, figura 2.20. Por último, en la figura 2.21 se pueden ver los resultados cuando se aplica una fuerza normal estática sobre el prisma, de 28.54 N con el calibrador triaxial.

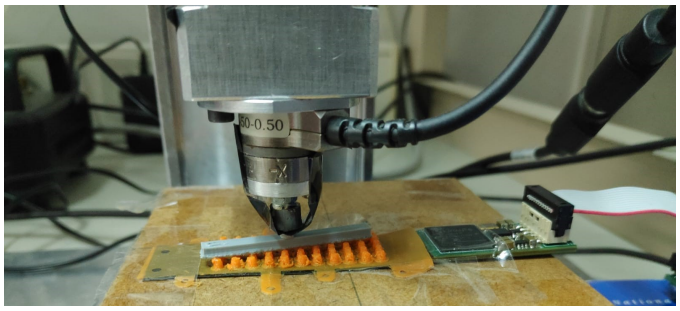


Figura 2.18. Aplicación de fuerzas sobre pieza en forma de prisma con el calibrador triaxial.

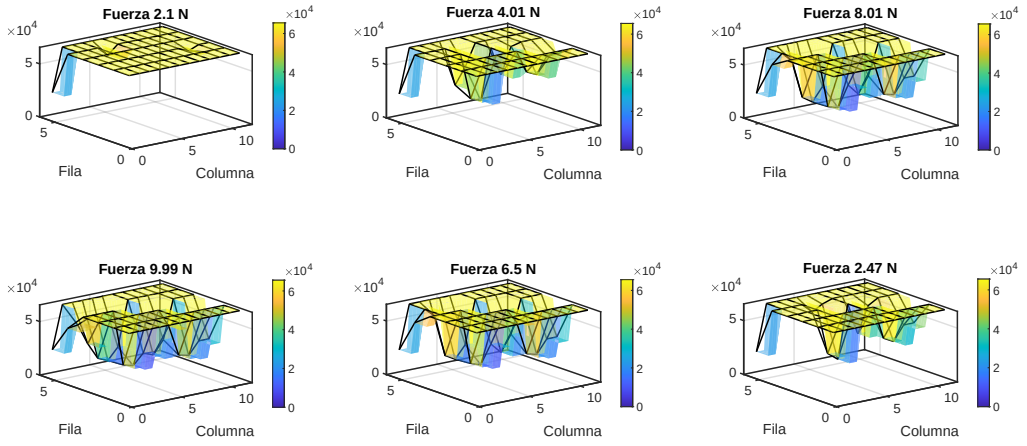


Figura 2.19. Secuencia de aplicación de fuerza progresiva de 0 N a 10 N en la pieza en forma de prisma de la figura 2.18. Se emplea una lámina piezoresistiva continua. La primera fila muestra fuerza creciente y la segunda, fuerza decreciente.

UNIVERSIDAD DE MÁLAGA

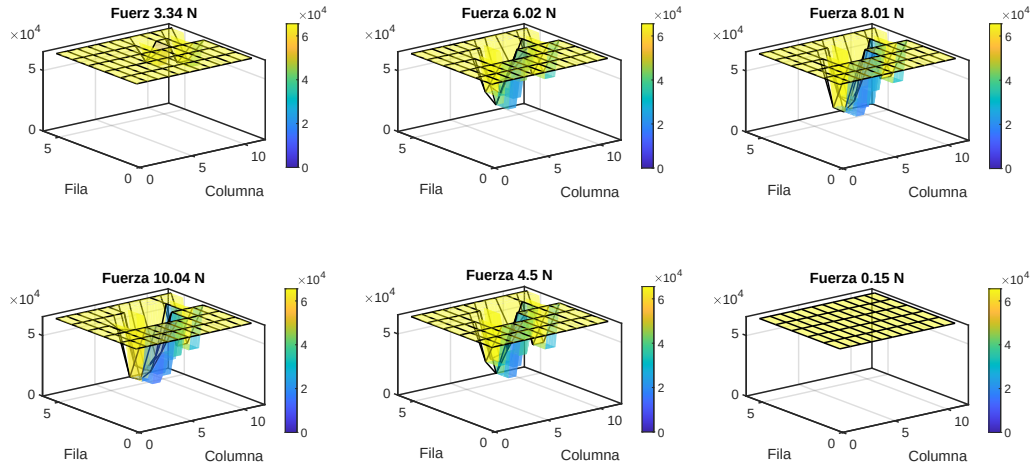
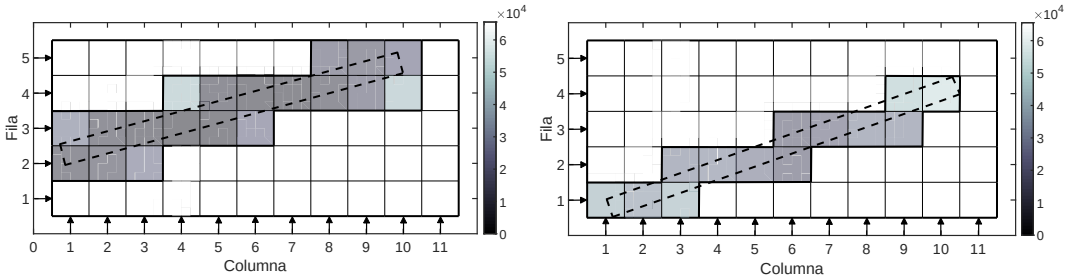


Figura 2.20. Secuencia de aplicación de fuerza progresiva de 0 N a 10 N en la pieza en forma de prisma de la figura 2.18. Se emplea una lámina piezoresistiva discretizada por láser. La primera fila muestra fuerza creciente y la segunda, fuerza decreciente.



(a) (b)

Figura 2.21. Salida de la matriz del sensor al realizar un experimento con la pieza-prisma de la figura 2.18; (a) con la capa de piezoresistivo continuo aplicando una fuerza con el calibrador triaxial de 28.54 N, y (b) con el piezoresistivo discretizado por láser con una fuerza de 28.55 N.

Sensibilidad a estímulos laterales

Aparte de las fuerzas normales a la superficie de sensado, en esta tesis es importante que la respuesta del sensor a estímulos tangenciales sea buena. Utilizando el *setup* experimental preliminar en la figura 2.22, se realiza una exploración con el dedo artificial completo sobre una tira de velcro.

El experimento consiste en definir una trayectoria lineal de 8.7 cm de longitud para el dedo, sobre el material a explorar, a dos velocidades diferentes de 3 mm/s y 6.2 mm/s. La lectura de la balanza en la figura 2.22 se encontraba en el rango de $[901 - 1140]$ g y $[970 - 1200]$ g para cada test, respectivamente. Con la lámina piezoresistiva discretizada, el espectro de frecuencias obtenido para el *cover* continuo y estructurado se muestra en la figura 2.23. La amplitud de las frecuencias del espectro en la figura 2.23 es mucho mayor para el caso del *cover* estructurado, demostrando así que es una buena opción para la exploración de texturas y objetos.

UNIVERSIDAD DE MÁLAGA

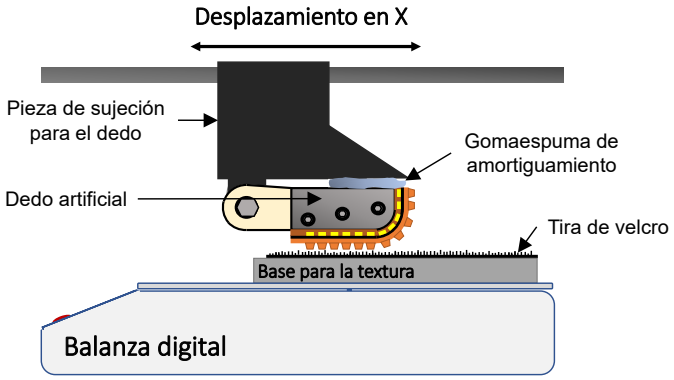


Figura 2.22. *Setup* preliminar para un test de estímulos laterales.

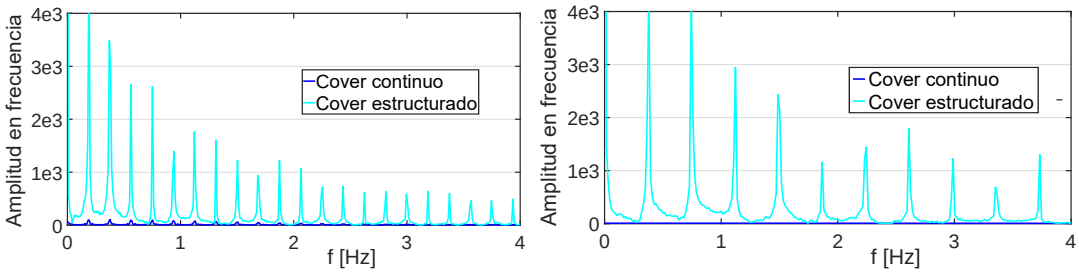


Figura 2.23. Comparación entre la magnitud del espectro de la salida de un táctel del sensor para el *cover* estructurado y el *cover* continuo, cuando un estímulo lateral se aplica, con dos velocidades de desplazamiento diferentes: (a) 3 mm/s y (b) 6.2 mm/s.

2.3.2. Sensor de palma artificial

El diseño de la palma no necesita un diseño mecánico complejo. Además, la matriz táctil de la palma no requiere de aislamiento láser entre tácteles en la capa piezoresistiva porque la electrónica se puede diseñar para reducir la diafonía, al haber más espacio. Por lo tanto, se utiliza una superficie de goma continua como *cover*, adherida a una capa de material piezoresistivo continuo del mismo tipo que el utilizado en el dedo. La figura 2.24 muestra el sensor completo para la palma.

El circuito correspondiente a la electrónica de la palma se hace a través de un único cable para el direccionamiento por filas, y un amplificador operacional por cada columna, para así generar una tierra virtual. De esta forma, se evita el *crosstalk* entre tácteles de una fila. La electrónica de la palma se encuentra en la PCB que se puede ver en la parte superior izquierda de la figura 2.24, está diseñada y fabricada de acuerdo a [37]. Esta placa también está basada en una FPGA de Xilinx® de la familia Spartan-6® y también contiene una electrónica de adquisición y comunicación para tres dedos, como el descrito en la subsección 2.3.1, que se pueden incorporar a una mano de Barrett [38]. Este dispositivo integra tanto la electrónica correspondiente al sensor de la palma como el

UNIVERSIDAD DE MÁLAGA

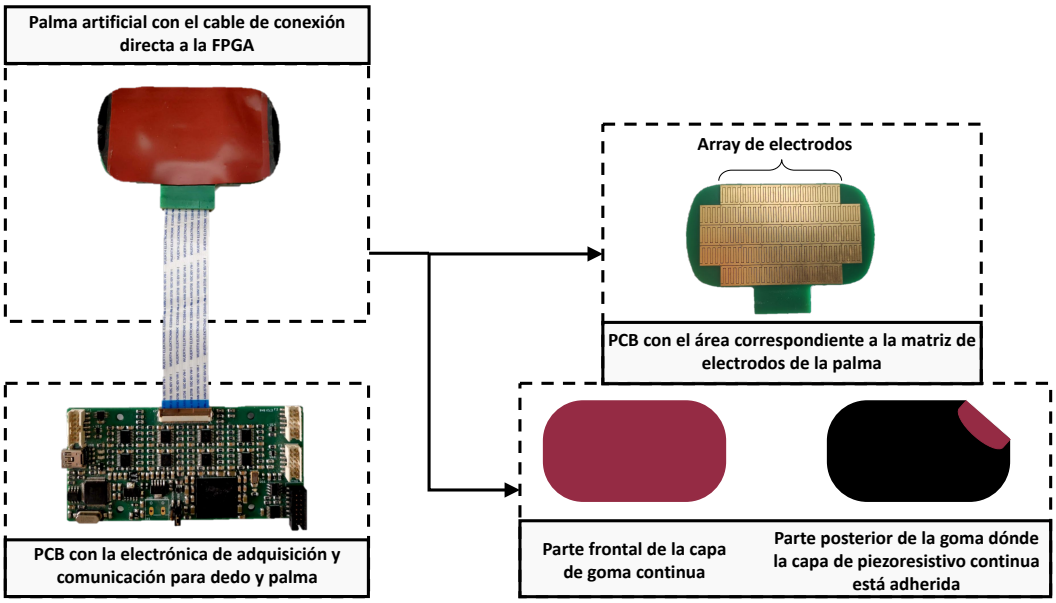


Figura 2.24. Palma artificial consistente en una PCB semirígida, con un elastómero continuo consistente en una goma, con módulo de Young $0.68 \cdot 10^6$ [61]. Las figuras no están a escala.

controlador del sistema táctil completo. La electrónica de adquisición se conecta a través de un cable plano con la matriz de electrodos de la palma. Para este trabajo, los datos proporcionados tanto por el dedo como por la palma se envían a través de un puerto USB 2.0 al *host* u ordenador, dónde, a través de una aplicación diseñada con LabVIEW®, se toman estos datos y se guardan en ficheros de texto.

2.4. Desarrollo del *setup* experimental

Para realizar los experimentos de exploración de texturas y objetos, se ha adaptado una impresora 3D Witbox® 2 como robot cartesiano. Esta opción presenta una ventaja en cuanto a coste, además de permitir programar trayectorias con distintos tipos de movimiento dentro del espacio de trabajo de la impresora, lo que resulta útil para la exploración de texturas y objetos. La impresora Witbox® 2 cuenta con un área de impresión de $(297 \times 210 \times 210)$ mm, lo que es más que suficiente para los propósitos de este proyecto, y los motores pueden alcanzar una velocidad lineal de hasta 200 mm/s, lo que también es adecuado para las condiciones de exploración planteadas. En los siguientes apartados se detallará la electrónica de control y las diferentes partes que intervienen en el *setup* tanto para la exploración de texturas como para el reconocimiento de objetos con diferente rigidez.

En la figura 2.25 se muestra un esquema general que incluye todos los elementos que forman parte del *setup* básico utilizado en este trabajo.

UNIVERSIDAD DE MÁLAGA

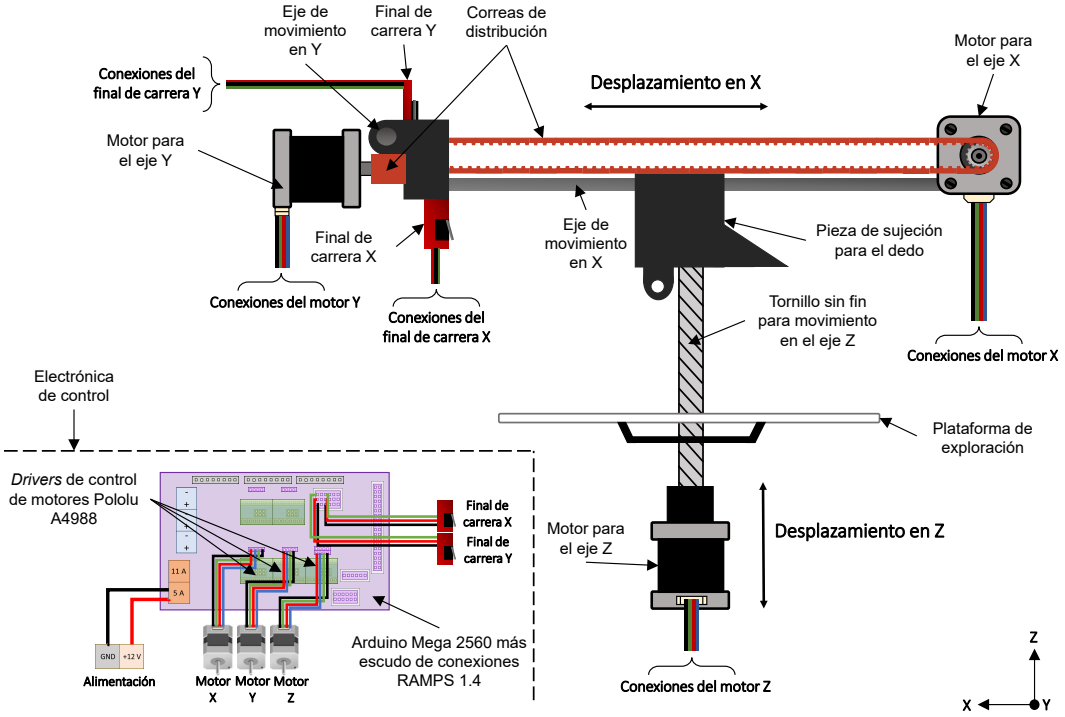


Figura 2.25. Disposición general del *setup* experimental adaptado de una impresora 3D.

2.4.1. Electrónica de control de los motores

La plataforma de control utilizada en los experimentos es la plataforma de hardware abierto Arduino®, la cual proporciona comodidad en el control de las trayectorias a seguir. En concreto, se ha utilizado la placa Arduino® Mega 2560 [63], un microcontrolador basado en la arquitectura ATmega1280 utilizado para diferentes aplicaciones gracias a su flexibilidad y ser de código abierto. Se elige este microcontrolador respecto a otros como el UNO, ya que dispone de más periféricos. Además, se ha montado una placa de ampliación RAMPS 1.4 [64] para facilitar y hacer más versátil el conexionado entre elementos. Esta placa de expansión permite conectar de manera sencilla los *drivers* Pololu A4988 [65] correspondientes a los motores NEMA 17 [66] que controlan los tres ejes de la impresora 3D. En la figura 2.26 se muestra el esquema de conexionado de todos los elementos involucrados en el control de los motores paso a paso.

Alimentación

La alimentación utilizada corresponde a la de la línea de 5 A y 12 V para no dañar la electrónica de la RAMPS 1.4 [64]. También es importante tener en cuenta que el diodo de protección D1 (1N4004) de la placa debe estar soldado siempre que se conecte a alimentación externa, ya que de lo contrario podría dañarse la electrónica. Además, se recomienda no utilizar tensiones superiores a 12 V.

UNIVERSIDAD DE MÁLAGA

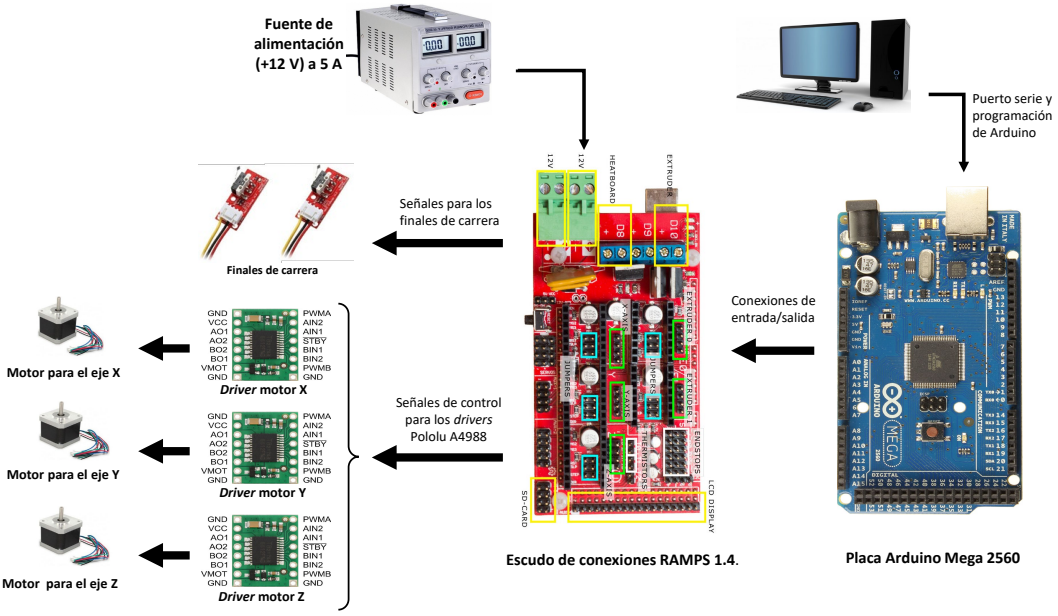


Figura 2.26. Esquema de conexión de todos los elementos a la electrónica de control para el *setup* de la figura 2.25.

Motores paso a paso NEMA 17

El modelo específico de motores paso a paso que se utilizan en la impresora Witbox® 2 es el NEMA 17-42BYGHW609 [66]. Es un motor muy robusto y ampliamente utilizado en aplicaciones de impresión 3D. Se trata de un motor bipolar [67] cuyas principales características se dan en la tabla 2.1. En el apéndice C se incluye información adicional acerca de las dimensiones, resolución en número de pasos por vuelta y frecuencia de resonancia de estos motores.

Tabla 2.1. Principales valores nominales del motor NEMA 17 utilizado [66].

Propiedad	Valor nominal
Ángulo de paso	1.8^o
Tensión nominal	3.4V
Intensidad nominal	1.7A
Resistencia de fase	2Ω
Inductancia de fase	3mH
Máximo torque	400mNm
Inercia del rotor	$54gcm^2$
Peso del motor	0.24kg

Estos motores soportan una corriente nominal máxima de 1.7 A. A través de *drivers* de protección se controla la cantidad de corriente necesaria que se envía a los motores.

UNIVERSIDAD DE MÁLAGA

Driver Pololu Shield - A4988

El uso del *driver* Pololu A4988 simplifica el control de los motores paso a paso al generar las señales necesarias para su funcionamiento y proporcionar protecciones de temperatura y corriente. Este dispositivo puede operar a una tensión máxima de 35 V y una intensidad máxima de 2 A. En el apéndice C se incluye información adicional acerca de la electrónica interna de este *driver* y de la técnica de *microstepping* utilizada.

Finales de carrera

En el *setup* de la figura 2.25, se han utilizado 2 finales de carrera, para los ejes X e Y, respectivamente. A través del uso de estos elementos se determina la posición y los límites de movimiento de los ejes. En el *setup* general de la figura 2.25, los finales de carrera X e Y detectan cuando las piezas de sujeción que se desplazan sobre los ejes de movimiento alcanzan los extremos de los ejes X e Y, respectivamente. Así, ayudan a garantizar una precisa ubicación durante las trayectorias, además de mantener la seguridad en los desplazamientos, evitando movimientos fuera de rango.

Estos finales de carrera tienen dos líneas de alimentación y tierra, respectivamente, y una tercera línea para la señal. Si está configurado como normalmente cerrado, al activarse el final de carrera, se obtendrá un valor de '0' lógico en la línea de la señal.

2.4.2. Software para el control de motores

Para realizar el control de los motores de la impresora se utiliza el entorno de desarrollo integrado (IDE) de Arduino® en conjunto con el IDE para el software de código abierto Processing® y una aplicación en LabVIEW® para guardar los datos en fichero. De forma más específica, a través del código en Arduino® se ejecutan los movimientos de las trayectorias necesarias para la exploración de texturas y palpación de objetos. Por otro lado, en Processing® se diseña una interfaz gráfica que permite enviar a través del puerto serie a Arduino®, los valores de velocidad, distancia a recorrer, sentido del movimiento, altura de la textura u objeto y número de iteraciones $\{v, d, s, h, N\}$.

Finalmente, cuando se están tomando datos del dedo y/o la palma a través de la PCB presentada en la figura 2.24, se sincronizan Arduino® y LabVIEW® a través de la tarjeta de adquisición 6009 de National Instruments® (ver figura A.1 del apéndice A de figuras). De esta forma, la aplicación en LabVIEW® escribe los datos en fichero cuando corresponde. En la figura 2.27 se presenta un diagrama de flujo en el que se puede ver la sincronización llevada a cabo entre Arduino®, Processing® y LabVIEW®. En este diagrama, n es la iteración actual.

UNIVERSIDAD DE MÁLAGA

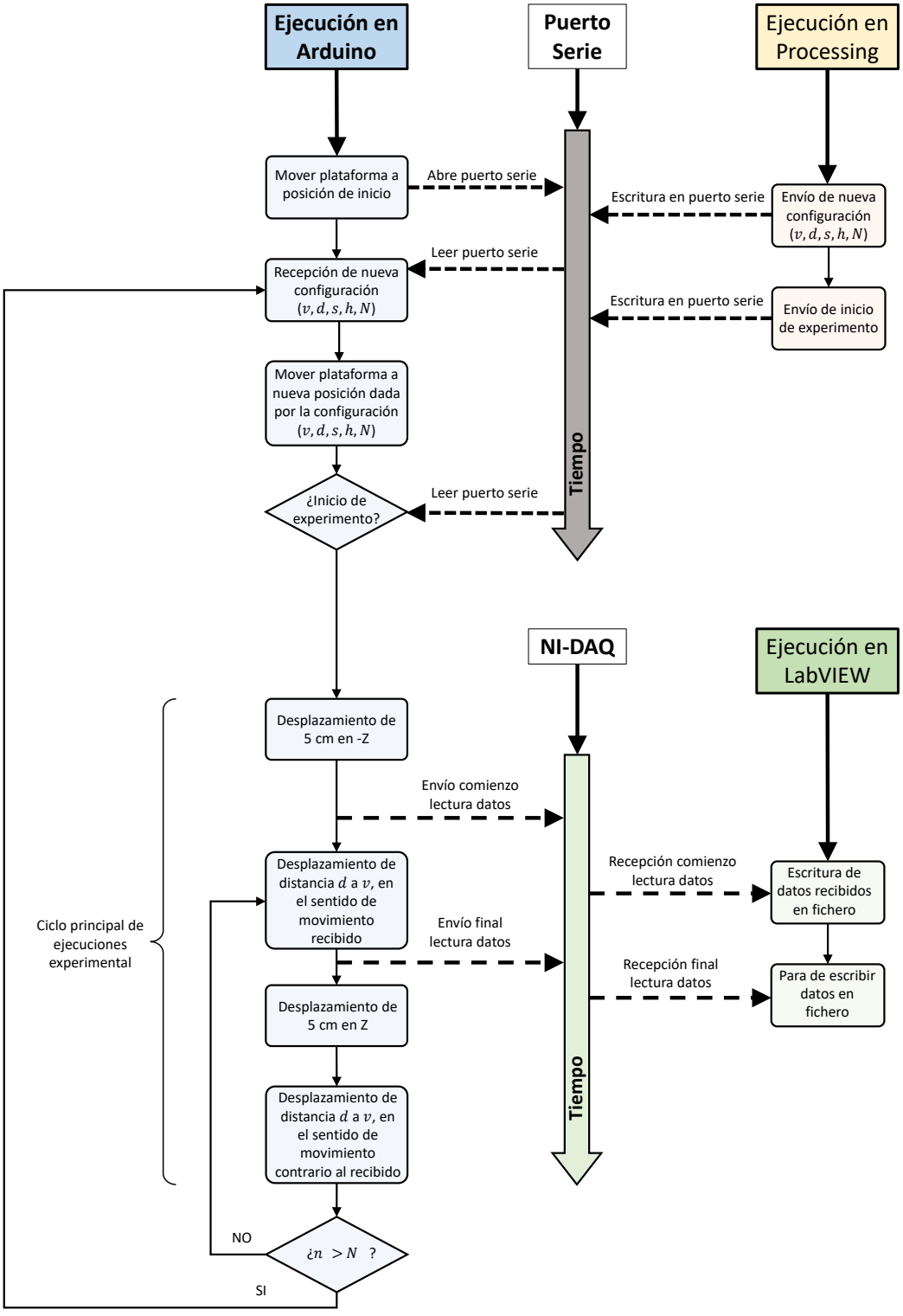


Figura 2.27. Diagrama de flujo de la ejecución en Arduino® y Processing® para el control de los motores del *setup*, y salvado de datos en fichero a través de LabVIEW®.

UNIVERSIDAD DE MÁLAGA

2.4.3. Modalidad para texturas

En la figura 2.28 se muestra el *setup* completo utilizado para llevar a cabo la exploración de texturas. Se parte de la disposición básica presentada en la figura 2.25, a la que se añade el sensor táctil del dedo artificial. El extremo del dedo dispone de una rosca, lo que permite atornillarlo a la pieza de sujeción (parte superior de la figura 2.28). En la figura, se puede observar que se ha incorporado un pequeño fragmento de gomaespuma entre el dedo y la pieza que lo sujeta. Esta estrategia ayuda a reducir las oscilaciones mecánicas que se producen durante la exploración de texturas.

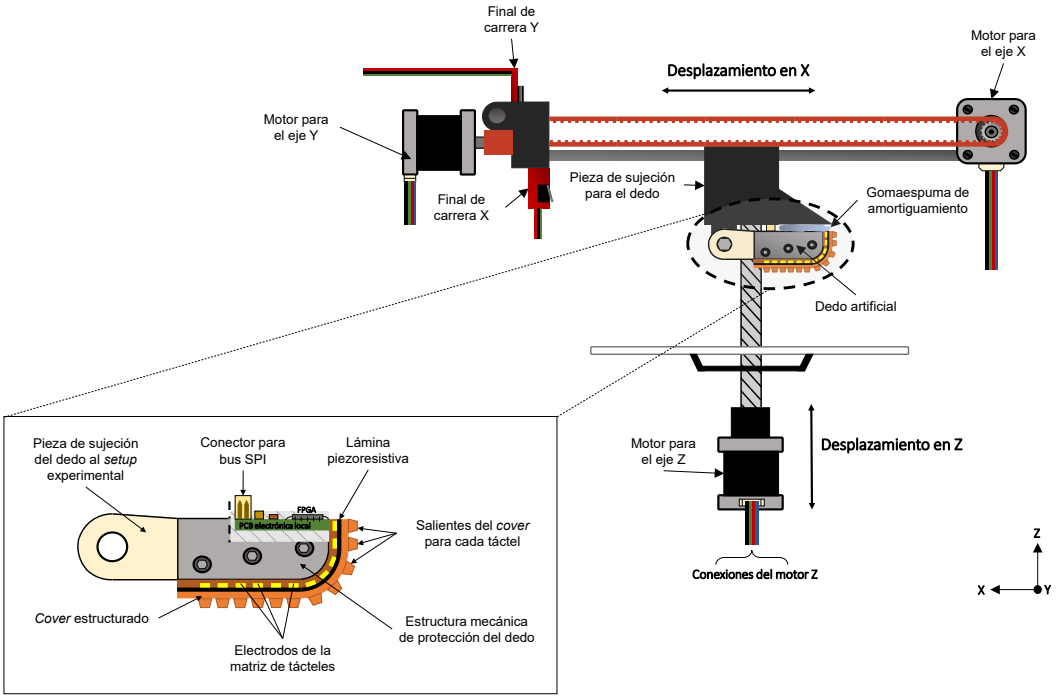


Figura 2.28. *Setup* para la exploración de texturas.

La figura 2.28 muestra un *zoom* en la parte del dedo artificial en el montaje. La electrónica local al dedo está basada en una FPGA de la familia Spartan-6® de Xilinx™. La placa de electrónica incluye un conector SPI, mediante el cual el dedo se puede conectar a la placa de adquisición y comunicación de la figura 2.1, o a la placa de desarrollo de AVNET® ZedBoard™ tal como se describe en el capítulo 3. Además, la electrónica local recibe información táctil a través de la matriz de electrodos con las capas de piezoresistivo y *cover*, detalladas en la subsección 2.3.1. El sensor está protegido mediante una estructura mecánica y una pieza de sujeción al *setup* experimental, como se muestra en la figura.

UNIVERSIDAD DE MÁLAGA

2.4.4. Modalidad para rigidez

La figura 2.29 muestra el *setup* utilizado para la clasificación de objetos según su rigidez. La disposición final parte del *setup* presentado en la figura 2.25. En este caso, se ha añadido una pieza en la base que incluye una célula de carga, con la que se pueden medir fuerzas de hasta 4 N, junto a un amplificador de la señal de la célula [68].

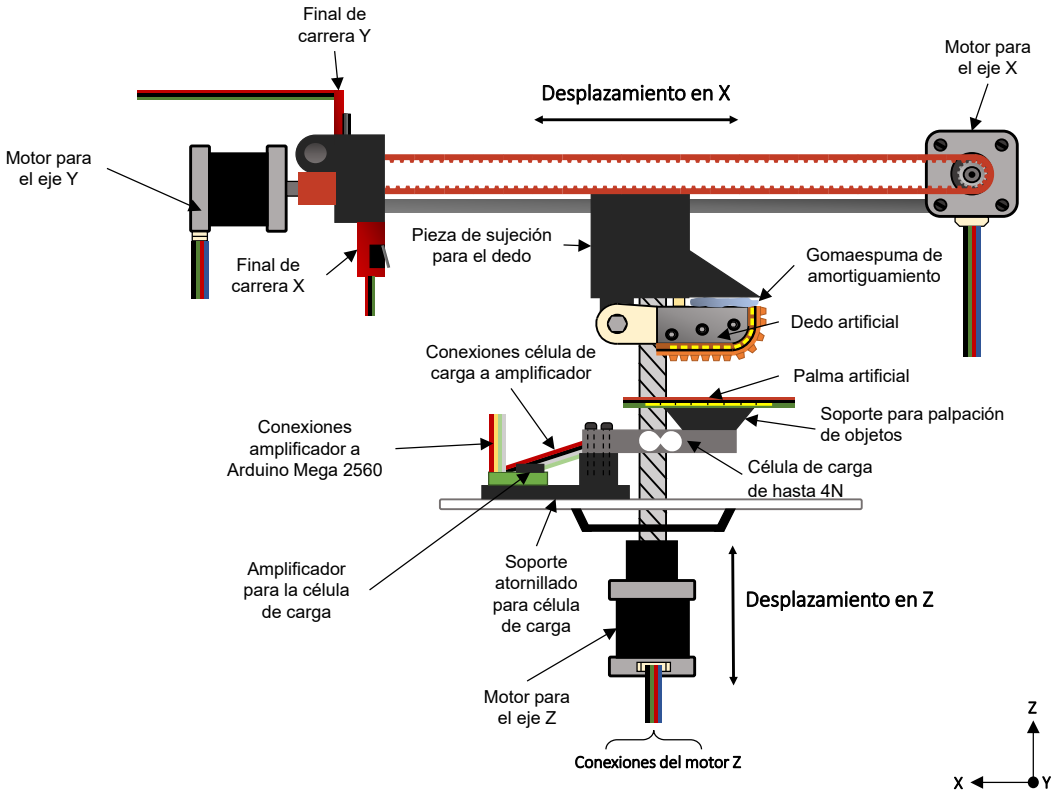


Figura 2.29. *Setup* para la palpación de objetos.

La célula de carga está conectada al conversor analógico-digital HX711 de 24 bits [68]. En la figura A.2 del apéndice A de figuras, se puede ver el esquema de conexionado correspondiente. Las señales DOUT/SCK se conectan directamente a pines digitales de Arduino®. Tras el calibrado inicial de la célula [69], Arduino® recibe los datos de la célula, que son monitorizados constantemente durante el experimento para evitar dañar los sensores o el objeto a explorar.

En el otro extremo de la célula de carga se coloca un soporte más pequeño sobre el que se encuentra la palma artificial, descrita en la subsección 2.3.2. Los objetos a evaluar se colocarán entre la palma y el dedo, y se someterán a un movimiento de compresión y descompresión en dirección vertical para extraer características que se envían al clasificador en tiempo real. Este clasificador estará embebido en el ARM® del SoC (ver capítulo 4).

UNIVERSIDAD DE MÁLAGA

2.5. Herramientas para la implementación en SoC

En esta sección se describe la plataforma hardware utilizada en esta tesis para la implementación de los algoritmos de pre-procesado. Además, se describe el entorno de desarrollo Vivado IDE® para la descripción del diseño en la FPGA, el entorno de desarrollo *System Development Kit* (SDK) para la programación del ARM® e implementación del sistema operativo en tiempo real FreeRTOS® junto a la pila lwIP.

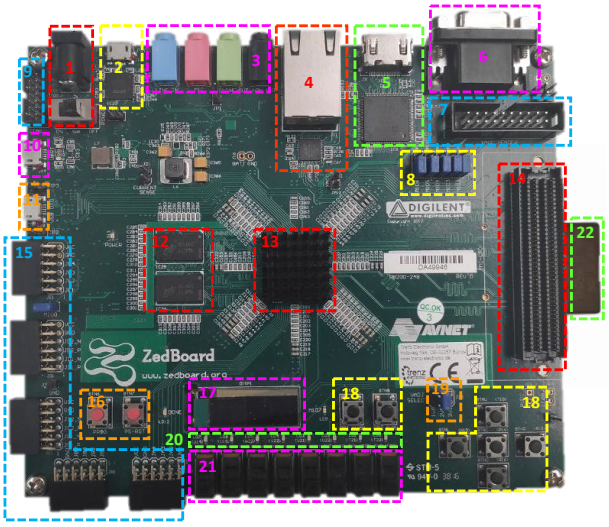
2.5.1. Placa de desarrollo ZedBoard™ y Zynq-7020®

Para validar cada algoritmo, existen diferentes opciones de dispositivos. En primer lugar, se puede considerar el uso de un procesador de propósito general o *General Purpose Processor* (GPP). Aunque este tipo de dispositivos es ampliamente utilizado, algunas aplicaciones en tiempo real requieren resultados inmediatos. Otra opción es el diseño de un *Application Specific Integrated Circuits* (ASIC). Un ASIC es un circuito específicamente diseñado para un propósito o tarea específica. La implementación en ASIC de un algoritmo garantiza un diseño óptimo en términos de restricciones de área y tiempo. Sin embargo, una vez que el ASIC se ha configurado, el diseño no se puede modificar. Por otro lado, las unidades de procesamiento gráfico o *Graphic Processor Units* (GPU), son herramientas muy potentes que permiten procesamiento en paralelo. Sin embargo, estos dispositivos requieren un alto consumo de energía, y en general su precio es mayor que otras opciones. Una solución intermedia entre un ASIC y un GPP es el hardware reconfigurable. Los dispositivos FPGA están compuestos de miles de células lógicas reconfigurables [70].

En el contexto de esta tesis, se utiliza la ZedBoard™ de AVNET® [71]. Se trata de una placa de desarrollo que permite la flexibilidad necesaria para evaluar diferentes alternativas para la implementación de los algoritmos de pre-procesamiento en dedo y palma. La placa está basada en el dispositivo SoC Zynq™ XC7Z020-CLG484-1. Este dispositivo combina las capacidades de una FPGA de hasta 85000 células lógicas programables o *programmable logic cells* (PLC), y las de un procesador *hard-core* de doble núcleo ARM® Cortex™ A9. La placa dispone de numerosos periféricos que permiten su utilización en diferentes aplicaciones. La Figura 2.30 muestra una imagen con las principales partes de la ZedBoard™.

La placa ZedBoard™ admite cuatro opciones de configuración de inicio (*boot mode*): a través de tarjeta SD, Quad-SPI, JTAG independiente o JTAG en cascada (ver cuadro

UNIVERSIDAD DE MÁLAGA



1. Alimentación 12V/5A
2. Conector de programación y *driver* FTDI
3. Líneas de entrada/salida para audio
4. Conector y controlador para Ethernet
5. Puerto HDMI y controlador
6. Puerto VGA
7. Puerto XADC
8. Jumpers de configuración en posición JTAG (cascada)
9. Conector JTAG de Xilinx
10. Conector para puerto serie (USB-UART)
11. Puerto USB-OTG
12. Memoria DDR3 de 512Mb
13. Dispositivo SoC Zynq-7000 más disipador de calor
14. Conector FMC
15. Conectores PMODs (x5)
16. Botones de programación y reset
17. Display OLED
18. Botones de usuario
19. Voltaje de unión ADJ, configurado a 2.5 V
20. LEDs de propósito general
21. Switches de propósito general
22. Tarjeta SD, con ranura en la parte posterior de la placa

Figura 2.30. Placa de desarrollo ZedBoard™ con sus principales módulos diferenciados.

8 en la figura). En la imagen se ha utilizado la configuración en cascada[4]. La gama Xilinx® de All-Programmable SoC (APSoC) además añade la versatilidad de la interfaz *Advanced eXtensible Interface* (AXI), que permite el interconexionado[5] de elementos entre la parte PL (lógica programable) y la parte PS (sistema de procesamiento). En concreto, la mayoría de módulos hardware desarrollados en la arquitectura del sistema planteada en esta tesis (ver sección 3.8.2 del capítulo 3) se comunican entre sí a través del buses AXI-Stream.

Los buses AXI-Stream se emplean para el intercambio de datos entre componentes de un sistema digital, permitiendo una transferencia eficiente unidireccional, sin la necesidad de recurrir a direcciones de memoria. La transmisión de datos se realiza a través de canales, donde cada canal tiene tres señales principales: una señal de datos *tdata* (configurable entre 16, 32 o 64 bits) y dos señales de control, *tvalid* y *tready* (de 1 bit),

[4]La configuración en cascada JTAG permite que una sonda acceda a todos los dispositivos subsiguientes a través de una sola conexión al puerto JTAG. El inicio mediante SD o SPI es útil en aplicaciones que requieren almacenamiento de datos en una tarjeta SD o cuando la configuración de hardware y *firmware* del SoC es definitiva.

[5]El protocolo de bus AXI forma parte de ARM® AMBA, una familia de buses para microcontroladores que se introdujo por primera vez en 1996. La primera versión fue la interfaz AXI3, lanzada en 2003, seguida de AXI4, que es la versión actual utilizada desde 2010. Hay tres tipos de interfaces AXI4:

- AXI4; para diseños con altos requerimientos de memoria y rendimiento.
- AXI4-Lite; para comunicaciones ligeras, con bajo *throughput*, como pueden ser el control de registros de estado.
- AXI4-Stream; para transmitir datos a una alta velocidad.

Xilinx® introdujo estas interfaces por primera vez como parte de las herramientas de software de la suite de diseño ISE®, en la versión 12.3 [72] [73].

UNIVERSIDAD DE MÁLAGA

facilitando un mecanismo de control de flujo que asegura la correcta sincronización entre el emisor y el receptor. Opcionalmente, se pueden incluir las señales *tstrb* para indicar si la información transmitida se procesa como bytes de datos o de posición, *tkeep* para indicar si el contenido asociado a un byte de información forma parte del flujo de datos, *tlast* que indica el final de una trama de datos, *tid/tdest/tuser* que proporcionan identificadores para las tramas de datos y *twakeup* que identifica cualquier actividad asociada con la interfaz AXI-Stream [74].

En la figura 2.31 se proporciona un diagrama de la negociación básica entre dos módulos actuando como maestro y esclavo en una comunicación por AXI-Stream. En el diagrama, el esclavo activa la señal *tready* indicando que está listo para recibir información. El maestro envía los datos en el ciclo de reloj (ACLK) T2, activando la señal de *tvalid*. El dato se procesará en el flanco de subida de reloj (ACLK) correspondiente al ciclo T3.

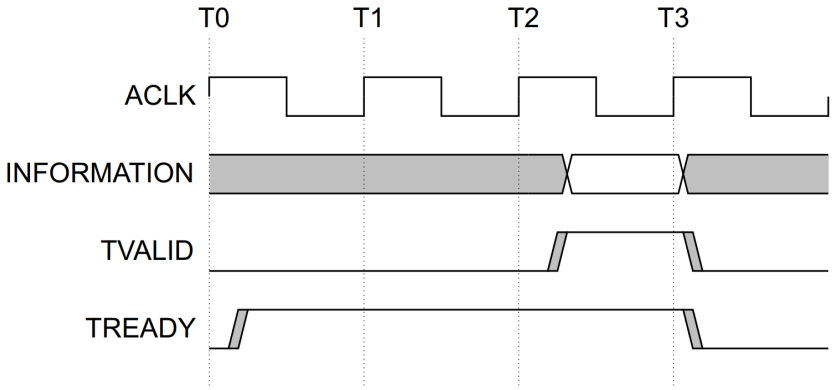


Figura 2.31. Negociación básica utilizando el protocolo de comunicación AXI-Stream [74].

La sección PL es ideal para la implementación de lógica de alta velocidad, aritmética y flujo de datos entre subsistemas. El hardware de esta parte se puede configurar a través de módulos IP Core proporcionados por la librería de Xilinx®, o bien mediante su diseño en *VHSIC Hardware Description Language* (VHDL) o Verilog (lenguajes de descripción de hardware). La gama Xilinx® de dispositivos SoC Zynq-7000® comparte arquitectura con las familias FPGA de Xilinx® Artix®-7 (gama baja), Kintex®-7 (gama media) y Virtex®-7 (gama alta). En concreto, la FPGA contiene *configurable logic blocks* (CLB) que se comunican entre sí y con los bloques de entrada/salida (I/O Blocks)[6] a través de una matriz programable de interconexiones. Cada CLB está compuesto por 2 *slices* (*slice* L y *slice* M). Cada *slice* a su vez contiene 4 *logic cells* (LC). Cada LC está compuesta por:

[6]Los *buffers* I/O se refieren a los pines físicos o bloques de hardware especiales del dispositivo que se utilizan para interactuar con el mundo exterior. Estos pines pueden ser programados para recibir información de entrada desde otros componentes electrónicos (entradas), o para enviar información a otros componentes (salidas).

UNIVERSIDAD DE MÁLAGA

- Una *Look-Up Table* (LUT) de 6 entradas, que, de forma general, permite su configuración como generador de funciones, como RAM distribuida de 64x1 bits o como registro de desplazamiento de 32 bits, *shift register LUT* (SRL).
- Dos elementos de memoria o *flip-flops* (FF).

Cada *slice* además incluye lógica de acarreo (CARRY), lógica aritmética y multiplexores para extensión de funciones. En la figura 2.32 se ilustra el interior de una CLB y de un *slice*.

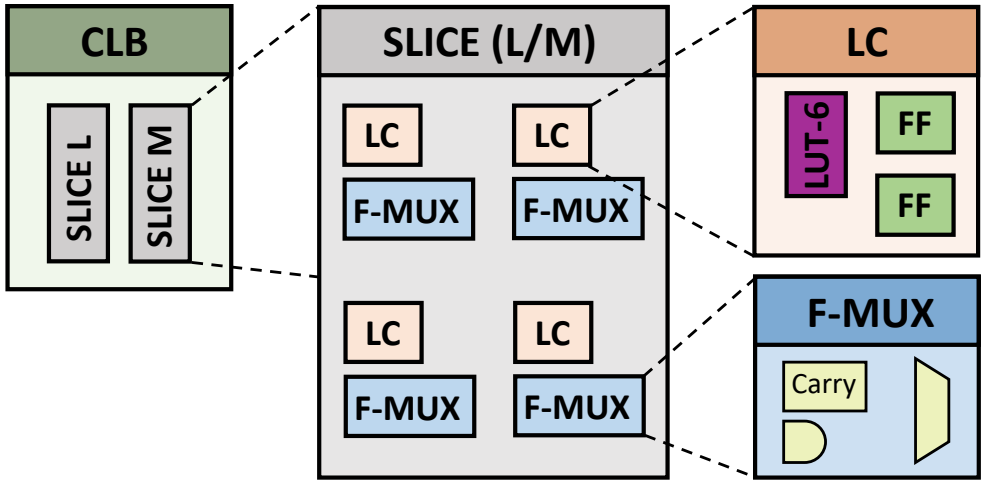


Figura 2.32. Esquema interno de una CLB, que muestra a su vez el interior de un SLICE.

Por otro lado, el SoC incluye también un número determinado de bloques empotrados *Block RAMs* (BRAM) y *digital signal processors* (DSP). Estos bloques permiten aumentar la capacidad de operación de la FPGA en términos de memoria y procesado de datos. A modo de resumen, en la tabla B.1 del apéndice B de tablas, se resumen las principales características para el dispositivo Zynq™ XC7Z020-CLG484-1 [75].

2.5.2. Entorno de desarrollo Vivado IDE ®

Vivado Design Suite® es un software de desarrollo de hardware integrado o *hardware description language* (HDL) producido por Xilinx®. Su primera versión se liberó en abril de 2012, sucediendo al software anterior de Xilinx® ISE. Esta *suite* proporciona un entorno completo para la elaboración, verificación y depuración de diseños en dispositivos FPGA. Vivado® incluye una amplia variedad de herramientas. Entre estas se pueden destacar un editor de código para los lenguajes de descripción hardware VHDL y Verilog, un simulador propio (XSIM) que permite simular y verificar el funcionamiento de los diseños digitales antes de su implementación en hardware, herramientas de depuración como los módulos *Integrated Logic Analyzer* (ILA) que permiten la captura

UNIVERSIDAD DE MÁLAGA

y visualización de señales internas del diseño mientras se ejecuta en hardware o herramientas de síntesis y enrutamiento. Todas estas opciones permiten a los ingenieros de hardware transformar diseños de alto nivel en representaciones de bajo nivel que pueden programarse en dispositivos FPGA.

Con su enfoque en facilitar la implementación de diseños de hardware complejos, Vivado® es particularmente notable por su capacidad para manejar diseños a nivel de sistema, con unas determinadas especificaciones de comportamiento para las entradas y salidas. Además, Vivado® es compatible con los estándares de la industria, lo que facilita su integración con otras herramientas de diseño hardware y software.

2.5.3. Entorno de desarrollo SDK

El entorno de Xilinx® SDK es un entorno de diseño integrado o *Integrated Design Environment* (IDE) para la creación de aplicaciones embebidas en microprocesadores de AMD: Zynq™ UltraScale+ PSoC, Zynq 7000 SoCs, y en el microprocesador *soft-core* MicroBlaze™. SDK es el primer entorno que proporciona herramientas de diseño, depuración y análisis de rendimiento. Entre algunas de sus principales ventajas destaca:

- Entorno incluido con el software Vivado Design Suite®, por lo que se puede interactuar directamente con las interfaces hardware desarrolladas.
- Está basado en Eclipse® 4.5.0.
- Incluye la posibilidad de crear aplicaciones 'bare-metal' (ejecutando la aplicación directamente en el ARM®) o en una pila de Linux embebido en el SoC.
- Incluye librerías completas para Xilinx y *drivers* de dispositivos.
- Incluye soporte para depuración y herramientas multi-procesador. En concreto, el 'AMD System Debugger' permite depurar el código de la aplicación a través de la línea de comandos y mediante la interfaz GUI de SDK. Este depurador proporciona las herramientas básicas de depuración (*breakpoints*, *watchpoints*, ejecución paso a paso del código, visualización de las variables de programa, de memoria y de la pila), así como la depuración del código que se ejecuta en los dos núcleos del procesador ARM®.
- El entorno SDK permite configurar el hardware del SoC mediante la cadena JTAG, o a través de la interfaz Q-SPI o el contenido de la tarjeta SD (ver subsección 2.5.1).

UNIVERSIDAD DE MÁLAGA

- Integra el sistema operativo en tiempo real o *real time operating system* (RTOS) FreeRTOS® para todas las plataformas.

2.5.4. Utilización de FreeRTOS® + lwIP Stack

Es importante recordar que el sensor táctil que se está utilizando consiste en un *array* de 55 electrodos (consultar sección 2.3). Por lo tanto, cuando una textura se explora en tiempo real, se obtiene una gran cantidad de datos por cada uno de estos tácteles. Aunque el análisis realizado en esta tesis se basa en el preprocesamiento de la señal obtenida por un táctel, el procedimiento es extrapolable a toda la matriz. Para asegurar un buen *throughput* y baja latencia, los datos obtenidos en tiempo real se envían a través de un puerto de conexión Ethernet. De esta forma, se puede alcanzar una velocidad de conexión física de hasta 1 Gbps, aunque en la práctica se logran 100 Mbps. Esto supera la limitación de una conexión serie a través de la interfaz UART en la ZedBoard™, donde la velocidad máxima es de 115200 baudios.

En lo respectivo al SoC se utiliza un RTOS embebido junto a una pila de bajo consumo o *lightweight* TCP/IP (lwIP *stack*). Específicamente, FreeRTOS® proporciona facilidad de uso, por lo que es apropiado para el propósito de este trabajo [76]. Teniendo en cuenta que el diseño no debe depender de una vasta cantidad de recursos, la implementación de FreeRTOS®+lwIP está justificada.

2.6. Resumen

En este capítulo se presenta el *setup* experimental que se ha construido para la realización de experimentos de este trabajo así como los sensores táctiles que se han utilizado para la adquisición de los datos durante las exploraciones. A continuación se expone un breve resumen de los contenidos del capítulo:

- En primer lugar, en la sección 2.1, se ha introducido la problemática existente en torno al uso de sensores táctiles en aplicaciones diversas. Específicamente, por qué es interesante la utilización de estos sensores para obtener características de texturas u objetos, y las opciones en cuanto a discretización de la parte sensible del sensor y estructuración de la superficie de la parte elastómera. Se comentan algunas ventajas e inconvenientes de estas configuraciones.
- A continuación, en la sección 2.2 se describe el principio de funcionamiento de los principales tipos de sensores utilizados en el desarrollo de aplicaciones industriales y robóticas.

UNIVERSIDAD DE MÁLAGA

- Posteriormente, en la sección 2.3 se describen los sensores táctiles utilizados en los experimentos de esta tesis. Se comenta el funcionamiento de la electrónica local al sensor del dedo y cómo se han diseñado y fabricado las capas que cubren la matriz de electrodos (capa de piezoresistivo discretizado más *cover* estructurado). Además, se presentan una serie de resultados en cuanto a la caracterización de los tácteles de la matriz para diferentes fuerzas y piezas de diferente geometría sobre el sensor. Por otra parte, también se explica en qué consiste el fenómeno perjudicial del *crosstalk* y qué efectos tiene en la medida, así como las estrategias implementadas para reducirlo. Todos estos resultados se presentaron en el congreso *Cairo International Biomedical Engineering Conference* (CIBEC) de 2018 [60].

- En todos los casos, se mejora la respuesta del sensor táctil siempre que se utiliza una lámina piezoresistiva discretizada y un *cover* estructurado. Como se ha visto, de esta manera, se consigue concentrar las fuerzas en los electrodos de los tácteles, aumentar la sensibilidad frente a estímulos tangenciales y reducir el efecto del *crosstalk*.

- También se comenta el uso de la palma artificial en los experimentos de rigidez. La palma se conecta de forma directa a la FPGA de una placa de adquisición y comunicación, a la que también se conecta el dedo.

- Después, en la sección 2.4 se presenta la disposición del *setup* experimental completo. Se comentan en mayor detalle todas las partes de la electrónica de control de los motores y el software necesario para definir trayectorias de experimentación. Para los experimentos de exploración de texturas se configura el *setup* básico sin la parte dedicada a la palma artificial y se configuran trayectorias lineales a lo largo del eje X de la impresora 3D, ver capítulo 3. Por otro lado, para los experimentos de rigidez, se incorporan los sensores del dedo y palma y se define un proceso de palpación, en el que los objetos se comprimen y descomprimen de forma incremental, ver capítulo 4.

- Por último, en la sección 2.5 se presentan las herramientas para la implementación en SoC del sistema definido en el marco de esta tesis, y se hace un resumen del capítulo en la sección 2.6.

UNIVERSIDAD DE MÁLAGA

3

Caracterización y clasificación de texturas

En este capítulo, se estudian en profundidad la caracterización y clasificación de texturas por medio del sensor táctil descrito en el capítulo 2. Se comienza con una introducción al uso de sensores táctiles en la detección de texturas, revisando algoritmos propuestos previamente para este propósito (sección 3.1). Se da una descripción detallada de nuestro algoritmo de extracción de características, algoritmo de barrido o *sweep*, y sus implicaciones en el rango de frecuencia a explorar, incluyendo una comparación de simulaciones y resultados reales con texturas de una frecuencia espacial específica. Además, se presenta una alternativa para detección en frecuencia, algoritmo de Goertzel (sección 3.2). Se presentan luego las especificaciones y el diseño de un filtro *antialiasing*, incorporado para asegurar la exploración en el rango establecido (sección 3.3). Posteriormente, se discuten dos simulaciones realizadas en Matlab® para analizar la clasificación con texturas de distintos armónicos. De este análisis, se seleccionan el conjunto de doce texturas a examinar en los experimentos (sección 3.4). Además, se describe el procedimiento experimental y presentamos los datos recogidos por el sensor táctil, incluyendo la aplicación del algoritmo de barrido propuesto y de una FFT, por un lado, y la búsqueda de los índices del algoritmo de Goertzel por otro (sección 3.5). Una vez obtenidos los vectores de características, se explica el algoritmo de entrenamiento y clasificación *k-means*, así como el proceso de eliminación de valores no deseados y la distribución estadística de las diferentes clases de texturas (sección 3.6).

UNIVERSIDAD DE MÁLAGA

Los resultados obtenidos mediante diferentes configuraciones del algoritmo de barrido, la FFT, y el de Goertzel, aplicadas a datos reales de exploración de texturas, se presentan en la sección 3.7. Tanto el algoritmo de barrido como la FFT, y el algoritmo de Goertzel, se implementan en un dispositivo SoC, que incluye una FPGA y un procesador ARM®. Se detalla la arquitectura global del sistema al implementar los algoritmos en el SoC, así como el consumo en términos de recursos lógicos, memoria, potencia y tiempo de procesamiento. También se describe la ejecución del código en el procesador ARM® (sección 3.8). Después se presenta una discusión sobre los resultados de clasificación obtenidos (sección 3.9). El capítulo concluye con un resumen del contenido del mismo (sección 3.10).

3.1. Introducción

La utilización de sensores táctiles robóticos para la detección de texturas ha permitido el desarrollo de aplicaciones en una variedad de campos, como la ingeniería biomédica, la robótica quirúrgica y la prostética [3]. La implementación de estrategias de tacto activo también ha permitido una mejor comprensión de las propiedades de las superficies de los objetos a explorar y una mayor precisión en su manipulación [2] [77].

La discriminación de patrones derivada de la exploración de distintas superficies y texturas puede abordarse de múltiples formas. Un enfoque inicial implica el procesamiento de imágenes táctiles utilizando sensores táctiles de alta resolución [78] [57] [79]. Sin embargo, una de las carencias de estos sistemas es la ausencia de eventos dinámicos derivados de la interacción entre el sensor y la superficie. Además, la implementación de estrategias de procesamiento en sistemas empotrados incrementa las demandas computacionales. A pesar de las estrategias propuestas para mejorar el rendimiento [80], persiste la necesidad de optimizar el uso de recursos computacionales, consumo de energía y tráfico de datos.

Más recientemente, un gran número de investigaciones en detección de texturas se han enfocado a estrategias de tacto activo, donde el sensor táctil y la superficie de exploración se mueven uno respecto a la otra [81] [82] [83]. De esta forma, al explorar diferentes superficies, las microvibraciones producidas por fricción debida a las fuerzas normales y tangenciales se utilizan como entrada para los algoritmos de detección de texturas [84] [85] [86]. Este es el mecanismo que el sentido del tacto humano utiliza para recopilar información sobre las características superficiales de un objeto[1].

[1]El sentido del tacto humano funciona de manera similar a través de la piel, que es el órgano más grande del cuerpo y contiene cuatro tipos de mecanoreceptores: células Merkel, terminaciones de Ruffini,

UNIVERSIDAD DE MÁLAGA

Con un sensor de entrada definido para el sistema robótico, los datos obtenidos a través de las estrategias de tacto activo se preprocesan para extraer un conjunto de descriptores o vectores de características. Estos sirven como entrada para los algoritmos de clasificación. Algunos investigadores [87] sugieren el uso directo de los datos del sensor táctil para la etapa de clasificación. Sin embargo, esto puede provocar que el algoritmo memorice patrones, aumentando el riesgo de sobreajuste durante el entrenamiento y requiriendo más recursos computacionales y de memoria. Por tanto, se adoptan estrategias para obtener un conjunto reducido de características a partir de los datos del sensor táctil [88].

La detección de texturas puede ser de patrones periódicos o no periódicos. Para derivar características relevantes, se emplean diversos enfoques. Algunos investigadores utilizan descriptores estadísticos clásicos, como la media, desviación estándar y medidas de fricción, así como el análisis de contenido espectral. Estos métodos proporcionan información sobre la potencia de la señal, las relaciones de energía entre las diferentes bandas de frecuencia y otros parámetros espectrales [89] [90] [91]. Otros autores utilizan la Transformada Wavelet [92], características obtenidas a partir de la densidad espectral como el centroide, la asimetría y la curtosis, o los Modelos Ocultos de Márkov para ver las variaciones de las características en el tiempo [93]. En [27], se sugiere el uso de descriptores resistentes a los cambios en los movimientos de exploración, definidos como actividad (potencia de la señal) y parámetros derivados de ésta, como la movilidad y complejidad. Estos, utilizados junto con correlaciones lineales y no lineales de señales obtenidas de diferentes sensores, permiten la clasificación de diferentes objetos y texturas. No obstante, los descriptores utilizados son complejos de obtener localmente en tiempo real.

Así, la mayoría de las propuestas para la detección de texturas con tacto activo se basan en el procesamiento de la respuesta en frecuencia obtenida de los sensores táctiles. En particular, en el cálculo de los coeficientes de la transformada rápida de Fourier (FFT) [59] [94] [47] [32] [18]. En tales estrategias de tacto activo, es necesario mantener constante la velocidad relativa entre el sensor y la superficie conocida. Para sistemas que utilizan la información de múltiples tácteles como entrada, se pueden adoptar estrategias de reducción de la dimensionalidad de los datos, como el análisis de componentes principales [95].

Aunque las estrategias basadas en el cálculo de la FFT se utilizan ampliamente en sistemas robóticos táctiles, su implementación requiere de muchos recursos hardware y

corpúsculos Meissner y corpúsculos Pacinian, que difieren en su tipo de respuesta, lenta o rápida, y en la detección de frecuencias frente a estímulos, bajas o altas, respectivamente [3].

UNIVERSIDAD DE MÁLAGA

un alto consumo de potencia. Además, como las respuestas son individuales para cada táctel, es necesario procesar grandes cantidades de datos. En contraste, el algoritmo propuesto en [96] ofrece más beneficios en sistemas embebidos de bajo costo a costa de sacrificar el rendimiento. Este algoritmo divide el espectro de frecuencia en bandas y utiliza el vector de potencia de cada banda como descriptor. Los cálculos se realizan en el dominio del tiempo y la complejidad del algoritmo en tiempo y espacio es lineal con el número de bandas.

Para resolver los problemas de memoria y tiempo, resulta útil buscar algoritmos de pre-procesado amigables con el hardware (hardware-*friendly*). En este contexto, [41] propone un algoritmo bioinspirado para medir la aspereza de diferentes superficies, que utiliza un filtrado adaptativo de paso-banda. Se trata de un algoritmo de gradiente-descendente que ajusta un factor de retardo hasta que el máximo de la raíz cuadrada de la salida del sensor alcanza su valor máximo, y de acuerdo con este valor máximo encontrado, se obtiene una frecuencia característica asociada a la superficie explorada.

Basándose en [41], en esta tesis se propone un algoritmo de barrido para buscar las principales frecuencias espaciales características de diferentes texturas. El algoritmo se puede aplicar a un solo táctel, lo que lo hace más sencillo y robusto ante la variabilidad entre diferentes tácteles. Dado que la matriz táctil está compuesta por diferentes tácteles, este algoritmo es adecuado para implementaciones que permiten la adquisición y procesamiento de datos en paralelo, y en particular, en dispositivos FPGA [37] [97]. En esta tesis, este algoritmo se implementa en la placa de desarrollo ZedBoard™, y específicamente en el dispositivo Zynq-7000® XC7Z020-CLG484-1, que combina una FPGA junto a un procesador de doble núcleo ARM® CortexA9™ (ver sección 2.5 del capítulo 2). Además, en esta tesis se propone la aplicación del algoritmo de Goertzel para la detección de texturas, y también se implementa y evalúa su rendimiento.

3.2. Algoritmos de extracción de características

En esta sección, se presentan algoritmos de detección de texturas que permiten la extracción de características propias de la superficie que se está explorando. Estas características pueden considerarse descriptores para la etapa de clasificación.

3.2.1. Algoritmo de barrido o *sweep*

Una aproximación adaptativa consistente en tacto activo se refiere al algoritmo de barrido o *sweep* que se presenta en este apartado. En la figura 3.1 se ilustra la exploración de una textura $S(x)$ a una determinada velocidad v con un sensor táctil.

UNIVERSIDAD DE MÁLAGA

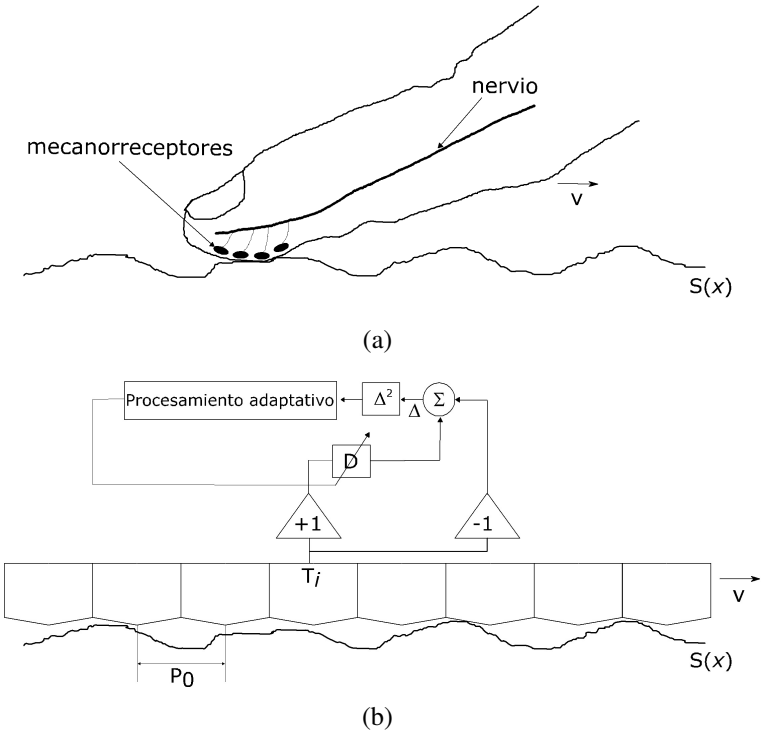


Figura 3.1. (a) Representación de la detección de una textura por un dedo humano y su realización bionspirada [41], y (b) su propuesta de realización simplificada [42].

La resolución de este sensor viene dada por el espacio entre tácteles, P_0. La salida del *array* de sensores táctiles viene dada por un filtro adaptativo que modifica el espaciado entre tácteles o *pitch* equivalente de acuerdo a la expresión (3.1).

$$P = P_0 + v \cdot D \tag{3.1}$$

donde P es el *pitch* equivalente de la textura y D es un retardo o *delay*. En [41] este parámetro de retardo se ajusta a partir de un algoritmo de gradiente descendente que trata de encontrar el máximo de la salida del filtro, explorando la superficie con un *array* lineal de 8 tácteles. Sin embargo, una implementación mucho más simple que utiliza la señal de un único táctel puede proporcionar una información similar. Si sólo se utiliza la salida de dos tácteles, $\mathbf{T_j}$ y $\mathbf{T_{j+1}}$, la raíz cuadrada de la salida del filtro (caso de [41]) viene dada por la expresión en (3.2).

$$Y = \sqrt{\frac{1}{M}\sum_{n=1}^{M}(T_j(vt_n) - T_{j+1}(vt_n + P))^2} \tag{3.2}$$

donde M es el número de muestras y t_n es el instante de tiempo para la muestra n. Tenga en cuenta que (3.2) es proporcional a la distancia euclídea entre los vectores

UNIVERSIDAD DE MÁLAGA

$\mathbf{T_j} = [T_j(vt_1), ...T_j(vt_M)]$ y $\mathbf{T_{j+1}} = [T_j(vt_1 + P), ...T_j(vt_M + P)]$. Se puede escribir la expresión:

$$V = \frac{1}{M}\sum_{n=1}^{M}(T_j(vt_n) - T_{j+1}(vt_n + P))^2 \tag{3.3}$$

que es más sencilla de calcular y dará una información similar. La textura $S(x)$ se puede modelar como una onda sinusoidal de amplitud A, de acuerdo a (3.4):

$$S(x) = Asin(\frac{2\pi}{\lambda}x) \tag{3.4}$$

donde λ es la longitud de onda y x la entrada. En este caso, $\mathbf{T_j}$ y $\mathbf{T_{j+1}}$ representan dos conjuntos de muestras de la misma forma de onda sinusoidal. En concreto, con una diferencia en fase de $\frac{2\pi}{\lambda}P$: $Asin(\frac{2\pi}{\lambda}vt_n)$ y $Asin(\frac{2\pi}{\lambda}(vt_n + P))$. Si ambos están en fase, entonces habrá un mínimo en V, es decir, habrá un mínimo cuando $P = \lambda \cdot k$, donde k es un entero. Por otro lado, si están en antifase, entonces $P = \lambda\left(\frac{1+2\cdot k}{2}\right)$. Teniendo en cuenta la expresión en (3.1), se pueden escribir estas condiciones como:

$$P_0 + v \cdot D = \lambda \cdot k \tag{3.5}$$

para los mínimos, y:

$$P_0 + v \cdot D = \lambda\left(\frac{1 + 2 \cdot k}{2}\right) \tag{3.6}$$

para los máximos.

Ahora bien, la aproximación que estamos considerando también es válida si se considerara una separación entre tácteles mínima, es decir, si solamente la salida de un único táctel se tuviera en cuenta. En este escenario $P_0 = 0$, ver figura 3.1. Esta estrategia es especialmente recomendable cuando se trabaja con *arrays* de sensores táctiles, puesto que puede existir una alta dispersión entre la respuesta de entrada-salida de diferentes tácteles. En este caso, las expresiones para los mínimos y máximos pasan a ser $v \cdot D = \lambda \cdot k$ y $v \cdot D = \lambda\left(\frac{1+2\cdot k}{2}\right)$, respectivamente. Estos mínimos y máximos se corresponden con los extremos de V, en (3.3). El uso de la estrategia propuesta en la figura 3.1 junto a un pre-procesado adaptativo permitirá la localización de estos mínimos y máximos, y por tanto, la caracterización de la textura que se explore.

Un procedimiento sencillo para la caracterización de la textura consiste en barrer o

UNIVERSIDAD DE MÁLAGA

hacer un *sweep* en diferentes valores del parámetro D, mientras que la textura se explora a una determinada velocidad v. Mientras que se hace este barrido, la expresión en (3.3) se calcula para cada valor de D en un rango de frecuencias espaciales previamente determinado. Además, los extremos de V pueden detectarse y almacenarse con unos pocos recursos lógicos basados en comparadores. De este modo, las longitudes de onda correspondientes a los armónicos principales de la textura, se pueden identificar a partir de (3.5) y (3.6). En esta tesis, se propone la localización de los mínimos en (3.5) como características de la textura a explorar.

Para encontrar los mínimos, un procedimiento simple consiste en calcular V, (3.3), para cada valor de D en un cierto intervalo, para después localizarlos con un algoritmo de detección de picos. El rango espacial de exploración máximo para este procedimiento viene dado por $D = mT_s$, con $1 \leq m \leq M$, siendo T_s el período de muestreo. Puesto que $t_n = nT_s$ y $1 \leq n \leq M$ en (3.3), el barrido se completa cuando $m = M$, y por tanto, el tiempo total de barrido para este algoritmo viene dado por (3.7). Esta idea también se muestra en la figura 3.2.

$$T_{sweep} = 2MT_s \tag{3.7}$$

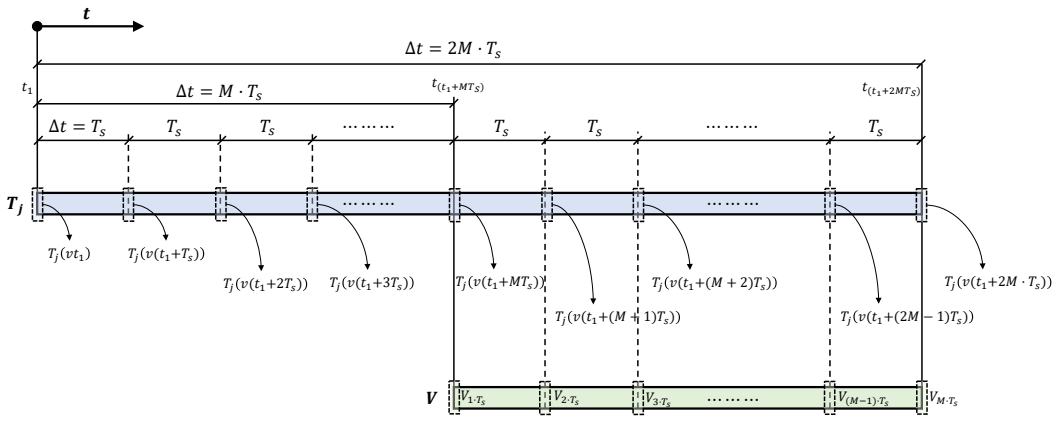


Figura 3.2. Diagrama temporal para el cálculo de V en (3.3), para $1 \leq m \leq M$. En el diagrama, t es el tiempo.

El número de muestras que deben adquirirse para calcular (3.3) en el rango $1 \leq m \leq M$ es $2M$. Hay que tener en cuenta que en esta aproximación existe un compromiso entre el tiempo de barrido (3.7) y la frecuencia espacial mínima, o equivalentemente la longitud de onda máxima, λ_{max}, que se puede detectar. Esta longitud de onda máxima vendrá dada por (3.8):

$$\lambda_{max} = MvT_s \tag{3.8}$$

UNIVERSIDAD DE MÁLAGA

Por otro lado, la frecuencia espacial máxima, o longitud de onda mínima, vendrá dada por el teorema del muestreo de Nyquist-Shannon [98], según (3.9):

$$\lambda_{min} = 2vT_s \tag{3.9}$$

Esta longitud de onda mínima es, en realidad, algo mayor cuando se considera una implementación real, ya que es necesario implementar un filtro *antialiasing* antes de calcular los términos de la suma agregada en (3.3). Esto es debido a que el cálculo de las diferencias $T_j(vt_n) - T_j(v(t_n + D))$ es sensible a ruido. El diseño de este filtro se explica en la sección 3.3.

Para terminar, nótese que aunque este algoritmo se puede aplicar a más de un táctel, nuestra propuesta se lleva a cabo con un solo táctel.

3.2.2. Resultados de simulación de barrido iniciales

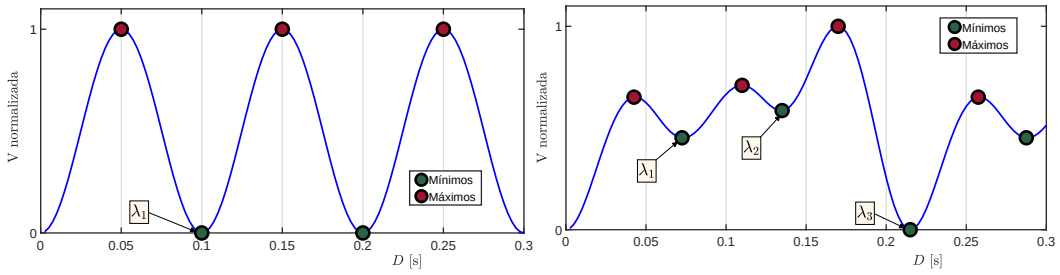
Para comprobar el funcionamiento del algoritmo de barrido, inicialmente se planteó la simulación sobre modelos de texturas ideales. Por ello, en este apartado, se muestran los resultados de simulación obtenidos sobre dos señales sinusoidales de una frecuencia y suma de tres frecuencias, respectivamente.

Para dos texturas ideales modeladas por un armónico y la suma de tres armónicos en frecuencia, respectivamente, se aplica el algoritmo de *sweep* dado por V, en (3.3). En concreto, las texturas vienen dadas por $S_1(x) = sin(\frac{2\pi}{\lambda_1} \cdot x)$ y $S_2(x) = sin(\frac{2\pi}{\lambda_1} \cdot x) + sin(\frac{2\pi}{\lambda_2} \cdot x) + sin(\frac{2\pi}{\lambda_3} \cdot x)$, respectivamente. Los resultados de aplicar el algoritmo *sweep* a estas texturas se visualizan en la figura 3.3, generada con Matlab®. Las frecuencias espaciales principales, o armónicos, de cada textura pueden deducirse a partir de los extremos en las ecuaciones (3.5) y (3.6). Específicamente, $k = 1$ para el primer mínimo y la expresión resultante $\lambda = v \cdot D$ (teniendo en cuenta que para nuestra propuesta de un solo táctel, $P_0 = 0$) proporciona los armónicos. La velocidad simulada es de $v = 10$ mm/s.

3.2.3. Algoritmo de gradiente descendente

Una alternativa para encontrar los extremos (3.5) y (3.6), es el uso de un algoritmo de gradiente descendente. Una posibilidad es la utilización del método *weight perturbation* [99]. Este procedimiento es un tipo de algoritmo de gradiente descendente simple de calcular y orientado a implementación en hardware. Si consideramos las diferencias en tiempo consecutivas dadas en la expresión (3.3), se puede escribir:

UNIVERSIDAD DE MÁLAGA



(a) (b)

Figura 3.3. Cálculo de V (3.3) normalizado con los extremos señalados para las texturas ideales: (a) $S_1(x) = sin(\frac{2\pi}{\lambda_1} \cdot x)$ y (b) $S_2(x) = sin(\frac{2\pi}{\lambda_1} \cdot x) + sin(\frac{2\pi}{\lambda_2} \cdot x) + sin(\frac{2\pi}{\lambda_3} \cdot x)$. Para $S_1(x)$, el primer mínimo indica el período espacial de la textura: $\lambda_1 = v \cdot D_{min} = 10\text{mm/s} \cdot 0.1\text{s} = 1\text{mm}$. Para $S_2(x)$, los tres primeros mínimos indican el período espacial correspondiente a las tres frecuencias espaciales que definen la textura: $\lambda_1 = v \cdot D_{min1} = 10\text{mm/s} \cdot 0.0725\text{ s} = 0.72\text{ mm}$, $\lambda_2 = v \cdot D_{min2} = 10\text{mm/s} \cdot 0.135\text{ s} = 1.35\text{ mm}$ y $\lambda_3 = v \cdot D_{min3} = 10\text{mm/s} \cdot 0.215\text{ s} = 2.15\text{ mm}$.

$$\frac{\Delta V}{T_s} = \frac{V(D_m + T_s) - V(D_m)}{T_s} \tag{3.10}$$

donde D se actualiza como:

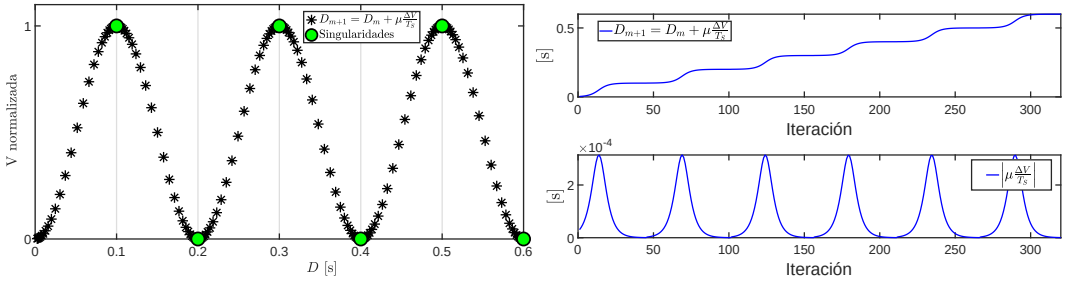
$$D_{m+1} = D_m + \mu \frac{\Delta V}{T_s} \tag{3.11}$$

siendo μ una constante.

Las figuras 3.4 y 3.5 muestran los resultados del algoritmo en la detección de los extremos, que corresponden a los armónicos principales de las texturas $S_1(x)$ y $S_2(x)$, antes utilizadas en la figura 3.3. Es relevante señalar que, dado que se necesita la muestra $T_j(v(t_n + D))$ para calcular V (ver ecuación 3.3) para cada frecuencia espacial posible, esta debe ser previamente obtenida. Por lo tanto, el tiempo requerido para finalizar la exploración y entregar un resultado es comparable al del algoritmo de barrido (ver ecuación 3.7). No obstante, la figura 3.4 demuestra que el algoritmo de gradiente descendente concluye el proceso en un número reducido de pasos, aunque la duración de cada paso pueda variar. Como consecuencia, es razonable prever un menor consumo de energía en una implementación con FPGA.

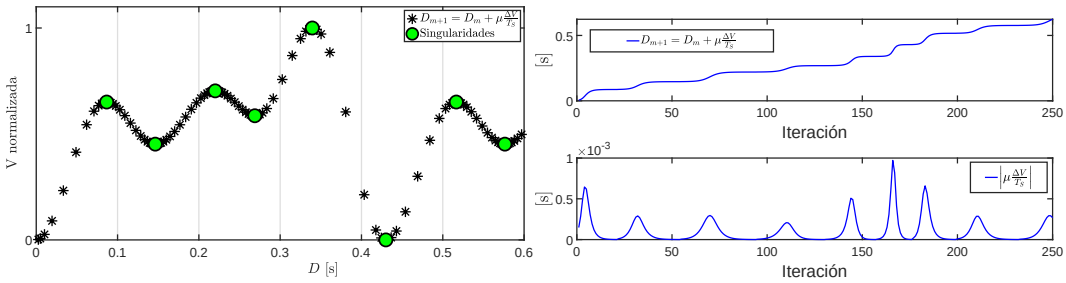
Por otro lado, el parámetro de aprendizaje μ tiene que ser lo suficientemente pequeño para que las frecuencias espaciales más altas se puedan identificar con éxito. Sin embargo, esto resulta en un mayor intervalo de tiempo para detectar la frecuencias espaciales más bajas. De hecho, la frecuencia de muestreo T_s es el incremento más pequeño posible para el parámetro de retardo D. En esos casos, la utilización del gradiente descendente no supone ningún beneficio respecto al *sweep*.

UNIVERSIDAD DE MÁLAGA



(a) (b)

Figura 3.4. (a) Cálculo de V (3.3) normalizado con singularidades proporcionadas por el algoritmo de gradiente descendente, de acuerdo con la expresión en (3.11), para una textura con un solo armónico en $\frac{1}{\lambda_1}$, donde $\lambda_1 = v \cdot D_{min} = 10\frac{mm}{s} \cdot 0.1$ s $= 1$ mm. (b) Actualización de D y gradiente de la varianza de acuerdo con la expresión en (3.10).



(a) (b)

Figura 3.5. (a) Cálculo de V (3.3) normalizado con singularidades proporcionadas por el algoritmo de gradiente descendente, de acuerdo con la expresión en (3.11), para una textura con tres armónicos en $\frac{1}{\lambda_1}$, $\frac{1}{\lambda_2}$ y $\frac{1}{\lambda_3}$, donde $\lambda_1 = v \cdot D_{min1} = 10\frac{mm}{s} \cdot 0.0725$ s $= 0.72$ mm, $\lambda_2 = v \cdot D_{min2} = 10\frac{mm}{s} \cdot 0.135$ s $= 1.35$ mm y $\lambda_3 = v \cdot D_{min3} = 10\frac{mm}{s} \cdot 0.215$ s $= 2.15$ mm. (b) Actualización de D y gradiente de la varianza de acuerdo con la expresión en (3.10).

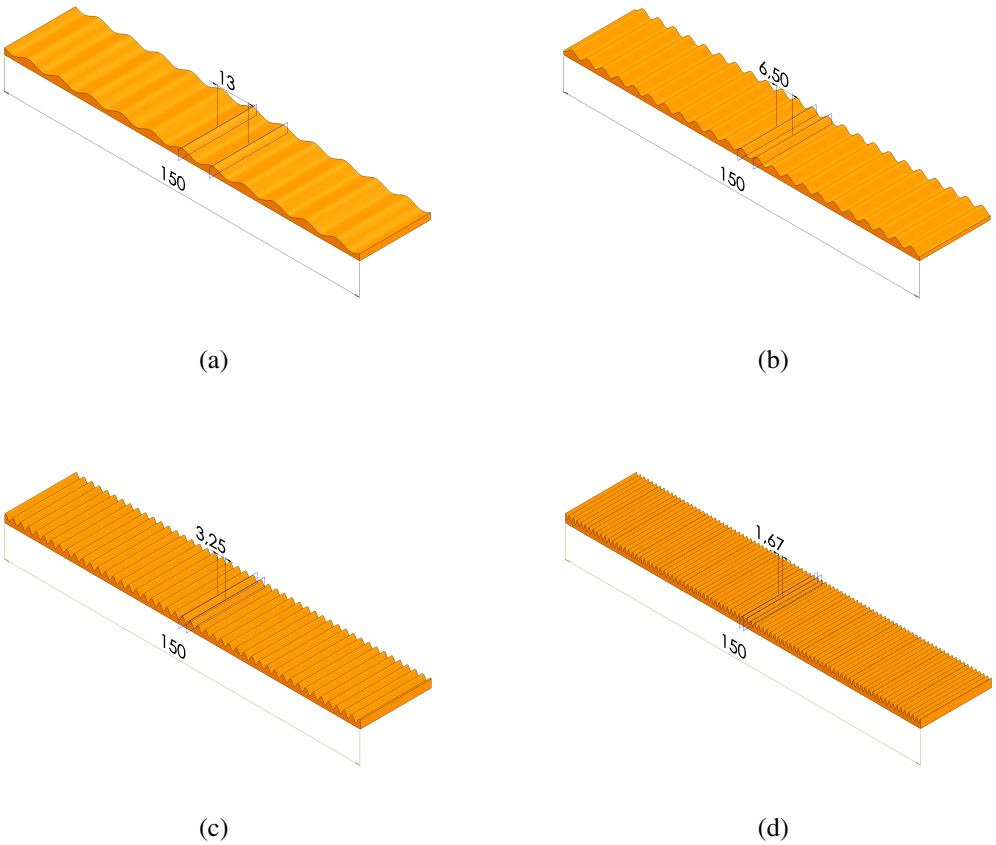
3.2.4. Resultados de aplicación del barrido para un conjunto sencillo de texturas

Inicialmente, con el objetivo de verificar que el algoritmo de barrido presentado en el apartado 3.2.1 tiene el comportamiento deseado en la búsqueda de las frecuencias principales, se diseñan cuatro texturas sinusoidales sencillas, de distinta frecuencia espacial, con el software de edición 3D SolidWorks®. Estas cuatro texturas están hechas de Filaflex®. En la imagen de la figura 3.6 se pueden ver las cuatro.

A continuación, con el *setup* experimental de la figura 2.28 (sección 2.4 del capítulo 2), las tres primeras texturas, 3.6(a), 3.6(b) y 3.6(c) se exploran a una velocidad de 10 mm/s, mientras que la última textura 3.6(d) se explora a 3 mm/s. Una vez se calcula V (3.3) sobre los datos recogidos de cada una, se obtienen las gráficas de la figura 3.7.

En la figura 3.7 la localización del primer mínimo en D da información sobre la frecuencia espacial principal de cada muestra, como se comentó en la subsección 3.2.1,

UNIVERSIDAD DE MÁLAGA



(a) (b)

(c) (d)

Figura 3.6. Muestras de texturas periódicas con período espacial λ de: (a) 13 mm, (b) 6.5 mm, (c) 3.25 mm y (d) 1.67 mm, [42]. La longitud de cada textura es de 150 mm. En la imagen, las unidades están en mm.

expresión (3.5). En concreto, para la textura dada en la figura 3.7(a) tenemos $\lambda = v \cdot D_{min} = 10\frac{mm}{s} \cdot 1.327\text{s} = 13.27$ mm. Para la textura de la figura 3.7(b) tenemos $\lambda = v \cdot D_{min} = 10\frac{mm}{s} \cdot 0.6407\text{s} = 6.407$ mm. Para la textura de la figura 3.7(c) tenemos $\lambda = v \cdot D_{min} = 10\frac{mm}{s} \cdot 0.3452\text{s} = 3.452$ mm. Finalmente, para la textura de la figura 3.7(d) tenemos $\lambda = v \cdot D_{min} = 3\frac{mm}{s} \cdot 0.5528\text{s} = 1.658$ mm.

Por otra parte, los resultados de simulación básicos presentados en la subsección 3.2.2 son coherentes con los que se obtienen en estos experimentos iniciales. Las singularidades de la curva dada por V (3.3) en cada caso, y más concretamente, la localización de mínimos, pueden utilizarse como características de entrada a los algoritmos de clasificación, como se verá más adelante.

UNIVERSIDAD DE MÁLAGA

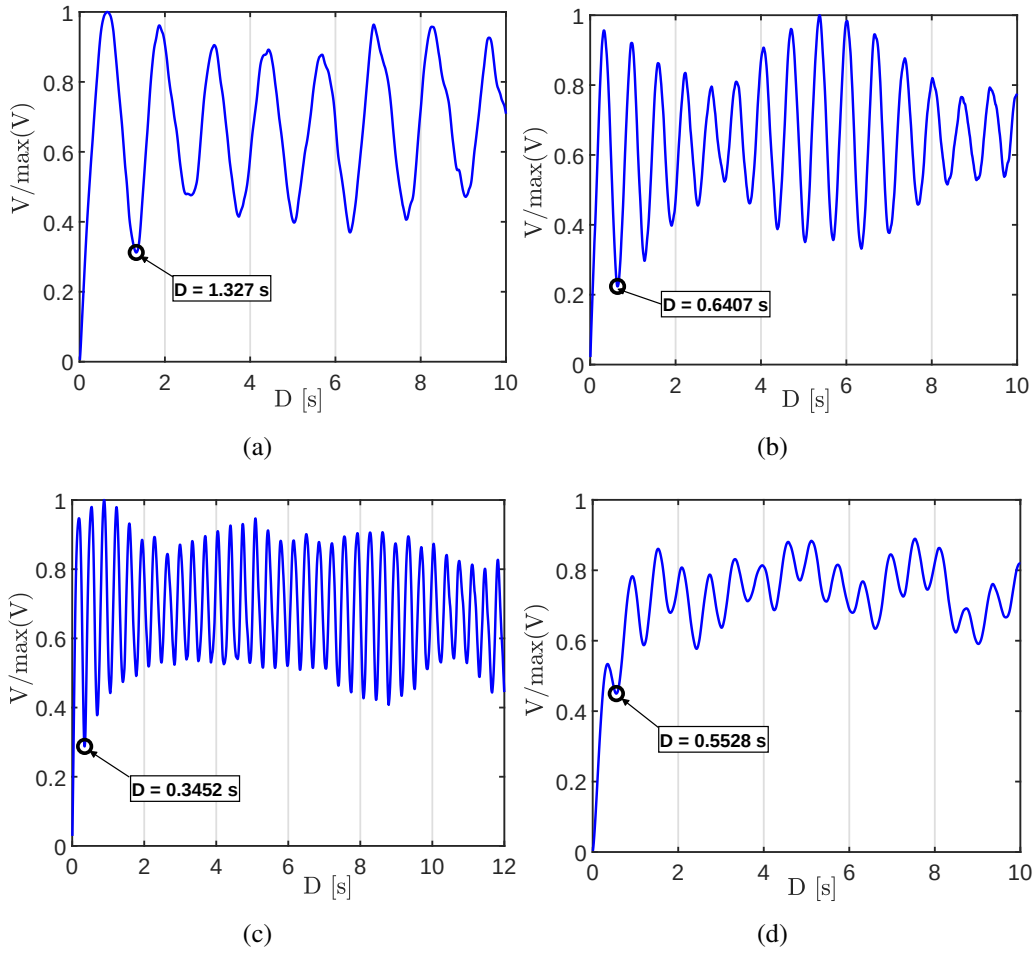


Figura 3.7. Para cada una de las texturas de la figura 3.6, se calcula V (3.3), y se representa de forma normalizada. El período espacial para cada una de ellas es de: (a) 13 mm, (b) 6.5 mm, (c) 3.25 mm y (d) 1.67 mm. Las texturas (a), (b) y (c) se exploran a una velocidad de $v = 10$ mm/s y la textura (d) se explora a una velocidad de $v = 3$ mm/s. En cada una de ellas se representa el valor para el parámetro D en la ecuación (3.3), a partir del cual se puede calcular la frecuencia espacial principal de cada una de ellas.

3.2.5. Algoritmo de Goertzel

En este apartado se propone la implementación de un algoritmo adicional de detección en frecuencia, denominado "algoritmo de Goertzel". Este método se deriva de la transformada discreta de Fourier o *Discrete Fourier Transform* (DFT) y permite detectar componentes específicas en una señal sin necesidad de analizar todo el espectro, lo que se traduce en un menor tiempo de ejecución y un ahorro de recursos. Resultados preliminares de este algoritmo se presentaron en el congreso internacional IWANN 2023 [111].

Introducción al algoritmo

El algoritmo de Goertzel calcula las K primeras componentes de la DFT de una señal muestreada en el tiempo, $\{x[n]\}$ de longitud N, cuya frecuencia de muestreo es F_s. De

UNIVERSIDAD DE MÁLAGA

esta forma, se pueden calcular únicamente aquellas frecuencias de interés, siendo por tanto más eficiente que la FFT para determinadas situaciones. A partir de la expresión para calcular la DFT de la señal:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi k \frac{n}{N}}, k = 0, ..., N-1 \tag{3.12}$$

se puede derivar la función de transferencia discreta dada por (3.13):

$$H_k(z) = \frac{1 - e^{-j\frac{2\pi k}{N}} z^{-1}}{1 - 2 \cdot cos\left(\frac{2\pi k}{N}\right) z^{-1} + z^{-2}} \tag{3.13}$$

Los cálculos intermedios para derivar (3.13) se pueden consultar en [112] y [113]. La expresión en (3.13) equivale a un sistema de segundo orden, cuya ecuación en diferencias es:

$$y_k[n] = x[n] - x[n-1]e^{-j\frac{2\pi k}{N}} + 2 \cdot cos\left(\frac{2\pi k}{N}\right) y_k[n-1] - y_k[n-2] \tag{3.14}$$

con $x[-1] = y[-1] = y[-2] = 0$. Esta estructura se puede describir utilizando las variables de estado:

$$s[n] = x[n] + 2 \cdot cos\left(\frac{2\pi k}{N}\right) s[n-1] - s[n-2], \tag{3.15}$$

mientras que la salida viene dada por:

$$y_k[n] = s[n] - e^{-j\frac{2\pi k}{N}} s[n-1] \tag{3.16}$$

siendo $s[-1] = s[-2] = 0$.

Esta descripción en espacio de estados se corresponde con la de un proceso de filtrado IIR, siendo ventajosa, ya que solamente el cálculo de la muestra $y_k[N]$ es de interés, y se corresponde con el valor del coeficiente espectral $X[k]$ buscado para el índice de frecuencia k. El gráfico de flujo del sistema de segundo orden para el algoritmo de Goertzel es el de la figura 3.8.

El algoritmo tradicional de la FFT, aplicado a una señal $\{x[n]\}$ de longitud N, tiene una demanda computacional proporcional a $N \cdot log_2(N)$. El número de operaciones

UNIVERSIDAD DE MÁLAGA

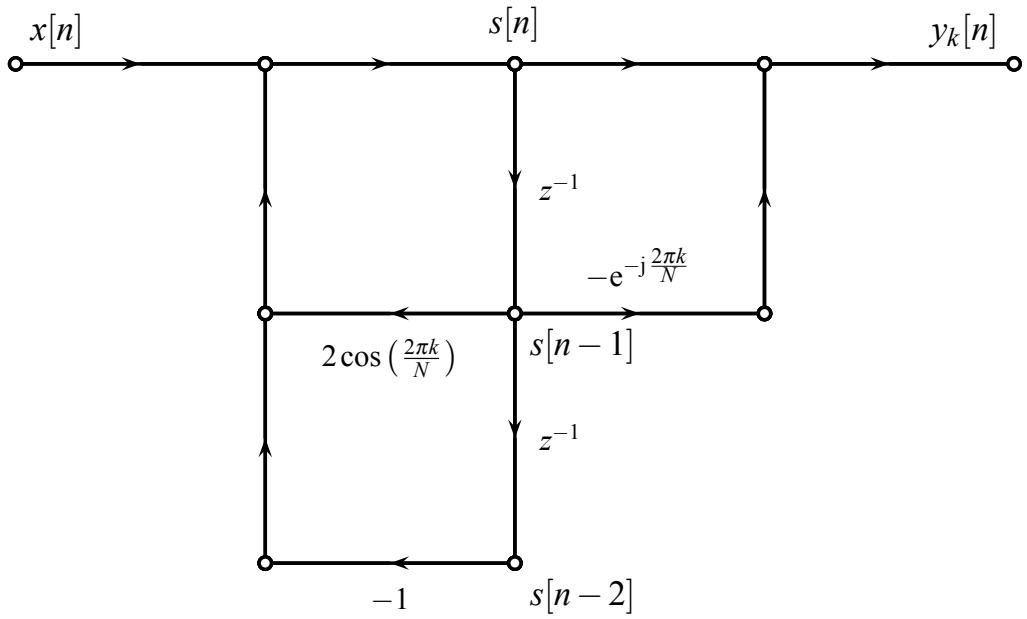


Figura 3.8. Gráfico para el flujo de señales correspondiente al sistema de segundo orden dado por la función de transferencia en (3.13), extraído de [112]. Las variables de estado se indican en la figura.

reales encontrado en la literatura es de aproximadamente $6N \cdot log_2(N)$ (tomando una multiplicación compleja como combinación de cuatro multiplicaciones y dos sumas). Para el algoritmo de Goertzel, son necesarias N multiplicaciones reales y $2N$ sumas reales para calcular una única frecuencia. Por tanto, se necesitan aproximadamente $3N$ operaciones para calcular una frecuencia. Si se extrapola al cálculo de K frecuencias, se necesitará un número de aproximadamente $3NK$ operaciones [112].

Teniendo en cuenta lo anterior, se puede comprobar para qué número de frecuencias K, es más ventajoso explotar el algoritmo de Goertzel frente a la FFT, en número de operaciones. Para ello, se puede escribir:

$$3NK < 6N \cdot log_2(N) \tag{3.17}$$

de donde se deriva (3.18):

$$K < 2 \cdot log_2(N) \tag{3.18}$$

Por otro lado, si se consideran señales reales, la FFT necesita un *buffer* de memoria de un total de $4N$ posiciones. Para el algoritmo de Goertzel, se necesitan $7K$ posiciones de memoria [112]. Por tanto, se puede escribir:

UNIVERSIDAD DE MÁLAGA

$$7K < 4N \tag{3.19}$$

de donde se deriva (3.20):

$$K < \frac{4}{7}N \tag{3.20}$$

Para una determinada implementación del algoritmo de Goertzel, ésta será más ventajosa que la FFT si cumple las condiciones dadas por las expresiones en (3.18) y (3.20). Más adelante, se presenta la metodología empleada para la búsqueda de los índices de Goertzel, subsección 3.5.3.

3.3. Diseño de un filtro *antialiasing*

En esta sección se describe el diseño del filtro *antialiasing* mencionado en la subsección 3.2.1. Como se vio en el capítulo 2, los datos proporcionados por el dedo se muestrean a una frecuencia de $F_s = 485$ Hz, es decir, con un período de muestreo de $T_s = 2.06$ ms. Para cumplir con el teorema del muestreo de Nyquist-Shannon en frecuencia, se utiliza la ecuación (3.21):

$$F_s > 2F_{max} \equiv 2B \tag{3.21}$$

siendo F_s la frecuencia de muestreo y $F_{max} = B$ la frecuencia más alta contenida en la señal de entrada.

Según la ecuación (3.21), la frecuencia máxima de trabajo del sistema es de $F_{max} \equiv B = 242.5$ Hz. Para evitar *aliasing*, se emplea un filtro paso bajo de Respuesta al Impulso Finita o *Finite Impulse Response* (FIR), lineal e invariante en el tiempo o *linear time-invariant* (LTI). Una de las principales ventajas de este filtro es su fase lineal, lo que significa que el retardo de grupo es constante, por lo que no afecta al resultado final de la clasificación de texturas [98].

3.3.1. Especificaciones del filtro *antialiasing*

Es importante tener en cuenta que la implementación del filtro *antialiasing* en la FPGA debe ser lo suficientemente sencilla para no comprometer el consumo de recursos. Por lo tanto, se ha elegido un número de *taps* u orden del filtro de 16, lo que corresponde al

UNIVERSIDAD DE MÁLAGA

número de coeficientes del filtro. Las especificaciones en frecuencia consideradas para el diseño del filtro se pueden consultar en la tabla 3.1:

Tabla 3.1. Especificaciones en frecuencia del filtro *antialiasing*.

Especificación	Valor
Rizado en la banda de paso	$\delta_{p1} = \delta_{p2} = 0.1$ $1 + \delta_{p1} = 1.1$ $1 - \delta_{p2} = 0.9$
Rizado en la banda de rechazo	$\delta_s = 0.001$
Frecuencia de corte a $-3dB$	$f_c = 217$ Hz
Frecuencia límite de la banda de paso en el intervalo $-\pi < \omega_p \leq \pi$	$f_p = 182$ Hz $\omega_p = \frac{2\pi f_p}{F_s} = 0.75\pi$ rad/s
Frecuencia de comienzo de la banda de rechazo en el intervalo $-\pi < \omega_p \leq \pi$	$f_s = 242$ Hz $\omega_s = \frac{2\pi f_s}{F_s} = \pi$ rad/s

La ganancia ideal del filtro en la banda de paso es igual a la unidad, pero puede variar entre $(1 + \delta_{p1})$ y $(1 - \delta_{p2})$, mientras que en la banda eliminada la ganancia varía entre 0 y δ_s. Es común expresar estas ganancias máximas en la banda de paso y en la banda eliminada en decibelios. En la tabla 3.2 se muestran los valores correspondientes a las especificaciones de ganancia del filtro *antialiasing* utilizado.

Tabla 3.2. Especificaciones de ganancia en dB del filtro *antialiasing*.

Especificación en ganancia	Valor
Ganancia ideal de la banda de paso	$G_{\text{pass band}} = 20\log_{10}(1) = 0$ dB
Ganancia máxima de la banda de paso	$G_{\text{max pass band}} = 20\log_{10}(1.1) = 0.83$ dB
Ganancia mínima en el límite de la banda de paso	$G_{\text{min pass band}} = 20\log_{10}(0.9) = -0.92$ dB
Ganancia máxima de la banda de rechazo	$G_{\text{max reject band}} = 20\log_{10}(0.006) = -60$ dB

En el siguiente apartado, y con las especificaciones de frecuencia y ganancia presentadas en las tablas 3.1 y 3.2, respectivamente, se construye el filtro *antialiasing* utilizado en este proyecto de tesis.

3.3.2. Obtención de los coeficientes del filtro con la herramienta FDATool® de Matlab®

A raíz de las especificaciones anteriores, se puede diseñar el filtro *antialiasing* en Matlab®, a través de su herramienta de diseño de filtros FDATool®. La respuesta al impulso del filtro así como sus coeficientes aparecen en la figura 3.9.

UNIVERSIDAD DE MÁLAGA

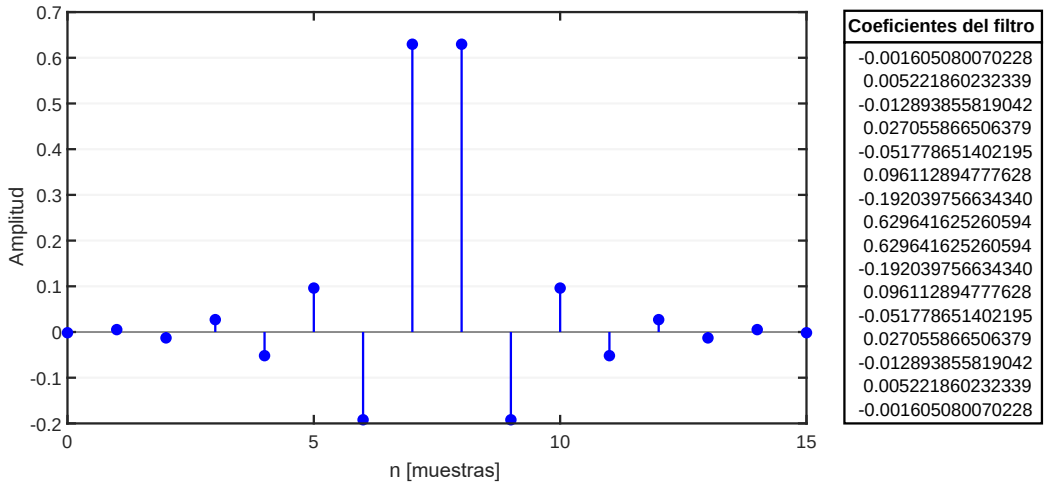


Figura 3.9. Respuesta al impulso del filtro *antialiasing* y coeficientes del filtro con precisión 'double' en Matlab®.

La implementación del filtro en hardware implica una cuantificación de los coeficientes, lo que puede generar una pérdida de precisión en la respuesta del filtro. En este caso, se ha adoptado una precisión en punto fijo tipo 'Fix16-frac15 bits'. Esto significa que cada coeficiente se ha cuantificado de forma que tendrá 15 bits de parte fraccional y 1 bit para la parte entera. En la figura 3.10 se muestra la respuesta en magnitud del filtro con precisión 'double' (Matlab®) y con precisión 'Fix16-frac15 bits'. También se incluye la respuesta en fase del filtro. Se puede comprobar que el sistema es de fase lineal. El retardo de grupo [98] de este filtro es 7.5, es decir, tiene un retraso de aproximadamente 8 muestras, o en tiempo 16.5 ms, teniendo en cuenta que $F_s = 485$ Hz.

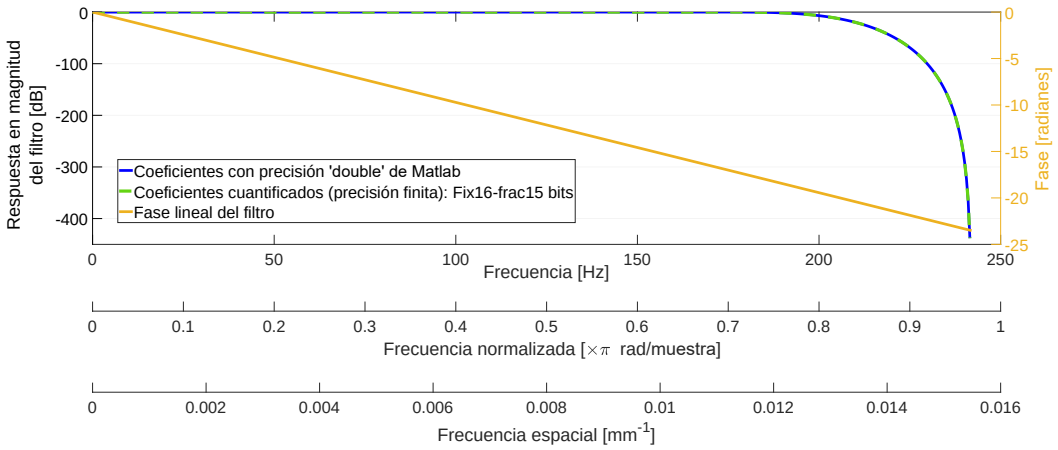


Figura 3.10. La figura muestra la superposición de la respuesta en magnitud del filtro *antialiasing* diseñado con precisión 'double' (Matlab®) y con precisión 'Fix16-frac15 bits', así como su respuesta en fase en radianes. Se han añadido los ejes equivalentes en π rad/muestras y frecuencia espacial mm^{-1}.

En la tabla 3.12 de la sección 3.8 se proporciona el consumo de recursos que este filtro requiere en la implementación del sistema completo. Por otro lado, en el siguiente

UNIVERSIDAD DE MÁLAGA

apartado se presenta la respuesta del filtro ante la exploración obtenida de una muestra ejemplo de textura.

Matlab® permite trabajar con funciones pre-compiladas que se comportan del mismo modo que algunos IP Core® que proporciona la librería de Xilinx®. Específicamente, para implementar el filtro en hardware, en Vivado® se hace uso del IP Core® FIR Compiler™. Su función análoga en Matlab® es '$fir_compiler_v7_2_bitacc$'. En este sentido, en la figura 3.11 se muestra la respuesta del filtro *antialiasing* frente a una señal de entrada de exploración de una textura con frecuencia espacial $1/1.3$ mm^{-1}.

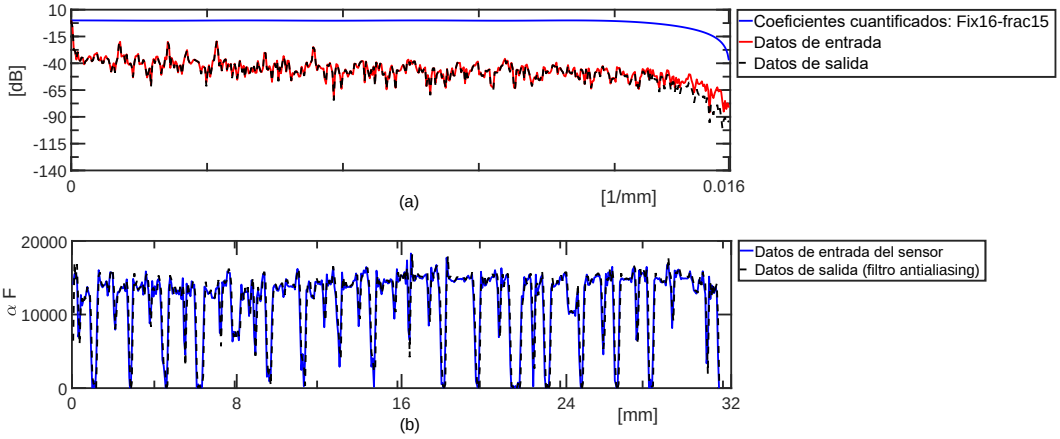


Figura 3.11. (a) Respuesta del filtro en frecuencia, y (b) en tiempo, para una muestra de exploración de una textura con frecuencia espacial $1/1.3$ mm^{-1}.

Se puede observar que el filtro es capaz de atenuar las frecuencias superiores a la frecuencia de Nyquist, y por tanto, cumplir con su función de *antialiasing*. Asimismo, se puede apreciar cómo la respuesta en magnitud del filtro con precisión 'double' (Matlab®) y precisión 'Fix16-frac15' prácticamente coinciden, validando el uso de una precisión de 16 bits en punto fijo para la implementación del filtro.

3.4. Caracterización de texturas

En esta sección se presenta una simulación sobre un conjunto ideal de texturas de distinta frecuencia espacial. También se describe el modelado de fuentes principales de ruido que afectan a la exploración de texturas. Por otro lado, con el objetivo de estudiar el comportamiento ante frecuencias espaciales muy próximas entre sí, se considera otro conjunto de texturas simuladas con modulación en frecuencia progresiva. Finalmente, a raíz de estas simulaciones, se selecciona un conjunto de texturas real para su exploración en tiempo real con sensores táctiles.

UNIVERSIDAD DE MÁLAGA

3.4.1. Simulación con diseños ideales de distintas texturas

La descripción e implementación del algoritmo de clasificación empleado se presenta más adelante, en la secciones 3.6 y 3.8. Sin embargo, con el objetivo de analizar el impacto en la clasificación, en esta sección se hace uso del clasificador implementado en esta tesis.

Previamente a escoger el conjunto definitivo de texturas para los experimentos de exploración, se realiza una simulación sencilla en Matlab® con modelos ideales de distintas frecuencias espaciales. Esta simulación se lleva a cabo tanto para el algoritmo de *sweep* como para la FFT.

Modelado de un conjunto preliminar de texturas

En la tabla 3.3 se presenta el diseño propuesto para doce texturas ideales con diferente frecuencia espacial. Estas texturas se van a utilizar para una simulación inicial en Matlab®.

Tabla 3.3. Conjunto preliminar de texturas ideales para llevar a cabo la simulación inicial

Textura	Modelada por	Definida a partir de la longitud de onda espacial (λ)
#1	Señal *chirp* $y = sin(\frac{20\pi x^2}{369} + \frac{x}{3})$, dónde x es distancia, de 0 a 12.3 mm	–
#2	Onda sinusoidal	$\lambda = 1.7$ mm
#3	Onda sinusoidal	$\lambda = 3.3$ mm
#4	Onda sinusoidal	$\lambda = 1.2$ mm
#5	Onda sinusoidal de 3 frecuencias: $y = sin(2\pi x/\lambda_1) + sin(2\pi x/\lambda_2) + sin(2\pi x/\lambda_3)$, donde x es distancia, de 0 a 21.5 mm	$\lambda_1 = 21.5$ mm $\lambda_2 = 10.8$ mm $\lambda_3 = 7.2$ mm
#6	Onda cuadrada	$\lambda = 1.6$ mm
#7	y = $\begin{cases} 1, si\ (0 \text{ mm} \leq x \leq 1 \text{ mm}, 2 \text{ mm} \leq x \leq 3 \text{ mm}) \\ 0, si\ (1 \text{ mm} \leq x \leq 2 \text{ mm}, 3 \text{ mm} \leq x \leq 5 \text{ mm}) \end{cases}$	–
#8	y = $\begin{cases} 1, si\ (0 \text{ mm} \leq x \leq 1 \text{ mm} \\ 0, si\ (1 \text{ mm} \leq x \leq 1.4 \text{ mm}) \end{cases}$	–
#9	Moduladora: Onda senoidal ($m_f = 7$) Portadora: Onda triangular ($m_a = 0.4$)	$\lambda_m = 7$ mm $\lambda_c = 1$ mm
#10	Moduladora: Onda senoidal ($m_f = 5$) Portadora: Onda triangular ($m_a = 0.4$)	$\lambda_m = 5$ mm $\lambda_c = 1$ mm
#11	Moduladora: Onda senoidal ($m_f = 7$) Portadora: Onda triangular ($m_a = 0.4$)	$\lambda_m = 21$ mm $\lambda_c = 3$ mm
#12	Moduladora: Suma de 2 ondas senoidales ($m_{f1} = 3, m_{f2} = 9$) Portadora: Onda triangular ($m_a = 0.4$)	$\lambda_{m1} = 6$ mm $\lambda_{m2} = 18$ mm $\lambda_c = 2$ mm

En la tabla 3.3, la textura #1 está definida como un barrido en frecuencia o *chirp* de 0 a 12.3 mm. Las señales #2, #3 y #4 son señales sinusoidales de una determinada frecuencia espacial. La textura #5 es una señal sinusoidal suma de 3 frecuencias espaciales. La textura #6 es una señal cuadrada. Las texturas #7 y #8 son señales definidas como funciones a trozos.

UNIVERSIDAD DE MÁLAGA

Las texturas #9, #10, #11 y #12 de la tabla 3.3 son señales diseñadas con modulación lineal en frecuencia. En particular, se utiliza una estrategia de control sinusoidal de modulación por ancho de pulso o *pulse-width modulation* (PWM). En este escenario, una señal sinusoidal de control (señal moduladora) de una determinada longitud de onda λ_m se utiliza para modular una señal repetitiva triangular (señal portadora) de longitud de onda λ_c. La señal de control puede variar su frecuencia, mientras que la señal portadora mantiene su frecuencia constante. Hay dos principales parámetros a considerar en este tipo de modulación [100] [101]:

- Relación de modulación en amplitud, m_a. Este parámetro se define como el cociente entre la amplitud de la señal moduladora y la señal portadora, de acuerdo a $m_a = \frac{A_m}{A_c}$, donde A_m y A_c son las amplitudes de la señal moduladora y portadora, respectivamente.
- Relación de modulación en frecuencia, m_f. Este parámetro se define como el cociente entre la longitud de onda de la señal moduladora y la señal portadora, $m_f = \frac{\lambda_m}{\lambda_c}$.

Modelado básico de la interacción entre el sensor y la textura

Para llevar a cabo una simulación simple pero aproximada a un comportamiento real de interacción entre los tácteles del sensor y la superficie de la textura, para cada una de las texturas en la tabla 3.3 utilizaremos una transformación sencilla. Si consideramos una textura de esta tabla como entrada x_{in}, la señal equivalente para hacer la simulación vendrá dada por una secuencia de valores entre 0 y 1 de acuerdo a (3.22):

$$y = \begin{cases} 1, si\ x_{in} > 0 \\ 0, si\ x_{in} \leq 0 \end{cases} \tag{3.22}$$

De esta forma, reducimos la complejidad de la interacción entre el dedo y la superficie a vectores cuya información indica si durante la exploración existe o no contacto entre el dedo y la superficie. Esto nos ayudará a llevar a cabo la simulación para tener una idea inicial del problema. No obstante, el comportamiento real, como veremos más adelante en la sección 3.5, es más complejo.

Modelado de fuentes de ruido

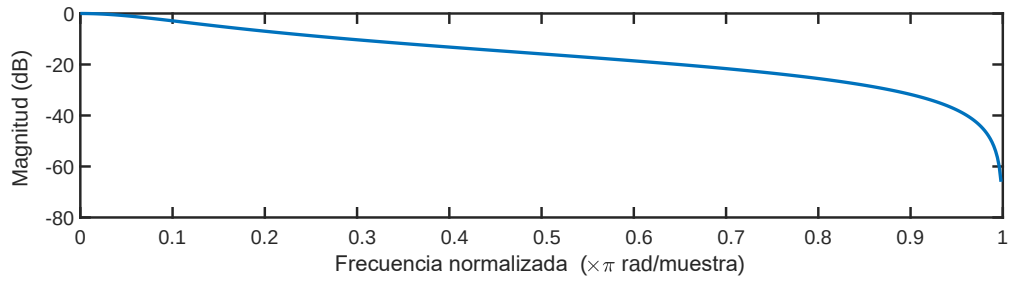
A la hora de realizar exploraciones sobre diferentes tipos de superficies, utilizando el *setup* de la figura 2.28 (sección 2.4 del capítulo 2), hay diversas fuentes de ruido que

UNIVERSIDAD DE MÁLAGA

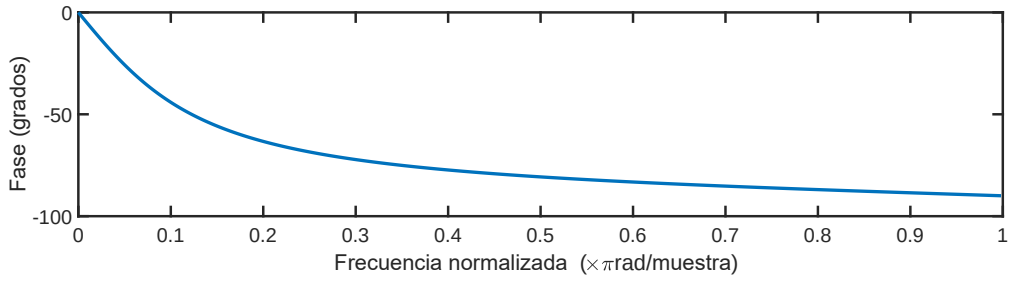
pueden afectar a la respuesta del sensor, y por tanto, a los datos que se adquieren. Las oscilaciones mecánicas durante el contacto por tacto activo entre el sensor y la superficie por un lado, y las que introduzcan los motores de la impresora por otro (ver capítulo 2, sección 2.4), hacen que en la toma de datos aparezcan picos o señales espúreas de amplitud pequeña. Esto último será analizado con mayor detalle en la sección 3.7 de este capítulo.

Para modelar las previsibles fuentes de ruido se han considerado tres posibles deformaciones que afectan a la señal táctil:

1. Deformación de la fase de la señal. Esto puede conseguirse mediante la aplicación de un filtro de Respuesta al Impulso Infinita o *Infinite Impulse Response* (IIR). Específicamente, para todas las señales de la tabla 3.3 se suma un ruido de fase que viene dado por la respuesta de un filtro de Butterworth de orden 1. Este tipo de filtros es de fase no lineal. Su respuesta en magnitud y fase se puede ver en la figura 3.12.



(a)



(b)

Figura 3.12. (a) Respuesta del filtro IIR de Butterworth diseñado para introducir deformación en fase. (b) Respuesta en fase para el mismo filtro.

2. Suma de ruido blanco (o gaussiano) a la señal. Utilizando el comando *awgn* de Matlab®, se ha añade de forma aleatoria un ruido gaussiano a la señal de $SNR = 15$ dB.

UNIVERSIDAD DE MÁLAGA

3. Deformación en amplitud de la señal. Para lograr esto aplicamos la técnica de modulación en amplitud a la señal con el comando $ammod$ de Matlab®.

Para llevar a cabo la simulación, se añaden las tres componentes de ruido descritas a los modelos de texturas de la tabla 3.3. Con fines ilustrativos, en la figura 3.13 se muestra el proceso de transformación de la textura $\#5$ en la tabla 3.3 para su uso en esta simulación.

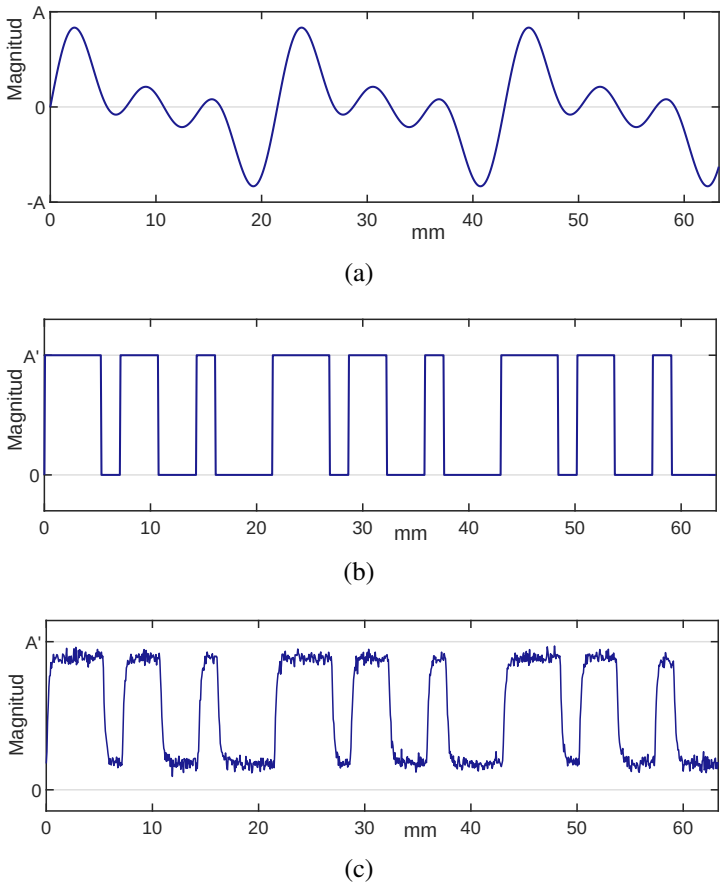


Figura 3.13. (a) Textura $\#5$ en la tabla 3.3, donde A es la amplitud máxima posible. (b) Textura $\#5$ una vez se le ha aplicado la transformación dada en (3.22), donde (0,A') es el rango de amplitud después de la transformación. (c) Textura $\#5$ final, una vez se le han introducido las tres componentes de ruido; de fase, ruido blanco y modulación en amplitud.

A continuación, se aplican los algoritmos de barrido y FFT en Matlab® (la implementación de estos algoritmos en Matlab® se detalla en la sección 3.7). El valor de la longitud del vector de salida para el barrido, expresión (3.3), es de $M = 1024$, y la FFT que se calcula es del mismo tamaño. Finalmente, se utiliza el algoritmo de entrenamiento y clasificación *k-means* (ver sección 3.6). Los resultados indican un $100\,\%$ de clasificación para el caso del barrido y un $82.7\,\%$ para la FFT. Las matrices de confusión obtenidas son las que se muestran en la figura 3.14, para el *sweep* y la FFT, respectivamente.

UNIVERSIDAD DE MÁLAGA

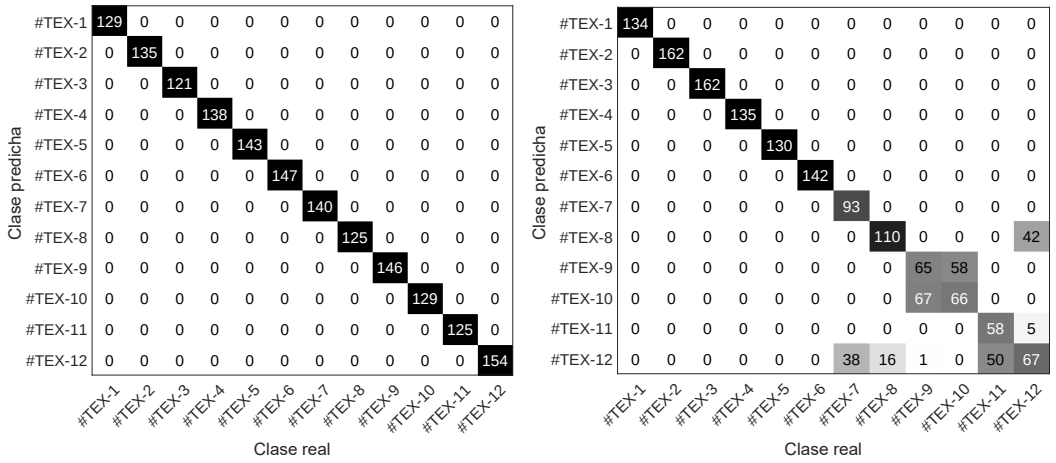


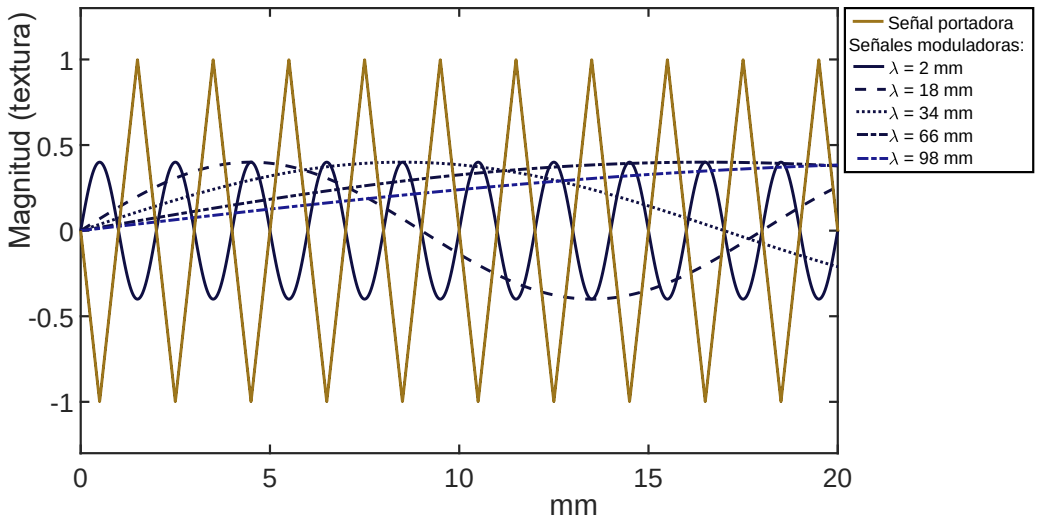
Figura 3.14. (a) Matriz de confusión obtenida para el caso del barrido. (b) Matriz de confusión obtenida para el caso de la FFT.

Como se ve en la figura 3.14, el porcentaje de clasificación para la FFT es más bajo que para el barrido. En esta simulación, las texturas escogidas están relativamente separadas entre sí. Por ello, la FFT podría ser más sensible a frecuencias introducidas por ruido, que producen armónicos de magnitud pequeña. Vemos en la matriz de confusión para la FFT, figura 3.14(b) que el clasificador se confunde más en las últimas 5 o 6 texturas, que son las que tienen frecuencia espacial más baja en la tabla 3.3. Por otro lado, el algoritmo de barrido no es tan sensible a ruidos de baja amplitud (como veremos en la sección 3.7), por lo que es capaz de clasificar correctamente las texturas en esta simulación.

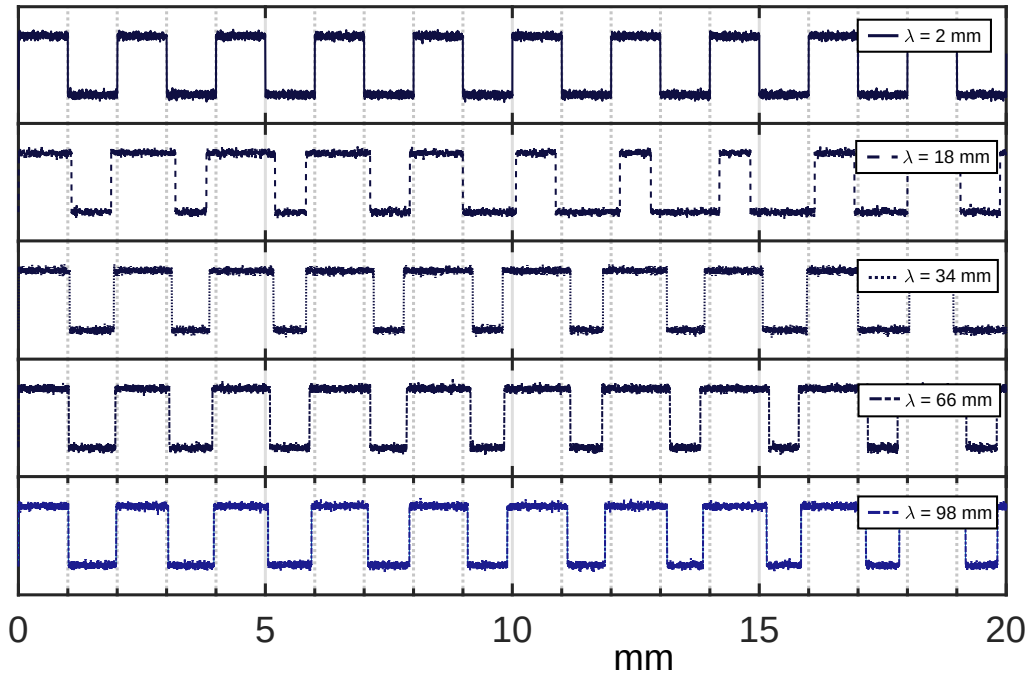
3.4.2. Simulación con una modulación en frecuencia progresiva

Aunque los resultados para la simulación previa pueden explicarse por la naturaleza de las texturas escogidas, es llamativo que la FFT arroje porcentajes inferiores al *sweep*. Frente a un escenario en el que a la señal de entrada se le añade algún tipo de ruido de baja amplitud, una posible hipótesis sugiere que la FFT tiene una mayor capacidad para distinguir entre texturas cuyas frecuencias espaciales se encuentran relativamente próximas entre sí, mientras que se ve más afectada en el caso de utilizar frecuencias espaciales más alejadas unas de otras, debido a la introducción de armónicos de alta frecuencia. Sin embargo, el algoritmo de *sweep* podría comportarse de forma contraria ante la misma situación. Si las frecuencias espaciales se encuentran muy próximas entre sí, el procedimiento podría clasificar dos casos distintos como el mismo. Para investigar esta hipótesis, se realiza una segunda simulación en la que se modifica progresivamente la frecuencia de una señal sinusoidal moduladora. En la figura 3.15 se muestra un ejemplo del procedimiento para esta simulación.

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura 3.15. (a) Ejemplo de modulación progresiva en frecuencia con SNR$=15$ dB. La longitud de onda de la señal portadora es $\lambda_c = 2$ mm para todos los casos y $m_a = 0.4$. Las longitudes de onda de la señal moduladora están indicadas en cada caso en la figura. (b) Resultado de la modulación, obtenida una vez se le añade el ruido. En todos los casos se simula una distancia de 63.3 mm con $M = 1024$, teniendo en cuenta que la velocidad de exploración se establece en $vel = 30$ mm/s y la frecuencia de muestreo es $F_s = 485$ Hz. En la figura se muestran los primeros 20 mm de la simulación.

UNIVERSIDAD DE MÁLAGA

Como se puede ver en la figura 3.15(b), a medida que la frecuencia espacial para la señal moduladora es más baja (o longitud de onda más alta), el resultado de la modulación es cada vez más similar. Para que el clasificador pueda actuar, se introduce en todos los casos ruido en la señal con una determinada relación señal/ruido o *signal-to-noise ratio* (SNR). La tabla 3.4 muestra los resultados obtenidos en la simulación para *sweep* y FFT, con diferentes relaciones señal/ruido y número de texturas. En todos los casos se simula una distancia de 63.3 mm con $M = 1024$, teniendo en cuenta que la velocidad de exploración se establece en $vel = 30$ mm/s y la frecuencia de muestreo es $F_s = 485$ Hz. La longitud de onda de la señal portadora es $\lambda_c = 2$ mm, con $m_a = 0.4$ para todos los escenarios. Por otra parte, se consideran longitudes de onda para la señal moduladora equiespaciadas en el rango $\lambda_m \in [2, 300]$ mm por el número de texturas considerado en cada caso.

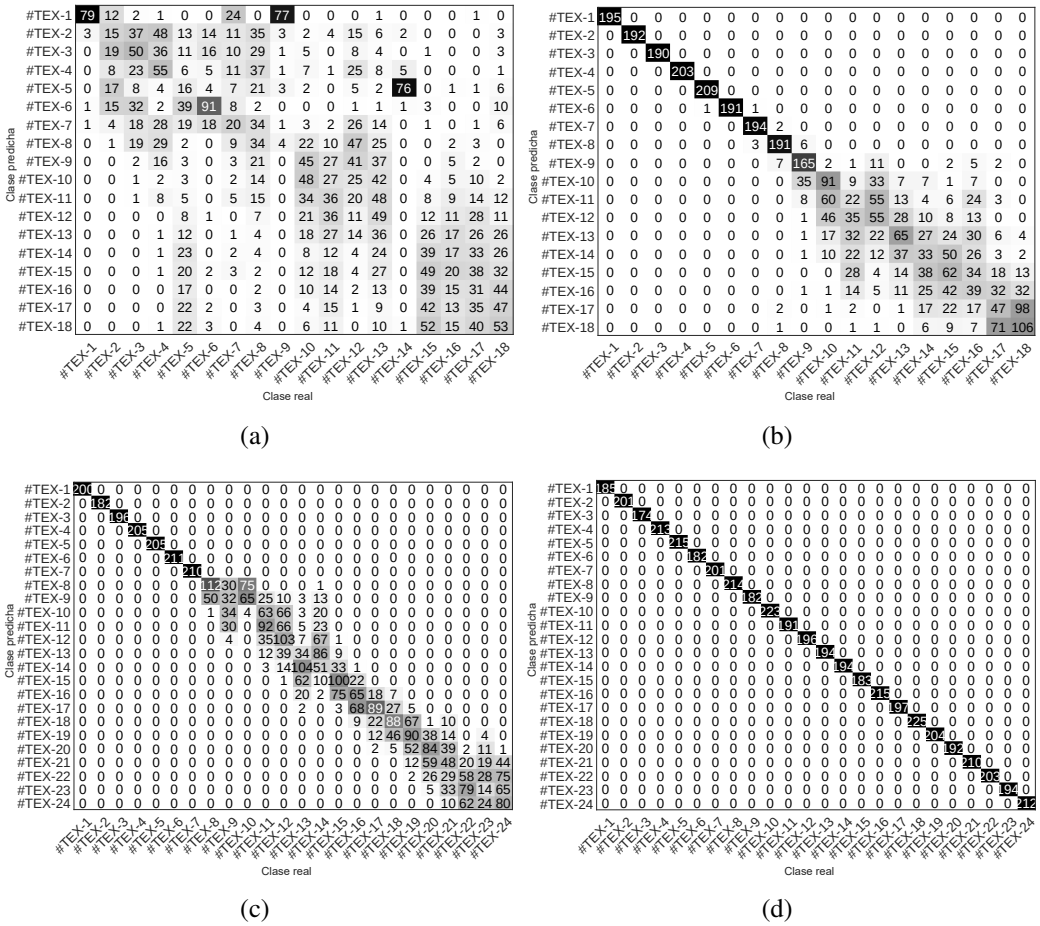
Tabla 3.4. Resultados de la simulación con modulación en frecuencia progresiva cuando se suma ruido blanco (se indica mediante la relación SNR), que indican el porcentaje de clasificación correcta en cada caso.

Relación señal/ruido (SNR)	Algoritmo	Número de texturas simulado		
		12	18	24
1 dB	Sweep	26.7 %	**17.9 %**	13.5 %
	FFT	72.3 %	**62.5 %**	55.5 %
5 dB	Sweep	29.3 %	24.3 %	20.8 %
	FFT	100 %	100 %	100 %
10 dB	Sweep	53.5 %	45 %	39.7 %
	FFT	100 %	100 %	100 %
15 dB	Sweep	71.8 %	59.6 %	**53.2 %**
	FFT	100 %	100 %	**100 %**

En la figura 3.16 se muestran las matrices de confusión para los casos de la simulación destacados en negrita en la tabla 3.4. Los resultados observados en las matrices de confusión de la figura 3.16 reafirman la hipótesis previamente planteada: la FFT demuestra mayor capacidad para distinguir entre frecuencias espaciales que se encuentran próximas entre sí. Como se aprecia en la esquina inferior derecha de las matrices de

UNIVERSIDAD DE MÁLAGA

confusión 3.16(a) y 3.16(b), cuando las frecuencias se acercan entre sí, la FFT incurre en menos errores de clasificación en comparación con el método de *sweep*.



(a) (b)

(c) (d)

Figura 3.16. (a) Matriz de confusión obtenida para barrido con 18 texturas y SNR=1 dB. (b) Matriz de confusión obtenida para FFT con 18 texturas y SNR=1 dB. (c) Matriz de confusión obtenida para barrido con 24 texturas y SNR=15 dB. (d) Matriz de confusión obtenida para FFT con 24 texturas y SNR=15 dB.

3.4.3. Selección del conjunto de texturas para experimentos

En este apartado se exponen las texturas que se han escogido para la realización de los experimentos de exploración llevados a cabo en esta tesis. En concreto, las cuatro texturas simples de la figura 3.6, se amplían a un conjunto mayor de texturas, más complejas en frecuencia y de distinto material y propiedades [39].

Rango de exploración de texturas

Como se indicó en la sección 3.2, la expresión (3.9) determina la frecuencia espacial máxima (o período espacial mínimo), dependiendo del valor de la velocidad de exploración v. Esto define la longitud mínima de exploración que se puede lograr. Por otro lado,

UNIVERSIDAD DE MÁLAGA

la frecuencia de corte de $f_c = 217$ Hz, del filtro *antialiasing* considerado en la sección 3.3, puede disminuir esta frecuencia espacial máxima (longitud mínima) alcanzable de acuerdo a (3.23).

$$\lambda_{minFilter} = \frac{1}{f_c} \cdot v \tag{3.23}$$

La tabla B.2 del apéndice B de tablas muestra el período espacial mínimo alcanzable en función de distintos valores de la velocidad de exploración v, teniendo en cuenta el efecto del filtro *antialiasing* ($\lambda_{minFilter}$) como sin él λ_{min}. El valor $\lambda_{minFilter}$ se calculó redondeando por exceso (3.23) al múltiplo más cercano del período de muestreo. Este último se calcula como $\frac{1}{F_s} \cdot v$, donde $F_s = 485$ Hz corresponde a la frecuencia de muestreo del sistema. Por otro lado, en la tabla B.3 del apéndice B de tablas, se muestra la longitud espacial máxima que se puede obtener de acuerdo a diferentes valores de M y v, teniendo en cuenta la expresión (3.8) de la sección 3.2. Para los experimentos llevados a cabo en esta tesis, los parámetros de exploración se han establecido en $M = 1024$ y $v = 30$ mm/s. Por consiguiente, de acuerdo a las tablas B.2 y B.3 del apéndice B, el rango de exploración viene dado por $\lambda_{minFilter} = 0.19$ mm (se recuerda que la longitud de onda mínima está afectada por el filtro *antialiasing*) y $\lambda_{max} = 63.3$ mm. Las frecuencias espaciales que caen fuera de este rango no se pueden identificar.

Ampliación a un conjunto complejo de texturas

Para tener mayor diversidad de texturas, el conjunto inicial de la figura 3.6, se extiende a 12 texturas que abarcan un espectro amplio de frecuencias espaciales. Esta cantidad de texturas está en consonancia con trabajos recientes que también han manejado números similares (por ejemplo, 7 texturas en [95] y [94], 9 texturas en [18], 12 en [96] o 17 en [102]). Por otro lado, en [95] y [96] los autores enfatizan la importancia de incluir texturas fabricadas con tela o materiales suaves. Estos tipos de texturas no sólo son comunes en la vida diaria, sino que también presentan patrones que resultan ser difíciles de distinguir sin la aplicación de métodos no destructivos. El conjunto ampliado completo de texturas viene dado por la imagen de la figura 3.17.

Las frecuencias espaciales de las 12 texturas de la figura 3.17 se encuentran dentro del rango de exploración dado por $\lambda_{minFilter}$ y λ_{max}. Se pueden dividir en 3 grupos dependiendo de sus características: texturas fabricadas mediante impresión 3D, texturas diseñadas a partir de la técnica de modulación en frecuencia (ver sección 3.4) y texturas de material textil. El primer grupo comprende seis muestras, cuatro de las cuales se imprimieron con Filaflex®, que es un material flexible. Las dos muestras restantes

UNIVERSIDAD DE MÁLAGA

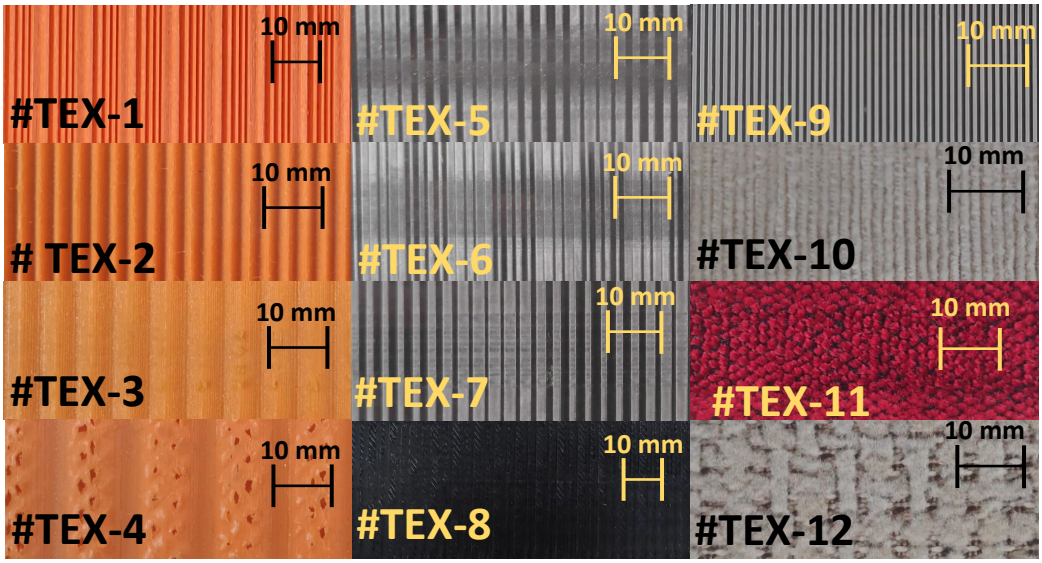


Figura 3.17. Fotografías de la parte superior de las 12 texturas utilizadas en los experimentos de exploración de texturas de esta tesis doctoral.

están fabricadas con PLA®, un polímero biodegradable derivado del maíz o la caña de azúcar, que es un material rígido [103]. La impresora utilizada es la Witbox®. El segundo grupo consta de tres texturas grabadas en una lámina de metacrilato. Por último, el tercer grupo consta de tres texturas de tela seleccionadas de un catálogo. Las principales características de estas texturas se presentan en la tabla 3.5.

Tabla 3.5. Principales propiedades de las doce texturas empleadas en este trabajo de tesis.

Textura	Material	Modelada por	Definida a partir de la longitud de onda espacial (λ)
#TEX-1	Filaflex	Señal *chirp* o de barrido en frecuencia $y = sin(\frac{20\pi x^2}{369} + \frac{x}{3})$, donde x es distancia, de 0 a 12.3 mm	–
#TEX-2	Filaflex	Onda sinusoidal	$\lambda = 3.25$ mm
#TEX-3	Filaflex	Onda sinusoidal	$\lambda = 6.5$ mm
#TEX-4	Filaflex	Onda sinusoidal	$\lambda = 13$ mm
#TEX-5	Metacrilato	Moduladora: Onda sinusoidal ($m_f = 7$) Portadora: Onda triangular ($m_a = 0.4$)	$\lambda_m = 21$ mm $\lambda_c = 3$ mm
#TEX-6	Metacrilato	Moduladora: Suma de dos senos ($m_{f1} = 3$, $m_{f2} = 9$) Portadora: Onda triangular ($m_a = 0.4$)	$\lambda_{m1} = 6$ mm $\lambda_{m2} = 18$ mm $\lambda_c = 2$ mm
#TEX-7	Metacrilato	Moduladora: Onda sinusoidal ($m_f = 13$) Portadora: Onda triangular ($m_a = 0.4$)	$\lambda_m = 39$ mm $\lambda_c = 3$ mm
#TEX-8	PLA	$y = \begin{cases} 1, si(0 \text{ mm} \leq x \leq 1 \text{ mm}, 2 \text{ mm} \leq x \leq 3 \text{ mm}) \\ 0, si(1 \text{ mm} \leq x \leq 2 \text{ mm}, 3 \text{ mm} \leq x \leq 5 \text{ mm}) \end{cases}$	–
#TEX-9	PLA	Onda cuadrada	$\lambda = 1.3$ mm
#TEX-10	Tela	Sin patrón espacial específico	$\lambda = \approx 1.5$ mm
#TEX-11	Tela	Sin patrón espacial específico	$\lambda = \approx 4.5$ mm
#TEX-12	Tela	Sin patrón espacial específico	$\lambda = \approx 8$ mm

En el diseño de las texturas presentadas en la tabla 3.5, es de crucial importancia el análisis de las frecuencias espaciales inherentes a cada una. A modo de ejemplo, la figura 3.18 exhibe el contenido armónico de las texturas #TEX-2, #TEX-3 y #TEX-4.

UNIVERSIDAD DE MÁLAGA

Por otro lado, la figura A.5 muestra el contenido armónico para la textura #TEX-7. Estas figuras muestran el espectro ideal de las texturas de acuerdo a su diseño original. En el apéndice A de figuras se muestra el espectro para las texturas #TEX-1, #TEX-5, #TEX-6, #TEX-8 y #TEX-9. Las texturas #TEX-10, #TEX-11 y #TEX-12 están definidas a partir de períodos espaciales aproximados de 1.5 mm, 4.5 mm y 8 mm (medidos en laboratorio), pero debido a su naturaleza no se puede conocer de antemano su espectro ideal. Cabe recordar también que la interacción entre el sensor y la superficie produce un contenido espectral más complejo, cuyos resultados se mostrarán en la subsección 3.4.4.

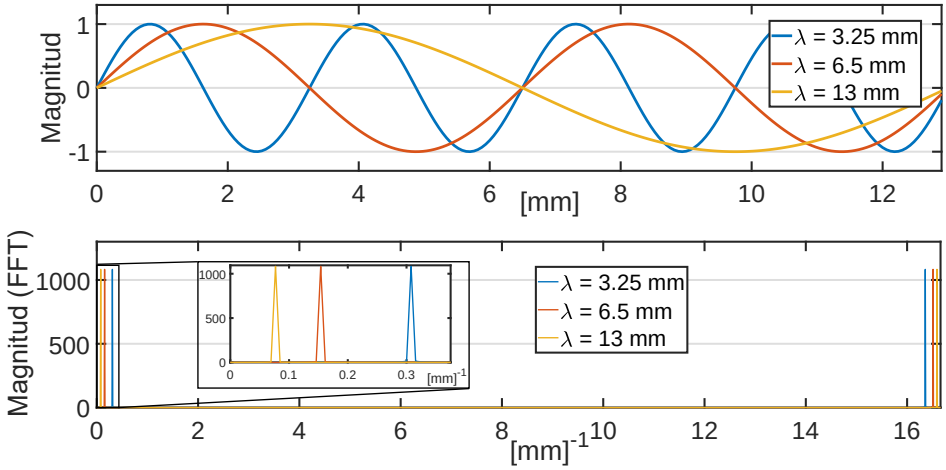


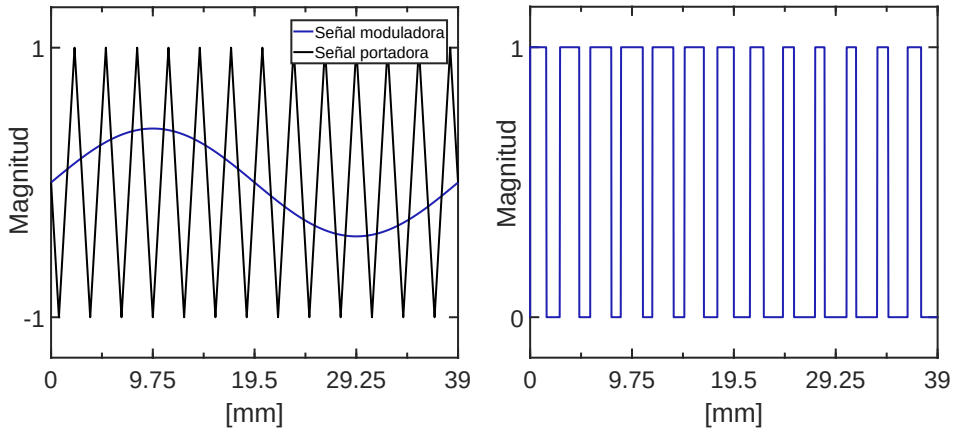
Figura 3.18. Texturas #TEX-2, #TEX-3 y #TEX-4 de la tabla 3.5, se representa un período de la textura #TEX-6 para las tres (parte superior de la figura) y espectro en frecuencia para con un *zoom* para los armónicos principales (parte inferior de la figura).

3.4.4. Exploración de texturas

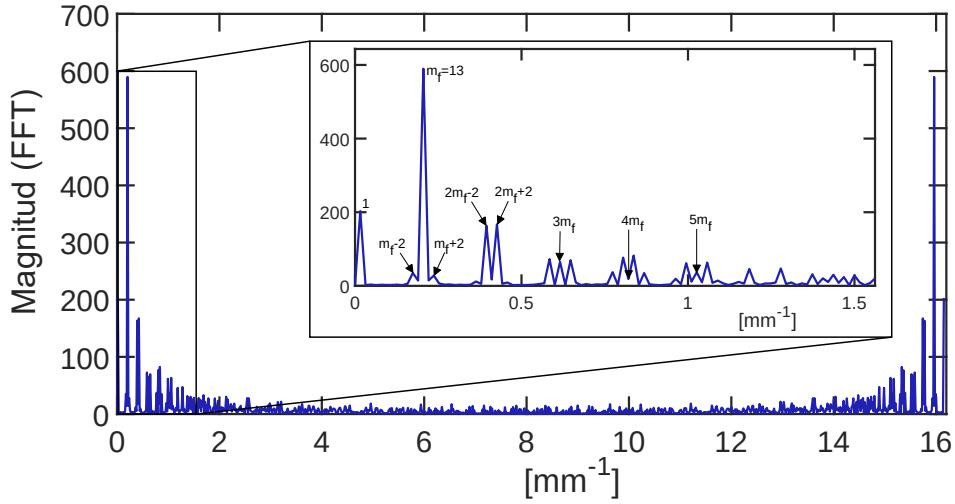
Los datos de las texturas se obtuvieron a partir del *setup* experimental representado en la figura 2.28 del capítulo 2. En la fotografía de la figura 3.20 se muestra dicho *setup* en el proceso de exploración de una textura.

Más concretamente, la figura 3.21 muestra la secuencia experimental para obtener datos de las texturas. En particular, la figura 3.21(a) muestra la posición inicial del dedo con el sensor táctil y la figura 3.21(b) muestra una captura del desplazamiento del sensor sobre la superficie hasta que alcanza la posición final, figura 3.21(c). Esta secuencia se ejecutó 200 veces por textura a una velocidad constante de $v = 30$ mm/s. Durante el desarrollo de este procedimiento, se obtuvieron datos en tiempo real que se registraron en archivos de texto utilizando una aplicación desarrollada en LabVIEW® (ver diagrama de flujo de la figura 2.27 en el capítulo 2). Cada exploración mostrada en la figura 3.21 proporcionó un vector de 2048 muestras que llamaremos *frame*. Cada una de estas muestras es una palabra de 16 bits de longitud. Puesto que se trabajó con 12 texturas, el volumen total de *frames* registrados es 12 texturas $\cdot 200$ *frames*/textura $= 2400$ *frames*,

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura 3.19. (a) Modulación lineal en frecuencia para la textura #TEX-7 de la tabla 3.5. Se representa un período espacial de la textura. (b) Espectro en frecuencia para dicha modulación con un *zoom* para los armónicos principales, que se muestran como múltiplos de $m_f = 13$.

lo cual equivale a 9.4 MB de datos. La figura 3.22 muestra un conjunto ejemplo de *frames* para cada textura, tal y como se obtuvieron tras la aplicación del filtro *antialiasing* comentado en la sección 3.3.

3.5. Extracción de características

Tal y como se comentó en la sección de introducción 3.1, la información obtenida a través del procedimiento descrito en la subsección 3.4.4 se debe procesar para reducir la cantidad de datos que se transmiten y almacenan. Esta etapa de procesado permite simplificar el funcionamiento del clasificador.

UNIVERSIDAD DE MÁLAGA

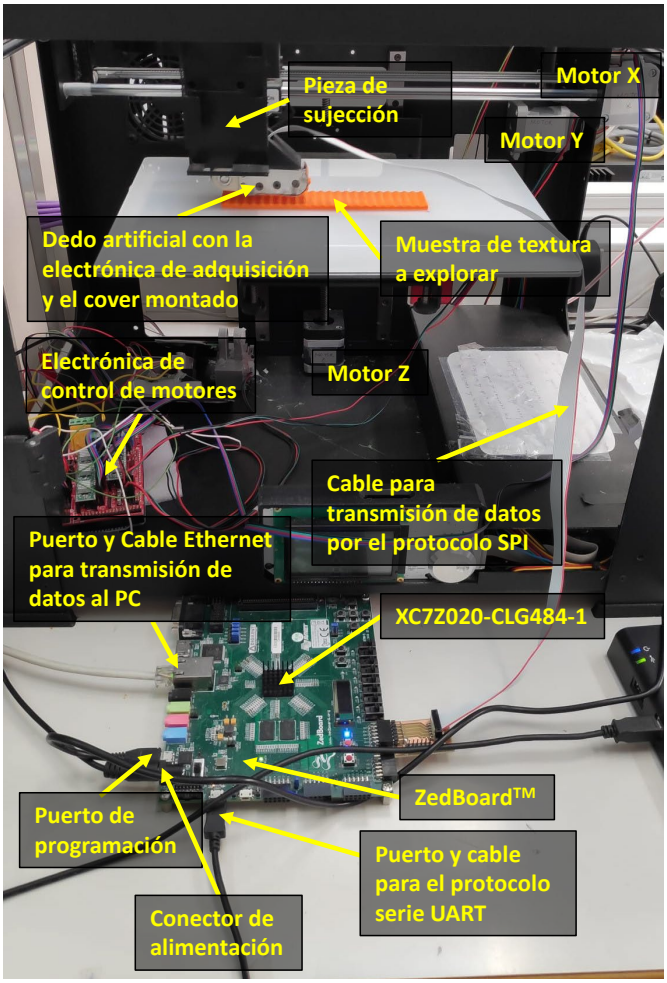


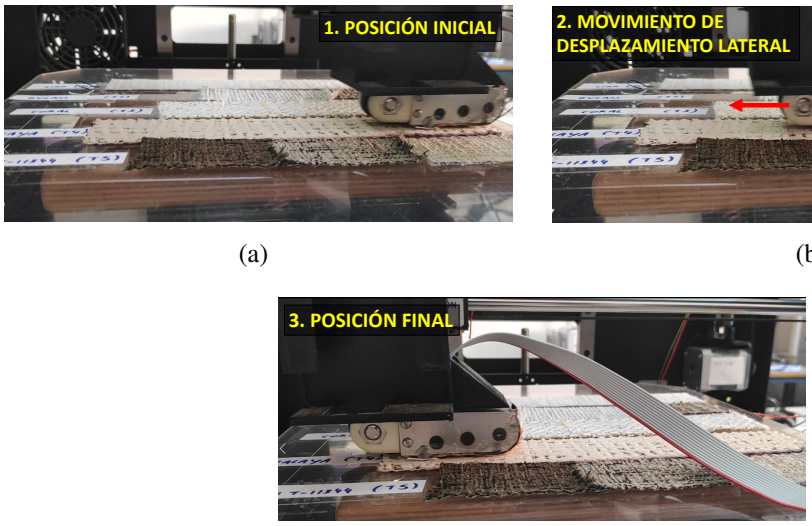
Figura 3.20. Fotografía del *setup* experimental con las diferentes partes usadas para explorar las texturas. La placa de desarrollo ZedBoard™ (descrita en la sección 3.8) con sus puertos de entrada/salida y el núcleo basado en un SoC Zynq-7000® también se indican en la imagen.

Como se comentó en la sección 2.5 del capítulo 2, la implementación de los algoritmos de extracción de características se lleva a cabo en un SoC, un tipo de dispositivo que incorpora una parte hardware basada en FPGA y una parte consistente en un microprocesador. A continuación, se examinan con mayor profundidad las opciones consideradas para la implementación del algoritmo de barrido y la FFT.

3.5.1. Algoritmo de barrido o *sweep*

En esta tesis se han explorado distintas posibilidades a partir de la aplicación del *sweep* sobre los datos táctiles del sensor. Con el propósito de llevar a cabo una fase de entrenamiento con restricciones de hardware similares a las que se tendrían en un experimento real, los datos obtenidos con el sensor táctil (de los que un ejemplo se mostraba en la figura 3.22), se procesaron con la herramienta de Co-Simulación de *High-Level Synthesis* (HLS) de Vivado®. HLS permite escribir código en C, C++ o System C, y a

UNIVERSIDAD DE MÁLAGA



(a) (b)

(c)

Figura 3.21. Secuencia de exploración. (a) Posición inicial. (b) Movimiento de desplazamiento lateral a $v = 30$ mm/s. (c) Posición final.

través de la aplicación de directivas de optimización, sintetizarlo para obtener la descripción *register-transfer level* (RTL) más próxima al sistema real (en el repositorio público *https://github.com/raulLkRb/thesis.git* se puede consultar un código genérico para estas simulaciones). La figura 3.23 muestra el resultado de la aplicación del algoritmo de barrido sobre las 12 texturas de la figura 3.17.

A parte de estas 12 texturas, es interesante comprobar qué ocurre para una textura de control basada en una superficie pulida. La figura 3.24 muestra el resultado cuando se aplica V (3.3) sobre los datos de una exploración sobre una lámina lisa de metacrilato. En este caso, como cabe esperar, la aplicación del barrido no proporciona ningún mínimo identificable.

Es importante destacar que, conforme a las expresiones proporcionadas en (3.5) y (3.6), sección 3.2.1, solamente se requiere la ubicación de los extremos para identificar la frecuencia espacial principal. Por tanto, la dimensión del vector de características puede reducirse drásticamente a M bits. Esto se logra marcando la posición de las singularidades en V (3.3) con el valor lógico '1' en la componente correspondiente del vector de características. En la figura 3.25 se ilustra este proceso para los datos de la textura #TEX-1 (ver figura 3.23), cuando se considera la localización de los mínimos en el vector de características. Como se discutirá en la sección de discusión de resultados 3.9, este subconjunto de componentes del vector de características permite alcanzar porcentajes satisfactorios de clasificación cuando se transmite al clasificador.

En esta tesis, además de considerar la ubicación de los mínimos, se han investigado

UNIVERSIDAD DE MÁLAGA

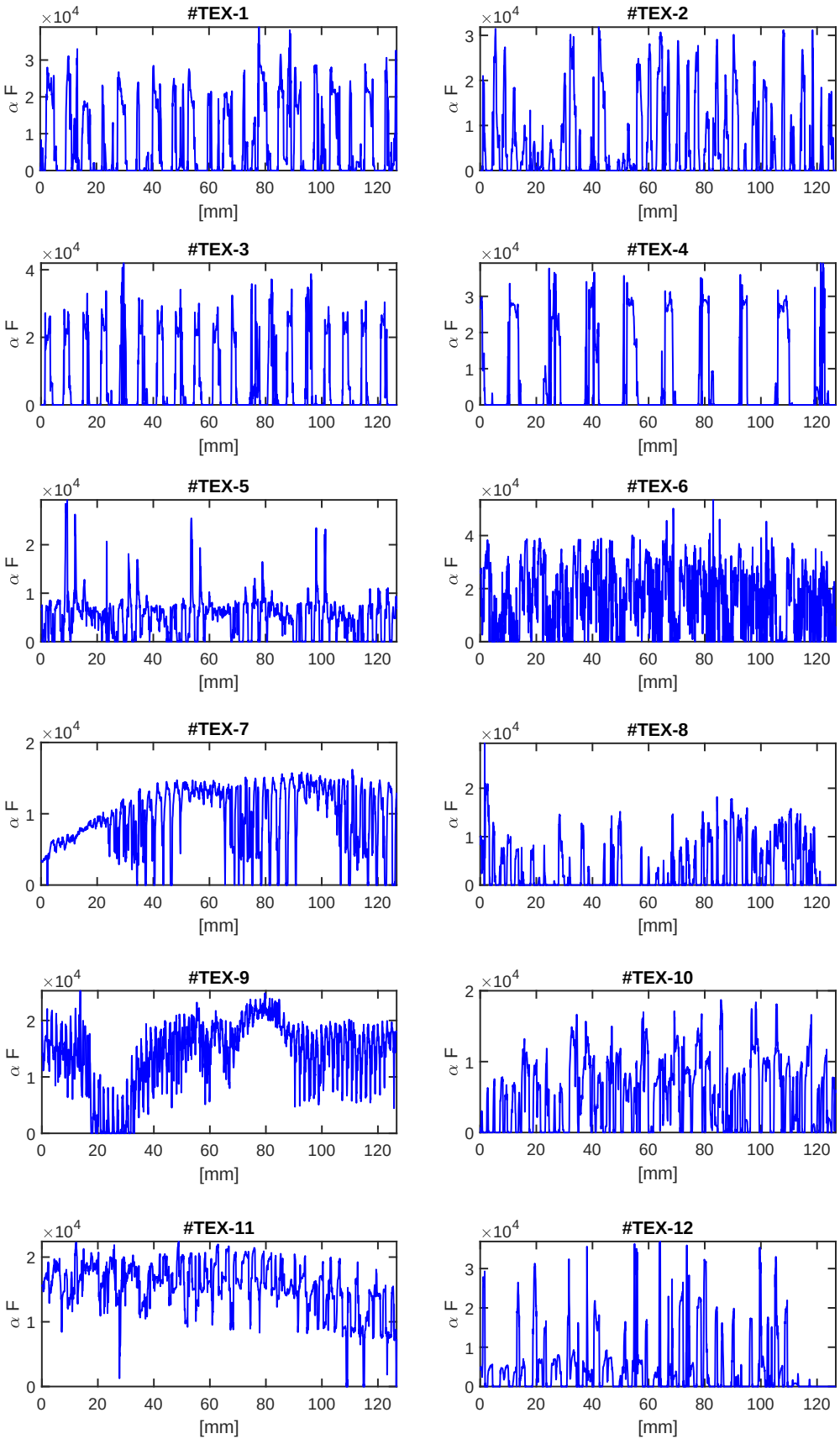


Figura 3.22. Salida digital del sensor táctil para cada una de las texturas del conjunto de la figura 3.17, cuando se exploran a una velocidad de $v = 30$ mm/s.

UNIVERSIDAD DE MÁLAGA

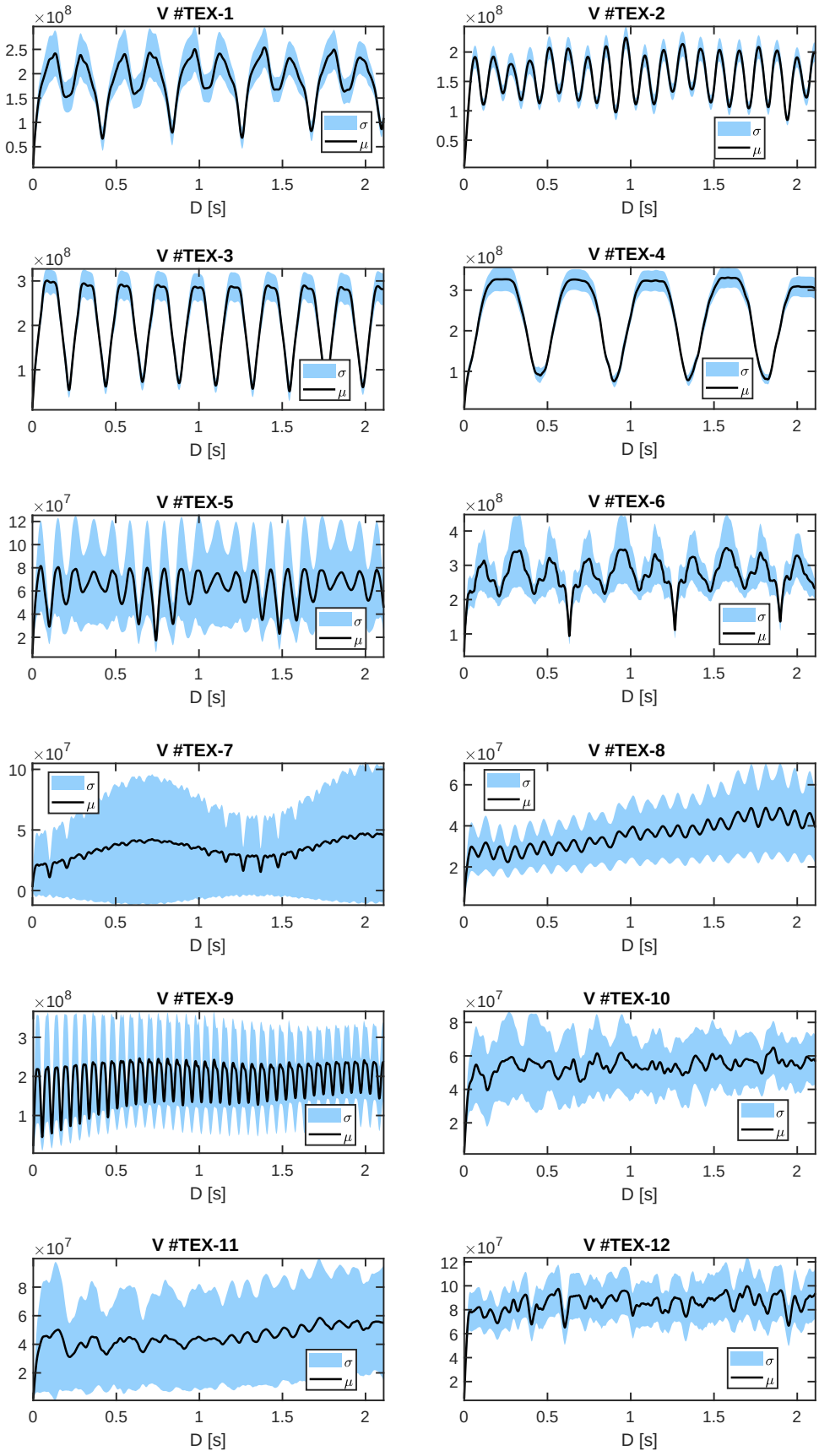
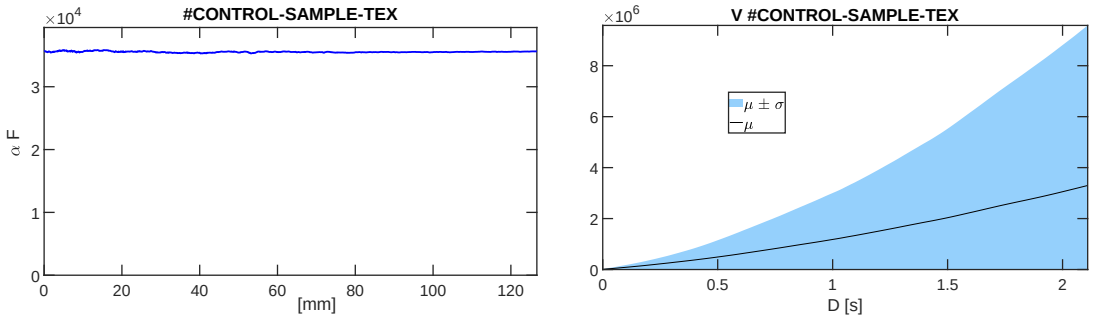


Figura 3.23. Salida de V (3.3) para $1 \leq m \leq M$ ($D = m \cdot T_s$) para el conjunto de texturas de la figura 3.17.

UNIVERSIDAD DE MÁLAGA



(a) (b)

Figura 3.24. (a) Salida digital del sensor táctil tomada a $v = 30$ mm/s para la superficie de control en metacrilato liso. (b) Salida de V (3.3) con $1 \leq m \leq M$ ($D = m \cdot T_s$) para esta superficie.

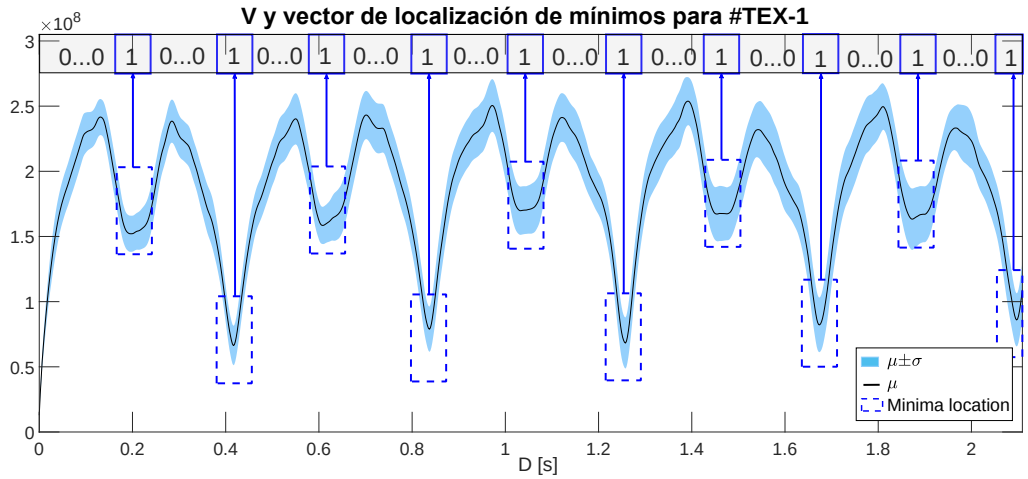


Figura 3.25. Salida de V (3.3) para $1 \leq m \leq M$ ($D = m \cdot T_s$), para la textura #TEX-1 del conjunto de texturas de la figura 3.17. Adicionalmente, una representación del vector de la localización de los mínimos se muestra en la parte superior de la figura.

otras dos estrategias. Una de las opciones consiste en aplicar un proceso de diezmado a V (3.3) antes de localizar los mínimos. Esta acción permite reducir aún más el tamaño del vector de características resultante. Además, debido a las posibles oscilaciones causadas por ruido y otros factores ajenos al algoritmo (sección 2.4 del capítulo 2), aplicar un filtro de media móvil sobre el resultado V (3.3) antes de implementar cualquier otra estrategia podría evitar que se detecten mínimos producidos por el ruido. La aplicación individual o conjunta de estas configuraciones se discute en la sección 3.7 de este capítulo.

3.5.2. *Fast Fourier Transform* (FFT)

En este trabajo también se implementa la transformada rápida de Fourier (FFT) para comparar su rendimiento con la propuesta del algoritmo de barrido, dado que la FFT es el algoritmo usado habitualmente para la extracción de características. La figura 3.26 muestra la salida de la FFT obtenida para el conjunto de texturas utilizado en este estudio.

UNIVERSIDAD DE MÁLAGA

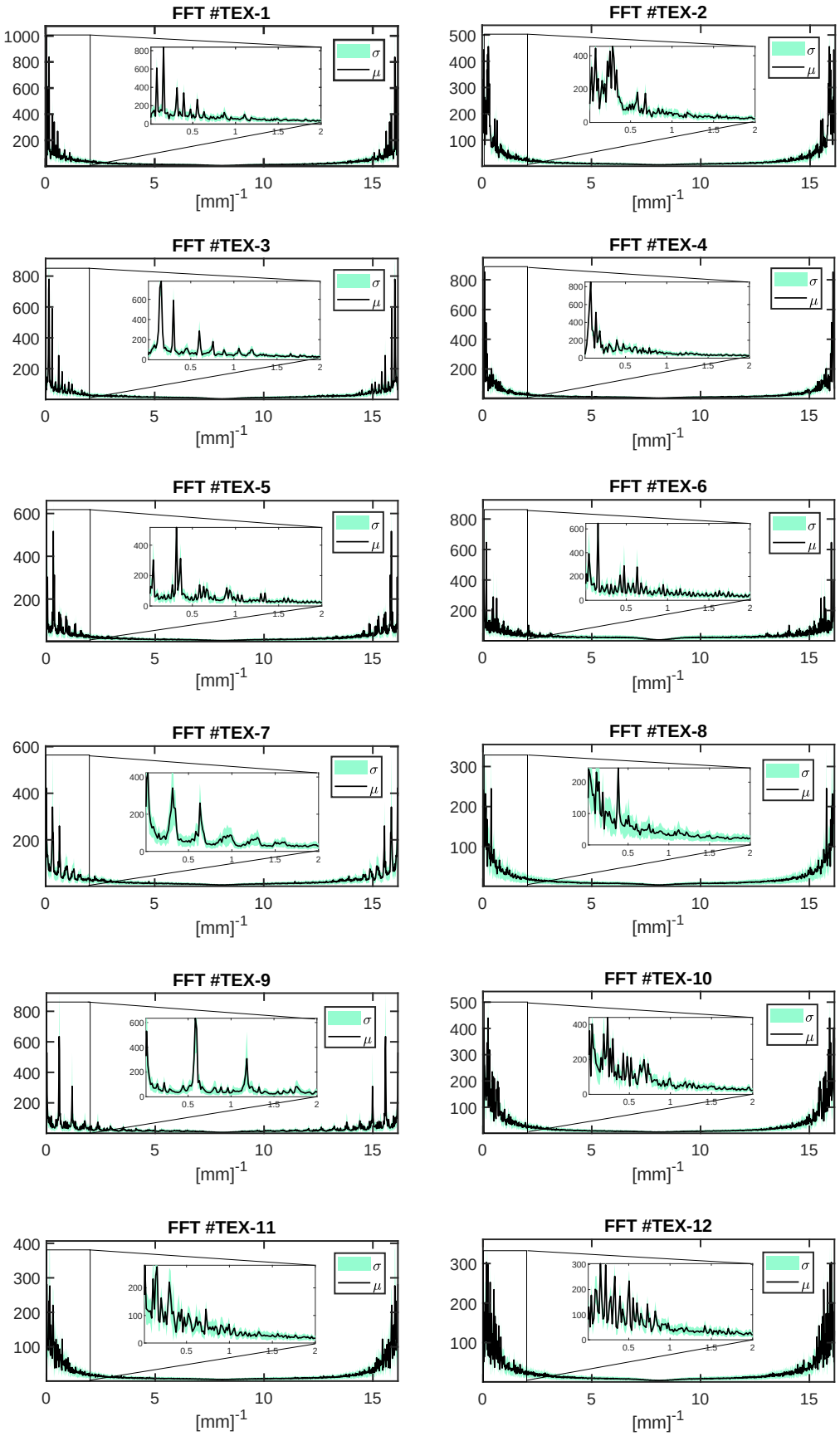


Figura 3.26. Salida de la aplicación de la FFT obtenida para el conjunto de texturas de la figura 3.17. Se ha hecho un *zoom* en los primeros valores del espectro.

UNIVERSIDAD DE MÁLAGA

Los datos de la figura 3.26 se han obtenido mediante la aplicación de las funciones *MATLAB Executable* (MEX) de Matlab®. El modelo MEX para la FFT produce resultados idénticos a nivel de precisión de bits respecto a los que se obtendrían utilizando el IP Core proporcionado con la librería de Xilinx®.

3.5.3. Búsqueda de los índices del espectro para el algoritmo de Goertzel

En el caso de la propuesta de Goertzel (ver subsección 3.2.5), teniendo en cuenta la naturaleza frecuencial de las texturas que se consideran en la figura 3.17, una posible aproximación es la extracción de aquellas frecuencias del espectro donde exista una mayor concentración de armónicos. De esta forma, estaríamos considerando las zonas del espectro que contienen la mayor parte de la información. En este sentido, si se examina la figura 3.26, que muestra la respuesta en frecuencia para estas texturas, se puede ver que la mayoría de información para las 12 texturas está en aproximadamente $1/4$ de la mitad del espectro. Por lo tanto, una posible estrategia es concentrar la búsqueda de los índices del algoritmo en dicha zona del espectro.

Las frecuencias consideradas no tienen por qué estar uniformemente distribuidas en el espectro. En esta tesis, para decidir qué índices de Goertzel se calculan, se propone la expresión no lineal dada en (3.24):

$$\frac{\pi}{2} + atan\left(\frac{y - N/2}{p}\right) \tag{3.24}$$

donde p es un factor para controlar la densidad de índices que quedan al principio o final del espectro en frecuencia. El valor $N/2$, que aparece restando en el numerador, hace que y sea un valor entero definido en el rango $(0, N)$, ya que la función arco-tangente se extiende desde $-\infty$ hasta ∞. Si se calcula el máximo para normalizar la expresión (3.24), se tiene que:

$$max\left(\frac{\pi}{2} + atan\left(\frac{y - N/2}{p}\right)\right) = \pi \tag{3.25}$$

Finalmente, para obtener el valor del índice en frecuencia k_{indice}, se puede escribir:

$$k_{indice} = N \cdot \frac{\left(\frac{\pi}{2} + atan\left(\frac{y-N/2}{p}\right)\right)}{\pi} \tag{3.26}$$

UNIVERSIDAD DE MÁLAGA

El uso de la función arco-tangente es apropiada para nuestro caso, ya que se puede controlar en qué zonas se quiere tener una mayor densidad de índices. El factor $\frac{\pi}{2}$ que aparece sumando en (3.24) hace que el rango de aplicación sea mayor que 0. Normalizando la expresión (3.24) y multiplicando por N, se consigue que el valor de los índices en (3.26) quede entre 0 y N.

En esta tesis, se viene considerando el caso en que el tamaño del vector de salida M es 1024. Para hacer un análisis equitativo con el barrido y FFT, el tamaño considerado para el cálculo del algoritmo de Goertzel es $N = 1024$. Por tanto, si se consideran las expresiones dadas en (3.18) y (3.20), se tiene que:

$$\begin{aligned} K < 2 \cdot log_2(1024) \rightarrow K < 20 \\ K < \frac{4}{7} \cdot 1024 \rightarrow K < 585 \end{aligned} \tag{3.27}$$

Por tanto, para que el cálculo del algoritmo de Goertzel con $N = 1024$ sea más ventajoso con respecto al de la FFT, el tamaño del vector de índices tiene que cumplir $K < 20$. Para hacer los cálculos más cómodos en una implementación en FPGA, se han considerado potencias de 2. Los valores más cercanos a la condición dada por $K < 20$ que se han usado son $K = \{4, 8, 16\}$. Considerando el espectro completo en el cálculo de los índices de Goertzel, se han analizado las siguientes posibilidades:

1. Separación de los índices de forma equiespaciada. En este caso, se tienen K valores separados por la misma distancia, entre 0 y N.
2. Separación de los índices de acuerdo a la expresión (3.26), con un parámetro de control $p = 1$.
3. Separación de los índices de acuerdo a la expresión (3.26), con un parámetro de control $p = 10$.
4. Separación de los índices de acuerdo a la expresión (3.26), con un parámetro de control $p = 25$.
5. Separación de los índices de acuerdo a la expresión (3.26), con un parámetro de control $p = 50$.
6. Separación de los índices de acuerdo a la expresión (3.26), con un parámetro de control $p = 100$.

UNIVERSIDAD DE MÁLAGA

Para los 6 casos anteriores, y con $K = 16$, en la figura 3.27 se representa el valor de los índices en frecuencia. El eje de abscisas de cada figura es el valor de y en la expresión (3.26), que se encuentra en el rango $(0, N)$, y el eje de ordenadas muestra el valor del índice k_{indice} en la expresión (3.26).

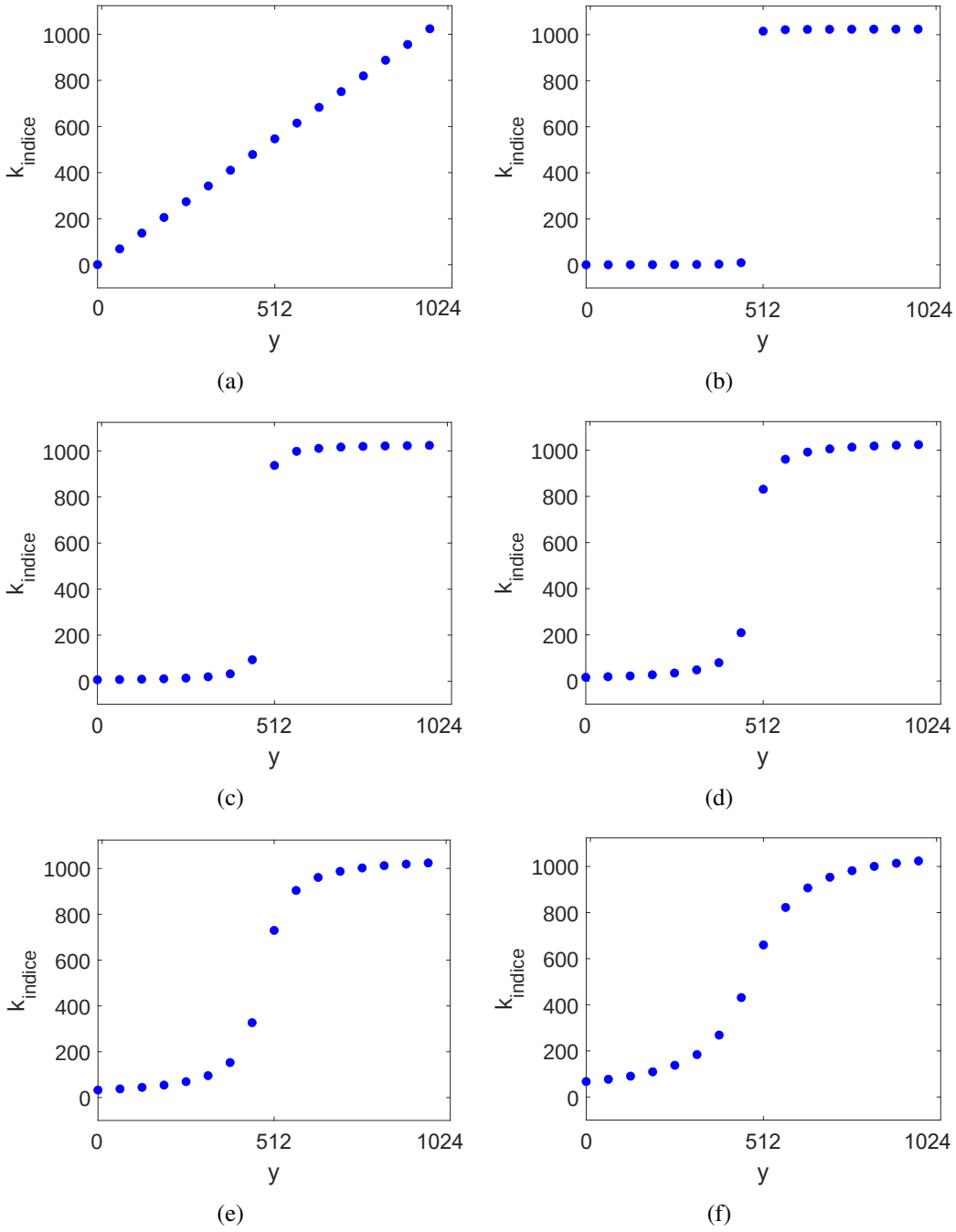


Figura 3.27. Representación del valor de los índices de Goertzel, para $K = 16$, considerando el cálculo de los mismos de forma: (a) equiespaciada, (b) con $p = 1$, (c) con $p = 10$, (d) con $p = 25$, (e) con $p = 50$ y (f) con $p = 100$.

UNIVERSIDAD DE MÁLAGA

En esta subsección se ha propuesto la aplicación de un algoritmo alternativo en frecuencia, capaz de obtener las K primeras componentes de la DFT de la señal táctil obtenida en la exploración de diferentes texturas. Para la selección de los índices de Goertzel, se propone la utilización de la expresión no lineal dada en (3.26), que concentra las frecuencias de interés en una determinada zona del espectro a partir de un parámetro de control p. Más adelante en el capítulo, en la subsección 3.7.3, se presentan los resultados de clasificación obtenidos para este algoritmo.

3.6. Algoritmo de entrenamiento y clasificación *k-means*

Para llevar a cabo la clasificación de las texturas, se entrena un algoritmo no supervisado de propósito general *k-means*. A diferencia de otras técnicas de aprendizaje automático o *machine-learning*, como *Support Vector Machines* (SVM) o *K-Nearest Neighbors* (KNN), que requieren almacenar una gran cantidad de datos para realizar tareas en tiempo real, la principal ventaja del *k-means* es su simplicidad y velocidad al converger a las diferentes clases. El proceso de entrenamiento del algoritmo *k-means* involucra la selección inicial de los centroides, que son vectores que representan a cada clase. Esta selección puede ser totalmente aleatoria o utilizando el método *k-means++* [104]. En la figura 3.28 se muestra un esquema que ilustra el procedimiento seguido para obtener los conjuntos de entrenamiento y test utilizados en los algoritmos de barrido y FFT.

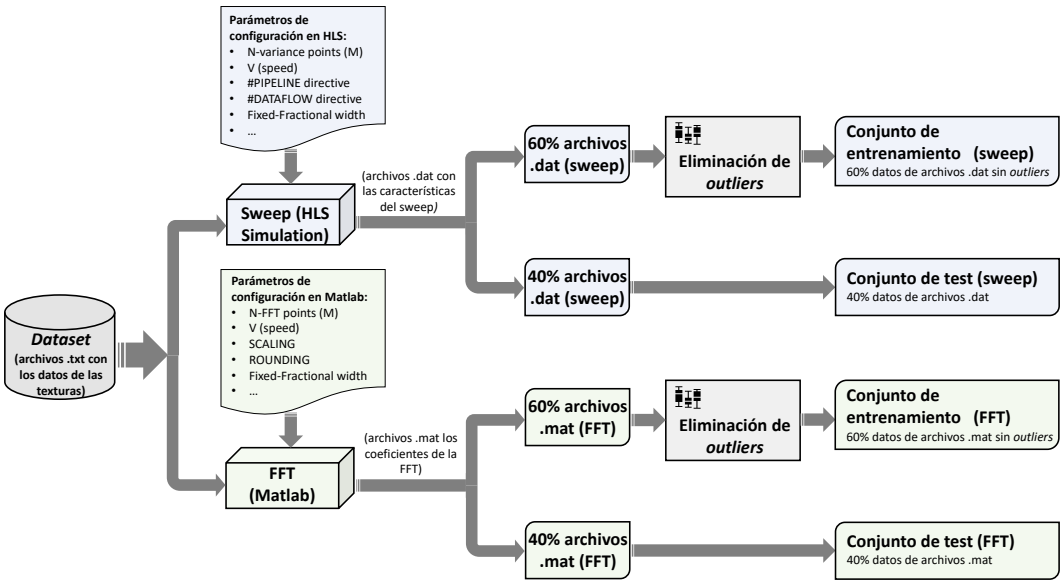


Figura 3.28. Procedimiento para la obtención de los conjuntos de entrenamiento y test, para los algoritmos de *sweep* y FFT, respectivamente.

Durante la exploración, hay momentos en los que el sensor del dedo se encuentra por encima de la superficie a explorar, antes del inicio y después del final del contacto (ver figura 3.21 en la sección 3.4). Por lo tanto, los datos obtenidos durante esos momentos no

UNIVERSIDAD DE MÁLAGA

son válidos. Para evitar esto, se sincronizan los instantes de inicio y final de exploración utilizando una aplicación en LabVIEW® junto con la tarjeta de adquisición 6009 de National Instruments™ (ver apéndice A de figuras). De esta manera, sólo se guarda en el archivo de texto la información válida durante el tiempo de exploración.

Por otro lado, en el instante de inicio, el sensor topa con la superficie a explorar, por lo que puede introducir oscilaciones mecánicas no deseadas debidas a dicho contacto inicial. Para solventar esto, se representan en Matlab® los datos obtenidos, y de forma manual se elige el inicio de contacto para cada una de las doce texturas, trabajando con los $2 \cdot M = 2 \cdot 1024 = 2048$ (para el *sweep*) o $M = 1024$ (para la FFT) valores subsiguientes de la exploración. Dado que se conoce la velocidad de exploración y la frecuencia de muestreo del sistema, una vez elegido este punto, se repite la operación de forma automática para cada una de las 200 iteraciones siguientes (consultar apartado 3.4.4, sección 3.4).

Finalmente, una vez se obtienen todos los datos, se representan las 200 tramas de cada textura en Matlab®. Hay momentos en que no se registra lectura y por tanto falla la adquisición de la señal. Con el objetivo de eliminar estas tramas dudosas, se realizó una inspección visual y se descartaron aquellas que presentaban anomalías o datos inconsistentes en la exploración de una textura específica. Se muestra un ejemplo en la figura 3.29.

Después de eliminar estas tramas, el conjunto final de texturas se reduce de 2400 a 2194 *frames*. Aunque el número de iteraciones por textura varía, la cantidad de datos resultante es lo suficientemente grande como para no tener problemas en la fase de clasificación. Los datos obtenidos para este conjunto de texturas son públicos y están disponibles en [105].

Una vez que los algoritmos de barrido y FFT han sido configurados *offline*, estos se aplican sobre los datos obtenidos de las exploraciones. El algoritmo de barrido se implementa mediante Co-Simulación en HLS, y los resultados se guardan en archivos .dat. Por otro lado, la FFT se calcula utilizando las funciones pre-compiladas MEX de Matlab®, y los datos se almacenan en estructuras .mat. Los resultados tanto del barrido como de la FFT se dividen aleatoriamente en un $60\,\%$ para entrenamiento y un $40\,\%$ para pruebas.

En un escenario real, la información que llega al clasificador puede contener valores atípicos o *outliers*, que son muestras estadísticamente distantes del resto de los datos. Por lo tanto, no tiene sentido eliminar estos valores del conjunto de test. Sin embargo, se pueden retirar estos datos para mejorar el resultado del aprendizaje en la fase de

UNIVERSIDAD DE MÁLAGA

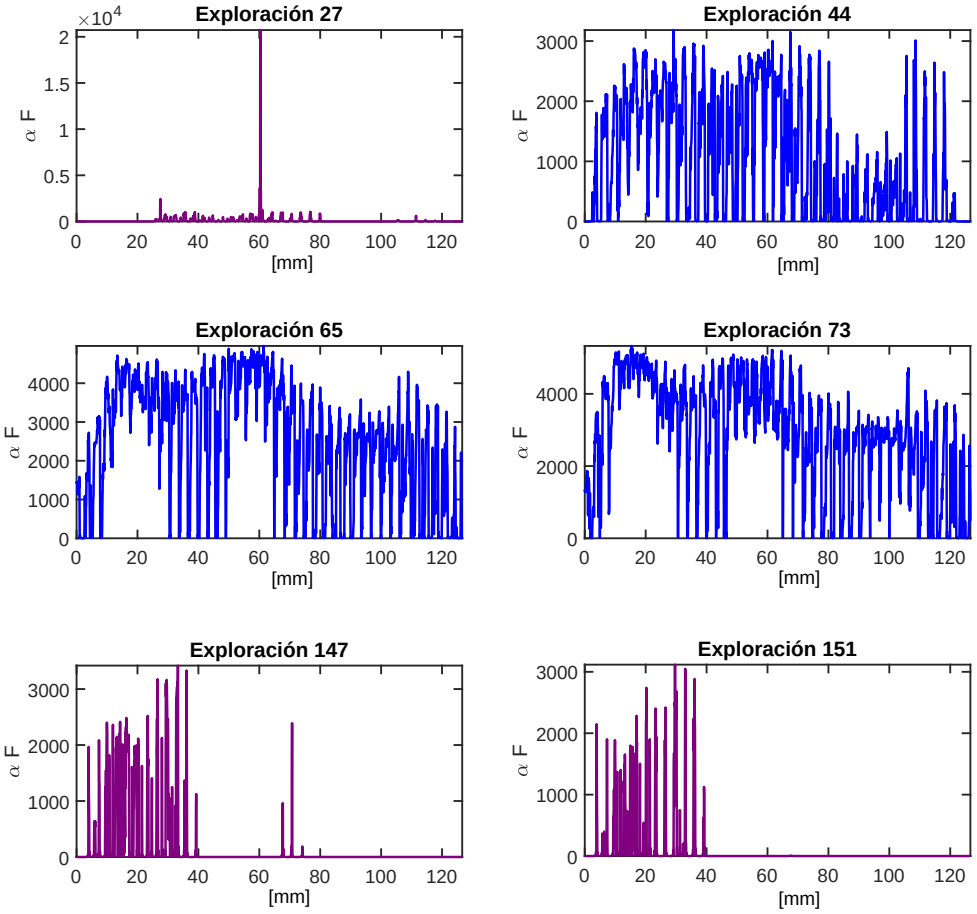


Figura 3.29. Ejemplo de inspección visual de tramas para la textura #TEX-9 en la figura 3.17. En la imagen, las exploraciones 27, 147 y 151 se descartaron.

entrenamiento. Así, se ha establecido un método para eliminar los valores espurios del conjunto de entrenamiento. A continuación, se detalla este procedimiento, además del funcionamiento del algoritmo de entrenamiento *k-means* para obtener los centroides y realizar la clasificación en tiempo real.

3.6.1. Eliminación de *outliers* en el conjunto de entrenamiento

Los valores atípicos o *outliers* del conjunto de entrenamiento son aquellos datos correspondientes a una clase, muy separados estadísticamente del resto. Para retirarlos, primero se calcula la distancia euclídea de todos los vectores de entrenamiento a un vector de referencia unitario $\boldsymbol{u}_{ref}$, ver figura 3.30, obteniendo así el vector de distancias $\boldsymbol{v}_d$. La imagen de la figura 3.31 ilustra esta idea para la matriz de características de cada clase **MC** (de n filas por c columnas).

Una vez se tiene el vector de distancias $\boldsymbol{v}_d$, se puede hacer una representación de la distribución asociada a cada clase, figura 3.32.

UNIVERSIDAD DE MÁLAGA

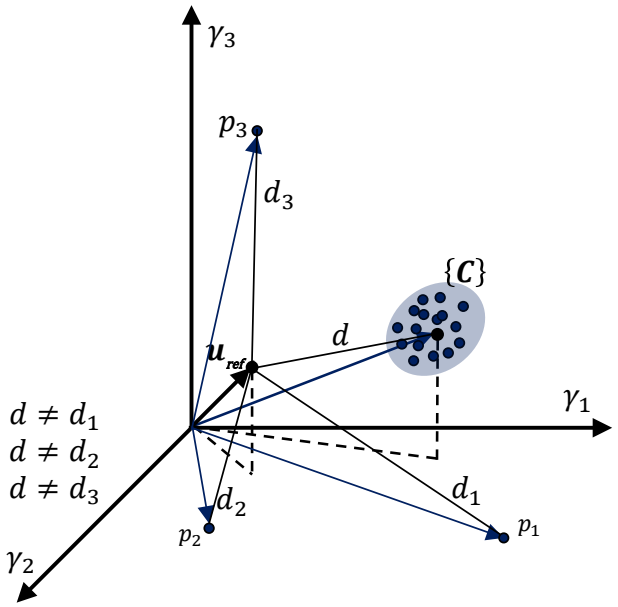


Figura 3.30. Representación para dimensión $n = 3$ de la distancia d entre el centroide del conjunto $\{C\}$ y el vector de referencia unitario $\boldsymbol{u_{ref}}$, y de las distancias d_1, d_2 y d_3 entre los *outliers* p_1, p_2 y p_3 y el vector $\boldsymbol{u_{ref}}$. Los ejes están representados como γ_1, γ_2 y γ_3.

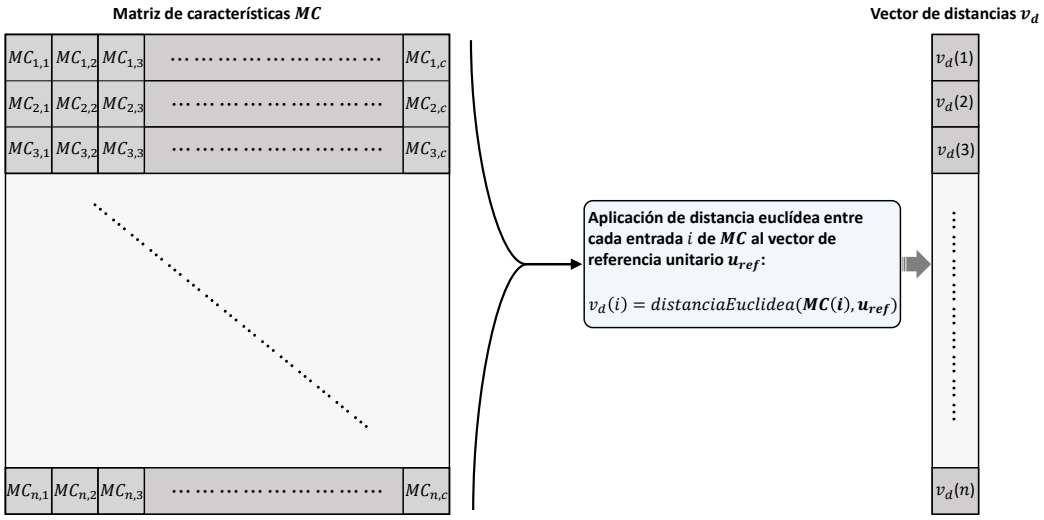


Figura 3.31. Cálculo de la distancia euclídea entre cada entrada i de la matriz de características **MC** y un vector de referencia unitario $\boldsymbol{u_{ref}}$, que da como resultado el vector de distancias $\boldsymbol{v_d}$.

Como se puede ver en la figura 3.32, las zonas de la distribución para cada clase con mayor densidad de puntos se encuentran dentro del rango $\mu \pm \sigma$. Siendo $\mu_{\boldsymbol{v_d}}$ y $\sigma_{\boldsymbol{v_d}}$ la media y desviación típica del vector de distancias, respectivamente. Cada entrada i de la matriz de características $\mathbf{MC}_{n\times c}$ se considera un *outlier* si se da alguna de las condiciones en (3.28).

UNIVERSIDAD DE MÁLAGA

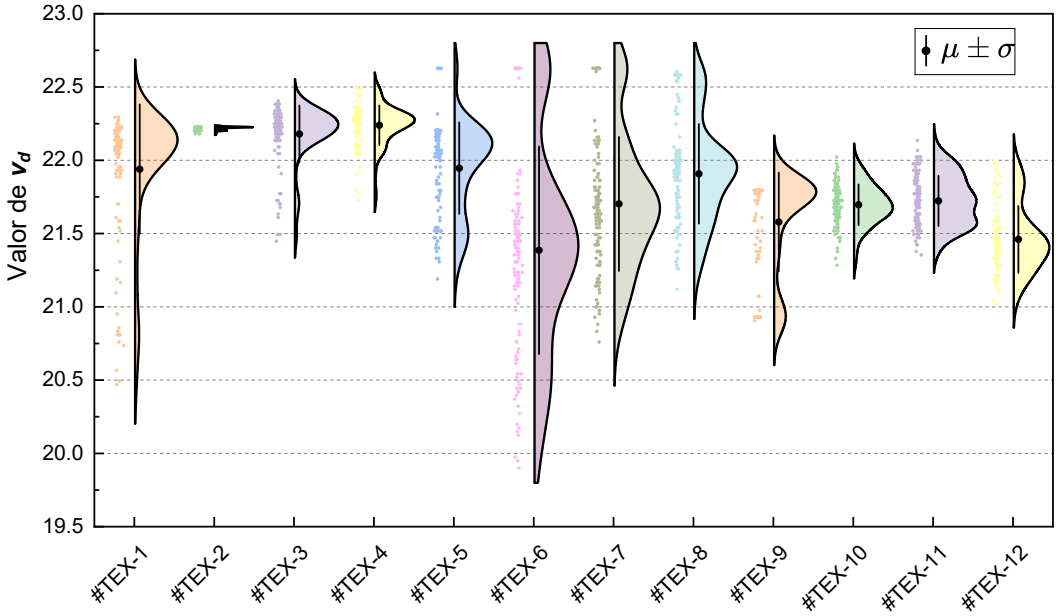


Figura 3.32. Diagrama de tipo *rain plot* que muestra la distribución (para el conjunto de entrenamiento) de todos los valores de $\boldsymbol{v_d}$ para cada clase, junto a su media y desviación típica. Las distribuciones se muestran para el algoritmo de *sweep*.

$$\begin{aligned} &\textbf{1): } \mathbf{MC}_{n\times c}(i) \text{ es } \textit{outlier} \text{ si: } \mathbf{MC}_{n\times c}(i) > (\mu_{\boldsymbol{vd}} + \sigma_{\boldsymbol{vd}}) \\ &\textbf{2): } \mathbf{MC}_{n\times c}(i) \text{ es } \textit{outlier} \text{ si: } \mathbf{MC}_{n\times c}(i) < (\mu_{\boldsymbol{vd}} - \sigma_{\boldsymbol{vd}}) \end{aligned} \tag{3.28}$$

3.6.2. Algoritmo *k-means*

En esta tesis, el entrenamiento del algoritmo *k-means* se realizó de forma *offline* con los datos del conjunto de entrenamiento (figura 3.28). El procedimiento se describe a continuación. Primero, se inicializaron de forma aleatoria los centroides. Después, estos centroides se actualizaron en la fase de aprendizaje, utilizando el conjunto de datos de entrenamiento. Este proceso se repitió iterativamente, barajando los datos del conjunto de entrenamiento hasta que los centroides dejaron de cambiar, o hasta que se alcanzaba un número de épocas de entrenamiento determinado (el número de épocas de entrenamiento se estableció en 13). Finalmente, se evaluó el rendimiento del algoritmo entrenado utilizando el conjunto de datos de test, obteniendo el porcentaje de clasificación correspondiente. El procedimiento completo se repitió 100 veces y los centroides que lograron el porcentaje de clasificación más alto fueron seleccionados para el clasificador entrenado final [106]. En la figura 3.33 se muestra un diagrama de flujo, correspondiente a este proceso (el código completo en Matlab® se puede consultar en el repositorio público en *https://github.com/raulLkRb/thesis.git*).

UNIVERSIDAD DE MÁLAGA

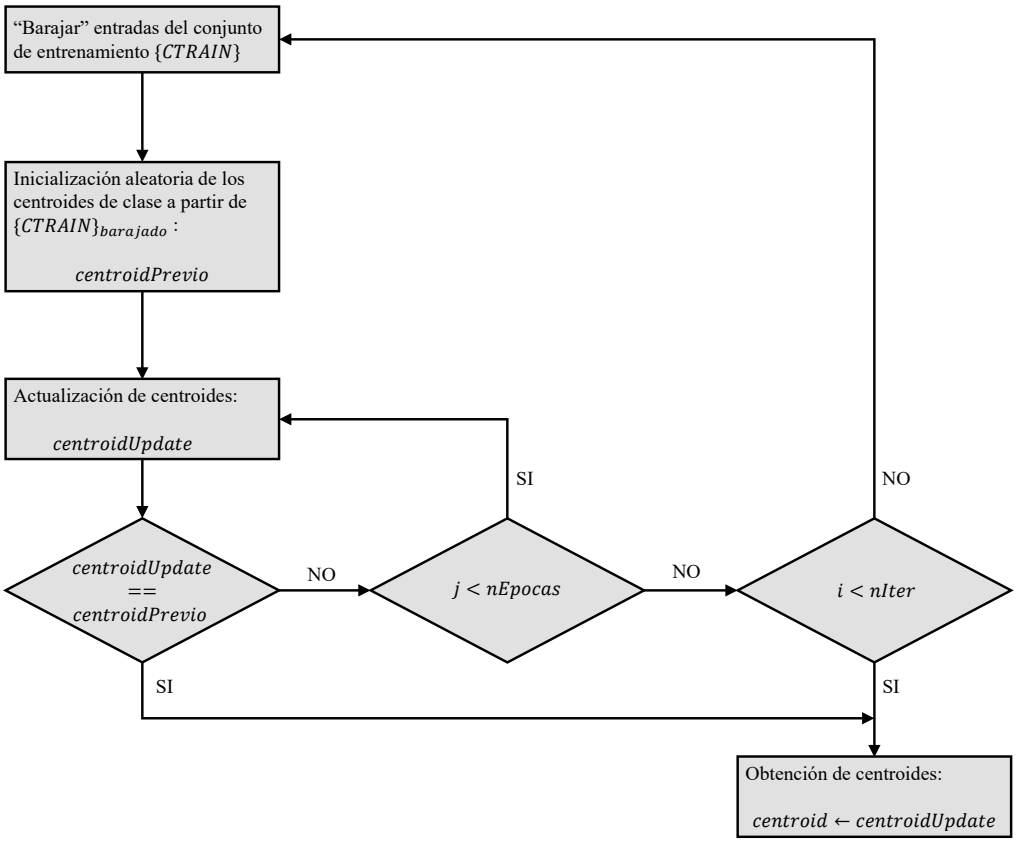


Figura 3.33. Diagrama de flujo simplificado para el proceso de entrenamiento y obtención de centroides óptimos con *k-means*. En la figura, los índices i y j recorren bucles hasta alcanzar el número de iteraciones total ($nIter$) del procedimiento, y el número de épocas de entrenamiento ($nEpocas$).

Una vez entrenado el algoritmo de clasificación *k-means*, se tiene un centroide óptimo por cada una de las 12 clases correspondientes a las texturas. Estos centroides se grabaron en la memoria del ARM® del SoC Zyn-7000® utilizado, ver sección 3.8. En tiempo real, cuando llega un nuevo vector de características al clasificador, se calcula la distancia euclídea del mismo a los centroides de todas las clases. El nuevo vector se clasifica en la clase cuya distancia calculada sea mínima. En la figura 3.34 se muestra el pseudocódigo correspondiente (consultar el repositorio público *https://github.com/raulLkRb/thesis.git* con el código en Matlab® y en SDK de Vivado®).

3.6.3. Visualización de la separación entre clases

Para visualizar la separación entre los grupos o clases en datos multidimensionales, se puede utilizar una representación estadística de la distancia de todos los puntos pertenecientes al conjunto de entrenamiento de una clase, con respecto al centroide de cada clase. En la figura 3.35 se muestran 12 diagramas de violín, cada uno correspondiente a

UNIVERSIDAD DE MÁLAGA

Entradas:
- $\boldsymbol{vNew}$ (vector de tamaño $1 \times M$ con una nueva entrada de características)
- k (número de clases)
- $\boldsymbol{centroide}$ (vector de tamaño $k \times n$ con los centroides óptimos para cada clase)

Variables:
- idx (índice de clase)
- $closetDistance$ (distancia más cercana entre $\boldsymbol{vNew}$ y cada centroide)

Procedimiento:
Calculamos la distancia de $\boldsymbol{vNew}$ a cada centroide en $\boldsymbol{centroide}$ y nos quedamos con la mínima de todas ellas para determinar la clase correspondiente:

for $i \leftarrow 1$ *to* n **do**
  // Inicializamos idx y $closetDistance$ a valores distintos a los que se obtendrán
  $idx = -1$ (valores a obtener están entre 1 y 12)
  $closetDistance = 1/0$ (inicializamos a valor muy grande $\rightarrow \infty$)
  for $j \leftarrow 1$ *to* k **do**
    // Calculamos distancia euclídea
    $distEuclidea = \sqrt{(\boldsymbol{vNew} - \boldsymbol{centroide}(j))^2}$
    // Nos quedamos con la distancia al grupo más cercano
    if $distEuclidea < closetDistance$
      $idx = j$
      $closetDistance = distEuclidea$
    end
  end
  $clusterIdx = idx$
end

Salida: $clusterIdx$ con el valor de la clase a la que pertenece $\boldsymbol{vNew}$

Figura 3.34. Pseudocódigo para el algoritmo de clasificación *k-means* en tiempo real ante nuevas entradas de características.

la distribución de estas distancias.

En cada gráfica, se observa que al representar la distancia de los valores del conjunto de entrenamiento de una clase con respecto a su propio centroide, la distribución resultante siempre es menor que en los demás casos. Esto se debe a que el centroide de una clase no está tan alejado de los puntos de su misma clase como de los de las demás.

UNIVERSIDAD DE MÁLAGA

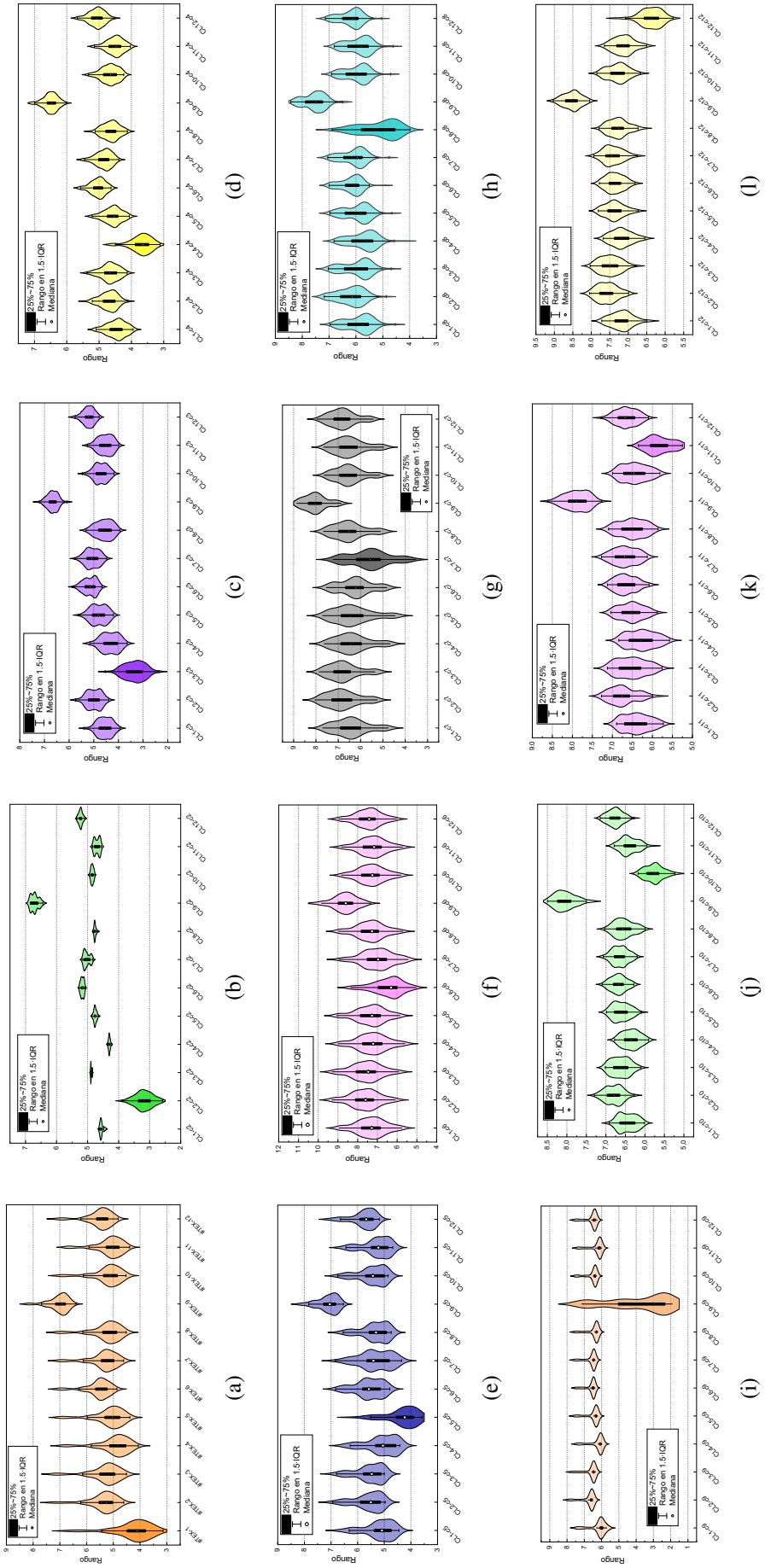


Figura 3.35. Representación de las distancia entre todos los valores del conjunto de entrenamiento de cada clase con respecto al centroide de la clase para la textura: (a) #TEX-1, (b) #TEX-2, (c) #TEX-3, (d) #TEX-4, (e) #TEX-5, (f) #TEX-6, (g) #TEX-7, (h) #TEX-8, (i) #TEX-9, (j) #TEX-10, (k) #TEX-11 y (l) #TEX-12.

UNIVERSIDAD DE MÁLAGA

3.7. Resultados en Matlab®

En esta sección se presentan los resultados obtenidos en Matlab® para diferentes configuraciones del algoritmo de barrido o *sweep*, así como para la FFT, y también para la propuesta de aplicación del algoritmo de Goertzel.

3.7.1. Resultados para el algoritmo de barrido o *sweep*

Cabe recordar que el algoritmo de barrido propuesto en la sección 3.2 calcula M valores a la salida para una señal de entrada de longitud $2 \cdot M$. En particular, se utiliza la expresión (3.3) para calcular el valor de V y después se calculan los extremos de V, marcando su posición en el vector de características resultante. Esto resulta ventajoso, ya que reducimos la cantidad de información que recibe el clasificador y el número de operaciones aritméticas para determinar la distancia euclídea a los centroides del problema. En este apartado, se presentan los resultados de clasificación al enviar como vector de características sólo las posiciones de los mínimos, sólo las posiciones de los máximos, o enviar conjuntamente las posiciones de los mínimos y máximos.

Estas tres posibilidades han sido evaluadas en Matlab®. Para las dos primeras opciones (posiciones de mínimos o posiciones de máximos), se puede utilizar el valor lógico '1' para indicar la presencia de una singularidad y el valor lógico '0' para indicar su ausencia. Por otro lado, para el envío conjunto de las posiciones de mínimos y máximos, se utiliza una codificación de dos bits. Para ello, se asigna el valor '01' (1 en decimal) a las posiciones de los mínimos, el valor '10' (2 en decimal) a las posiciones de los máximos, y el valor '00' (0 en decimal) a las posiciones donde no se encuentren ni mínimos ni máximos. De acuerdo con este criterio, se envían vectores de M posiciones al clasificador, donde cada posición contiene un valor de '0' o '1' (si se elige una de las dos primeras estrategias), o bien los valores '00', '01', o '10' (si se opta por la opción de enviar conjuntamente las posiciones de mínimos y máximos).

Mediante el entrenamiento del clasificador *k-means* descrito en la sección 3.6, se presentan en la tabla 3.6 los porcentajes de clasificación obtenidos en Matlab® para estos tres escenarios, teniendo en cuenta que $M = 1024$.

Como se puede ver en la tabla 3.6, los porcentajes de clasificación que se obtienen para estos casos son muy pobres, superando apenas el $50\,\%$, en el caso de enviar las posiciones de los mínimos. Esto denota la alta sensibilidad al ruido que presentan los algoritmos basados en la detección de singularidades. En los siguientes apartados, se proponen estrategias de mejora a este problema.

UNIVERSIDAD DE MÁLAGA

Tabla 3.6. Porcentajes de clasificación con *k-means* en Matlab® para el envío al clasificador de las posiciones de los mínimos, máximos, o mínimos y máximos conjuntamente. En todos los casos se proporciona la desviación típica entre los mejores casos de clasificación. Los datos proporcionados son para $M = 1024$ e incluyen la clasificación de *outliers* en el conjunto de test (ver sección 3.6).

Envío de posiciones de	Resultado de clasificación
Mínimos	$(52.7 \pm 2.7)\,\%$
Máximos	$(46.5 \pm 2.2)\,\%$
Mínimos y máximos	$(47.8 \pm 2.6)\,\%$

Filtro de media móvil de tamaño de ventana N

Una primera idea consiste en la aplicación en tiempo real de un filtro de media móvil a los datos de la salida de V (3.3), que se vaya calculando durante la exploración de la textura. Un filtro de media móvil consiste en desplazar una ventana de muestras de longitud N a lo largo del vector de entrada de datos, y calcular la media aritmética de los datos que se encuentren dentro de esta ventana. Aplicando este filtro se puede suavizar la señal obtenida para V, (3.3). En este trabajo se han considerado tamaños de ventana para el filtro de media móvil de $N = \{2, 4, 8, 16\}$. En la subsección 3.8.2 se comenta la implementación de este filtro en tiempo real.

Diezmado del barrido por un factor d

Aplicar un diezmado o submuestreo a una señal consiste en tomar una muestra cada n valores de la señal de entrada. Esto equivale a bajar la frecuencia de muestreo original de F_s a $F'_s = F_s/d$, siendo d un valor entero. Es importante recordar que los motores paso a paso utilizados en la configuración experimental pueden introducir oscilaciones durante la exploración, como se discutió en la sección 2.4 del capítulo 2. Además, otros factores mecánicos no controlados, relacionados con la interacción entre el sensor y la superficie, pueden generar picos de alta frecuencia. El diezmado ayuda a mitigar los efectos negativos de estas componentes espurias. Por otra parte, un submuestreo de la señal también puede introducir ruido de baja frecuencia debido al *aliasing*. Afortunadamente, el algoritmo de barrido muestra un comportamiento robusto frente a este tipo de fuentes, ya que este ruido es de baja amplitud y no altera la salida del algoritmo [107] [108], como se comprobará también en la sección 3.7. En la figura 3.36 se muestra un detalle del cálculo de V (3.3) para la textura #TEX-4 (ver figura 3.17), con un *zoom* en una zona con mínimos locales. Como se puede ver, si se envía la posición de los mínimos en la figura 3.36, se estarían localizando dos mínimos en lugar de uno, lo que afectaría a la clasificación. Por otro lado, en la figura 3.37 se puede ver el efecto de diferentes factores de diezmado d en la reducción de estas perturbaciones.

UNIVERSIDAD DE MÁLAGA

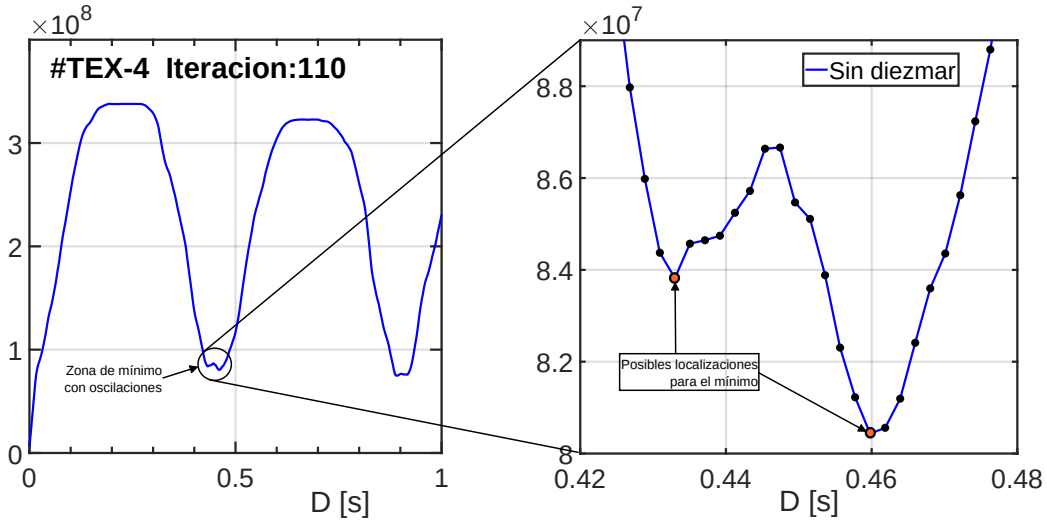


Figura 3.36. Detalle del cálculo de V (3.3) para la textura #TEX-4 de la figura 3.17. Se muestra el rango de $D = 0$ s a $D = 1$ s. Como se puede ver en el *zoom* aplicado, aparecen algunos picos de baja amplitud debidos a la interacción entre el sensor y la superficie.

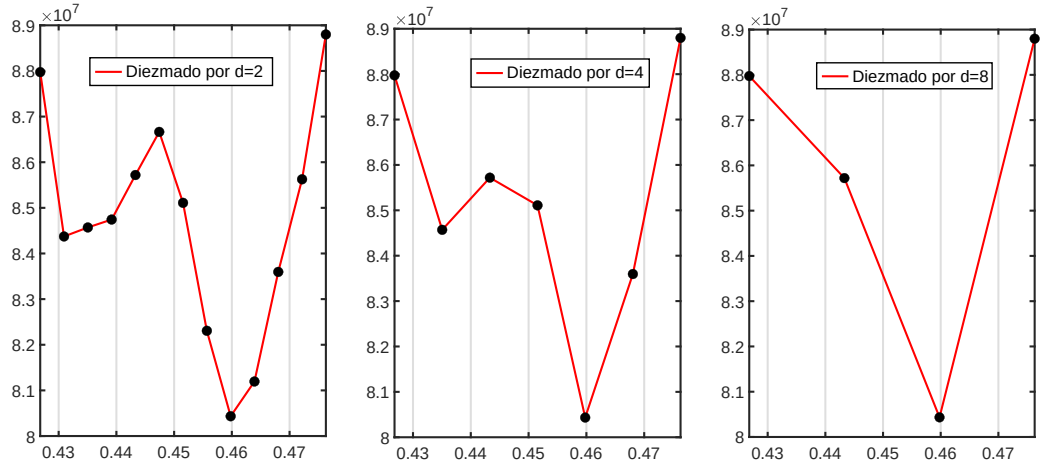


Figura 3.37. Diezmado sobre el cálculo de V (3.3) mostrado en la figura 3.36 para la textura #TEX-4. Se aplica un factor de diezmado de: (a) $d = 2$, (b) $d = 4$, (c) $d = 8$. Se puede ver que la localización del mínimo mejora conforme el diezmado es mayor.

Como se puede comprobar en la figura 3.37, cuando se aplica un diezmado de $d = 8$ sólo se detecta un mínimo en lugar de dos. Aunque la localización de este mínimo tendrá error por efecto del diezmado, el efecto global en la clasificación será positivo.

Resultados en Matlab® en la aplicación de las estrategias de mejora del filtro de media móvil y diezmado

En la tabla 3.7 se dan los resultados obtenidos en Matlab® para cada una de las opciones mencionadas anteriormente (filtro de media móvil, diezmado y su combinación) de forma individual y conjunta.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Tabla 3.7. Porcentajes de clasificación obtenidos utilizando el algoritmo *k-means* en Matlab® al enviar las posiciones de los mínimos, máximos o de los mínimos y máximos conjuntamente como características de entrada al clasificador. Además, se utilizan distintos factores de diezmado d, indicando entre paréntesis el número de componentes del vector de características correspondiente (comp. VC). En todos los casos se proporciona la desviación típica de los mejores resultados de clasificación. Los datos proporcionados incluyen la clasificación de *outliers*.

Posiciones de	Tamaño de ventana del filtro de media móvil, N	Factor de diezmado d						
		128 (8 comp. VC)	64 (16 comp. VC)	32 (32 comp. VC)	16 (64 comp. VC)	8 (128 comp. VC)	4 (256 comp. VC)	2 (512 comp. VC)
Mínimos	2	$(44.3 \pm 3.9)\,\%$	$(60.2 \pm 3.9)\,\%$	$(82.2 \pm 0.9)\,\%$	$(87.8 \pm 1.5)\,\%$	$(89.6 \pm 1)\,\%$	$(90.6 \pm 1)\,\%$	$(82.5 \pm 3.2)\,\%$
	4	$(44.4 \pm 2.8)\,\%$	$(63 \pm 3)\,\%$	$(82.8 \pm 1.11)\,\%$	$(87.4 \pm 1.3)\,\%$	$(88.9 \pm 1.2)\,\%$	$(89.9 \pm 0.8)\,\%$	$(85.2 \pm 2.4)\,\%$
	8	$(41.1 \pm 2.5)\,\%$	$(69.2 \pm 1.9)\,\%$	$(83 \pm 0.9)\,\%$	$(88.4 \pm 1)\,\%$	$(90.6 \pm 0.7)\,\%$	$(90.8 \pm 0.8)\,\%$	$(85.1 \pm 2.2)\,\%$
	16	$(40.1 \pm 3.8)\,\%$	$(66.2 \pm 3.2)\,\%$	$(79.7 \pm 1)\,\%$	$(89 \pm 0.9)\,\%$	$(90 \pm 0.8)\,\%$	$(87.3 \pm 1)\,\%$	$(82.2 \pm 3)\,\%$
	Sin filtro	$(44.8 \pm 2.9)\,\%$	$(58.5 \pm 4.5)\,\%$	$(81 \pm 1)\,\%$	$(88.3 \pm 1.5)\,\%$	$(90.5 \pm 0.8)\,\%$	$(90.1 \pm 1.1)\,\%$	$(81.1 \pm 2.5)\,\%$
Máximos	2	$(42.3 \pm 3)\,\%$	$(64.7 \pm 2.7)\,\%$	$(80.4 \pm 1.1)\,\%$	$(84.4 \pm 1)\,\%$	$(83.3 \pm 1.2)\,\%$	$(81 \pm 1.6)\,\%$	$(70.5 \pm 2.8)\,\%$
	4	$(43.1 \pm 2.6)\,\%$	$(65.4 \pm 2.2)\,\%$	$(80 \pm 1.13)\,\%$	$(84.1 \pm 1)\,\%$	$(82.6 \pm 1.3)\,\%$	$(80.4 \pm 1.5)\,\%$	$(71.6 \pm 2.4)\,\%$
	8	$(44.9 \pm 3.6)\,\%$	$(65.7 \pm 2.6)\,\%$	$(81.1 \pm 1)\,\%$	$(84.6 \pm 1)\,\%$	$(85.3 \pm 1.3)\,\%$	$(80.9 \pm 1.6)\,\%$	$(72 \pm 2.5)\,\%$
	16	$(45.8 \pm 4)\,\%$	$(67.1 \pm 2.2)\,\%$	$(83.6 \pm 1.12)\,\%$	$(86.8 \pm 0.9)\,\%$	$(83.3 \pm 1.4)\,\%$	$(80.4 \pm 1.7)\,\%$	$(67.9 \pm 1.9)\,\%$
	Sin filtro	$(41.7 \pm 2.3)\,\%$	$(65.2 \pm 2.8)\,\%$	$(80.3 \pm 0.9)\,\%$	$(85.4 \pm 1.3)\,\%$	$(83.9 \pm 1.3)\,\%$	$(81.6 \pm 1.3)\,\%$	$(69.1 \pm 2.7)\,\%$
Mínimos y máximos	2	$(49.8 \pm 1.7)\,\%$	$(53.2 \pm 3.6)\,\%$	$(76.3 \pm 1.3)\,\%$	$(84.8 \pm 1.12)\,\%$	$(84.7 \pm 1)\,\%$	$(85.7 \pm 1)\,\%$	$(79 \pm 3.9)\,\%$
	4	$(49.8 \pm 1.7)\,\%$	$(55.2 \pm 3.6)\,\%$	$(76.3 \pm 1.3)\,\%$	$(84.8 \pm 1.12)\,\%$	$(84.7 \pm 1)\,\%$	$(85.7 \pm 1)\,\%$	$(79 \pm 3.9)\,\%$
	8	$(52.5 \pm 2.3)\,\%$	$(61.5 \pm 3.5)\,\%$	$(76.4 \pm 1.1)\,\%$	$(82.8 \pm 1.2)\,\%$	$(84.12 \pm 1.1)\,\%$	$(83.13 \pm 1.2)\,\%$	$(72.7 \pm 3.6)\,\%$
	16	$(49.8 \pm 1.7)\,\%$	$(55.2 \pm 3.6)\,\%$	$(76.3 \pm 1.3)\,\%$	$(84.8 \pm 1.12)\,\%$	$(84.7 \pm 1)\,\%$	$(85.7 \pm 1)\,\%$	$(79 \pm 3.9)\,\%$
	Sin filtro	$(49.8 \pm 1.7)\,\%$	$(55.2 \pm 3.6)\,\%$	$(76.3 \pm 1.3)\,\%$	$(84.8 \pm 1.12)\,\%$	$(84.7 \pm 1)\,\%$	$(85.7 \pm 1)\,\%$	$(79 \pm 3.9)\,\%$

3.7.2. Resultados para *Fast Fourier Transform* (FFT)

La transformada de Fourier se puede considerar como la herramienta ideal o *gold standard* para obtener información en frecuencia. El cálculo de los coeficientes de la FFT viene dado por un espectro en frecuencia simétrico. Esto quiere decir que, para una FFT de M puntos, la mitad del espectro (de tamaño $M/2$) contendrá toda la información.

Si consideráramos como vector de características el espectro completo, es decir, los M coeficientes de la FFT, el clasificador podría beneficiarse de las redundancias de información introducidas. Sin embargo, al duplicar la información, también estamos duplicando el tamaño de envío. Por tanto, para mantener una comparación equitativa con el *sweep*, nos quedamos con los M/d primeros valores de la primera mitad del espectro de la FFT, donde d es el factor de diezmado utilizado en el *sweep* (que para la FFT actúa como una fracción o divisor de M). En la tabla 3.8, se presentan los porcentajes de clasificación obtenidos en Matlab® para el caso de la FFT.

Tabla 3.8. Porcentajes de clasificación obtenidos utilizando el algoritmo *k-means* en Matlab® al enviar los coeficientes de la FFT como características de entrada al clasificador. En todos los casos se proporciona la desviación típica de los mejores resultados de clasificación, indicando entre paréntesis el número de componentes del vector de características correspondiente (comp. VC). Los datos proporcionados incluyen la clasificación de *outliers*.

Divisor d						
128 (8 comp. VC)	64 (16 comp. VC)	32 (32 comp. VC)	16 (64 comp. VC)	8 (128 comp. VC)	4 (256 comp. VC)	2 (512 comp. VC)
$(72.3 \pm 1.1)\,\%$	$(82.8 \pm 1.4)\,\%$	$(88.6 \pm 0.9)\,\%$	$(90.1 \pm 1)\,\%$	$(90.2 \pm 0.9)\,\%$	$(90.7 \pm 0.9)\,\%$	$(90.5 \pm 0.9)\,\%$

En el caso de enviar todos los coeficientes correspondientes a la mitad del espectro de la FFT, es decir, $M = 1024$ valores, se logra un porcentaje de clasificación del $(90.5 \pm 0.9)\,\%$. Sin embargo, esto no ocurre en el caso del barrido, donde se obtenía un porcentaje máximo de $52.7\,\%$ (ver tabla 3.6). Una posible explicación radica en que la FFT es menos sensible a los ruidos espurios de alta frecuencia ya que su amplitud es relativamente baja, como también se argumentó en las simulaciones de la subsección 3.4.2. Esto hace que tenga un impacto mínimo en el espectro de las diferentes texturas durante la exploración (ver figura 3.26 en la subsección 3.5.2).

Por otro lado, en el caso del barrido, al aplicar un algoritmo de detección de mínimos, estas pequeñas fluctuaciones de ruido pueden generar pasos por cero continuos al comparar los cambios de signo en la derivada de la señal. A pesar de esto, los resultados obtenidos mediante las estrategias de mejora propuestas demuestran porcentajes de clasificación prometedores en el análisis *offline*. Aunque existen otras técnicas como la normalización o el uso de filtros más efectivos que podrían mejorar la clasificación,

UNIVERSIDAD DE MÁLAGA

se debe tener en cuenta que a medida que el procesamiento se vuelve más complejo en hardware, la propuesta del algoritmo puede resultar menos atractiva, ya que su rendimiento podría ser incluso peor que el de la FFT en términos de consumo de recursos, potencia, tráfico de datos y escalabilidad. En la siguiente sección 3.8 se explicará cómo se han implementado estos algoritmos en el dispositivo SoC utilizado.

3.7.3. Resultados para el algoritmo de Goertzel

En esta subsección se muestran los resultados de clasificación obtenidos cuando se aplica el algoritmo de Goertzel con $N = 1024$, sobre las 12 texturas de la figura 3.17. Para llevar a cabo la clasificación, se ha seguido el mismo procedimiento para el entrenamiento del algoritmo de clasificación propuesto para el barrido y FFT, ver sección 3.6.

El espectro completo de la señal muestreada $\{x[n]\}$ está replicado respecto a $F_s/2$. Por tanto, sólo es necesario calcular los índices para la primera mitad del espectro, que se utilizarán para obtener las componentes de frecuencia que se envíen al clasificador. La ventaja computacional y de memoria que ofrece Goertzel frente a la FFT, expresiones (3.18) y (3.20), sólo depende del tamaño del vector de índices K que se considere. La tabla 3.9 muestra los resultados obtenidos para $K = \{4, 8, 16\}$ y diferente separación espectral de los índices.

Tabla 3.9. Porcentajes de clasificación con *k-means* cuando se considera el algoritmo de Goertzel propuesto para $N = 1024$, para diferentes valores del número de índices K y del parámetro de control p en (3.26). En todos los casos se proporciona el valor máximo de clasificación encontrado. Los datos proporcionados incluyen la clasificación de *outliers* en el conjunto de test (consultar sección 3.6).

K	Equiespaciado	$p = 1$	$p = 10$	$p = 25$	$p = 50$	$p = 100$
4	26.7 %	59.8 %	58.9 %	56.3 %	46.2 %	41.6 %
8	36.7 %	69.9 %	79.8 %	65.6 %	67.9 %	43.4 %
16	43 %	59.8 %	**84.8 %**	79 %	81.4 %	64 %

Como se puede ver en la tabla 3.9, en todos los casos en los que se toma la expresión (3.26) para escoger los índices, los resultados son mejores que en el caso de equiespaciarlos. Por otra parte, al aumentar el valor del parámetro de control p, los porcentajes disminuyen respecto a valores más bajos. Esto es debido a que se seleccionarían más frecuencias de la zona alta del espectro (frecuencias espaciales más altas, donde apenas hay información, ver figura 3.26). Este efecto se mitiga al aumentar el valor de K. El mejor caso en la tabla 3.9 es para $K = 16$ y $p = 10$ (obteniendo un $84.8\,\%$ de precisión). En la subsección 3.8.3, se describe la implementación en SoC del algoritmo de Goertzel propuesto, y se compara a las implementaciones del barrido y FFT.

UNIVERSIDAD DE MÁLAGA

3.8. Implementación de algoritmos en SoC

En esta sección se profundiza en la integración de los algoritmos de extracción de características en un marco global de implementación del diseño. Se explica el camino de datos o *datapath* para la arquitectura de sistema planteada en hardware. Se describe cada parte y módulos de la arquitectura por separado. Finalmente, se describe la implementación de la parte de clasificación en el ARM®.

Arquitectura global del diseño

El diagrama de bloques de la figura 3.38 muestra la implementación del diseño en el SoC para el algoritmo de *sweep*, FFT, o Goertzel respectivamente. Las tres opciones se representan mediante líneas discontinuas. En la figura se puede ver la arquitectura global del sistema así como el camino de datos completo. Además, en el apéndice E se incluye una captura en Vivado® con el diagrama de bloques equivalente a este diseño. En la figura 3.38 aparecen las dos partes del SoC, PL y PS (ver sección 2.5 del capítulo 2). En esta tesis, la estrategia adoptada consiste en ubicar los algoritmos de preprocesamiento en la parte PL (FPGA), mientras que el clasificador se ejecuta en el ARM®. A continuación, se describe el camino de datos entre los módulos de la parte PL, y el enfoque de la parte PS.

Comenzando con la parte PL de la figura 3.38, en un primer paso, un controlador basado en el protocolo serie de comunicación SPI, toma los datos proporcionados por el sensor táctil y los reordena a un formato de palabra de 16 bits. Estos datos se envían a un filtro *antialiasing*, ver sección 3.3. La salida del filtro consiste en palabras de 32 bits que se guardan en una memoria. A su vez, esta memoria se comunica con el módulo de interfaz AXISTREAM. El protocolo de comunicación AMBA® AXI-Streaming se utiliza para comunicar el resto de módulos del sistema en la parte PL [109] (ver sección 2.5 del capítulo 2). A continuación, ya sea utilizando el algoritmo de barrido *HLS-Vivado Sweep Algorithm*, el IP Core de la FFT, o el algoritmo de Goertzel *HLS-Vivado Goertzel Algorithm*, se calculan los vectores de características correspondientes. Estos valores se envían a una memoria DDR externa a través de un *buffer* implementado como una *First-Input First-Output* (FIFO) y un módulo *Direct Memory Access* (DMA). De esta manera, en la figura 3.38, la parte del núcleo ARM® puede acceder a la información almacenada en la memoria sin preocuparse por el tráfico de datos.

El entorno de desarrollo Xilinx® SDK (ver sección 2.5 del capítulo 2) se utiliza para implementar el clasificador en el núcleo ARM®. Como se comentó en la sección 3.6,

UNIVERSIDAD DE MÁLAGA

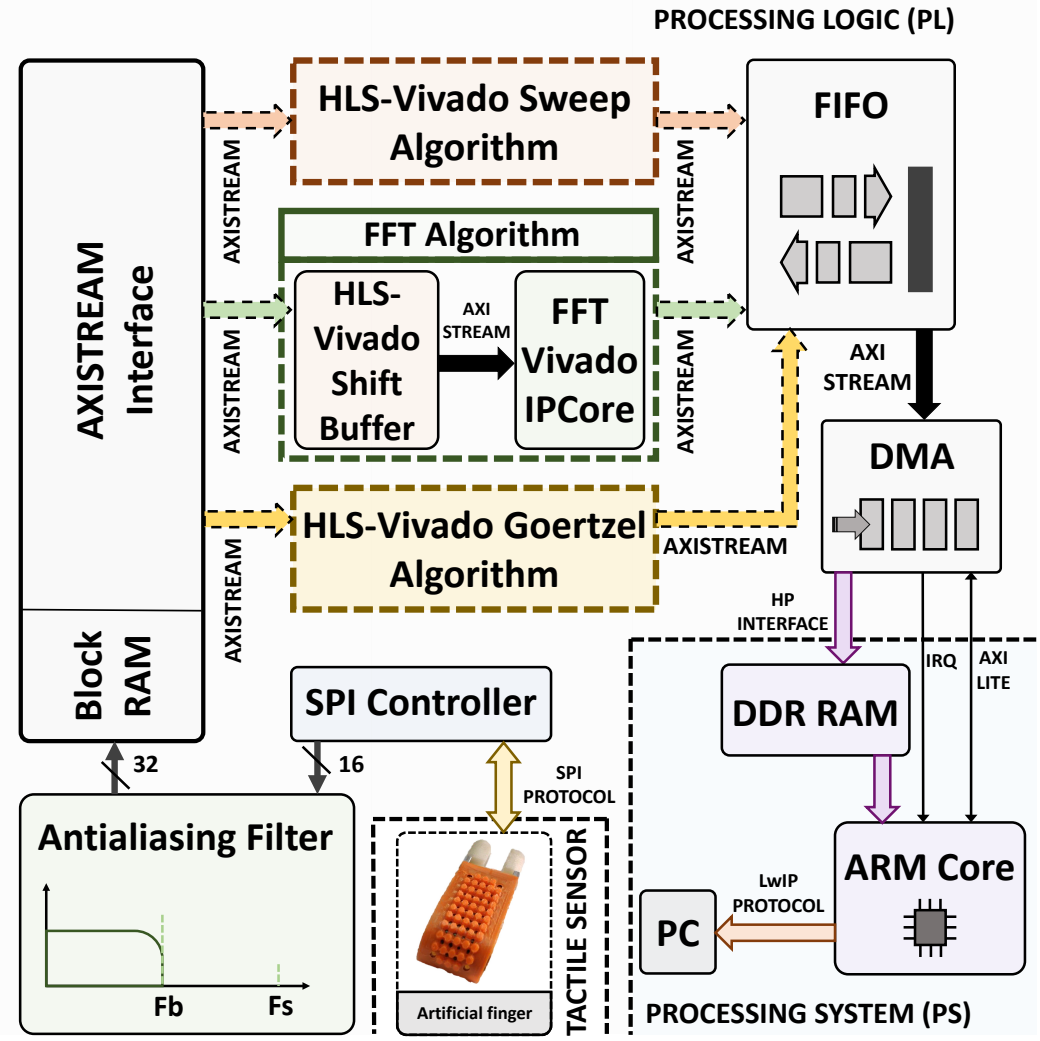


Figura 3.38. Diagrama general de la arquitectura en la Zynq-7000® XC7Z020 para el algoritmo de barrido *sweep*, FFT, o algoritmo de Goertzel. Cada módulo está representado mediante líneas discontinuas y equivale al IP Core diseñado o utilizado para cada algoritmo.

los centroides se guardan en la memoria del procesador. Por otro lado, se mejora el *throughput* (cantidad de datos transmitida por unidad de tiempo) al trabajar con el sistema operativo en tiempo real FreeRTOS® integrado en el ARM®, en combinación con una pila ligera de TCP/IP llamada lwIP, también implementada en el ARM®, ver sección 2.5 del capítulo 2. Cada uno de estos módulos, tanto para PL como para PS, se describen en mayor detalle en el apartado 3.8.1.

3.8.1. Módulos hardware en la parte PL del diseño

En este apartado se describen cada uno de los módulos que aparecen en la parte PL en la figura 3.38. El diseño presentado en la figura admite la recepción de hasta 55 datos, uno por táctel, provenientes del sensor táctil. En esta tesis, se selecciona un único táctel para trabajar de forma unidimensional. Así, verificamos de forma óptima el funcionamiento

UNIVERSIDAD DE MÁLAGA

del algoritmo, aunque en cualquier caso, la parte de recepción de la matriz completa se ha dejado inalterada.

SPI Controller

Este módulo está diseñado mediante el lenguaje de descripción hardware VHDL. El control del módulo se hace con una máquina de 7 estados (el código correspondiente al módulo se puede consultar en el repositorio público *https://github.com/raulLkRb/thesis.git*). El controlador SPI se encarga de recibir los datos que provienen del sensor táctil y enviarlos al filtro *antialiasing* en formato punto fijo de 16 bits. En concreto, este módulo trabaja con dos frecuencias de reloj distintas. Por un lado, para el control por SPI se tiene un reloj de frecuencia $f_{sclk} = 12.5$ MHz. En este caso, el módulo *SPI Controller* es el maestro y el sensor táctil es el esclavo. El diagrama de la figura 3.39 muestra un esquema de las conexiones entre los dos módulos.

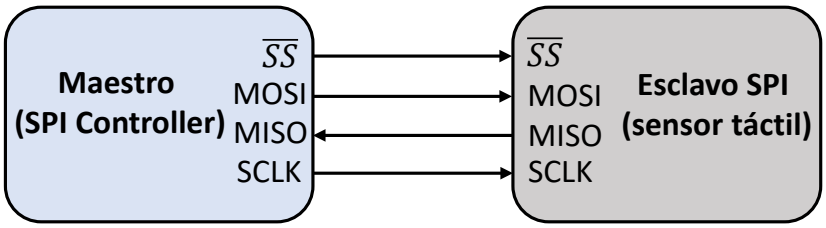


Figura 3.39. Esquema básico de comunicación mediante el protocolo bus serie SPI.

El resto del módulo trabaja con el reloj de frecuencia $f_{clk} = 100$ MHz. El módulo recibe progresivamente todas las filas y columnas por cada nuevo *frame*[2]. De esta manera, podemos enviar los datos de todos los tácteles al filtro *antialiasing* durante el intervalo de tiempo entre *frames*. Cabe destacar que este tiempo equivale a la frecuencia de muestreo que, como se ha comentado anteriormente, es de $F_s = 485$ Hz, o un período de muestreo de $T_s = 2.06$ ms. El tiempo que se tarda en recibir la matriz completa no supera los 1800 ciclos de reloj a $f_{sclk} = 12.5$ MHz. Esto equivale a aproximadamente 0.14 ms, que es bastante más pequeño que el período de muestreo. Además, el resto del sistema trabaja a 100 MHz, y como se puede ver en la simulación presentada en el apéndice F, el algoritmo de barrido actúa en un tiempo mucho menor al período de muestreo.

Filtro *antialiasing*

Este módulo se ha diseñado a partir del IP Core de Vivado® FIR Compiler. Las interfaces de entrada y salida se implementan mediante el protocolo AXI-Streaming, descrito en el 2.5 del capítulo 2.. Los coeficientes del filtro son los ya presentados en la figura 3.9

[2]Un *frame*, en este caso, debe entenderse como los datos correspondientes a la matriz táctil completa.

UNIVERSIDAD DE MÁLAGA

en la sección 3.3. La arquitectura del filtro es de tipo *Systolic Multiply Accumulate*. A la salida del filtro se trabaja con palabras de 32 bits en punto fijo, que se envían a una interfaz de control, donde se reciben y guardan los datos, antes de pasarlos al algoritmo de barrido.

Block RAM + AXISTREAM Interface

Este módulo se ha diseñado mediante VHDL. El módulo consta de tres partes (consumidor de datos, productor de datos y una memoria). El código correspondiente a este módulo se puede encontrar en el repositorio público *https://github.com/raulLkRb/thesis.git*. Las entradas para este módulo son el dato de 32 bits filtrado, una palabra de 7 bits con la fila y columna correspondiente al táctel del dato, una señal binaria de duración 1 ciclo del reloj a $f_{clk} = 100$ MHz, la señal de reloj y el reset del sistema. La parte del consumidor se encarga de leer los datos y del control de escritura en la dirección de memoria indicada por fila y columna. Por otro lado, se ha diseñado una memoria de 128 posiciones[3] para almacenar los datos procedentes del filtro *antialiasing*. La memoria se ha descrito en VHDL para que se sintetice en un bloque empotrado de memoria BRAM, dejando así parte de la memoria distribuida o LUTRAM libre para otras funcionalidades. Finalmente, se implementa una interfaz de salida mediante el protocolo AXI-Streaming para la transmisión de los datos hacia el algoritmo de preprocesamiento.

Algoritmo de barrido

Este módulo contiene la descripción hardware para el algoritmo de barrido. En la subsección 3.8.2 se da un análisis detallado en cuanto al número de recursos lógicos, memoria, consumo de potencia y latencia de las diferentes configuraciones exploradas para este algoritmo. Además, se hace una comparación con las opciones equivalentes para el módulo FFT. A partir de dicho análisis se decide qué opción se implementa en el SoC.

Módulo para la FFT

Este módulo se ha diseñado a partir del IP Core de Vivado® 'xfft'. El módulo está configurado para calcular la FFT de un vector de M posiciones. Para una entrada de 1024 posiciones devuelve una salida de 1024 valores. Para hacer el pre-procesado en tiempo real equivalente al *sweep*, es necesario calcular la FFT cada vez que llegue un nuevo dato de entrada. Para lograr esto, se coloca un *buffer* de desplazamiento de M posiciones

[3]La memoria es de 128 posiciones, aunque sólo se escribe/lee en 55 de ellas, debido al direccionamiento por fila y columna. Al haber 5 filas y 11 columnas, podemos formar una palabra de 7 bits que actúa como dirección de memoria, concatenando los 3 bits correspondientes a la fila y los 4 bits correspondientes a la columna.

UNIVERSIDAD DE MÁLAGA

previo al IP Core de la FFT, ver figura 3.38. Este módulo está diseñado con la herramienta HLS® de Vivado y es similar al *buffer* de desplazamiento empleado para el caso del barrido (ver subsección 3.8.2). De esta forma, para cada dato nuevo a la entrada, la salida obtendrá M nuevos valores, correspondientes a la transformada de Fourier del nuevo valor de entrada más los $M-1$ anteriores. El IP Core para la FFT se ha diseñado de acuerdo a las opciones de configuración que se pueden consultar en el apéndice E.

Algoritmo de Goertzel

Este módulo contiene la descripción hardware para la aplicación del algoritmo de Goertzel propuesto. En la subsección 3.8.3 se proporciona un análisis detallado en cuanto al número de recursos lógicos, memoria, consumo de potencia y latencia de las diferentes configuraciones exploradas para este algoritmo.

Uso del módulo DMA con la memoria DDR

Este módulo se ha diseñado a partir del IP Core de Vivado® 'axi_dma'. El módulo de DMA permite la transferencia de datos de una parte del sistema a otra sin utilizar recursos del procesador, que puede dedicarse a realizar otras tareas. En este diseño, se utiliza la DMA para transferir los datos táctiles pre-procesados a la memoria DDR, para que estén disponibles en el procesador. Como se puede ver en la figura 3.38, el procesador (ARM Core) y el controlador para la memoria DDR (DDR RAM) forman parte de la parte PS del diseño. El módulo DMA y el algoritmo que le proporciona los datos están implementados en la parte PL. El bus AXI LITE permite al procesador comunicarse con la DMA para configurarla, e iniciar y monitorizar la transferencia de datos. Cada vez que haya datos disponibles el módulo DMA activará un flag de interrupción, que iniciará la ejecución de una rutina de interrupción o *interrupt service routine* (ISR) en el procesador. La interfaz de comunicación AXI-Streaming de la DMA está mapeada en memoria y le proporciona acceso a la memoria DDR. Por lo que, una vez activada la interrupción, el módulo DMA envía un flujo o *stream* continuo de datos (sin direcciones) al procesador. El código en C para la comunicación y transferencia por DMA se puede consultar en el repositorio público *https://github.com/raulLkRb/thesis.git*.

FIFO de salida

Este módulo se ha diseñado a partir del IP Core de Vivado® 'fifo_generator'. Las interfaces de entrada y salida se implementan mediante el protocolo AXI-Streaming. El módulo FIFO se implementa para desacoplar las frecuencias entre la parte PL y la parte PS. De esta forma, si el procesador opera a una frecuencia más baja que el hardware, la FIFO evita la pérdida de datos.

UNIVERSIDAD DE MÁLAGA

3.8.2. Implementación del barrido. Comparación en términos de consumo de recursos, potencia y latencia con la FFT

En este apartado se presenta una discusión entre algoritmo de barrido y la FFT en términos de consumo de recursos lógicos, de memoria, potencia y latencia, variando el parámetro correspondiente al tamaño del vector de salida, M. En particular, para cada caso se proporciona el consumo para las distintas variantes del *sweep*, presentadas en la sección 3.7.

Diseño del algoritmo de barrido utilizando la herramienta HLS de Vivado

A partir de la tabla 3.7, en este proyecto se decidió trabajar con el envío de las posiciones de los mínimos de V, (3.3). En dicha tabla se comprobó que, en general, esta opción era la que proporcionaba mejores resultados de clasificación en comparación con el envío de posiciones de máximos, o de mínimos y máximos conjuntamente. El cálculo de la posición de los máximos es similar al proceso seguido para calcular la posición de los mínimos, ya que consiste en comparaciones básicas. Y finalmente, la combinación de ambos supondría un aumento en términos de recursos y/o latencia. El pseudocódigo correspondiente a las distintas configuraciones del *sweep* se da en la figura 3.40.

El algoritmo de barrido básico, inicialmente adquiere una nueva muestra de la exploración a velocidad $v = 30$ mm/s y la almacena en un registro de desplazamiento de $2 \cdot M$ posiciones. A continuación, hay un bucle iterativo, paso 3, donde se calculan los M valores de V (que se corresponden con la salida de V (3.3)), donde el último es la muestra correspondiente al índice $2 \cdot M$ de la exploración. El cálculo de V consiste en diferencias consecutivas elevadas al cuadrado. Por lo tanto, haciendo uso del registro $ibuf$ de tamaño $2 \cdot M$ valores, la expresión (3.3) se puede dividir en los tres pasos de la ecuación (3.29):

$$\begin{cases} add \leftarrow (ibuf[M] - ibuf[M + (n+1)]) \times (ibuf[M] - ibuf[M + (n+1)]) \\ sus \leftarrow (ibuf[0] - ibuf[n+1]) \times (ibuf[0] - ibuf[n+1]) \\ V[n] \leftarrow V[n] + add - sus \end{cases} \tag{3.29}$$

donde n representa la iteración actual, de las M totales. El modelado hardware aproximado para el cálculo de V se representa en la figura 3.41. Como se podrá comprobar en el consumo de recursos, sólo se requiere de 2 multiplicadores y 3 sumadores, que se sintetizan en 2 DSPs empotrados y recursos lógicos. Por otra parte, es necesario un *buffer* de memoria de $2M$ posiciones y registros para guardar valores de cálculos intermedios. En HLS se pueden aplicar "pragmas" consistentes en directivas de optimización para

UNIVERSIDAD DE MÁLAGA

Entradas:
- x (nuevo dato de entrada)

Variables:
- p (variable auxiliar para nueva muestra)
- $ibuf$ (*buffer* interno para almacenar las muestras de datos)
- add (sumatorio en el cálculo de V)
- sus (factor de sustracción en el cálculo de V)
- V (factor de sustracción en el cálculo de V)
- $\boldsymbol{decV}$ (vector con los valores diezmados del array de V)
- M (longitud del vector de salida)
- n (índice de la muestra actual)
- d (factor de diezmado)
- $wndSize$ (ventana para el filtro de media móvil)
- $counterDecimate$ (contador de diezmado)

Procedimiento:

1. Obtenemos nueva muestra p de los datos de entrada:

$p \leftarrow x$

2. Almacenamos p_m y desplazamos una posición a la izquierda el *buffer* interno de desplazamiento $ibuf$ de tamaño $2M$:

$i = 0;$
$while\ \ i < 2M\ \ do$
Aplicamos → directiva HLS PIPELINE
$ibuf[i] \leftarrow ibuf[i+1]$
Accedemos al siguiente índice i
end
$ibuf[2M-1] \leftarrow p$

3. Extracción de características:

$n = 0;$
$while\ \ n < M\ \ do$
3a. Calculamos barrido básico V:
Aplicamos → directiva HLS PIPELINE
$add \leftarrow (ibuf[M] - ibuf[M+(n+1)]) \times (ibuf[M] - ibuf[M+(n+1)])$
$sus \leftarrow (ibuf[0] - ibuf[n+1]) \times (ibuf[0] - ibuf[n+1])$
$V[n] \leftarrow V[n] + add - sus$

3b. Opción: aplicamos filtro de media móvil previo a V:
$movingAvergFilter(V'[n], V[n], wndSize)$

3c. Opción: diezmamos V por un factor d y calculamos posiciones de mínimos:
#ifdef $decimateV$ → if $(counterDecimate == d)$ then
$computeMinimaLocation(outFeatureVector[n], V'[n])$
#endif
$counterDecimate++$

Accedemos al siguiente índice n
end

Salida: vector $\boldsymbol{outFeatureVector}$ de tamaño M con 1 bit por posición indicando la presencia '1' o no '0' de un mínimo local

Figura 3.40. Pseudocódigo para el algoritmo de barrido o *sweep*, que se aplica para cada nueva muestra de entrada.

UNIVERSIDAD DE MÁLAGA

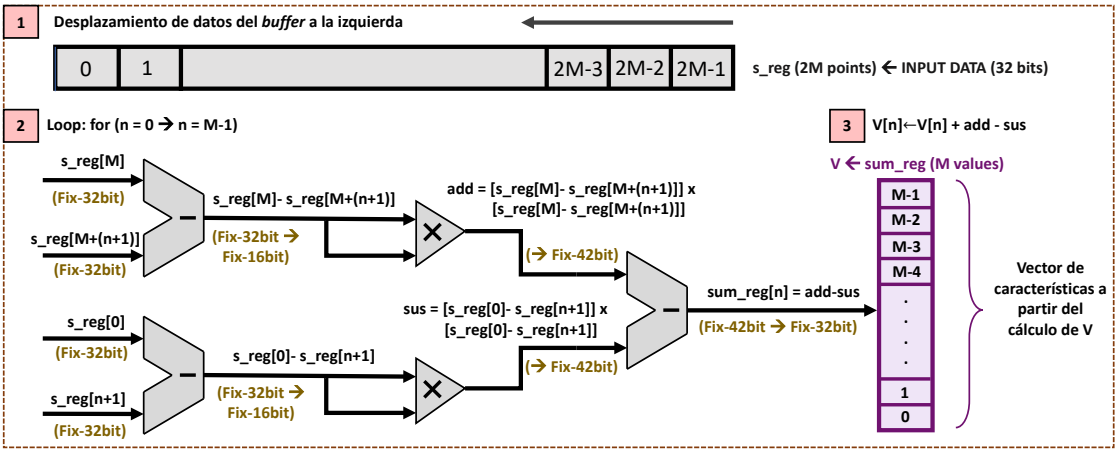


Figura 3.41. Modelado hardware aproximado para el cálculo de V (3.3) en tiempo real.

la síntesis RTL del algoritmo. Aunque existen varias opciones de optimización, en esta tesis se han explorado directivas que, en la síntesis lógica, permiten la concurrencia de procesos en mayor o menor medida. En concreto se ha trabajado con las siguientes:

- Directiva DATAFLOW. En una descripción escrita en System C, las operaciones se ejecutan de manera secuencial. Sin embargo, HLS automáticamente trata de minimizar la latencia y mejorar la concurrencia entre procesos. Con esta directiva, HLS además puede crear RAMs o FIFOs que permitan a funciones o bucles consumidores empezar a ejecutarse antes de que los módulos productores se hayan completado. De esta forma, estos métodos pueden operar en paralelo, disminuyendo la latencia y mejorando el *throughput* del diseño RTL.

- Directiva PIPELINE. Cuando esta directiva se aplica dentro de un bucle, la herramienta de HLS intenta que dentro del mismo, se procese una muestra cada cierto intervalo. Por defecto, este intervalo se corresponde a un ciclo de reloj, pero puede configurarse (parámetro *Interval*). Si no es posible aplicar dicho intervalo, HLS muestra un *warning*. Cuando esta directiva se aplica con la opción *rewind*, HLS permite a las tareas internas a la sección de código donde se aplique reiniciar su ejecución.

- Directiva UNROLL. Esta directiva permite crear múltiples copias del cuerpo de un bucle en la síntesis RTL. Con la opción *factor*, se pueden tener 2 accesos por ciclo y, de esta forma, explotar mejor las memorias de doble puerto que se sinteticen.

A modo ilustrativo, en la tabla 3.10 se muestra una estimación de recursos y latencia para el paso 3a del pseudocódigo en la figura 3.40 aplicando estas directivas. En cada síntesis, es importante que el diseño cumpla el *timing* (o retraso combinacional en un

UNIVERSIDAD DE MÁLAGA

camino de operaciones) dado por el período de reloj. En nuestro diseño, la frecuencia de reloj es de $f_{clk} = 100$ MHz, equivalente a un período o ciclo de reloj de 10 ns.

Tabla 3.10. Estimación de recursos hardware y latencia para diferentes combinaciones de las directivas de optimización DATAFLOW, PIPELINE y UNROLL.

Combinación de directivas	Estimación de recursos				Latencia en μs (peor caso)	Cumple *timing*
	BRAM (18kb) (de 280)	DSP (48b) (de 220)	FF (de 106400)	LUT Logic + LUTRAM (de 53200)		
Sin directivas	10	2	2567	2115	297	SI
DATAFLOW	10	2	1133	1820	307	SI
PIPELINE (Interval = 1)	10	2	2770	2310	195	SI
DATAFLOW + PIPELINE (Interval = 1)	10	2	1432	2017	195	SI
UNROLL	10	2048	16122	63535	256	NO
PIPELINE (rewind) + UNROLL (factor = 2)	10	4	5300	3644	195	SI
DATAFLOW + PIPELINE (rewind) + UNROLL (factor = 2)	10	4	2423	2810	195	SI

A partir de los datos presentados en la tabla 3.10, se observa que la aplicación de la directiva DATAFLOW reduce el uso de Flip-Flops y LUTs en comparación con un diseño sin directivas. Sin embargo, la latencia del algoritmo aumenta ligeramente. Por otro lado, al aplicar la directiva UNROLL, que busca maximizar la paralelización, se genera un diseño que no cumple el *timing* especificado por la frecuencia de reloj y utiliza más recursos de los permitidos por el SoC. El uso de la directiva PIPELINE combinado con la directiva global DATAFLOW consigue el diseño más concurrente sin comprometer el uso de recursos. En la implementación de este algoritmo, se ha optado por esta opción.

En el pseudocódigo mostrado en la figura 3.40, los pasos 3a, 3b y 3c corresponden a la aplicación en tiempo real de un filtro de media móvil, un diezmado por un factor d y el cálculo de las posiciones de los mínimos, respectivamente. Los pasos 3a y 3b, que involucran el filtro y el diezmado, son opcionales. Para este trabajo en particular, la implementación final prescinde del uso del filtro de media móvil, como se explicará más adelante. Los pseudocódigos correspondientes se presentan en la figura 3.42.

En la figura 3.42(a), el bucle del filtro de media móvil hace uso de la directiva PIPELINE. El filtro se ha implementado de acuerdo a cómo se explica en la sección 3.7. Para el diezmado, la implementación consiste en tomar un dato cada d muestras a la entrada. Por último, el cálculo de los mínimos consiste en comparaciones básicas para encontrar los cambios de signo al restar muestras consecutivas de la señal de entrada, 3.42(b).

UNIVERSIDAD DE MÁLAGA

Método: $movingAvergFilter(y, x, wndSize)$
Entradas:
- x (nueva muestra de entrada)
- sum (sumatorio para el cálculo del filtro)
- $wndSize$ (ventana para el filtro de media móvil, potencia de 2)

Variables:
- sum (sumatorio para el cálculo del filtro)
- $\boldsymbol{auxArray}$ (*buffer* de desplazamiento auxiliar de tamaño $wndSize$)

Procedimiento:
1. El *buffer* de desplazamiento $\boldsymbol{auxArray}$ lo declaramos para que se sintetice en una memoria BRAM:
Aplicamos → directiva HLS RESOURCE variable=$\boldsymbol{auxArray}$ core=RAM_2P_BRAM

2. Añadimos nueva muestra de datos x al *buffer* de desplazamiento y sumamos todos los valores del mismo en sum:
for $i \leftarrow 1\ to\ (wndSize - 1)$ do
    # Aplicamos → directiva HLS PIPELINE
    $auxArray[i] = auxArray[i + 1]$
    $sum = sum + auxArray[i]$
end
$auxArray[wndSize] = x$
$sum = sum + auxArray[wndSize]$

3. Dividimos por el tamaño de ventana $wndSize$ (desplazamos a la derecha tantas posiciones como indique la potencia de 2):
$y = sum \gg \log_2(wndSize)$

Salida: Nueva muestra calculada para el filtro de media móvil en tiempo real y

(a)

Método: $computeMinimaLocation(minimo, x)$
Entradas:
- x (nueva muestra de entrada)

Variables:
- $minimo$ (variable *booleana* para indicar si se encuentra o no un mínimo en la posición actual)
- $currentSample$ (índice indicativo de la posición actual del dato)
- $wndSize$ (ventana para el filtro de media móvil)
- $\boldsymbol{derivada}$ (*array* de dos posiciones para el cálculo de la derivada)
- $\boldsymbol{auxVector}$ (*buffer* auxiliar de dos posiciones)
- $flagMovAvrgFilt$ (*flag* indicando la aplicación del filtro de media móvil previo)

Procedimiento:
1. Guardamos el valor del dato de entrada:
$auxVector[1] \leftarrow x$

2. Calculamos el valor de la derivada como la diferencia entre el valor actual y el anterior:
$derivada[1] = auxVector[1] - auxVector[0]$

3. Detección de mínimo. Colocamos un '1' lógico a la salida si se halla un mínimo local:
if $(derivada[0] < 0\ \ \&\&\ \ derivada[1] > 0)$ then
    // Si se ha aplicado la opción de filtro de media móvil a V
    #ifdef $flagMovAvrgFilt$ → if $(currentSample > wndSize)$ then
        $minimo = 1$
    #endif
end

4. Actualizamos valores para siguiente iteración:
$derivada[0] \leftarrow derivada[1]$
$auxVector[0] \leftarrow auxVector[1]$

Salida: Variable $minimo$ indicando si para la posición actual existe un mínimo local '1', o no '0'

(b)

Figura 3.42. (a) Pseudocódigo para la opción de aplicación de un filtro de media móvil al cálculo de V, expresión (3.3). (b) Pseudocódigo para el cálculo de las posición de mínimos en V.

UNIVERSIDAD DE MÁLAGA

Estimación para el consumo de recursos hardware, de potencia y latencia de las diferentes opciones de barrido y comparación con FFT

Es importante recordar que en esta tesis, el rango de exploración de texturas está determinado por las condiciones de experimentación. En concreto, por la velocidad de exploración $v = 30$ mm/s, la aplicación del filtro *antialiasing* y la longitud del vector de salida M. Aunque los resultados experimentales de esta tesis corresponden a una longitud del vector de salida de $M = 1024$, la implementación en hardware puede independizarse del rango de exploración. De esta forma, se puede variar el valor de M para obtener el consumo de recursos, potencia y latencia en determinados escenarios. Con este fin, se ha realizado la síntesis e implementación del *sweep* básico (expresión (3.3)) y la transformada de Fourier para los valores de $M = \{8, 16, 32, 64, 128, 256, 512, 1024, 2048\}$. De esta manera, se obtiene una idea general del coste en tiempo, potencia y recursos para cada opción.

Por otro lado, para $M = 1024$, se decidió una implementación del barrido calculando las posiciones de mínimos como características (ver sección 3.7). En la tabla 3.7 veíamos que el mayor porcentaje de clasificación se obtenía para un factor de diezmado de $d = 8$ (correspondiente a 128 componentes del vector de características a la salida). Teniendo en cuenta esto, aquí además se muestra el consumo para la síntesis e implementación añadiendo el filtro de media móvil con tamaños de ventana de $N = \{2, 4, 8, 16\}$. En todos los casos, la implementación del algoritmo de barrido sigue la dada en el pseudocódigo de las figuras 3.40 y 3.42. Para la FFT se utiliza el IP core presintetizado *FFT Compiler* proporcionado por la librería de Xilinx®, como se describió en el apartado 3.8.1.

La herramienta HLS de Vivado® proporciona una estimación de los recursos que el algoritmo utilizará en una implementación en el SoC. Extrayendo estos datos para el *sweep* y los correspondientes a la FFT en Vivado®, se puede comparar el consumo de ambos algoritmos. Las figuras 3.43 y 3.44 muestran esta comparación en cuanto al número de recursos lógicos (LUT Logic y CARRY Logic), las figuras 3.45, 3.46 y 3.47 para elementos de memoria (Flip-Flops, LUTRAM, BRAM), la figura 3.48 muestra la suma de LUTRAM y BRAM en kb[4], la figura 3.49 para bloques DSP empotrados, la figura 3.50 el consumo de potencia y la figura 3.51 la latencia. En el eje de abscisas de cada figura se muestra el valor del tamaño de vector de salida M. En cada figura también se proporciona la información de los diferentes algoritmos de barrido para $M = 1024$.

[4]Cada LUT puede entenderse como una memoria de 64x1 bits y cada bloque empotrado de memoria RAM (BRAM) es de 18 kb (36 kb para doble puerto).

UNIVERSIDAD DE MÁLAGA

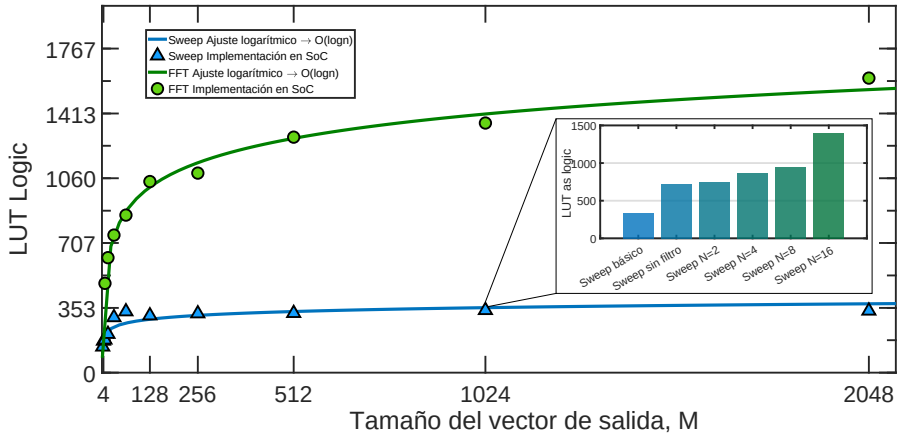


Figura 3.43. Comparación para el número de LUTs utilizadas como lógica o LUT Logic entre seis configuraciones del algoritmo de barrido, y la FFT.

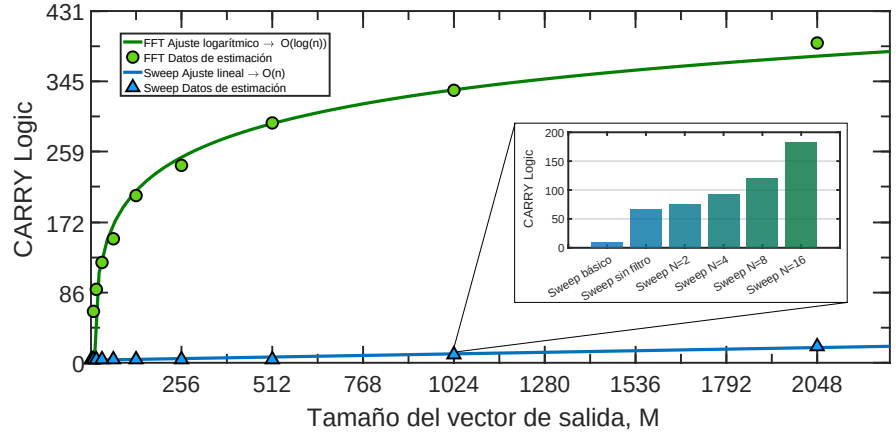


Figura 3.44. Comparación para el número de elementos lógicos de acarreo (CARRY Logic) entre seis configuraciones del algoritmo de barrido, y la FFT.

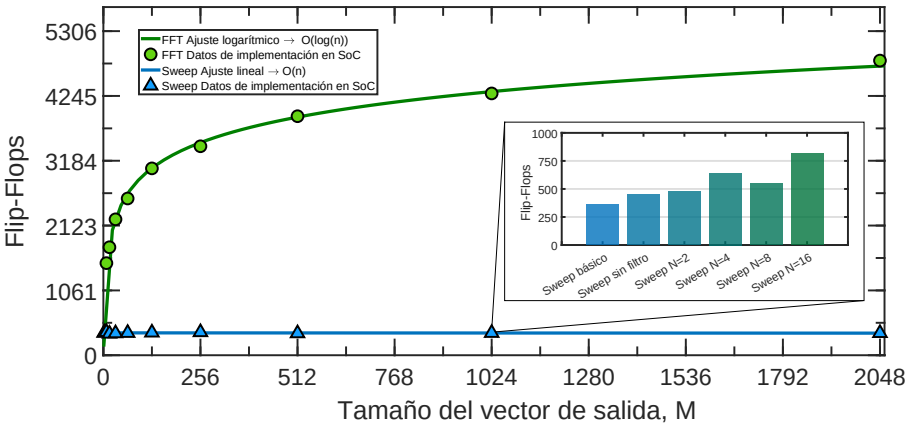


Figura 3.45. Comparación para el número de Flip-Flops o registros utilizados entre seis configuraciones del algoritmo de barrido, y la FFT.

UNIVERSIDAD DE MÁLAGA

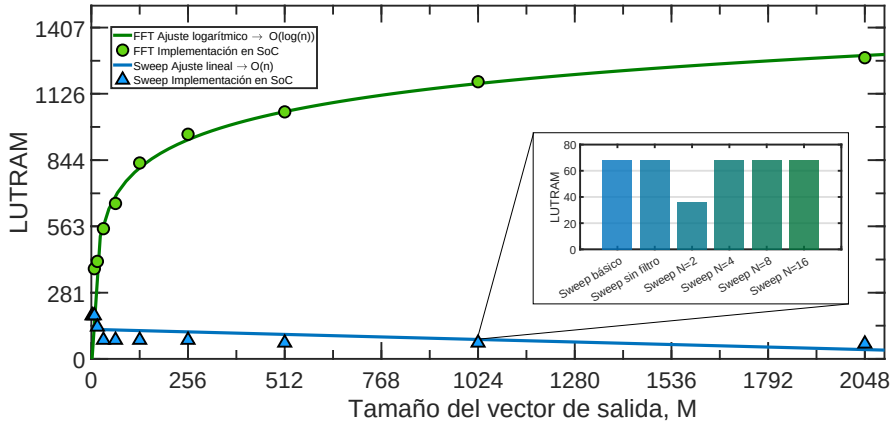


Figura 3.46. Comparación para la memoria distribuida o LUTRAM entre seis configuraciones del algoritmo de barrido, y la FFT.

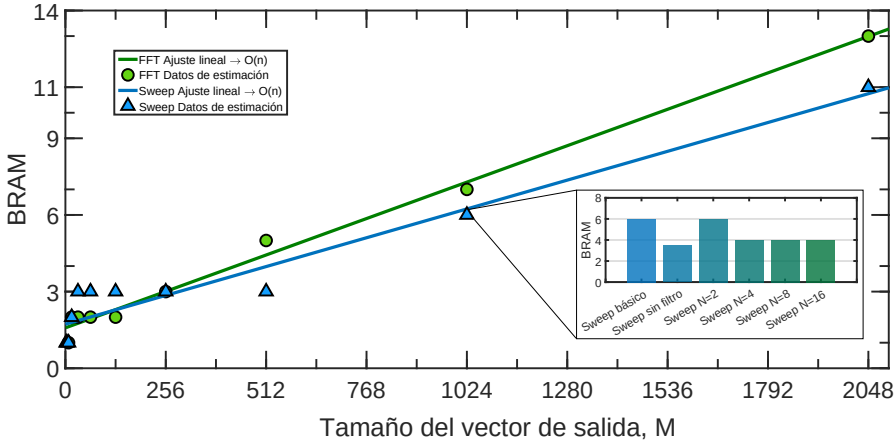


Figura 3.47. Comparación para los bloques BRAM empotrados entre seis configuraciones del algoritmo de barrido, y la FFT.

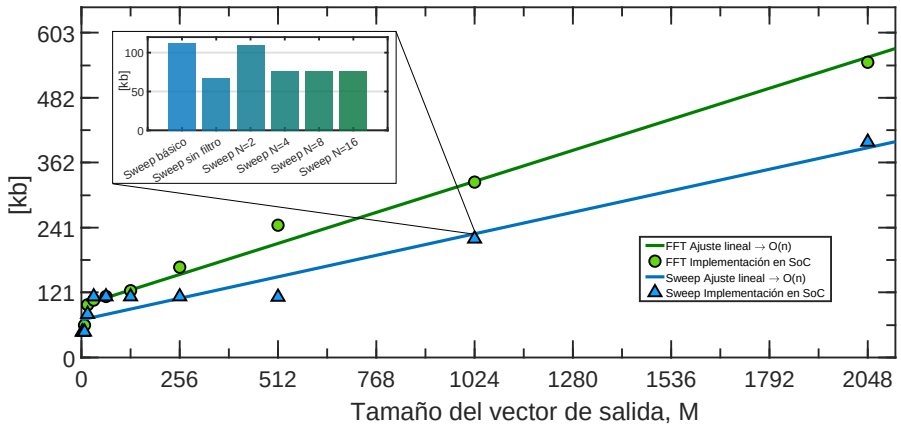


Figura 3.48. Comparación para la suma de memoria distribuida (LUTRAM) y bloques empotrados BRAM en kb, entre seis configuraciones del algoritmo de barrido, y la FFT.

UNIVERSIDAD DE MÁLAGA

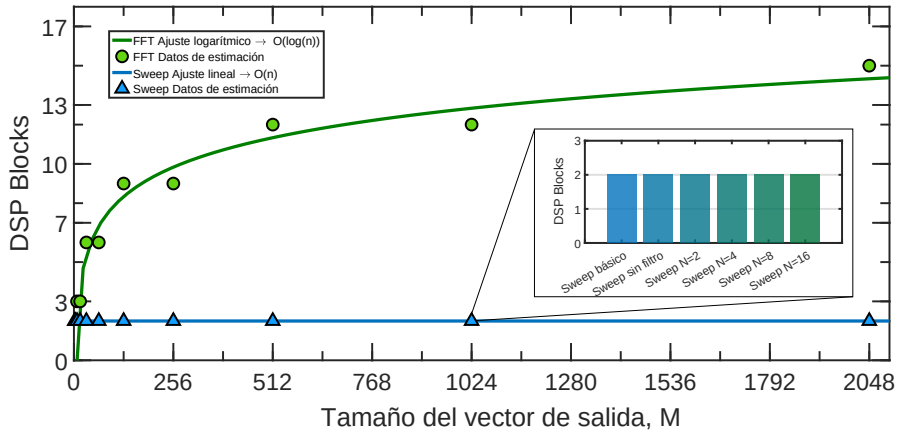


Figura 3.49. Comparación para los bloques DSP empotrados entre seis configuraciones del algoritmo de barrido, y la FFT.

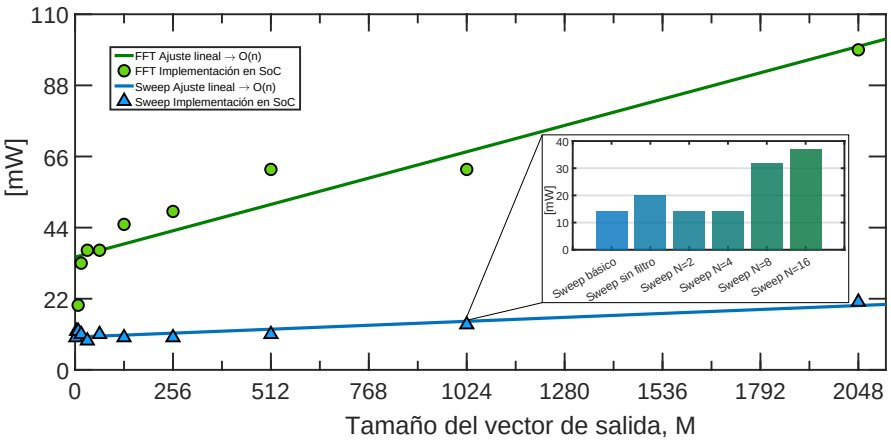


Figura 3.50. Comparación del consumo de potencia en mW entre seis configuraciones del algoritmo de barrido, y la FFT.

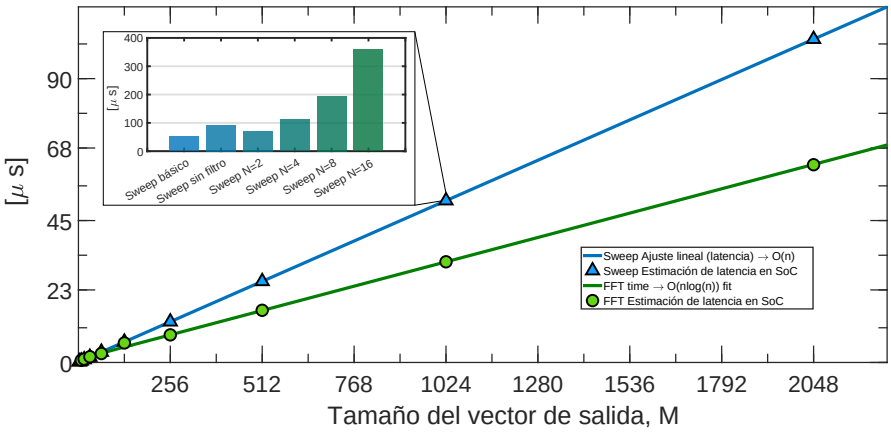


Figura 3.51. Comparación del tiempo de extracción de características t_{fe} (o latencia) entre seis configuraciones del algoritmo de barrido, y la FFT.

UNIVERSIDAD DE MÁLAGA

Como se puede comprobar en las figuras 3.43 y 3.44, el barrido consume menos recursos lógicos que la FFT en todos los casos. En cuanto al consumo de memoria, aunque los bloques BRAMs, figura 3.47, son similares en ambos casos, la FFT necesita más registros (Flip-Flops) y memoria distribuida (LUTRAM), figuras 3.45 y 3.46, que en el caso del barrido. Además, en la figura 3.48 vemos que la suma de LUTRAM y BRAM en kb es menor a la de la FFT siempre. Por otro lado, si nos fijamos en el caso $M = 1024$, vemos que con respecto al barrido básico, en general, el resto de configuraciones del barrido aumentan ligeramente el consumo. No obstante, éste sigue siendo muy bajo respecto a la FFT.

En cuanto al uso de DSPs empotrados, figura 3.49, se puede ver que el algoritmo de barrido sólo requiere de 2 DSPs, mientras que en el caso de la FFT se incrementa hasta 15, si se consideran 2048 componentes para el vector de salida. El consumo de potencia, figura 3.50 también es más pequeño para el barrido respecto a la FFT, del orden de unas decenas de mW. La latencia (o tiempo que tarda el algoritmo en proporcionar la salida una vez le llega la entrada) requiere de complejidad $\mathcal{O}(n)$ en el caso del barrido y crece como $\mathcal{O}(nlogn)$ para la FFT. En este caso, la latencia es mayor en el barrido. En cualquier caso, está en el orden de décimas de microsegundo, que es un tiempo pequeño para la identificación de texturas, teniendo en cuenta que la adquisición de las $2 \cdot M$ muestras de la superficie toma 4.22 segundos (expresión (3.7), sección 3.2).

Atendiendo a los datos de estimación de recursos en las figuras 3.43, 3.44, 3.45, 3.46, 3.47, 3.48, 3.50 y 3.51, por un lado, y los resultados de clasificación en la tabla 3.7 por otro, en esta tesis se decide implementar la opción para el barrido correspondiente a la no utilización de filtro de media móvil, cálculo de las posiciones de los mínimos, y aplicación de diezmado. Los resultados de clasificación de la implementación real se muestran en la subsección 3.9. En la subsección 3.8.4, se dan datos del consumo de todos los elementos de la arquitectura hardware en la figura 3.38.

3.8.3. Implementación del algoritmo de Goertzel en SoC

La implementación del algoritmo de Goertzel considerada es para señales reales en el tiempo. Para llevarla a cabo, a partir de la descripción en espacio de estados dada por las ecuaciones (3.15) y (3.16), la estructura de la figura 3.8, y con la ayuda de la información proporcionada en [113], se escribe un código con la herramienta de Vivado® HLS, que se puede consultar en el repositorio público *https://github.com/raulLkRb/thesis.git*. El pseudocódigo correspondiente es el que se proporciona en la figura 3.52.

La directiva de optimización 'HLS ALLOCATION' mostrada en el pseudocódigo de la figura 3.52 se utiliza para que el número de DSPs utilizados en la síntesis lógica sea

UNIVERSIDAD DE MÁLAGA

Entradas:
- x (nuevo dato de entrada)

Variables:
- $ibuf$ (*buffer* interno para almacenar las muestras de datos)
- N (longitud para el cálculo del algoritmo de Goertzel)
- K (número de frecuencias a calcular)
- $variableAuxiliar$ (variable auxiliara para el cálculo de la parte real de la salida)
- $coefReal$ (vector de coeficientes reales de tamaño K para el cálculo de las frecuencias)
- $coefImag$ (vector de coeficientes imaginarios de tamaño K para el cálculo de las frecuencias)

Procedimiento:
Aplicamos → directiva HLS DATAFLOW

1. Obtenemos nueva muestra p_m de los datos de entrada:
$p_m \leftarrow x$

2. Almacenamos p_m y desplazamos una posición a la izquierda el *buffer* interno de desplazamiento $ibuf$ de tamaño N:
$i = 0;$
$while\ \ i < N\ \ do$
Aplicamos → directiva HLS PIPELINE
$ibuf[i] \leftarrow ibuf[i+1]$
Accedemos al siguiente índice i
end
$ibuf[N-1] \leftarrow p_m$

3. Calculamos los K índices correspondientes
for $i \leftarrow 1\ to\ K$ do
# Aplicamos → directiva HLS ALLOCATION instances=mul limit=1 operation
$s1 = 0$
$s2 = 0$
for $i \leftarrow 1\ to\ N$ do
$s0 = auxArray[i] + coefReal[ind] \times s1 - s2$
$s2 = d1$
$s1 = s0$
end

// Cálculo de la salida del algoritmo (parte real y parte imaginaria)
$variableAuxiliar = coefReal[ind] \times s1$
$variableAuxiliar = variableAuxiliar \gg 1$
$salidaReal[ind] = variableAuxiliar - s2$
$salidaImag[ind] = coefImag[ind] \times s1$
end

Salida: vector $\boldsymbol{salidaReal}$ y vector $\boldsymbol{salidaImag}$, ambos de tamaño K, que contienen la parte real e imaginaria de los coeficientes en frecuencia calculados.

Figura 3.52. Pseudocódigo correspondiente al algoritmo de Goertzel.

del menor número posible. Más adelante se mostrarán resultados con esta directiva y sin ella.

Otra puntualización importante en el pseudocódigo de la figura 3.52 es la aparición de los dos vectores *coefReal* y *coefImag*. Estos vectores, de tamaño K, contienen coeficientes reales e imaginarios, necesarios para el cálculo de las componentes en frecuencia de Goertzel [112]. El cálculo de los mismos se muestra a continuación. Para cada índice k, los coeficientes reales vienen dados por (3.30):

UNIVERSIDAD DE MÁLAGA

$$\textit{coefReal}[k] = 2 \cdot cos\left(\frac{2\pi k}{N}\right) \tag{3.30}$$

Por otra parte, también para cada índice k, los coeficientes imaginarios vienen dados por (3.31):

$$\textit{coefImag}[k] = sen\left(\frac{2\pi k}{N}\right) \tag{3.31}$$

En la implementación del algoritmo, estos coeficientes se pueden calcular de manera *offline*, por lo que no es necesario calcularlos en cada iteración. Esto supone una ventaja respecto a la FFT, ya que de otra forma, el número de operaciones y, por tanto, la complejidad computacional del algoritmo aumentaría considerablemente.

Una vez que el código en HLS (ver figura 3.52) se sintetiza, se genera un IP Core que se puede utilizar en el diseño de la arquitectura en Vivado®. La arquitectura y camino de datos utilizados son los mismos que se presentaron en la figura 3.38. En este caso, se sustituye el algoritmo de barrido, o el cálculo de la FFT, por la propuesta de algoritmo de Goertzel. Una vez se implementa el diseño, se comprueba con el entorno de desarrollo SDK de Vivado® que la clasificación proporcionada es correcta, ver subsección 3.8.5. En la tabla 3.11 se muestra el consumo de recursos lógicos y de memoria en hardware, así como de potencia y latencia para tres implementaciones diferentes del algoritmo de Goertzel. Además, a modo de comparativa, se incluyen los datos de consumo para el barrido y FFT, ya mostrados con anterioridad en la tabla 3.12, así como los datos de latencia y porcentajes de clasificación (estos últimos mencionados en la subsección 3.7.3).

A partir de los datos de la tabla 3.11, se puede ver que el algoritmo de Goertzel consume menos recursos lógicos, de memoria y potencia que la implementación de la FFT. Sin embargo, el tiempo de extracción de características o latencia sí es superior a la FFT, aunque en menos de $116\mu s$ se obtienen las características que se envían al clasificador.

Un aspecto interesante es la utilización de la directiva de optimización 'HLS ALLOCATION'. Cuando ésta se aplica, se consigue reducir el número de DSPs en la implementación de 24 a 8. Aunque esto tiene un efecto en el uso de memoria distribuida (LUTRAM), este incremento no afecta apenas al consumo global. Además, el resto de valores bajan respecto a la versión sin directiva. Finalmente, si se compara el algoritmo de Goertzel con $K = 16$ frente al barrido, se puede ver que la mayoría de valores son similares.

UNIVERSIDAD DE MÁLAGA

Tabla 3.11. Datos del consumo real de recursos lógicos y de memoria en hardware, así como el consumo de potencia y latencia, para la implementación del algoritmo de Goertzel con $N = 1024$, con $K = 4$, $K = 8$ y $K = 16$, y un factor de control $p = 10$. Para poder comparar con barrido y FFT también se incluye el consumo de ambos para $M = 1024$, así como los porcentajes de clasificación para cada caso. Estos datos se recogen tal y como los proporciona el informe de utilización de Vivado IDE®.

Algoritmo	LUTRAM (de 53200)	BRAM 18kb (de 280)	Pares de FF (de 53200)	LUT Logic (de 53200)	DSP 48b (de 220)	F7 MUXES (de26600)	Consumo de potencia (mW)	Latencia (μs)	Porcentaje de clasificación
Goertzel $K = 16$	0	3	405	726	24	0	51	116	84.8 %
Goertzel $K = 16$ (directiva 'HLS ALLOCATION')	48	1	282	598	8	0	20	114.8	84.8 %
Goertzel $K = 8$	0	3	379	678	24	0	45	58	79.8 %
Goertzel $K = 8$ (directiva 'HLS ALLOCATION')	48	1	282	609	8	0	20	57.4	79.8 %
Goertzel $K = 4$	2	3	402	1048	48	61	2	34	58.9 %
Algoritmo *sweep* ($d = 8$, posiciones de mínimos)	36	5.5	227	706	2	0	23	92.4	89.2 %
Módulo HLS-Vivado Shift Buffer + FFT	846	4.5	2306	1965	12	0	62	31.9	91.4 %

3.8.4. Consumo real de recursos hardware y potencia en la parte PL

Los informes de utilización de recursos (*Utilization Report*), proporcionados por Vivado® muestran el consumo real para todos los módulos hardware de la arquitectura de la figura 3.38. A modo de resumen, la tabla 3.12 muestra el consumo global de recursos y potencia en el dispositivo Zynq-7020® para los diferentes módulos en la arquitectura de la figura 3.38.

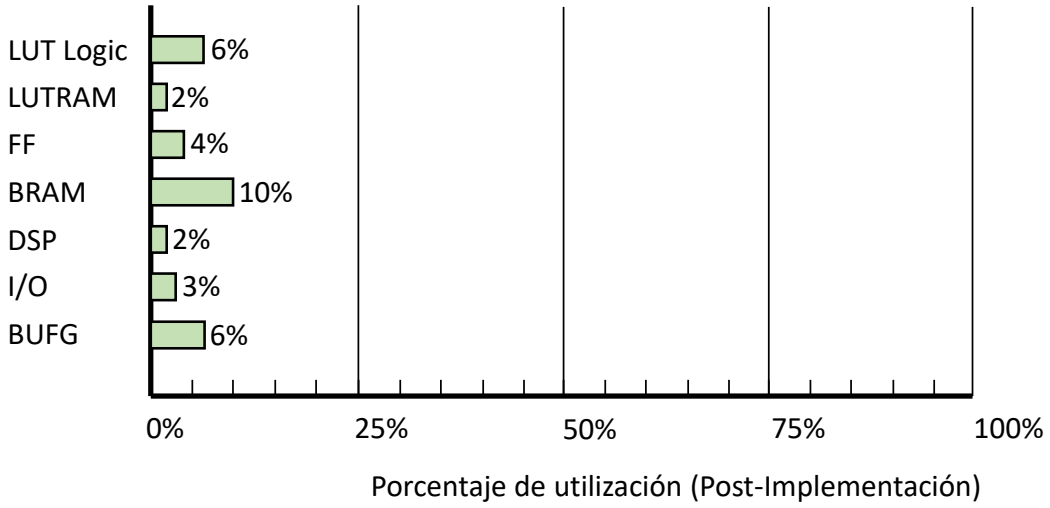
Tabla 3.12. Datos del consumo real de recursos lógicos y de memoria para la implementación de los diferentes módulos hardware en el diseño global de la figura 3.38, tal y como los proporciona el informe de utilización o 'Utilization Report' de Vivado IDE®.

Módulo hardware	LUTRAM	BRAM	Pares de FF	LUT Logic	DSP	F7 MUXES	Consumo de potencia (mW)
Controlador SPI	0	0	21	41	0	0	< 1
Filtro *antialiasing*	133	1	109	42	3	16	10
Block RAM/AXISTREAM Interface	0	0.5	14	35	0	0	2
FIFO de salida	0	5.5	29	67	0	0	4
Módulo DMA	62	1	607	846	0	1	5
Algoritmo *sweep* ($d = 8$, posiciones de mínimos)	36	5.5	227	706	2	0	23
Módulo HLS-Vivado Shift Buffer + FFT	846	4.5	2306	1965	12	0	62

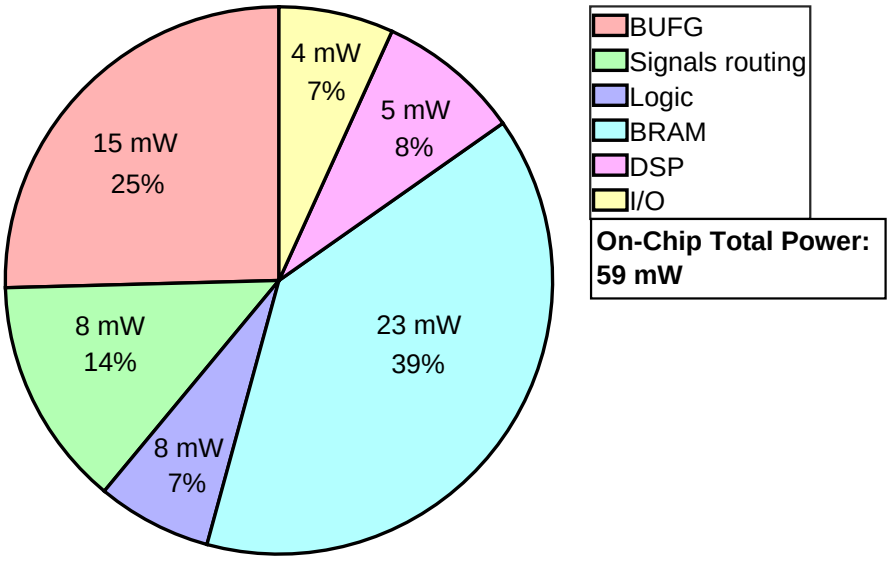
Por otro lado, la opción de 'Project Summary' en Vivado® proporciona el consumo global post-implementación, en porcentajes de utilización respecto al número de recursos total. Este consumo se calcula considerando también la utilización de *buffers* de

UNIVERSIDAD DE MÁLAGA

entrada/salida (I/O) y los de BUFG[5]. La figura 3.53 muestra estos porcentajes extraídos directamente de Vivado® para la implementación real en SoC. Como se puede ver, el consumo global de recursos no excede del $10\,\%$ (bloques BRAMs) del total para ninguno de los campos. La potencia total global en hardware también se mantiene baja, siendo de apenas 59 mW.



(a)



(b)

Figura 3.53. Consumo global 'Post-Implementation' para la implementación del *sweep* en SoC, en porcentajes de utilización respecto al número total de recursos que proporciona el dispositivo Zynq-7020®. (a) Consumo global de recursos hardware. (b) Consumo de potencia en hardware para las diferentes partes del diseño.

[5]Los *global buffers* (BUFG) son componentes diseñados para gestionar y distribuir eficientemente señales de alta *fan-out*, es decir, señales que necesitan llegar a numerosos destinos dentro del dispositivo. Son especialmente críticos para el manejo de las señales de reloj, ayudando a minimizar problemas de sincronización y tiempos de propagación.

UNIVERSIDAD DE MÁLAGA

3.8.5. Implementación de la parte PS del diseño

El entorno de desarrollo SDK de Vivado® (sección 2.5 del capítulo 2) permite escribir el código en C que se ejecutará en el ARM® de doble núcleo Cortex-A9™ embebido en el SoC Zynq7000®. Se comentan la metodología llevada a cabo y cómo se ha implementando. El código en C correspondiente a esta parte se puede consultar en el repositorio público *https://github.com/raulLkRb/thesis.git*.

Ejecución en el ARM® del SoC

En este apartado se describe el código utilizado para la clasificación cuando se están recibiendo los vectores de características en tiempo real, calculados por el algoritmo de barrido, FFT, o algoritmo de Goertzel, que están implementados en el hardware. La ejecución del código en el ARM® funciona como una interfaz entre el productor (parte PL) y el consumidor de datos (parte PS). Gracias a FreeRTOS® el proceso completo se puede llevar a cabo en diferentes hilos o *threads*. Estos hilos operan emulando una ejecución en paralelo en la que acceden a memoria y recursos compartidos. El *scheduler* del RTOS se encargará de multiplexar en tiempo los accesos de cada *thread* a estos recursos, según se requiera en cada momento.

En la figura 3.54 se puede ver el diagrama de flujo para la clasificación en diferentes clases, llevada a cabo por el ARM® del SoC. En este caso, el procedimiento consta de dos *threads* principales y un *thread* secundario que puede utilizarse o no. Inicialmente, se establecen todos los parámetros del sistema, incluyendo memoria y periféricos. Después, se configura el FreeRTOS® junto a la pila lwIP. A continuación se inicializan un semáforo y dos colas de datos que se utilizarán para el intercambio de datos entre hilos. Después se inicializan la DMA así como su ISR. FreeRTOS® permite establecer prioridades a la ejecución de los diferentes *threads*. El hilo más crítico es el THREAD 1, ya que se encarga de la recepción y encolado de datos, por lo que se ha establecido con mayor prioridad respecto a los otros dos. Por último, se inicializan los centroides obtenidos en la fase de entrenamiento (ver sección 3.6). De forma más específica, la ejecución de cada *thread* se describe a continuación:

1. El primer hilo (THREAD 1) comprueba primero si hay un número determinado de grupos de datos en la FIFO de salida, ver figura 3.38. Esta FIFO evita la pérdida de datos por parte del hardware si la frecuencia del mismo es más alta que en el ARM®. Por tanto, cada vez que se obtengan M muestras a la salida del algoritmo, estas se almacenarán en la FIFO, hasta que el consumidor esté disponible para recibir los últimos NG grupos de M datos que se hayan guardado. Para pedir una

UNIVERSIDAD DE MÁLAGA

transferencia de M datos a través de la DMA, el procesador primero baja el flag de interrupción ($RxDone = 0$). A continuación activa las interrupciones para la DMA, y toma el semáforo creado al inicio para asegurarse de que la recepción de datos es segura y no va a ser interrumpido por otro hilo. Después, pide la transferencia de datos desde la parte PL a la parte PS. En este momento, la DMA envía los datos desde la memoria DDR hacia el procesador. En el ARM® se activa la ISR para la DMA, donde primero se desactivan las interrupciones de DMA. Una vez se produce la interrupción de recepción ($IrqStatus\&AxiDmaInterrupt$), se activa el flag de interrupción ($RxDone = 1$) y se libera el semáforo. Finalmente, en la ejecución principal del THREAD 1, se encolan los M datos recibidos del hardware en la primera de las colas creadas al inicio (cola1). El espacio que estaba ocupado en la FIFO finalmente queda libre de nuevo para continuar almacenando nuevas muestras procedentes de la salida del algoritmo.

2. Este hilo (THREAD 2) se encarga de la clasificación. Primero obtiene el vector de características de M datos de la cola1. A continuación, con los centroides guardados en la memoria del ARM®, aplica el algoritmo *k-means* en tiempo real, tal y como se mostró en el pseudocódigo de la figura 3.34, sección 3.6, para inferir la clase (el centroide) más próxima al vector de datos, que finalmente se encola en la segunda de las colas creadas al inicio (cola2).

3. Este hilo (THREAD 3) es opcional en esta aplicación. Si no se utiliza, el THREAD 2 puede enviar la clase por el puerto serie en lugar de encolarla. En caso de utilizarse, el hilo inicialmente lee los datos de clasificación de la cola2. A continuación se envía la clase correspondiente mediante el protocolo lwIP en formato cadena de caracteres. Para lograr esto es necesario ejecutar un terminal Telnet en el ordenador.

Como se comentó previamente, en una etapa inicial de la tesis se monitorizó la recepción de los datos brutos procedentes de todos los tácteles de la matriz del sensor. Para este escenario, la utilización de la pila lwIP es necesaria, ya que el puerto serie limita la velocidad de transmisión de los datos correspondientes a los 55 tácteles de la matriz. El THREAD 2 no se utiliza en este caso, puesto que sólo nos interesa la información táctil, no la clasificación. El THREAD 1 se encarga de transferir de la parte PL a PS tramas consecutivas de lectura de la matriz del sensor, es decir transmisiones en grupos de 55 datos. Estos datos se envían a la única cola creada al inicio para este caso, y finalmente, se desencolan y envían por lwIP en el THREAD 3.

UNIVERSIDAD DE MÁLAGA

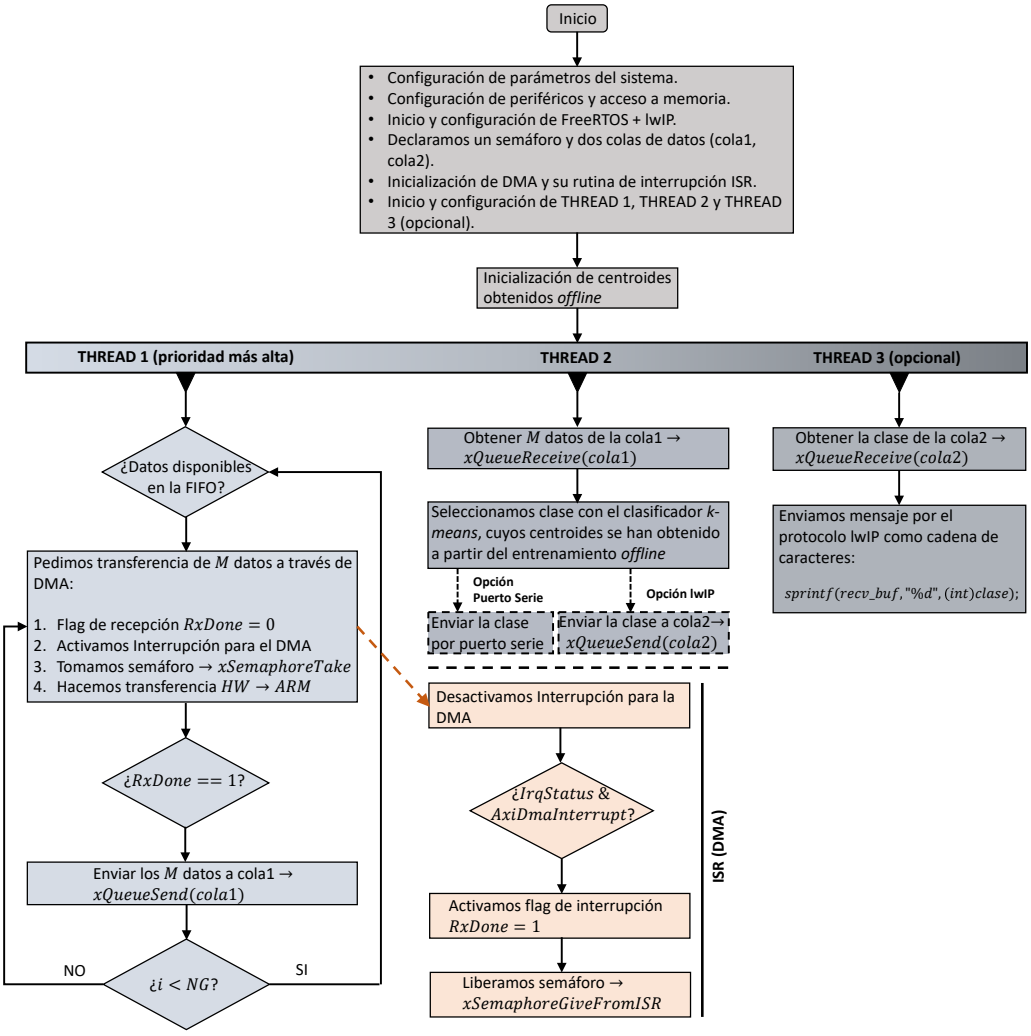


Figura 3.54. Diagrama de flujo resumido del código en C implementado en el ARM® a partir del entorno de desarrollo SDK. En el diagrama se muestra la ejecución de los tres hilos o *threads* utilizados y la ISR correspondiente al módulo DMA.

3.9. Resultados de clasificación

Esta sección se centra en los resultados obtenidos a partir de la clasificación en tiempo real en el SoC. Se presenta primero una comparación entre barrido y FFT, y a continuación, se presenta el resultado para la mejor implementación del algoritmo de Goertzel.

3.9.1. Comparativa entre barrido y FFT

Como se indicó en la subsección 3.8.2, se selecciona la implementación del algoritmo de barrido, aplicando diezmado y sin filtrado, y se calculan las posiciones de los mínimos como vector de características. La figura 3.55 muestra el porcentaje de clasificación resultante de esta implementación. Es importante tener en cuenta que, en el caso de la FFT, el espectro de frecuencias es simétrico y toda la información frecuencial

UNIVERSIDAD DE MÁLAGA

se encuentra en su primera mitad. Para cada factor de diezmado utilizado en el *sweep* ($d = \{128, 64, 32, 16, 8, 4, 2\}$), nos quedamos por tanto, con una fracción de la primera mitad del espectro FFT.

El eje de abscisas de la figura 3.55 representa el tamaño del vector de características y el eje de ordenadas muestra el porcentaje de clasificación para el barrido y la FFT. Para ambos métodos se presentan dos curvas, una de las cuales se obtuvo con restricciones de hardware no consideradas en la otra. Para el cálculo de los coeficientes de la FFT, estas restricciones implicaron el uso de las funciones MEX pre-compiladas de Matlab® (ver subsección 3.5.2). En cuanto al algoritmo de barrido, los resultados para la curva etiquetada como "Sweep FPGA test" se obtuvieron en tiempo real con el algoritmo de extracción de características y el clasificador implementado en el SoC, aunque el entrenamiento se realizó *offline*, tal como se ha mencionado previamente.

Como se puede apreciar en la figura 3.55, el porcentaje de clasificación obtenido con los coeficientes de la FFT como vectores de características, se estabiliza después de los primeros 64 coeficientes. Este resultado corrobora que la mayoría de la información táctil está contenida en frecuencias espaciales bajas del espectro. No obstante, es importante tener en cuenta que la respuesta en frecuencia de la señal depende tanto de las texturas como del sensor táctil, específicamente de la forma y composición del *cover* en contacto con las texturas. La figura 3.55 muestra que también se logra un buen porcentaje de clasificación con el algoritmo de barrido. El resultado se estabiliza a partir de 128 componentes del vector de características. Para un tamaño menor, los porcentajes caen, lo que puede ser debido entre otros factores, a la aplicación de mayores factores de diezmado, ver sección 3.7. Por otro lado, el espacio requerido para el caso del *sweep* es menor que para la FFT porque solamente se envían las posiciones de los mínimos al clasificador. En la curva de la figura 3.55, el punto de máximo rendimiento se obtiene al enviar 512 componentes del vector de características al clasificador, al aplicar el *sweep* en el SoC en tiempo real. En este punto, se logra un porcentaje de clasificación de $89.17\,\%$ para el *sweep* y de $91.4\,\%$ para la FFT.

3.9.2. Evaluación del algoritmo de barrido a través de las matrices de confusión

Para evaluar la eficacia del algoritmo de barrido propuesto, se utilizan las matrices de confusión. Estas permiten comparar las predicciones realizadas por el clasificador con los valores reales de los datos del conjunto de prueba. Las figuras 3.56 y 3.57 muestran las matrices de confusión correspondientes al *sweep* y a la FFT, respectivamente. En ambos casos, los datos del conjunto de test incluyen *outliers* (ver sección 3.6). La precisión

UNIVERSIDAD DE MÁLAGA

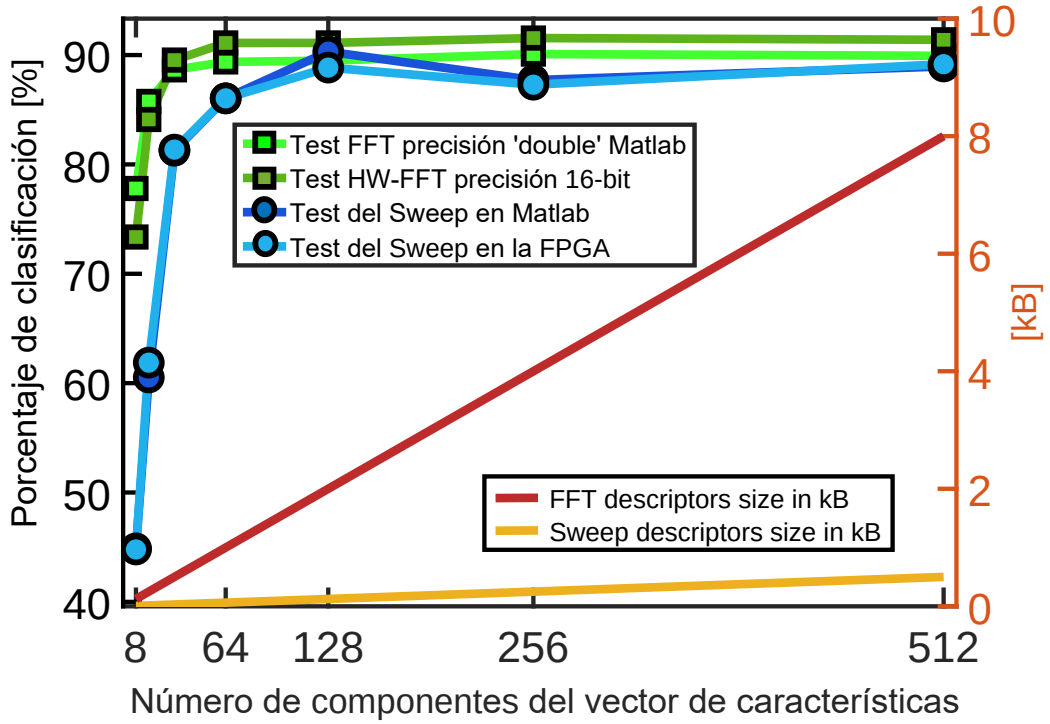


Figura 3.55. Porcentajes de clasificación para el algoritmo de *sweep* y FFT, dependiendo del número de componentes del vector de características que se envían al clasificador, y tamaño en kB del vector de características para ambos casos [39]. La longitud del vector de salida es $M = 1024$.

obtenida para cada caso se indica encima de cada figura. La velocidad de exploración se estableció en 30 mm/s y $M = 1024$. A partir de los resultados de las figuras 3.56 y 3.57, se puede ver que la FFT proporciona un porcentaje de clasificación superior en todos los casos, pero aún así el *sweep* obtiene también buenos resultados a pesar de consumir muchos menos recursos. En el siguiente apartado analizaremos las métricas de sensibilidad y especificidad para el *sweep*, que nos proporcionarán información sobre el rendimiento del sistema a la hora de clasificar entre las diferentes clases.

UNIVERSIDAD DE MÁLAGA

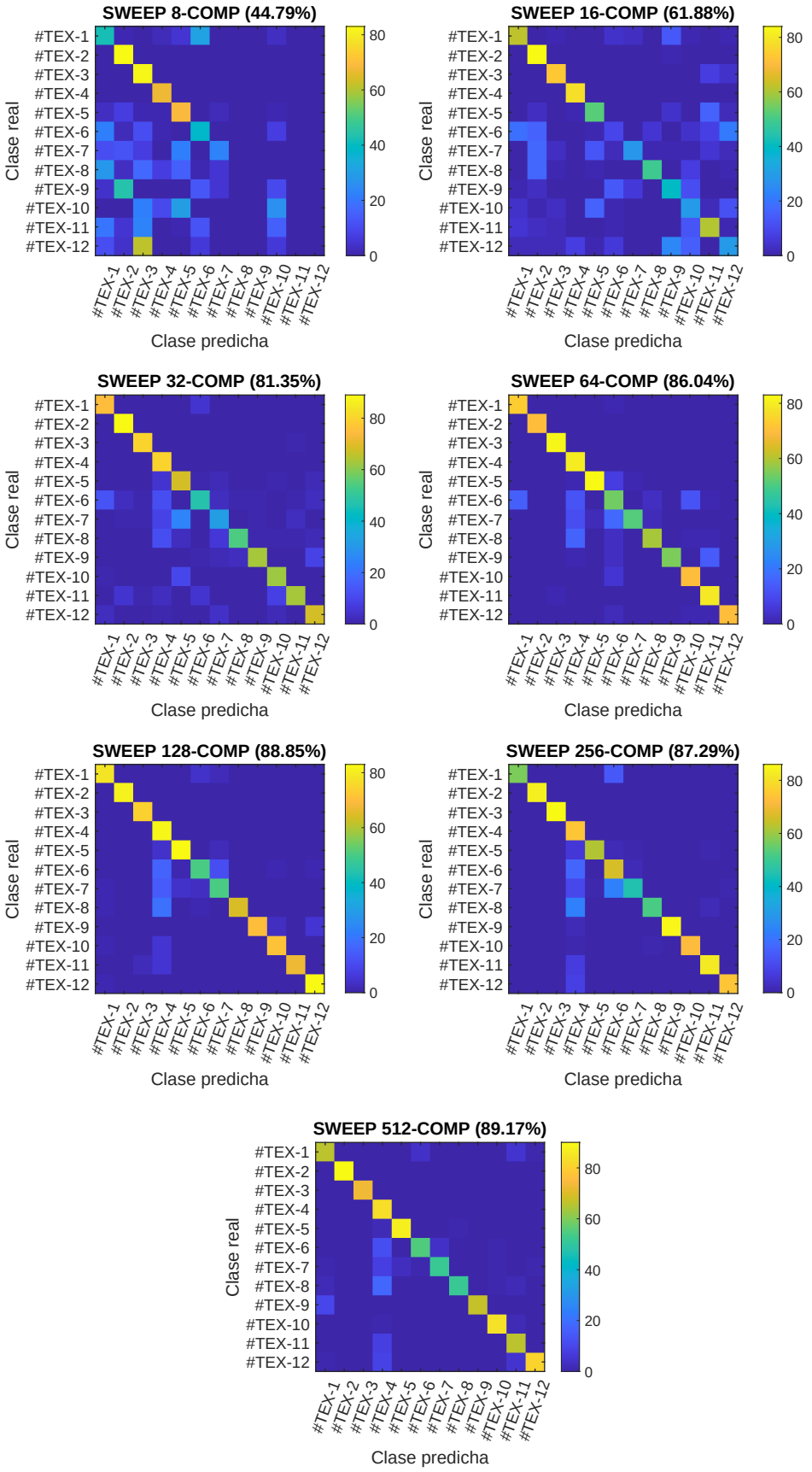


Figura 3.56. Matrices de confusión con los resultados del algoritmo de *sweep* para las texturas mostradas en la figura 3.17 y cada punto del gráfico 3.55, con $M = 1024$. Estos resultados se han extraído directamente a partir de un test en tiempo real en el chip Zynq-7020®.

UNIVERSIDAD DE MÁLAGA

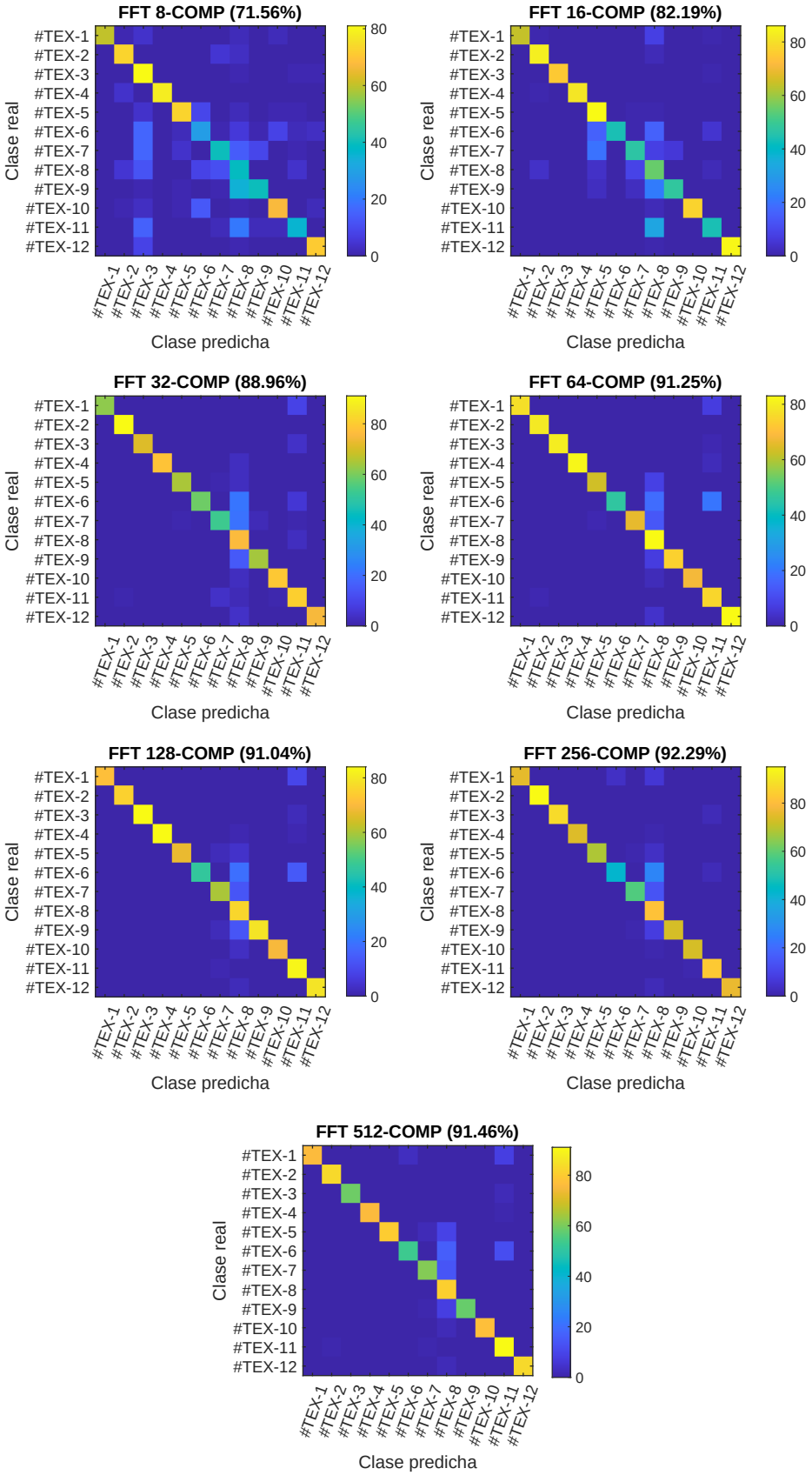


Figura 3.57. Matrices de confusión con los resultados de la FFT para las texturas mostradas en la figura 3.17 y cada punto del gráfico 3.55, con $M = 1024$. Estos resultados se han extraído a partir de las funciones pre-compiladas MEX de Matlab®.

UNIVERSIDAD DE MÁLAGA

3.9.3. Análisis de la sensibilidad y especificidad para cada textura en un espacio *Receiver Operating Characteristic* (ROC)

Para cada una de las 12 texturas de este estudio se pueden calcular cuatro métricas. De forma individual, para una textura i tendremos:

- Verdaderos positivos (VP): se refiere a las ocasiones en que la textura i es correctamente identificada como la textura i.
- Falsos positivos (FP): se produce cuando una textura distinta a la i es incorrectamente identificada como la textura i.
- Verdaderos negativos (VN): corresponde a las instancias donde una textura que no es la textura i, se identifica correctamente como una textura no i.
- Falsos negativos (FN): sucede cuando la textura i es incorrectamente clasificada como una textura diferente.

Sobre la base de estos valores, se derivan dos parámetros adicionales: sensibilidad y especificidad. La sensibilidad, también conocida como tasa de verdaderos positivos, mide la proporción de clasificaciones correctas de la textura i. Su cálculo se realiza según la ecuación (3.32):

$$Sensibilidad = \frac{VP}{VP + FN} \tag{3.32}$$

Por otro lado, la especificidad, o tasa de verdaderos negativos, refleja la habilidad de identificar correctamente las texturas que no son la i. Esta métrica se calcula a través de la ecuación (3.33):

$$Especificidad = \frac{VN}{VN + FP} \tag{3.33}$$

Estas métricas son de gran utilidad para evaluar la eficacia de un modelo en la correcta identificación de clases. Un espacio ROC provee una representación gráfica de la habilidad de un modelo clasificador para discriminar entre clases positivas y negativas en todos los posibles umbrales de clasificación. En la figura 3.58 se presenta el espacio ROC obtenido para cada textura según el número de componentes del vector de características enviadas al clasificador, cuando se aplica el algoritmo de barrido.

UNIVERSIDAD DE MÁLAGA

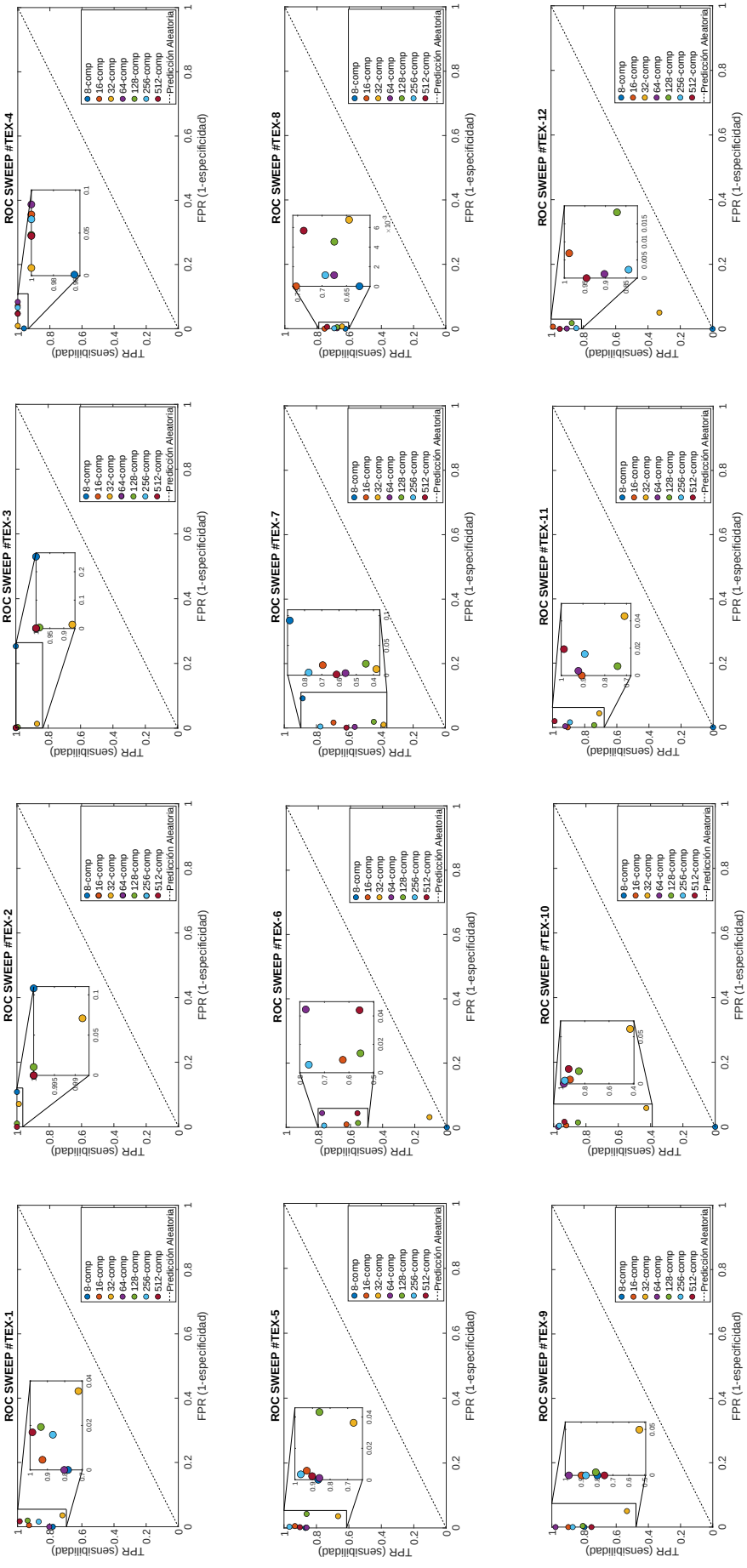


Figura 3.58. Representación en el espacio ROC para cada una de las 12 texturas, cuando se aplica el algoritmo de *sweep*. En cada gráfico, el eje vertical representa la sensibilidad (la proporción de verdaderos positivos), y el eje horizontal muestra la tasa de falsos positivos (1-especificidad). Un modelo perfecto se ubicaría en la esquina superior izquierda del gráfico.

UNIVERSIDAD DE MÁLAGA

3.9.4. Conclusiones en base a los resultados de clasificación del barrido

A partir de las matrices de confusión y las representaciones ROC para el algoritmo de barrido (consultar figuras 3.56 y 3.58) y diferentes tamaños del vector de características, podemos deducir varias conclusiones:

1. El rendimiento de la clasificación se incrementa al aumentar el número de componentes del vector de características. Se aprecia una correlación positiva entre el número de componentes enviadas al clasificador y la precisión en la clasificación. Esto se evidencia en las matrices de confusión, donde el número de casos incorrectamente clasificados (fuera de la diagonal principal) disminuye a medida que se incrementa el número de componentes.

2. De forma individual, el modelo clasifica mejor unas texturas que otras en diferentes escenarios. Por ejemplo, la textura #TEX-2 parece tener un rendimiento muy bueno en casi todos los casos, mientras que otras como la #TEX-6 y la #TEX-12 mejoran al aumentar el número de componentes del vector de características.

3. Los resultados indican que la sensibilidad aumenta a medida que se incrementa el tamaño del vector de características, siendo más notorio en el intervalo de 8 a 256 componentes. Esto sugiere que el modelo se beneficia de una mayor resolución, ya que proporciona una representación más detallada y precisa de la textura. Por otro lado, para tamaños del vector de características más pequeños (8 y 16 componentes), la sensibilidad varía bastante entre texturas, lo que indica que el clasificador tiene dificultades para distinguir entre texturas para estas resoluciones más bajas. Finalmente, a partir de 256 a 512 componentes, la mejora en la sensibilidad se estabiliza, indicando que a partir de este punto el aumento en el número de componentes del vector de características no aporta más información relevante para la clasificación de las texturas.

4. En cuanto a la especificidad, ésta es notablemente alta para la mayoría de los casos, con valores cercanos a la unidad para las texturas #TEX-2, #TEX-3 y #TEX-4. Esto sugiere que el modelo es cauteloso al clasificar las texturas, es decir, evita eficazmente los falsos positivos. A diferencia de la sensibilidad, no hay una tendencia de mejora con el número de componentes del vector de características, sino que se mantiene alta en todos los casos.

En conclusión, el análisis de los resultados sugiere que el tamaño del vector de características tiene un impacto significativo en la sensibilidad del modelo, pero no tanto en

UNIVERSIDAD DE MÁLAGA

la especificidad. Aumentar el número de componentes enviadas al clasificador mejora la capacidad del modelo para identificar las texturas, pero sólo hasta un límite (alrededor de 256 componentes), después del cual, este incremento no aporta beneficios adicionales. Además, el modelo muestra robustez al evitar falsos positivos. De manera general, el rendimiento del clasificador es satisfactorio al distinguir entre las 12 texturas estudiadas.

3.9.5. Resultados del algoritmo de Goertzel

En este apartado se dan los resultados obtenidos para la aplicación de la propuesta del algoritmo de Goertzel en el momento de presentación de la tesis.

El mejor caso en la tabla 3.9 (subsección 3.7.3) se dio para $K = 16$ y $p = 10$ (obteniendo un $84.8\,\%$ de precisión). La matriz de confusión para la implementación de este escenario es la que se presenta en la figura 3.59.

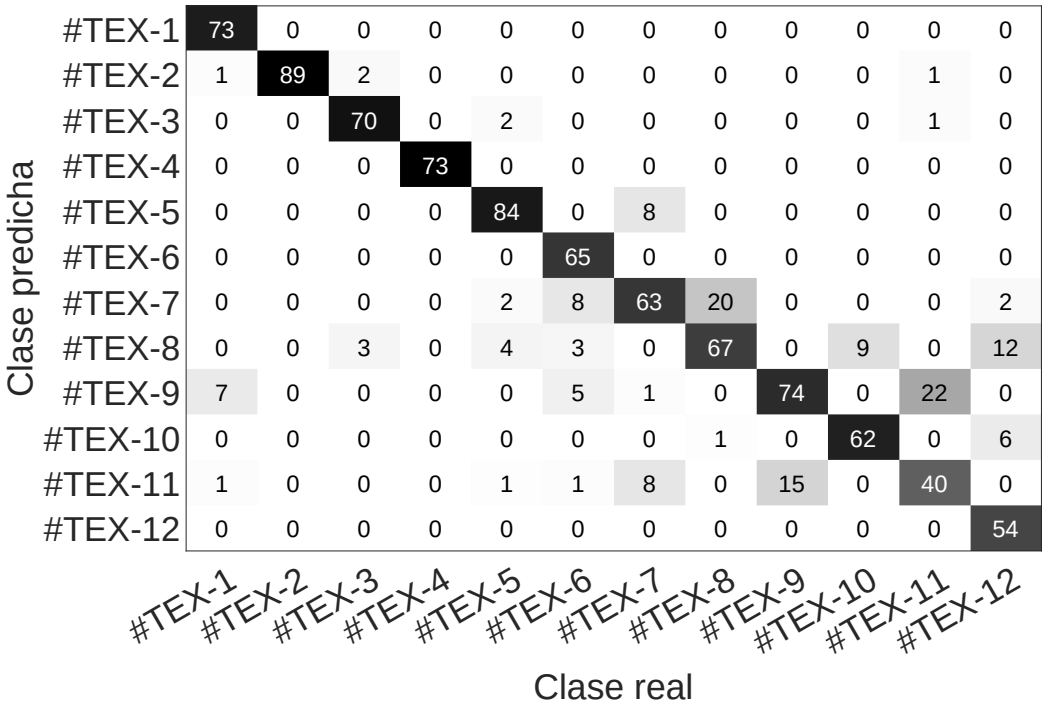


Figura 3.59. Matriz de confusión cuando se aplica el algoritmo de Goertzel con $N = 1024$, $K = 16$ y $p = 10$ sobre las 12 texturas de la figura 3.17.

Aquí, cabe recordar el porcentaje de clasificación alcanzado para las implementaciones del barrido y FFT consideradas en esta tesis. En el caso de la implementación óptima del barrido seleccionada ($d = 8$, envío de posición de mínimos, sin filtro de media móvil), se obtuvo un porcentaje de clasificación máximo de $89.2\,\%$. Con la FFT se tenía $91.4\,\%$. Estos números no están muy lejos del $84.8\,\%$ alcanzado con Goertzel.

UNIVERSIDAD DE MÁLAGA

3.9.6. Discusión general

Como se comentó en la introducción de este capítulo, sección 3.1, en el trabajo propuesto en [96], también se propone un algoritmo de identificación de texturas orientado a sistemas empotrados. En dicho trabajo, las señales de un acelerómetro 3D y un micrófono de sonido se usaban como entradas al algoritmo de extracción de características. El algoritmo proporciona la potencia de la señal en un conjunto de bandas de frecuencia espectrales, que se tomaban como características para identificar la textura. Dicho algoritmo no requiere almacenar todas las entradas complejas del vector, por lo que consume menos memoria. Por otra parte, es más complejo que el algoritmo de barrido que se propone en esta tesis. Por lo tanto, el consumo de recursos para su cálculo, así como el número de multiplicadores o DSPs, es mayor. Los autores presentaron una alternativa en [110] que mejora el *throughput* paralelizando, pero con el coste de aumentar los recursos computacionales. En resumen, esta propuesta requiere más recursos computacionales pero menos recursos de memoria que en nuestro caso.

No obstante, la comparación con otras propuestas no es fácil, puesto que depende de muchos aspectos. Primero, el rendimiento depende de la tecnología de sensor utilizada, porque los datos registrados se producen por la interacción entre el sensor y las superficies, a través de la presión del contacto, y con ciertos tipos de acciones de exploración [95]. Además, el tamaño y naturaleza del conjunto de texturas para testear el rendimiento también influye en los resultados. Sin embargo, el diseño del sensor y la elección del algoritmo de extracción de características pueden adaptarse según la aplicación final a la que estén destinados. Algunas propuestas superan el rendimiento humano a la hora de diferenciar entre texturas [90], pero podría no ser necesario implementarlas en manos artificiales al menos que se centren en tareas más especializadas. Los autores en [95] argumentan que con pocas características frecuenciales se pueden distinguir texturas, mientras que en el trabajo en [96] y en este trabajo [39] también se muestra lo mismo, puesto que el porcentaje de clasificación es incluso alto cuando las características extraídas son pocas. Por lo tanto, el algoritmo de extracción de características podría simplificarse.

Nuestra propuesta de algoritmo de barrido proporciona información acerca de la posición de las componentes de frecuencia en el espectro, pero no acerca de su potencia. La simplificación del algoritmo puede limitar su rendimiento, pero, como se dice en [110], el uso de más sensores pueden compensar la pérdida de precisión con respecto al método de extracción basado en FFT. Este podría ser el caso del sensor táctil considerado en esta tesis. Solamente con un canal se ha comprobado que la aproximación es factible, pero se podrían usar más canales (16 canales se usan en [95]). En este caso, la simplicidad

UNIVERSIDAD DE MÁLAGA

del hardware de extracción de características lo hace más escalable. La adquisición de la matriz táctil se puede hacer en paralelo con el cálculo de otras características como los momentos de la imagen [37] y las características para la detección de texturas. Finalmente, un rendimiento mejor en la clasificación se puede lograr si se cambian las velocidades y direcciones de exploración [95] [90], lo que es posible en un sistema robótico.

Por otra parte, la aplicación de la implementación del algoritmo de Goertzel sobre las texturas de la figura 3.17 muestra que, con las adecuadas directivas de optimización, el algoritmo presenta resultados comparables a los obtenidos mediante la implementación del barrido y la FFT. Específicamente, se ha visto que consume menos recursos lógicos, de memoria y potencia que la FFT en todos los casos, y similar al uso del hardware que hace el barrido. La latencia (o tiempo que tarda el algoritmo en proporcionar la salida cuando recibe una entrada), sí es mayor que la FFT, y sólo ligeramente mayor al barrido. El porcentaje de clasificación alcanzado para el caso óptimo es del $84.8\,\%$, valor que se encuentra en sintonía a los alcanzados con el barrido ($89.2\,\%$) y FFT ($91.4\,\%$).

Teniendo en cuenta esto, se puede concluir que la implementación del algoritmo de Goertzel es de interés para la clasificación de texturas, teniendo un rendimiento similar al algoritmo de barrido propuesto, para las texturas y condiciones de experimentación consideradas en esta tesis. Queda abierta la posibilidad de comparar ambas propuestas, en el caso de aumentar el número de texturas o tácteles en el estudio. Por otro lado, la implementación de Goertzel resulta muy ventajosa en cuanto al consumo de recursos y potencia respecto a la FFT. La latencia que muestra el algoritmo es superior a la FFT, pero no obstante, se encuentra en el orden de decenas de μs.

3.10. Resumen

En este capítulo se lleva a cabo la caracterización y clasificación de doce texturas seleccionadas para su exploración. Se implementa el algoritmo de barrido, propuesto para la extracción de características, en un dispositivo SoC, y se compara con la implementación de una FFT. A continuación, se presenta un resumen de los contenidos abordados en este capítulo:

- En la sección 3.1, se presenta el contexto y la motivación del desarrollo de un algoritmo de extracción de características para la detección de texturas en sistemas táctiles inteligentes. Se discuten las limitaciones y desafíos actuales en la detección de texturas, como la complejidad del procesamiento de señales y la necesidad de una gran cantidad de datos para entrenar a los clasificadores. Además, se resaltan las ventajas potenciales de utilizar un sistema embebido basado en FPGA, que

UNIVERSIDAD DE MÁLAGA

permite el procesamiento de datos en paralelo y se adapta a diversos requisitos del sistema. Se presenta una visión general del enfoque propuesto, que consiste en un algoritmo simple pero efectivo que escanea el espectro de frecuencia y proporciona información sobre la ubicación de las componentes frecuenciales como características. En resumen, esta sección establece el marco teórico y conceptual para el resto del documento.

- En la sección 3.2, se presenta con mayor detalle la propuesta de algoritmo. Se describe cómo se pueden encontrar las frecuencias fundamentales de la superficie explorada a partir de las posiciones de singularidades del cálculo en (3.3). El algoritmo propuesto se basa en una estrategia de barrido a una velocidad de exploración específica, donde se ajusta un parámetro de retardo hasta encontrar la frecuencia espacial característica de la textura. Para analizar el comportamiento del algoritmo, se realizan simulaciones básicas utilizando texturas ideales y se compara con otro algoritmo adaptativo basado en la técnica de gradiente descendente. Además, se llevan a cabo pruebas con cuatro texturas sinusoidales fabricadas mediante impresión 3D para verificar el funcionamiento básico del algoritmo. Por otro lado, se introduce un algoritmo alternativo de detección en frecuencia, denominado "algoritmo de Goertzel", que calcula las K primeras componentes frecuenciales de la DFT de una señal, permitiendo así identificar componentes particulares del espectro sin estudiar todas las frecuencias espaciales. Este algoritmo es más rentable que la FFT cuando se cumplen las condiciones dadas por (3.18) y (3.20).

- El diseño de un filtro *antialiasing* para evitar componentes de alta frecuencia y asegurar el límite de Nyquist se propone en la sección 3.3.

- La sección de caracterización de texturas (sección 3.4) aborda la simulación de 12 texturas ideales mediante el uso de Matlab®. Se discuten tres fuentes distintas de ruido que pueden afectar la exploración, como la deformación de la fase de la señal, el ruido blanco y la deformación en amplitud de la señal. Además, se presenta una simulación con modulación en frecuencia progresiva. Esto permite evaluar la robustez del algoritmo de barrido frente a ruidos de alta frecuencia con baja amplitud, en comparación con la FFT, cuyo espectro se ve afectado en tales casos. Por último, se selecciona el conjunto final de texturas utilizado para los experimentos, donde se exponen sus características de diseño principales y se muestran ejemplos de datos obtenidos durante la exploración en tiempo real.

- La sección de extracción de características de texturas (sección 3.5) presenta los resultados obtenidos al aplicar la expresión V (3.3) al conjunto final de texturas seleccionado. Además, se proporcionan los resultados de la aplicación de la FFT

UNIVERSIDAD DE MÁLAGA

utilizando las funciones pre-compiladas MEX de Matlab® para el mismo conjunto de texturas. Estos resultados permiten comparar el desempeño de ambos enfoques en términos de detección y descripción de las características espaciales de las texturas. Por otra parte, se añade un último apartado, en el que se indica el método de selección de los índices de Goertzel. Para la búsqueda de estos índices, se propone la utilización de la expresión no lineal dada en (3.26), que concentra las frecuencias de interés en una determinada zona del espectro a partir de un parámetro de control p.

- En la sección 3.6, se presenta el algoritmo de entrenamiento y clasificación *k-means*, junto con su implementación en el contexto de los datos obtenidos mediante la aplicación de (3.3) y FFT a las texturas. Se explica el proceso de entrenamiento, que incluye la eliminación de valores atípicos o *outliers*. Además, se realiza una representación estadística de los datos de entrenamiento del *sweep* mediante diagramas de violín, lo que permite visualizar la separación entre las diferentes clases. Esta representación gráfica proporciona una visión general de la distribución de los datos y la capacidad del algoritmo para distinguir entre las clases de texturas.

- En la sección 3.7, se detallan las diferentes configuraciones evaluadas para el algoritmo de barrido. Estas configuraciones incluyen el envío de las posiciones de las singularidades de V (3.3) al clasificador, considerando la aplicación de un filtro de media móvil previo y la aplicación de un proceso de diezmado. Se discute cómo estas técnicas impactan en la reducción de oscilaciones espurias en la forma de la señal de V. Además, se presentan los resultados de aplicar estas técnicas de manera individual o combinada, así como los resultados obtenidos mediante la aplicación de la FFT, y los resultados de aplicación de la propuesta de aplicación de Goertzel para distintos valores del tamaño del vector de índices K, y diferentes maneras de espaciar los índices. Los resultados se presentan en forma de tablas, lo que permite una comparación objetiva y cuantitativa de los diferentes enfoques.

- En la sección de implementación 3.8, se implementa el algoritmo de detección de características espaciales de texturas (algoritmo de barrido) en un SoC y se compara con la implementación de una FFT. Se presentan los principales módulos de la placa de desarrollo ZedBoard™ y las características del dispositivo SoC Zynq™-XC7Z020-CLG484-1. Se describe la arquitectura global del sistema y el funcionamiento de los módulos implementados en la parte PL. También se profundiza en las distintas opciones de configuración para el barrido y se compara con la implementación basada en FFT, en términos de recursos, potencia y tiempo de ejecución. A continuación, para la implementación de Goertzel, se compara el

UNIVERSIDAD DE MÁLAGA

rendimiento de este algoritmo con el que se obtenía con el barrido y la FFT. A partir de estos resultados, se verifica que su implementación logra un menor consumo de recursos y potencia respecto a la de la FFT. Además, se muestra el consumo real de la implementación global en el SoC y se presenta la implementación en la parte PS utilizando el entorno de desarrollo SDK y el sistema operativo en tiempo real FreeRTOS®.

- En la sección 3.9, se presentan los resultados de clasificación obtenidos en tiempo real mediante la implementación del algoritmo de barrido en el SoC. Se realiza una comparación detallada entre estos resultados y los obtenidos utilizando la FFT. Se muestran las matrices de confusión correspondientes a ambos algoritmos, considerando diferentes valores del parámetro de longitud del vector de salida M. Para el algoritmo de barrido, se proporcionan representaciones gráficas en un espacio ROC para cada una de las clases de texturas analizadas. En general, se observa que a partir de un cierto tamaño del vector de características, se obtienen porcentajes de clasificación robustos en comparación con la implementación basada en la FFT. A continuación, se presentan los resultados de aplicación de la propuesta de algoritmo de Goertzel. Finalmente, se hace una discusión en torno a las diferentes alternativas de implementación propuestas y las presentadas por otros autores. La simplificación del vector de características que se envía al procesador es crítico a la hora de reducir el consumo de recursos, pero a la vez, lograr altos porcentajes de clasificación.

- Por último, en la sección 3.10, se hace un resumen del capítulo.

UNIVERSIDAD DE MÁLAGA

4

Reconocimiento y clasificación de objetos por rigidez

La rigidez es otra de las propiedades más relevantes en la interacción y manipulación de robots con el entorno. Después de abordar la detección de características espaciales de texturas en el capítulo anterior, es natural profundizar en la capacidad de un sistema táctil para distinguir la dureza de diferentes objetos. Este capítulo se centra en el reconocimiento y clasificación de objetos a partir de su rigidez. En primer lugar, se describen las aplicaciones que involucran sensores táctiles para detectar esta propiedad. Se presentan estudios relevantes sobre este tema y se introduce la propuesta de extracción de características (sección 4.1). A continuación, se propone el uso de características simples para la etapa de clasificación, que se obtienen a partir de las curvas de momentos de la imagen táctil (sección 4.2). Luego, se presenta un conjunto de 42 clases de objetos de distinta rigidez que se utilizaron en los experimentos de palpación. Además, se describe el procedimiento experimental y la clasificación de los datos obtenidos (sección 4.3). El algoritmo de extracción de características se implementa en el dispositivo SoC Zynq™-7020. Se describe la arquitectura propuesta y se introduce la técnica de reducción de dimensionalidad mediante el análisis de componentes principales o *principal component analysis* (PCA) (sección 4.4). A continuación, se presentan los resultados obtenidos sin PCA y con PCA para diferentes combinaciones de momentos de la imagen, características y número de bits por característica. También se realiza un análisis de los recursos hardware, consumo de potencia y latencia del algoritmo. Se

UNIVERSIDAD DE MÁLAGA

seleccionan los escenarios con mejor rendimiento y se calculan los parámetros de sensibilidad y especificidad para ellos (sección 4.5). Por último, se presentan las conclusiones (sección 4.6) y un resumen del capítulo (sección 4.7).

4.1. Introducción

La manipulación e interacción de objetos con el entorno requiere la detección de propiedades como el coeficiente de fricción, la textura, la geometría y la rigidez [2] [114]. La detección de estas propiedades implica el desarrollo de dispositivos con pieles electrónicas (*e-skin*) que ofrecen oportunidades en campos como la atención médica, las interfaces humano-máquina, la realidad virtual, la inteligencia artificial y la robótica [115].

La rigidez de un objeto puede estimarse directamente mediante la medición de la fuerza y el desplazamiento cuando se aplica presión sobre el mismo [116]. Sin embargo, la curva fuerza-desplazamiento de un objeto deformable puede ser no lineal y bastante compleja [117]. En un estudio anterior [117], se utilizó la técnica *functional principal component analysis* (FPCA) para aproximar la curva fuerza-desplazamiento de dos sensores de presión mediante tres funciones base. Se lograron clasificar ocho objetos y se obtuvieron mejores resultados con un clasificador KNN. Otro estudio [118] calculó ocho parámetros a partir de las lecturas de dos sensores de fuerza al presionar una verdura hasta que se colapsaba. Estos parámetros incluían la media, la varianza, la desviación estándar, la fuerza máxima, el factor de calidad, los cuartiles y el factor de cuartil. Todos los datos registrados se redujeron a estos ocho parámetros, y mediante un algoritmo de árbol de decisión se obtuvieron buenos resultados al clasificar tomates en tres categorías diferentes según su rigidez. La rigidez también puede estimarse mediante el análisis de la fuerza de reacción al explorar un objeto mediante movimientos predefinidos [119] [120].

Por otro lado, la deformación de un objeto se puede estimar a partir de la salida de sensores táctiles, especialmente utilizando manos artificiales avanzadas equipadas con pinzas (*grippers*). La imagen táctil completa puede utilizarse como entrada para arquitecturas basadas en redes neuronales complejas, como las redes neuronales convolucionales o *convolutional neural networks* (CNN), que estiman la rigidez a partir de secuencias de imágenes adquiridas durante la deformación del objeto [57] [121]. En estos casos, el pre-procesado de imágenes táctiles consiste esencialmente en filtrado, normalización y reorganización (*reshaping*), para adaptar las imágenes a la entrada de la red neuronal.

Otra aproximación consiste en utilizar características específicas de las imágenes táctiles en lugar de la imagen completa. Esto reduce el volumen de datos que deben almacenarse y procesarse, lo que también simplifica su procesado. En [122] se utilizaron la

UNIVERSIDAD DE MÁLAGA

fuerza máxima medida por un táctel (unidad de sensado de fuerza en el sensor táctil), el área de contacto y el desplazamiento de su centroide como características. Se exploraron diferentes combinaciones de características y algoritmos, obteniendo buenos resultados con modelos ocultos de Márkov para clasificar ocho objetos en cuatro categorías.

Los dos primeros momentos de color de la imagen táctil, es decir, el valor medio y la desviación estándar de los tácteles se propusieron como características en [123]. En este estudio, un clasificador KNN se utilizó con la métrica de distancia basada en la técnica de *dynamic time warping* (DTW). Una estrategia similar se siguió en [124], con un sensor táctil centrado en medir grandes variaciones del desplazamiento y deformación a través de sus tácteles.

Diversas aproximaciones en la literatura han logrado un buen rendimiento con un limitado número de objetos y categorías. Sin embargo, aunque las características propuestas lograron reducir el almacenamiento de datos y la complejidad de computación, estos permanecen altos en el caso de sistemas embebidos. Estos sistemas pueden llevar a cabo computación distribuida con sensores inteligentes, lo que mejora la latencia y reduce el tráfico de datos en buses de comunicación. El sensor táctil inteligente con interfaz de conexión directa con la FPGA propuesto en [37], proporciona los primeros momentos de la imagen táctil, que ofrecen información sobre el área de contacto, la fuerza de contacto, la ubicación, la forma y la orientación de la imagen táctil. Por lo tanto, las series temporales de estos momentos, tomadas mientras que un objeto está siendo comprimido y descomprimido, proporcionan mucha información acerca de cómo se deforma.

Considerando estas curvas de momentos de la imagen, en esta tesis se amplía un estudio preliminar con 9 objetos de geometría básica y diferente rigidez, a un conjunto de hasta 42 clases de objetos de distinta rigidez. En este segundo análisis, se propone el uso de cuatro características simples extraídas directamente de las curvas temporales de los momentos de la imagen.

4.2. Características propuestas para la clasificación

En el trabajo presentado en el congreso internacional *IEEE International Symposium on Industrial Electronics* de 2020, [43], propusimos utilizar las secuencias temporales de momentos de la imagen táctil como vectores de características de entrada a un clasificador KNN, con $k = 1$, $k = 3$ y $k = 5$. Con esta estrategia, se lograron clasificar nueve objetos diseñados con SolidWorks® con una precisión de hasta el $86.7\,\%$. Estos objetos se fabricaron mediante impresión 3D con Filaflex® y PLA®. La dureza de estos objetos

UNIVERSIDAD DE MÁLAGA

se controló variando la rejilla o mallado interno del mismo. De este modo, se consiguieron objetos de densidad baja, media o alta. Una representación de la sección de estos objetos se da en la figura 4.1.

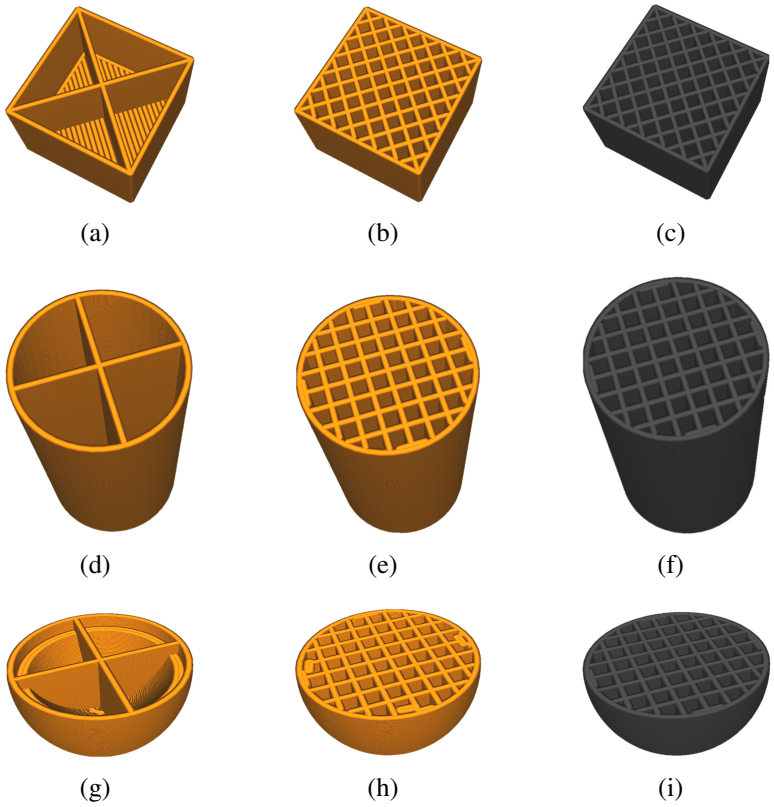


Figura 4.1. Sección del diseño de las nueve piezas preliminares [43]: (a) cubo de densidad baja, (b) cubo de densidad media, (c) cubo de densidad alta, (d) cilindro de densidad baja, (e) cilindro de densidad media, (f) cilindro de densidad alta, (g) esfera de densidad baja, (h) esfera de densidad media y (i) esfera de densidad alta.

Puesto que estas secuencias de datos ocupan un tamaño grande de memoria, en esta tesis se avanza un paso más, con el objetivo de reducir su complejidad y manejo. Para lograr esto, se sigue una estrategia en la cual se extraen características similares a las que se utilizan en escenarios donde no se emplean sensores táctiles [125] [126]. De esta manera, no es necesario realizar cálculos complejos como los empleados en el cálculo de distancias basado en la técnica DTW. Simplemente se calcula la distancia euclídea entre vectores de características, cuyas componentes son las características seleccionadas, lo cual es más sencillo.

Como se mencionó en la introducción (sección 4.1), los momentos de la imagen táctil se calculan de forma local en el dedo robótico y la palma [43] [37]. En particular, los momentos de orden $\{p, q\}$ de la imagen táctil vienen dados por (4.1).

UNIVERSIDAD DE MÁLAGA

$$M_{p,q} = \sum_{x=1}^{N} \sum_{y=1}^{M} x^p y^q I(x,y) \tag{4.1}$$

donde $I(x,y)$ es la salida del táctel para la fila x y columna y.

Mediante la adquisición en tiempo real de las imágenes táctiles, se pueden obtener curvas de momentos de las imágenes táctiles versus tiempo mientras que los objetos se están comprimiendo o descomprimiendo. La figura 4.2 muestra estas curvas para un ejemplo de objeto (berenjena) y el momento $M_{0,0}$. El siguiente conjunto de características están definidas a partir de esta gráfica, para reducir de forma significativa el consumo de recursos y las dimensiones del vector de entrada al clasificador:

- AR (figura 4.2a): Área bajo la curva de momento de la imagen.
- MAX (figura 4.2b): Valor en magnitud para el máximo desplazamiento.
- $H1$ (figura 4.2c): Valor en magnitud, anterior a un tercio del tiempo necesario para alcanzar el pico máximo de la curva.
- $H2$ (figura 4.2d): Valor en magnitud, posterior a un tercio del tiempo necesario para alcanzar el pico máximo de la curva.

Las características $H1$ y $H2$ están relacionados con la histéresis del objeto, que puede ser significativa en objetos hechos de materiales elastómeros. Esta histéresis hace que la gráfica no sea simétrica respecto al eje vertical definido por el pico máximo de la curva. Los instantes de tiempo correspondientes a los tres valores de las características MAX, $H1$, y $H2$ se conocen con antelación, puesto que la secuencia completa de movimientos se controla mediante el sistema robótico. Por tanto, las características correspondientes no requieren ningún cálculo y pueden ser directamente registradas. Además, el área bajo la curva sólo requiere de una suma acumulativa.

Una manera de comprobar que las características propuestas son efectivas para el objetivo de clasificación, es representar el valor de las mismas para las diferentes clases y sensores. Para ello, en las figuras 4.3(a) y 4.3(b) se muestra una representación estadística de las cuatro características propuestas en la figura 4.2, en forma de diagrama de

UNIVERSIDAD DE MÁLAGA

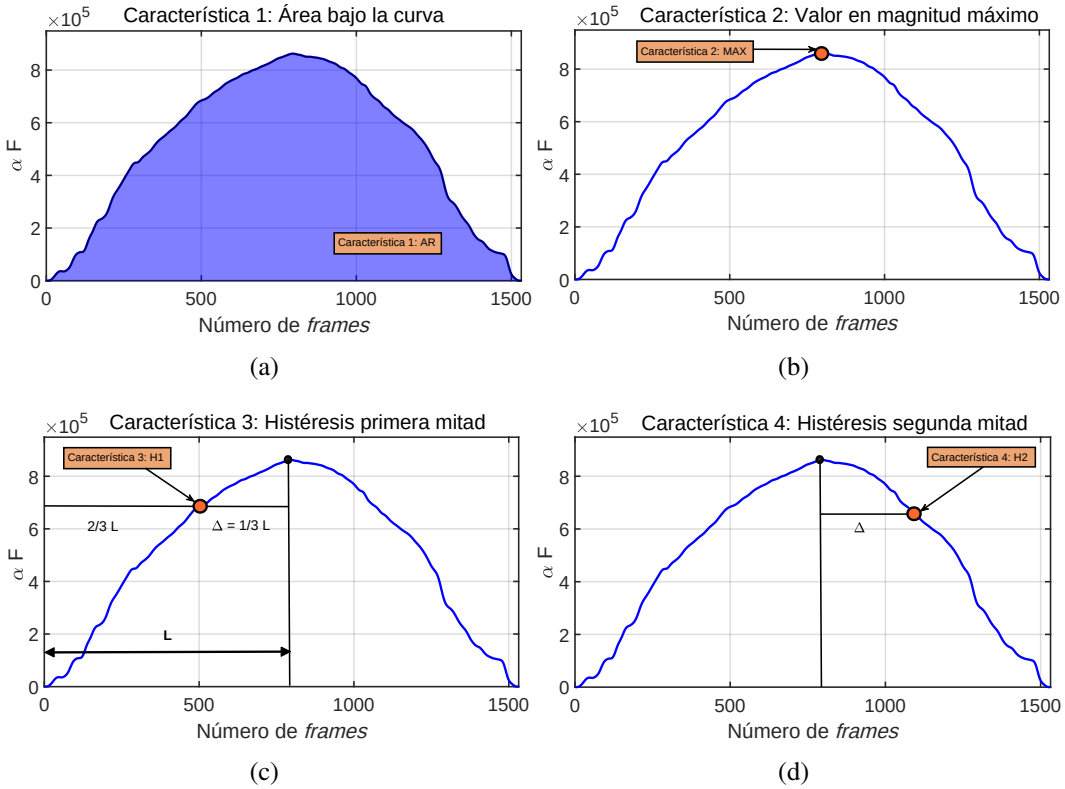


Figura 4.2. Características propuestas para el momento $M_{0,0}$ y el objeto #OBJ-2 (berenjena): (a) área bajo la curva (AR), (b) valor máximo de la curva (MAX), (c) valor en magnitud a $2/3$ del tiempo para alcanzar el máximo en la parte ascendente de la curva ($H1$) y (d) valor en magnitud a $1/3$ del mismo tiempo después de que el máximo sea alcanzado ($H2$).

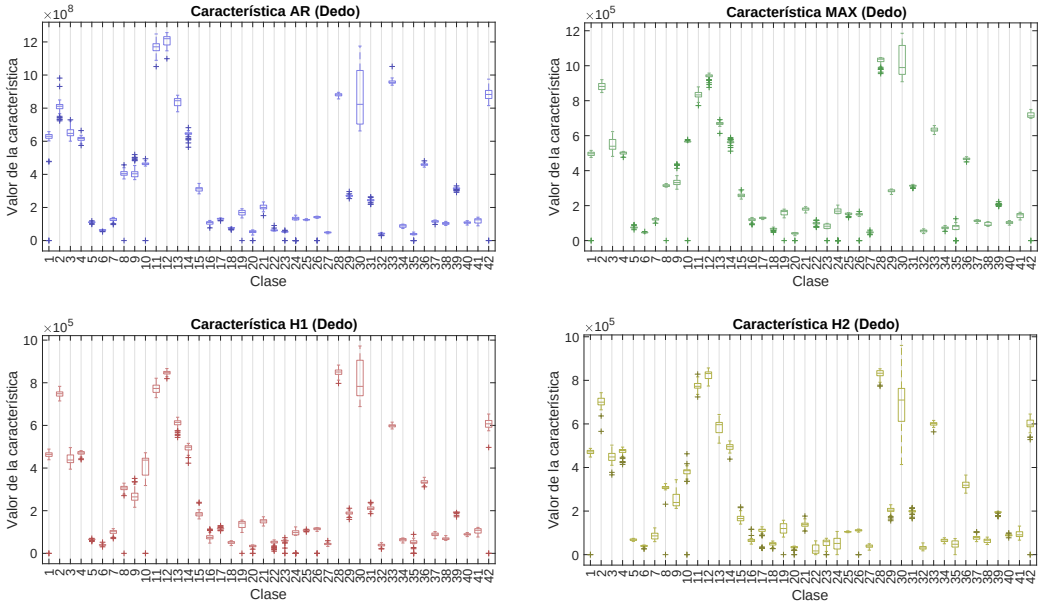
cajas y bigotes o *boxplot*[1], para el dedo y la palma, respectivamente. En estos diagramas se puede ver que, en general, las distribuciones para las características propuestas son compactas, es decir, todos los valores se encuentran próximos entre sí para cada clase. Esto es beneficioso, ya que las clases de objetos serán más distinguibles entre sí, como también comprobaremos en la sección de resultados 4.5.

Cabe resaltar que se calculan seis curvas de momentos diferentes ($M_{p,q}$) para cada exploración, lo que involucra tanto al dedo (f) como a la palma (p), ver capítulo 2. Cuando consideramos los momentos para ambos sensores y los cuatro valores de características (AR, MAX, $H1$ y $H2$), hay un total de 48 características para cada exploración. Para evitar el problema de tener información redundante, en este trabajo se incluye la opción

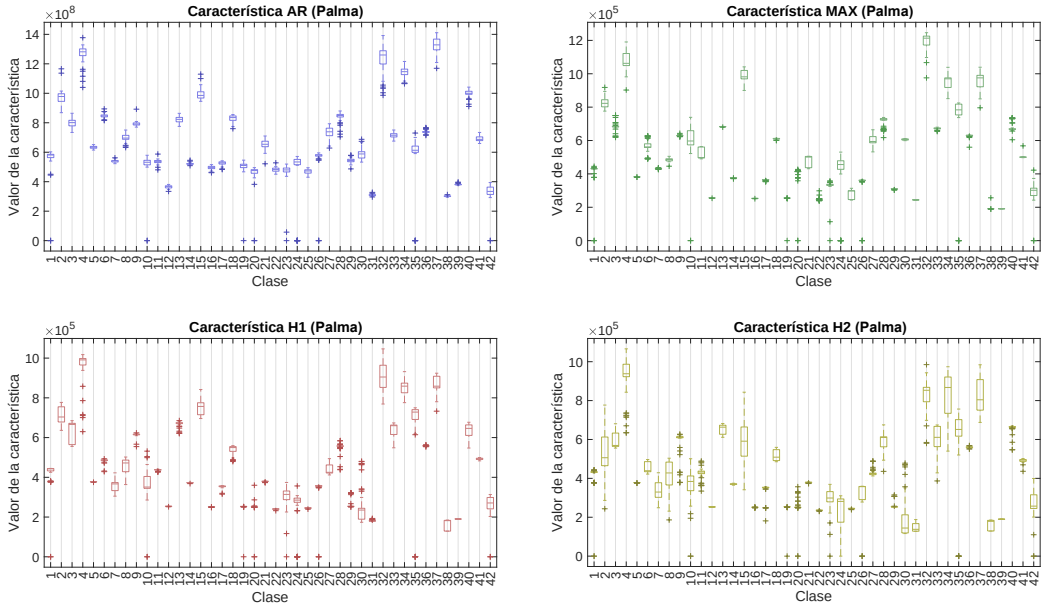
[1]Un *boxplot*, también conocido como diagrama de cajas y bigotes, es una representación gráfica que proporciona información sobre la distribución de un conjunto de datos numéricos. Consiste en un rectángulo (la caja) que abarca el rango intercuartil o IQR (el $50\,\%$ central de los datos), con una línea en su interior que marca la mediana. Dos líneas (los bigotes) se extienden desde la caja hasta los valores más extremos en un rango aceptable (que viene dado por $[Q1 - 1.5 \cdot IQR, Q3 + 1.5 \cdot IQR]$ donde $Q1$, $Q3$ son el valor para el primer y tercer cuartil, respectivamente). El *boxplot* permite identificar la simetría de la distribución, la presencia de valores atípicos como puntos separados y comparar diferentes grupos de datos de manera visual.

UNIVERSIDAD DE MÁLAGA

de reducir las dimensiones de los datos mediante el empleo de técnicas de análisis de componentes principales (PCA).



(a)



(b)

Figura 4.3. Diagramas de cajas y bigotes para las diferentes características propuestas, para: (a) el sensor correspondiente al dedo artificial y (b) el sensor correspondiente a la palma. En ambos casos, la representación está hecha para el momento $M_{0,0}$.

UNIVERSIDAD DE MÁLAGA

4.3. Exploración de objetos

En esta sección se explican de forma más detallada los experimentos realizados para clasificación de objetos. Además, se muestran los objetos con los que se realizaron los experimentos y se muestran las curvas de momentos de la imagen obtenidas para cada uno de ellos. Finalmente, se describe cómo se extraen las características propuestas en la figura 4.2 a partir de estas curvas, las cuales se utilizan como entrada a un clasificador *k-means*.

4.3.1. *Setup* experimental

La figura 4.4 muestra una fotografía equivalente al montaje experimental utilizado para las exploraciones de objetos que se presentó en la figura 2.29 del capítulo 2. En la imagen, se pueden identificar las distintas partes del montaje, incluyendo la electrónica de control, así como la electrónica de adquisición y comunicación a cuya FPGA se conectan el dedo y la palma artificial. Los objetos se colocan en el espacio entre ambos sensores.

4.3.2. Objetos a explorar

Para llevar a cabo los experimentos, se seleccionaron 17 piezas de fruta y verdura y 12 objetos de diferente forma y propiedades. Variando la posición de agarre de estos objetos, se exploraron y clasificaron hasta 42 clases distintas (ver figura 4.5 y tabla 4.1). Estas clases se agruparon en cuatro conjuntos. El primero de ellos consistió en objetos fabricados mediante impresión 3D, identificados como #OBJ-7, #OBJ-31, #OBJ-32 y #OBJ-37 a #OBJ-41. Estos objetos se fabricaron utilizando materiales termoplásticos como Filaflex® y TPU®. El segundo conjunto comprendió objetos en forma de ovoides en 3D, destinados a la rehabilitación terapéutica de la mano. Cada uno de estos ovoides presenta una rigidez específica. Dependiendo de la posición de agarre, se consideraron hasta doce clases diferentes en este grupo, desde #OBJ-16 a #OBJ-18 con rigidez media-alta, #OBJ-19 a #OBJ-21 con rigidez alta, #OBJ-22 a #OBJ-24 con rigidez baja, y #OBJ-25 a #OBJ-27 con rigidez media-baja. El tercer conjunto consistió en objetos seleccionados de escenarios del mundo real, identificados como #OBJ-4, #OBJ-5, #OBJ-6 y #OBJ-29. Por último, se incluyó un cuarto conjunto con el objetivo de extrapolar los resultados a aplicaciones como la agricultura inteligente o el procesado automático de alimentos. Este conjunto estaba compuesto por 17 piezas de frutas y verduras en diferentes estados de conservación (verde, madura o podrida). Las piezas en este conjunto se identificaron como #OBJ-1, #OBJ-2, #OBJ-3, #OBJ-9 al #OBJ-15, #OBJ-28, #OBJ-30, #OBJ-33 al #OBJ-36 y #OBJ-42.

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Tabla 4.1. Tabla con las 42 clases empleadas en este trabajo, ver también figura 4.5.

Clase	Descripción de la clase	Posición de exploración	Clase	Descripción de la clase	Posición de exploración
#OBJ-1	Aguacate	Horizontal	#OBJ-22	Ovoide naranja para terapia de mano	Boca abajo
#OBJ-2	Berenjena	Horizontal	#OBJ-23	Ovoide naranja para terapia de mano	Boca arriba
#OBJ-3	Ciruela	Estándar	#OBJ-24	Ovoide naranja para terapia de mano	Horizontal
#OBJ-4	Cubo de gomaespuma	Estándar	#OBJ-25	Ovoide verde para terapia de mano	Boca abajo
#OBJ-5	Cubo de gomaespuma	Vertical	#OBJ-26	Ovoide verde para terapia de mano	Boca arriba
#OBJ-6	Esfera para terapia de mano	Estándar	#OBJ-27	Ovoide verde para terapia de mano	Horizontal
#OBJ-7	Esfera impresa en 3D con Filaflex® verde	Estándar	#OBJ-28	Patata	Horizontal
#OBJ-8	Gel hidroalcohólico	Horizontal	#OBJ-29	Pelota de pádel	Estándar
#OBJ-9	Kiwi	Horizontal	#OBJ-30	Pera	Horizontal
#OBJ-10	Lechuga	Horizontal	#OBJ-31	Pirámide impresa en 3D con TPU® azul	Boca abajo
#OBJ-11	Limón maduro	Horizontal	#OBJ-32	Pirámide impresa en 3D con TPU® azul	Boca arriba
#OBJ-12	Limón verde	Horizontal	#OBJ-33	Plátano verde	Horizontal
#OBJ-13	Mandarina madura	Horizontal	#OBJ-34	Plátano maduro	Horizontal
#OBJ-14	Mandarina verde	Horizontal	#OBJ-35	Plátano podrido	Horizontal
#OBJ-15	Nectarina podrida	Horizontal	#OBJ-36	Tomate	Horizontal
#OBJ-16	Ovoide azul para terapia de mano	Boca abajo	#OBJ-37	Toroide impreso en 3D con Filaflex® verde	Horizontal
#OBJ-17	Ovoide azul para terapia de mano	Boca arriba	#OBJ-38	Toroide impreso en 3D con Filaflex® verde	Vertical
#OBJ-18	Ovoide azul para terapia de mano	Horizontal	#OBJ-39	Triángulo impreso en 3D con TPU® azul	Boca abajo
#OBJ-19	Ovoide morado para terapia de mano	Boca abajo	#OBJ-40	Triángulo impreso en 3D con TPU® azul	Boca arriba
#OBJ-20	Ovoide morado para terapia de mano	Boca arriba	#OBJ-41	Triángulo impreso en 3D con TPU® azul	Horizontal
#OBJ-21	Ovoide morado para terapia de mano	Horizontal	#OBJ-42	Zanahoria	Horizontal

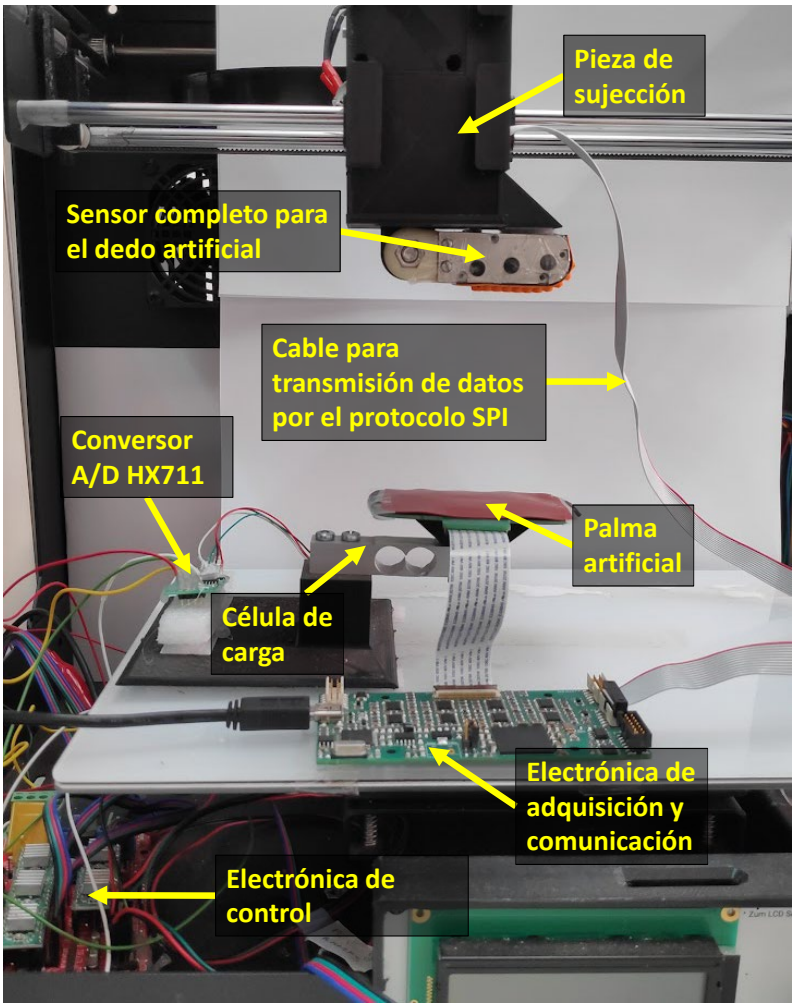


Figura 4.4. Fotografía del *setup* experimental para la exploración de objetos con distinta rigidez.

4.3.3. Procedimiento para obtener los datos

El procedimiento seguido para obtener los datos de entrenamiento y test se describe a continuación:

1. Manualmente, se coloca un objeto del conjunto mostrado en la figura 4.5 o tabla 4.1, en el espacio existente entre el dedo y la palma.

2. Se controla el movimiento del eje Z de la impresora, para mover verticalmente la palma hacia arriba (sentido positivo del eje Z) hasta que la célula de carga detecta un umbral de fuerza de bajo nivel, establecido en $F_{init} = 0.1$ N. Esta posición se guarda como la posición inicial de contacto.

3. Se controla el movimiento de la palma para que continúe moviéndose verticalmente hacia arriba, lo que comprime el objeto hasta que la palma alcanza una distancia máxima, respecto a la posición inicial de contacto, de aproximadamente ≈ 1.2 cm

UNIVERSIDAD DE MÁLAGA

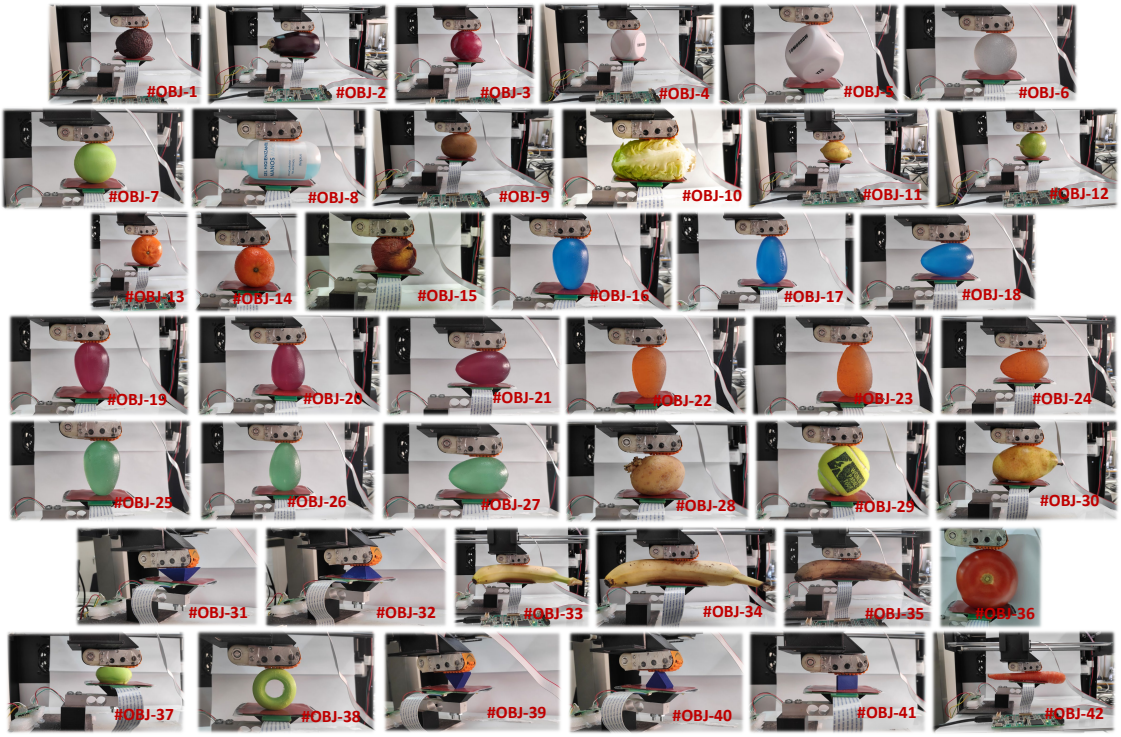


Figura 4.5. Conjunto de las 42 clases de objetos utilizadas para llevar a cabo las exploraciones de palpación, ver también tabla 4.1.

(aunque el agarre entre el dedo y la palma permite una cierta deformación, este límite se estableció empíricamente para evitar dañar el sistema al explorar objetos rígidos).

4. Por último, después de alcanzar el punto máximo del paso anterior, se controla el movimiento vertical de la palma, para que se mueva en dirección opuesta (sentido negativo del eje Z). De esta manera, el objeto se descomprime gradualmente hasta alcanzar de nuevo la posición inicial de contacto definida en el paso 2.

Este procedimiento se repitió 47 veces por objeto, a una velocidad de $v = 10$ mm/s. Durante las secuencias de compresión-descompresión, los sensores táctiles del dedo y la palma obtuvieron datos en tiempo real. Conforme se registraban estos datos, se enviaban por puerto serie al ordenador, donde una aplicación desarrollada en LabVIEW® los almacenaba en ficheros de texto (ver figura 2.27 del capítulo 2). En la figura 4.6 se presenta una descripción general de la metodología seguida para la exploración del objeto #OBJ-17 de la figura 4.5 o tabla 4.1.

Una vez se completaron los experimentos con los 42 objetos de la figura 4.5, se obtuvieron las curvas correspondientes a los seis primeros momentos de la imagen táctil (4.1) de estos objetos. La figura 4.7 muestra un ejemplo de estas seis primeras curvas de

UNIVERSIDAD DE MÁLAGA

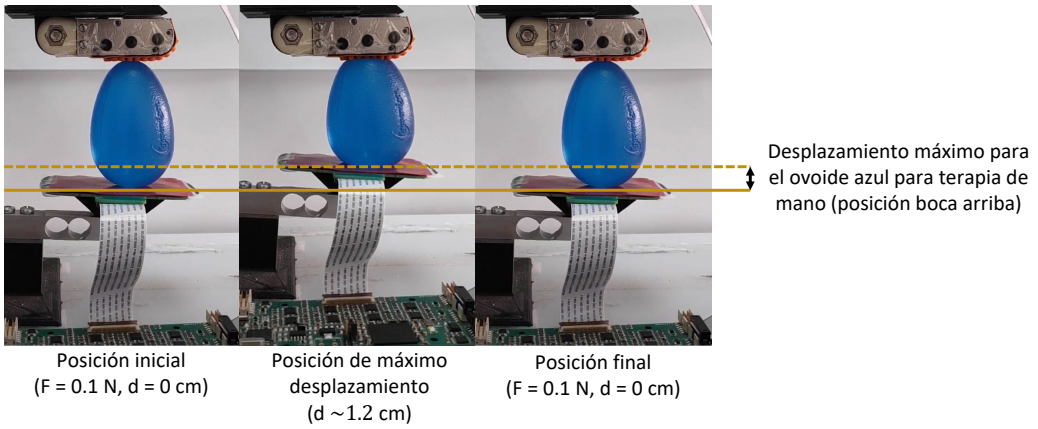


Figura 4.6. Ilustración de la secuencia de exploración para la clase $\#OBJ$-17.

momentos $M_{p,q}$ para el dedo y la palma, y para la clase #OBJ-17 y #OBJ-38 de la figura 4.5, respectivamente. En las figuras A.9, A.10, A.11, A.12, A.13 y A.14 del apéndice A de figuras se muestran las curvas de momentos correspondientes a todos las clases de objetos de la tabla 4.1.

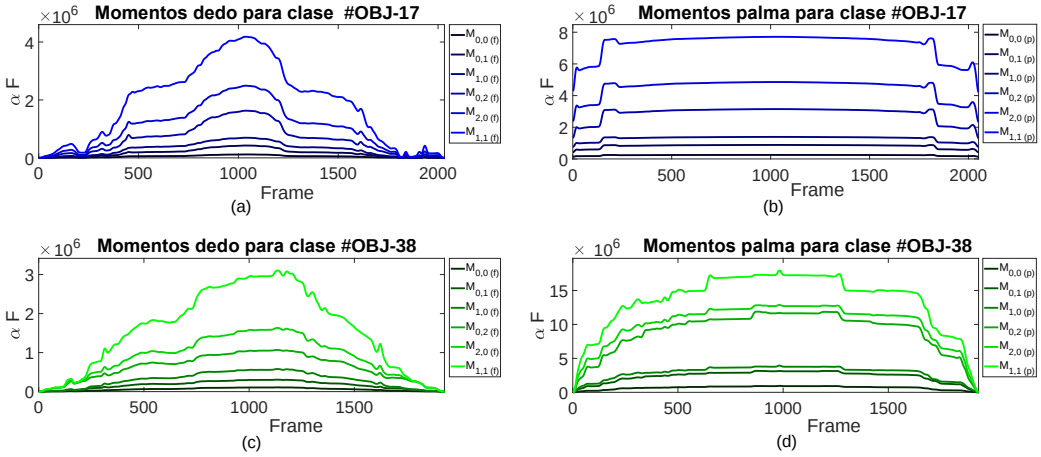


Figura 4.7. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (a) y (c), y con la palma (b) y (d), calculadas a partir de la exploración de las clases de objetos #OBJ-17 y #OBJ-38.

4.3.4. Algoritmo de entrenamiento

Finalmente, para llevar a cabo la clasificación se utiliza de nuevo el algoritmo de entrenamiento *k-means* presentado en la sección 3.6 del capítulo 3 para la exploración de texturas. El conjunto de características obtenido de las curvas de momentos se dividen en $60\,\%$ para entrenamiento y $40\,\%$ para test, de igual forma que en el caso de las texturas (ver sección 3.6 del capítulo 3).

UNIVERSIDAD DE MÁLAGA

4.4. Implementación en SoC

Para demostrar la viabilidad de la propuesta y estimar su rendimiento, se siguió un procedimiento en dos pasos. Primero, se adquirieron datos utilizando los sensores del dedo y la palma, como se describe en la sección 4.3. Las secuencias temporales de momentos de la imagen se transfirieron a la memoria DDR3 externa de la ZedBoard™ (ver sección 2.5 del capítulo 2). El proceso de extracción de características descrito en la sección 4.2 se implementó en la FPGA del SoC, mientras que el algoritmo de clasificación se implementó en el núcleo ARM®. El uso de la ZedBoard™ añade flexibilidad para explorar diversas alternativas de implementación en un sistema final. Las características calculadas en el dedo y la palma se transfieren a un procesador central, donde se ejecuta el clasificador. La figura 4.8 ilustra este sistema y el camino de datos o *datapath* seguido en la arquitectura global. A la entrada se coloca una cola tipo FIFO, que proporciona una salida de 32 bits. Estos datos se transmiten a una interfaz serie-paralelo a través de los buses del protocolo AMBA® AXI-Streaming ('AXISTREAM Serial to Parallel Interface', en la figura 4.8). Esto permite que los datos de entrada (momentos $M_{p,q}$ del dedo y palma) de la FIFO sean distribuidos a los diferentes módulos de pre-procesamiento ('VHDL Computing Module', en la figura 4.8). Por otra parte, las señales de sincronización aseguran la ejecución en paralelo de estos módulos en el hardware (en la subsección 4.4.1 se detalla el módulo 'VHDL Synchronization Modules' de la figura 4.8). El hecho de trabajar en paralelo es ventajoso, ya que se optimiza la latencia del pre-procesado en la FPGA. En un escenario donde se utilizan ambos sensores (dedo y palma), junto al máximo número de momentos de la imagen y características, el tiempo de ejecución se mantiene inalterado comparado a la utilización de un único módulo. Esto podría ser particularmente útil en situaciones en las que la complejidad del sistema se vea incrementada.

Para implementar los módulos de pre-procesamiento de la figura 4.8 se utilizó el entorno de diseño Vivado Design Suite™ (ver sección 2.5 del capítulo 2). Los bloques 'VHDL Features Computing Module' (detallados en la subsección 4.4.2) producen datos de salida ($D_{p,q}$) sincronizados para el dedo y la palma. Estos datos se transmiten luego a una interfaz paralelo-serie mediante el protocolo AXI-Streaming ('AXISTREAM Parallel to Serial Interface', en la figura 4.8). El módulo 'HLS Principal Component Analysis (PCA) Computing Module' (representado con línea discontinua en la figura 4.8) actúa en caso de aplicar la técnica PCA. La implementación de este módulo se explica en la subsección 4.4.3. Posteriormente, los vectores de características se transfieren en serie, a través de un *buffer* FIFO y un módulo DMA, a la memoria DDR externa de la ZedBoard™, donde se almacenan. El uso del módulo de DMA permite al núcleo ARM®

UNIVERSIDAD DE MÁLAGA

acceder a la información almacenada sin lidiar con el flujo de datos, como se comentó en la sección 3.8 del capítulo 3. El módulo FIFO de salida permite desacoplar la frecuencia entre la parte de lógica programable (PL) y la parte de sistema de procesamiento (PS).

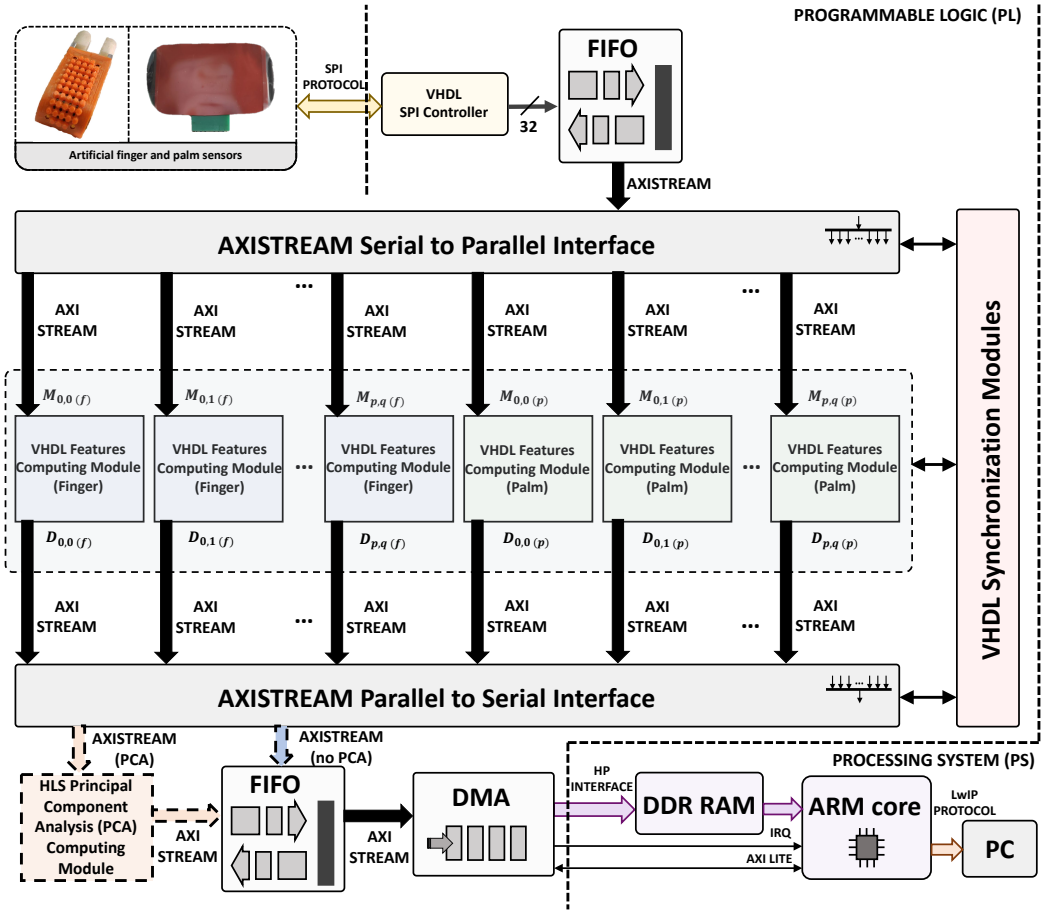


Figura 4.8. Arquitectura general de la implementación en el SoC Zynq-7000 XC7Z020®. La parte con línea discontinua en color anaranjado sólo se implementa si se aplica PCA, [40].

4.4.1. VHDL Synchronization Modules

La señal de salida $t_{validOut}$ del módulo 'VHDL Synchronization Modules' asegura la sincronización de las señales $t_{validOut}$ de todos los módulos 'VHDL Features Computing Modules' incluidos en el diseño, tanto para el dedo como para la palma. Las señales de $t_{validOut}$ se activan (valor lógico '1') durante un ciclo de reloj a la salida de 'VHDL Synchronization Modules' para indicar que el dato es válido. Es importante que todos los datos de salida sean válidos al mismo tiempo para asegurar su correcta recepción por parte del módulo esclavo 'AXISTREAM Parallel to Serial Interface'. De esta forma, la señal de t_{ready} del esclavo, se activará después de haber obtenido correctamente los datos, para indicar que el módulo ya está listo para una nueva recepción.

UNIVERSIDAD DE MÁLAGA

Para lograr correctamente la sincronización deseada, se propone el uso de la máquina de estados de la figura 4.9. Se trata de una máquina de estados de tipo Mealy. Esta máquina opera por pares de señales a la entrada, es decir, tiene dos señales de entrada y una de salida. Las señales de entrada corresponden a las de dato válido para dos módulos 'VHDL Synchronization Modules' de la figura 4.8. La señal de salida sincroniza estas dos señales de entrada, para que ambas se activen en el mismo ciclo de reloj. La máquina de estados consta de tres estados: st1, st2 y st3. Al iniciarse el sistema, se encuentra en el estado st1. En este estado, la salida será '1' si ambas entradas se encuentran al valor lógico '1', y será '0' si ambas entradas se encuentran al valor lógico '0'. Si una de las dos entradas cambia, la salida será '0', y la máquina de estados transitará a st2 o st3. Ambos estados operan de la misma manera: independientemente del valor de la señal que haya propiciado la transición, la señal de salida será '0' hasta que la otra entrada cambie a '1', momento en el cual, la salida es '1' y se vuelve al estado st1. Un ejemplo del funcionamiento de la máquina de estados se muestra en la imagen de la figura 4.10, para señales medidas con un osciloscopio MDO4104B-6 Mixed Domain Tektronix® en laboratorio. Como se puede ver, todas las señales de entrada están sincronizadas con la señal de salida $t_{validOut}$. El código VHDL correspondiente a este módulo se puede consultar en el repositorio público *https://github.com/raulLkRb/thesis.git*.

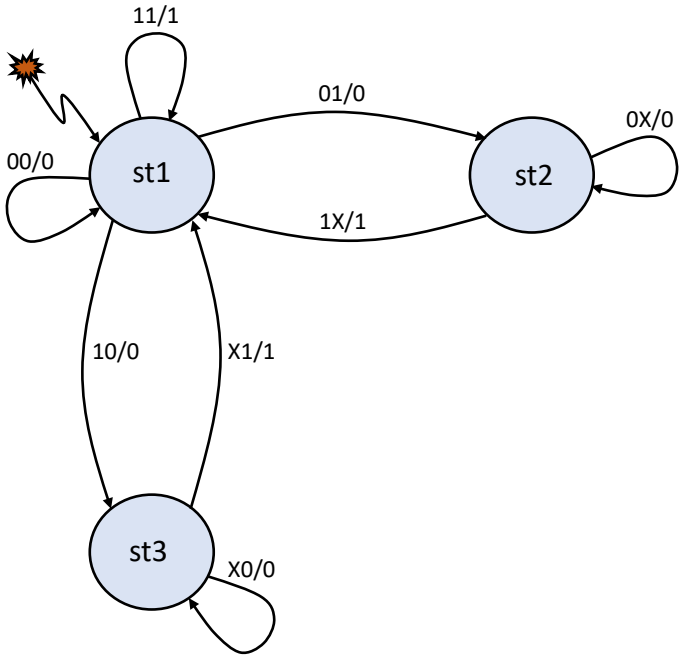


Figura 4.9. Máquina de estados para el módulo 'VHDL Synchronization Modules' de la figura 4.8. En cada transición de estados, el formato indicado es: $t_{validOut1}$,$t_{validOut2}$/$t_{validOut}$, donde $t_{validOut1}$ y $t_{validOut2}$ son las entradas al módulo, que se corresponden con las señales de dato válido del módulo 'VHDL Synchronization Modules', y $t_{validOut}$ es la señal de salida.

La máquina de estados representada en la figura 4.9 se puede escalar para trabajar con más de 2 señales de entrada. Esto se logra mediante una configuración en cascada,

UNIVERSIDAD DE MÁLAGA

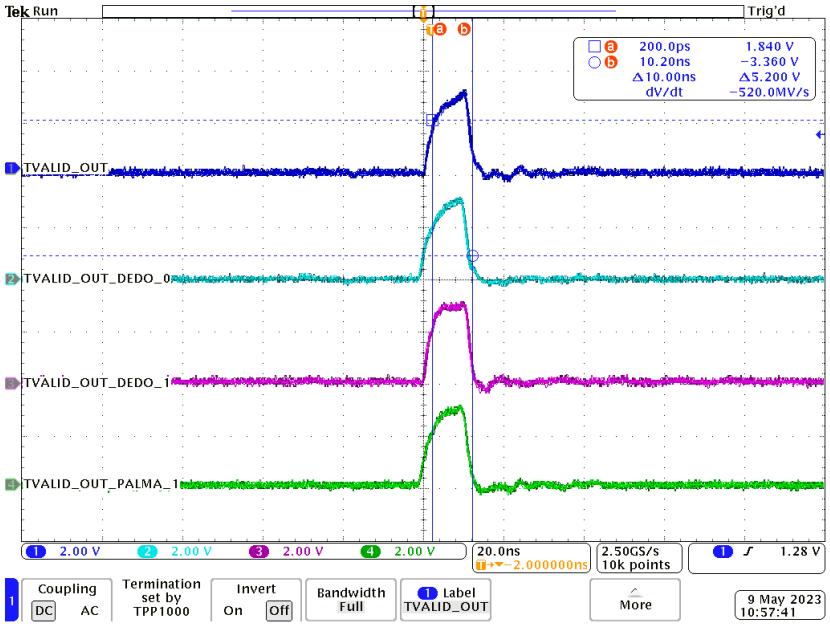


Figura 4.10. Ejemplo de sincronización de las señales de dato válido $t_{validOutDedo0}$, $t_{validOutDedo1}$, $t_{validOutPalma0}$. Las señales se midieron con el osciloscopio MDO4104B-6 Mixed Domain Tektronix®. Los cursores de medida están colocados en los umbrales de tensión ascendente o descendente, respectivamente.

donde las salidas de dos módulos de sincronización se convierten en las dos entradas del siguiente módulo. La figura 4.11 ilustra esta configuración. Esta es la disposición utilizada para los distintos casos abordados en esta parte de la tesis.

4.4.2. VHDL Features Computing Module

La figura 4.12 ilustra el hardware propuesto para el módulo 'VHDL Features Computing Module', encargado de calcular las características AR, MAX, $H1$ y $H2$ de las imágenes táctiles de los momentos de orden $\{p, q\}$ según la ecuación (4.1). El dato de entrada en un instante de tiempo tiene un ancho de palabra de 32 bits, en la que los tres bits más significativos indican si el dato, codificado en los bits del 0 al 28, corresponde a la característica $H2$, $H1$ o MAX, respectivamente. La activación de estos tres bits se puede determinar gracias al control de agarre durante la secuencia de exploración. Por su parte, el cuarto bit más significativo, END, indica el final de la secuencia de momentos de la imagen. La lectura de estos valores únicamente requiere comparaciones sencillas implementadas con lógica (LUT Logic). Por otro lado, la característica AR se calcula con un sumador que toma un dato de entrada por ciclo de reloj, agregándolo al valor global de la suma guardada en un registro. Finalmente, los valores de las características se concatenan en una única palabra de salida de 32 bits, que se envía a través de AXISTREAM. Este módulo también hace uso de las señales t_{valid} y t_{ready} del protocolo AMBA® AXI-Streaming, para asegurar la sincronización entre la entrada y la salida. La

UNIVERSIDAD DE MÁLAGA

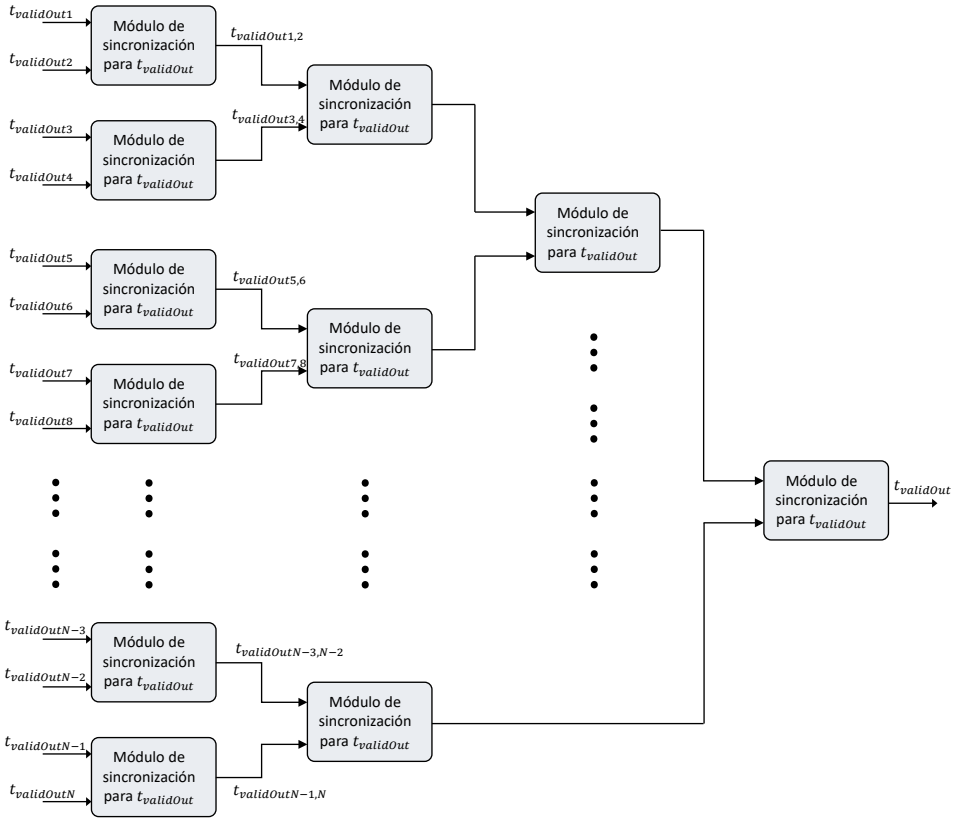


Figura 4.11. Configuración en cascada para el módulo 'VHDL Synchronization Modules' de la figura 4.8, para la sincronización de N señales de entrada.

frecuencia de reloj del sistema es de $f_{clk} = 100$ MHz. El dato de salida con los valores de las características se produce sólo tres ciclos de reloj después de que el último vector de entrada se haya leído.

El hardware de la figura 4.12 se sintetiza a partir del diseño en VHDL de una máquina de estados que controla el flujo de datos entre la entrada y la salida. La figura 4.13 muestra esta máquina de estados. En particular, se ha diseñado como una máquina de estados de tipo Moore. Cuando el sistema se inicia, la máquina de estados comienza en el estado st1. En este estado, se calculan las características AR, $H1$, MAX y $H2$ a partir de los datos de entrada. Las señales $t_{validOut} = 0$ y $t_{readyOut} = 1$ indican que el módulo está preparado para recibir información. Si el dato que llega es válido ($t_{validIn} = 1$) y se corresponde con el último de la secuencia temporal del momento de la imagen ($END = 1$), la máquina de estados avanza al estado st2. En este estado, se prepara la salida de 32 bits. La señal $t_{validOut} = 0$ indica que esta no es válida aún, y la señal $t_{readyOut} = 0$ avisa de que el módulo tampoco puede recibir datos. Por último, la máquina de estados transitará al estado st3 cuando el módulo que actúa como esclavo ('AXISTREAM Parallel to Serial Interface') notifique que está listo para la recepción de datos, activando la señal $t_{readySlave} = 1$. En el estado st3, las señales $t_{validOut} = 1$ y $t_{readyOut} = 0$ indican que el

UNIVERSIDAD DE MÁLAGA

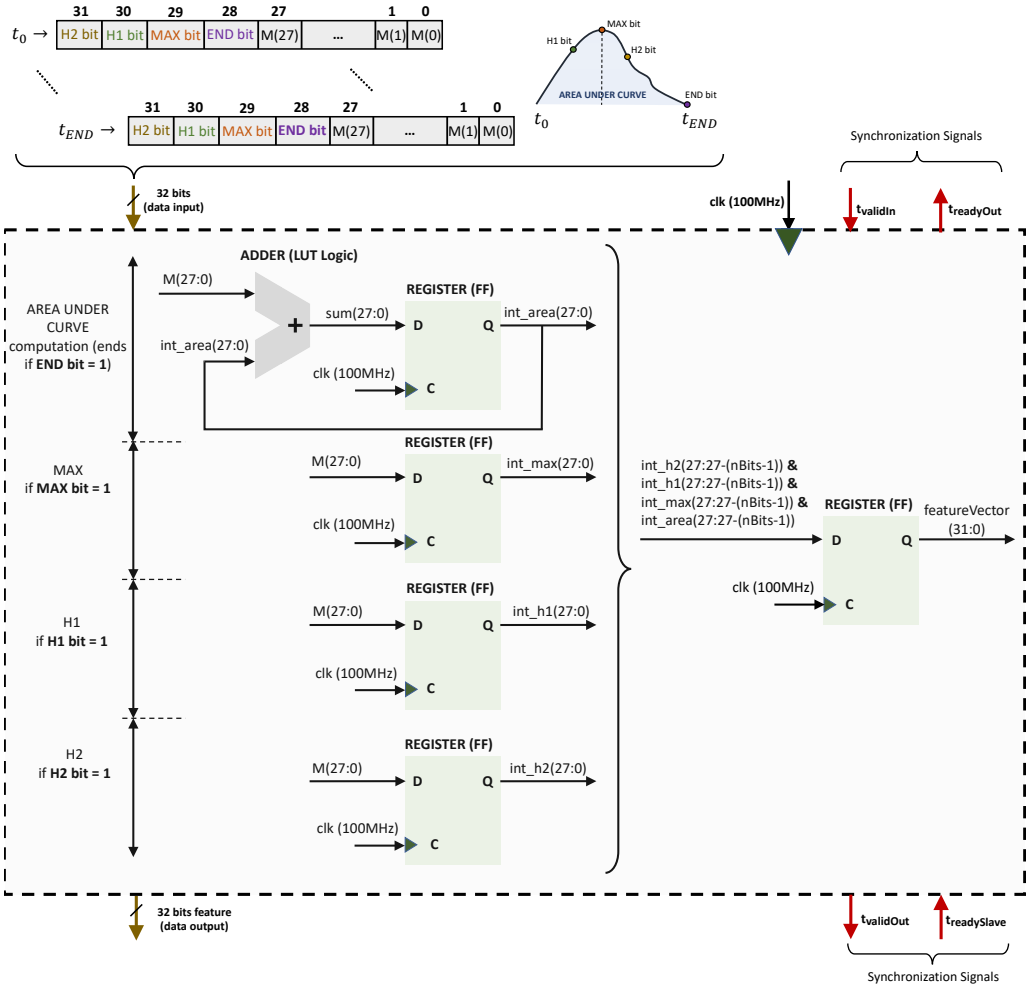


Figura 4.12. Propuesta de implementación para el módulo 'VHDL Features Computing Module' de la figura 4.8, [40].

dato a la salida es válido durante un ciclo de reloj. Como se puede ver, hay que esperar tres ciclos de reloj (correspondientes a la transición entre los estados st1, st2 y st3) para dar la salida, una vez se ha recibido el último dato de la secuencia (esto también se puede comprobar en las imágenes de la figura 4.14). El código en VHDL para este módulo se puede consultar en el repositorio público *https://github.com/raulLkRb/thesis.git*.

Como se aprecia en la figura 4.14, las señales $t_{validInDedo0}$ y $t_{validInPalma0}$ se activan en el mismo ciclo de reloj. Esto ocurre para todas las señales de entrada de tipo $t_{validIn}$. Los cambios de valor de las señales en la figura 4.14, corroboran el comportamiento de la máquina de estados de la figura 4.13. En particular:

a) El intervalo temporal entre el inicio (flanco de subida al valor lógico '1') de la señal que indica dato válido a la entrada ($t_{validIn}$) y el inicio de la señal de dato válido a la salida ($t_{validOut}$), es de $\Delta t = 30$ ns, es decir, tres ciclos de reloj. La activación

UNIVERSIDAD DE MÁLAGA

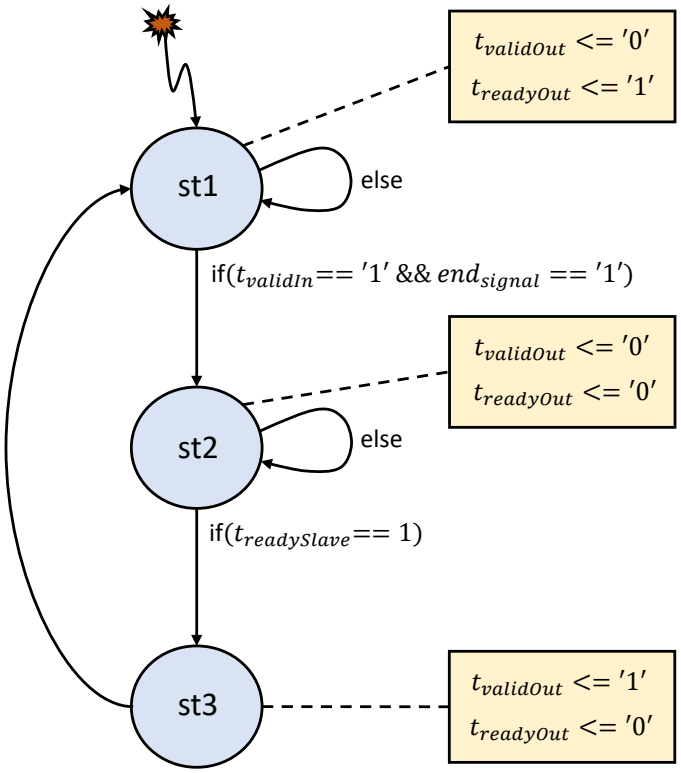


Figura 4.13. Máquina de estados para el módulo 'VHDL Features Computing Module' en la figura 4.8.

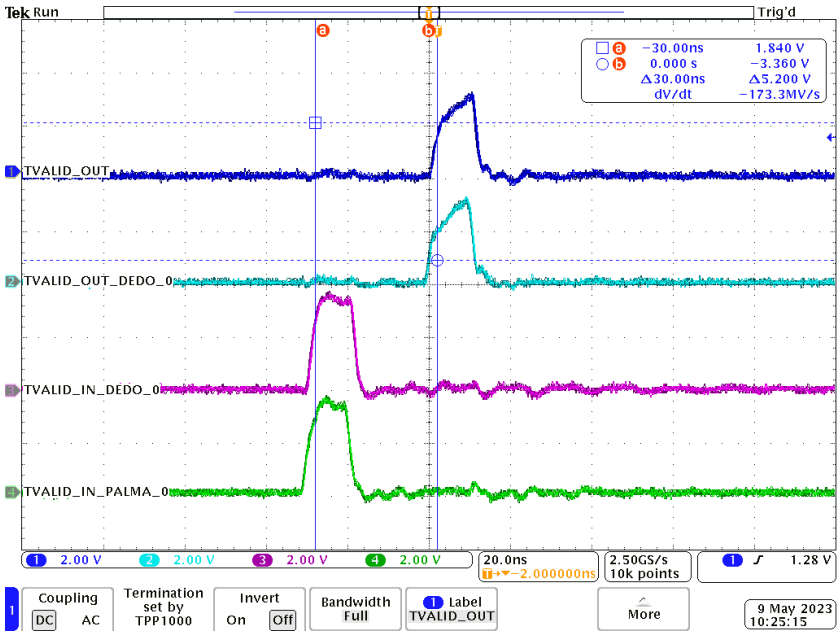
a '1' de la señal $t_{validIn}$ propicia el cambio de estado de st1 a st2. Cuando $t_{validOut}$ se activa a '1', nos encontramos en el estado st3.

b) Por otro lado, el intervalo temporal entre el inicio de la señal que indica dato válido a la entrada ($t_{validIn}$) y el final (flanco de bajada al valor lógico '0') de la señal que indica dato válido a la salida ($t_{validOut}$), es de $\Delta t = 40$ ns, es decir, tres ciclos de reloj, más otro ciclo de reloj correspondiente a 10 ns de duración de la señal $t_{validOut}$. Además, vemos que la señal de t_{ready}, indicativa de que el módulo está preparado para recibir datos, baja al nivel lógico '0' un ciclo después del final de la señal de $t_{validIn}$. En este momento, nos encontramos en el estado st2 de la figura 4.13. Cuando la máquina de estados transita a st3, la señal $t_{validOut}$ se activa a '1'. En este proceso, vemos que t_{ready} vuelve a ponerse al valor lógico '1' al transcurrir dos ciclos de reloj, momento en el cual la máquina de estados vuelve al estado st1.

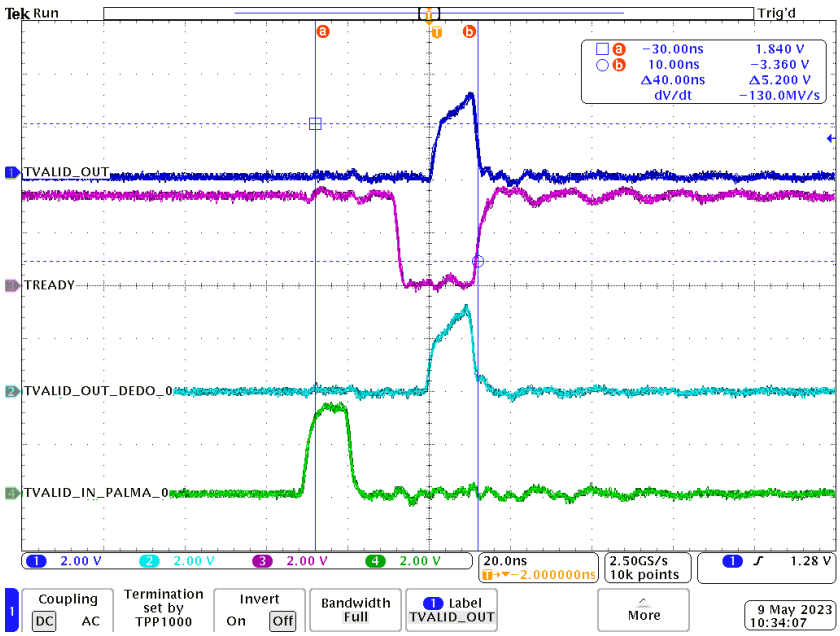
4.4.3. HLS Principal Component Analysis (PCA) Computing Module

El análisis de componentes principales o técnica PCA es un método riguroso para reducir la dimensionalidad de los datos. Tomemos **X** como una matriz cuyas filas son las

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura 4.14. Medida con el osciloscopio MDO4104B-6 Mixed Domain Tektronix® del conjunto de los siguientes conjuntos de señales: (a) $t_{validOut}$, $t_{validOutDedo0}$, $t_{validInDedo0}$, $t_{validInPalma0}$ y (b) $t_{validOut}$, t_{ready}, $t_{validOutDedo0}$, $t_{validInPalma0}$. En cada caso, los cursores de medida están colocados en los umbrales de tensión ascendente o descendente, respectivamente.

observaciones, es decir, los momentos de orden $\{p, q\}$ de la imagen táctil, y las columnas las variables, que se corresponden con el número de características que hay en una determinada exploración. Supongamos que la matriz $\mathbf{X}$ contiene n filas y P columnas. Por tanto, habrá una cantidad total de $n \times P$ datos.

UNIVERSIDAD DE MÁLAGA

En casos donde n es un valor alto, tendríamos un número elevado de datos, y por tanto, los recursos para una implementación en tiempo real se incrementarían. Para estos escenarios, la aplicación de técnicas PCA ayuda a reducir la dimensionalidad de los datos, obteniendo *arrays* m-dimensionales en lugar de n-dimensionales, siendo $n \gg m$. En particular, al restar el vector de medias $\boldsymbol{\mu}$ a cada fila de $\mathbf{X}$, se obtiene la matriz normalizada $\mathbf{X}_n$. La elección de un subconjunto de estos vectores, ordenados de mayor a menor de acuerdo a sus autovalores asociados, se corresponden con las 'componentes principales'.

Finalmente, la matriz reducida m-dimensional $\mathbf{X}_{red}$, se obtiene multiplicando la matriz de coeficientes $\mathbf{coeff}$ por la matriz traspuesta de $\mathbf{X}_n$, según la ecuación (4.2):

$$\mathbf{X}_{red} = \mathbf{coeff} \cdot \mathbf{X}_n^t \tag{4.2}$$

Una de las ventajas de PCA es que, una vez se ha calculado la matriz de coeficientes $\mathbf{coeff}$, si llegan nuevas series de datos táctiles de los sensores, $\mathbf{X}_{new}$, el conjunto reducido de características correspondiente a estos datos, $\mathbf{X}_{redNew}$, se puede calcular también, simplemente sustrayendo el vector de medias previamente mencionado $\boldsymbol{\mu}$, y luego multiplicando otra vez por la matriz de coeficientes $\mathbf{coeff}$, como se puede ver en la expresión (4.3):

$$\mathbf{X}_{redNew} = (\mathbf{X}_{new} - \mu) \cdot \mathbf{coeff} \tag{4.3}$$

En la figura 4.8 se incluye un módulo opcional que permite implementar la técnica PCA. La inclusión de este módulo requiere más memoria y recursos lógicos (como se comenta en la sección 4.5), pero se puede enviar un número más pequeño de características al clasificador que se implementa en el núcleo ARM®. El módulo 'HLS Principal Component Analysis (PCA) Computing Module' de la figura 4.8 se desarrolló en System C utilizando la herramienta HLS de Vivado®. El pseudocódigo correspondiente a este módulo se muestra en la figura 4.15. La directiva de optimización #HLS DATAFLOW se incluyó para mejorar la concurrencia en la descripción RTL. La implementación en *pipeline* se logra a través del uso de las directivas #HLS PIPELINE y #HLS INLINE. A cada nuevo vector de características $\mathbf{X}_{new}$ se le resta el vector de media pre-calculado $\boldsymbol{\mu}$ y después, este resultado se multiplica por la matriz de coeficientes $\mathbf{coeff}$, de acuerdo a (4.3).

UNIVERSIDAD DE MÁLAGA

Entrada: Vector de observaciones $\boldsymbol{X_n}$ (vector de tamaño n, con las n características calculadas)
Variables:
- $\boldsymbol{coeff}$ (matriz interna de tamaño $m \times n$ con los coeficientes del PCA, dónde m es el número de componentes principales)
- $\boldsymbol{\mu}$ (*buffer* interno de tamaño n con el vector de media)
- $\boldsymbol{X_\mu}$ (*buffer* interno de tamaño n al que se le ha substraído el vector de media $\boldsymbol{\mu}$)
- sum (suma interna para cada iteración en la multiplicación de $\boldsymbol{X_\mu}$ por $\boldsymbol{coeff}$)

Procedimiento:
Aplicamos → directiva HLS DATAFLOW

1. Restamos el vector de media $\boldsymbol{\mu}$ de $\boldsymbol{X_n}$:
    for $i \leftarrow 1\ to\ n$ do
        # Aplicamos → directiva HLS PIPELINE
        $X_\mu(i) = X_n(i) - \mu(i)$
    end

2. Multiplicamos $\boldsymbol{X_\mu}$ por la matriz de coeficientes del PCA $\boldsymbol{coeff}$:
    #Aplicamos → directiva HLS INLINE
    for $a \leftarrow 1\ to\ m$ do
        for $b \leftarrow 1\ to\ n$ do
            $sum = 0$
            $sum = sum + coeff(a,b) \times X_\mu(b)$
        end
        $X_{red}(a) = sum$
    end

Resultado: Vector de observaciones reducido $\boldsymbol{X_{red}}$ de tamaño m, con las m componentes principales

Figura 4.15. Pseudocódigo para el módulo *HLS Principal Component Analysis (PCA) Computing Module* en la figura 4.8.

Finalmente, se implementa el clasificador *k-means*, de la misma forma en que se presentó en el pseudocódigo de la figura 3.34 (sección 3.6 del capítulo 3). Asimismo, los centroides se obtuvieron a partir del entrenamiento *offline* en Matlab® del algoritmo, y se almacenaron en la memoria del ARM® del SoC.

4.5. Resultados y discusión

Esta sección muestra los resultados obtenidos de las palpaciones de objetos descritas en la sección 4.3. Concretamente, los procesos de obtención de datos, extracción de características y entrenamiento para la clasificación se explicaron en la subsección 4.3.3, sección 4.2 y subsección 4.3.4, respectivamente.

4.5.1. Resultados obtenidos sin PCA

Para determinar la mejor implementación, es necesario considerar diferentes combinaciones de sensores (dedo, palma), características (ver tabla 4.2), número de momentos de la imagen por sensor y bits por característica. Se utilizó Matlab® para obtener los resultados de la tabla 4.3, para diferentes combinaciones de momentos $M_{p,q}$, características y bits por característica (nbits/caract), respectivamente.

UNIVERSIDAD DE MÁLAGA

Tabla 4.2. Etiquetas para la combinación de características: área bajo la curva (AR), valor máximo (MAX), valor de magnitud en la parte ascendente a los $2/3$ del tiempo hasta alcanzar el máximo ($H1$) y valor de magnitud a $1/3$ de ese mismo tiempo después de alcanzar el máximo ($H2$).

Combinación de características	Etiqueta	Combinación de características	Etiqueta
AR	d1	MAX y $H2$	d9
MAX	d2	$H1$ y $H2$	d10
$H1$	d3	AR, MAX y $H1$	d11
$H2$	d4	AR, MAX y $H2$	d12
AR y MAX	d5	AR, $H1$ y $H1$	d13
AR y $H1$	d6	MAX, $H1$ y $H2$	d14
AR y $H2$	d7	AR, MAX, $H1$ y $H2$	d15
MAX y $H1$	d8		

Tabla 4.3. Porcentajes de clasificación sin PCA.

Sensor	nbits/ caract	Combinación de los momentos de orden $M_{p,q}$					
		$M_{0,0}$	$M_{0,0}$,$M_{0,1}$	$M_{0,0}$,$M_{0,1}$, $M_{1,0}$	$M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$	$M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$	$M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$,$M_{1,1}$
Dedo	8	$< 23\,\%$ (todos los casos)	$< 45\,\%$ (todos los casos)	$68.8\,\%$ (d15)	$82.3\,\%$ (d15)	$88.4\,\%$ (d11)	$89.1\,\%$ (d5)
	12	$63.2\,\%$ (d11)	$80.5\,\%$ (d5)	$86.8\,\%$ (d11)	**$91.7\,\%$ (d5)**	$89.6\,\%$ (d5)	$90.5\,\%$ (d5)
	16	$75\,\%$ (d11)	$83.6\,\%$ (d11)	$88.5\,\%$ (d11)	$90.9\,\%$ (d11)	$90.2\,\%$ (d5)	$89.9\,\%$ (d5)
Palma	8	$< 50\,\%$ (todos los casos)	$79.2\,\%$ (d11)	$86.2\,\%$ (d11)	$86.9\,\%$ (d11)	$86.9\,\%$ (d5)	$89\,\%$ (d5)
	12	$84.8\,\%$ (d11)	$87.5\,\%$ (d5)	$88.2\,\%$ (d5)	$88\,\%$ (d5)	$87.7\,\%$ (d5)	$89.3\,\%$ (d5)
	16	$82\,\%$ (d5)	$84.2\,\%$ (d11)	$88.4\,\%$ (d5)	$87.1\,\%$ (d5)	$87.8\,\%$ (d5)	**$89.6\,\%$ (d5)**
Dedo y palma	8	$68.6\,\%$ (d11)	$93.3\,\%$ (d11)	$97.3\,\%$ (d11)	**$95.5\,\%$ (d11)**	**$97.2\,\%$ (d11)**	**$98.5\,\%$ (d11)**
	12	**$94.5\,\%$ (d5)**	**$96.4\,\%$ (d5)**	$97\,\%$ (d5)	**$95.5\,\%$ (d5)**	$95.7\,\%$ (d5)	$98.2\,\%$ (d5)
	16	$93.5\,\%$ (d5)	$96.3\,\%$ (d5)	**$97.5\,\%$ (d5)**	$95.1\,\%$ (d5)	$96.1\,\%$ (d5)	$98.2\,\%$ (d8)

Para evaluar el consumo de recursos, primero se identificaron los casos con un alto porcentaje de clasificación (en negrita en la tabla 4.3). Estos casos se implementaron en el chip Zynq™-7000. De manera similar a cómo se hacía en las texturas (ver sección 3.8 del capítulo 3), el consumo de recursos se registró a partir del informe de utilización de Vivado IDE®. Los resultados se muestran en la tabla 4.4, observándose que solamente se requiere un número limitado de elementos de memoria (memoria distribuida (LUTRAM), bloques de memoria RAM empotrados (BRAM) y pares de Flip-Flops (FF)) para estas implementaciones. Más específicamente, la implementación únicamente utiliza *slice registers* (Flip-Flops). Cabe destacar que no se utilizan bloques BRAM ni DSP empotrados en el cálculo de las características, lo que hace más fácil la migración de este diseño a otras plataformas FPGA.

UNIVERSIDAD DE MÁLAGA

Tabla 4.4. Datos de rendimiento para la implementación de la extracción de características sin PCA (módulo 'VHDL Features Computing Module' en la figura 4.8), tal y como se proporciona de los informes de utilización de Vivado IDE®. Las dos últimas columnas indican el retardo de entrada-salida del módulo (en ns), y su consumo de potencia (en mW).

Mejor caso	LUTRAM	BRAM	Pares de FF	LUT Logic	DSP	F7 MUXES	t_{fe} (ns)	Consumo de potencia (mW)
Dedo: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$ (d5, 12 bits/caract)	0	0	14	19	0	0	30	4
Dedo y palma: $M_{0,0}$ (d5, 12 bits/caract)	0	0	28	38	0	0	30	2
Dedo y palma: $M_{0,0}$,$M_{0,1}$ (d5, 12 bits/caract)	0	0	56	76	0	0	30	4
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$ (d5, 16 bits/caract)	0	0	108	136	0	0	30	6
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$ (d11, 8 bits/caract)	0	0	72	128	0	0	30	12
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$ (d5, 12 bits/caract)	0	0	112	152	0	0	30	8
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$ (d11, 8 bits/caract)	0	0	90	160	0	0	30	15
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$,$M_{1,1}$ (d11, 8 bits/caract)	0	0	108	192	0	0	30	18
Palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$,$M_{1,1}$ (d5, 16 bits/caract)	0	0	84	114	0	0	30	6

Puesto que el número de características para cada caso de la tabla 4.4 es mayor que 3, es difícil visualizar la separación entre clases en el espacio de características. Para evitar este problema, calculamos la distancia euclídea de cada vector del conjunto de entrenamiento al correspondiente centroide asociado a su clase. El resultado se muestra en la figura 4.16, donde se pueden ver los histogramas de distancias para el centroide asociado a cada clase de la tabla 4.4.

En la figura 4.16 vemos que, en general, la mayoría de vectores de características se encuentran próximos al centroide de su respectiva clase, lo que hace que el porcentaje de clasificación sea alto, superando el $90\,\%$ en todos los casos. Por otro lado, aunque el caso 'Dedo y palma (nMom6-8bits/caract-d11)' proporciona una clasificación del $98.51\,\%$, otras implementaciones están más optimizadas en términos de recursos hardware y de uso de memoria, como ocurre para 'Dedo y palma (nMom1-12bits/caract-d5)'.

UNIVERSIDAD DE MÁLAGA

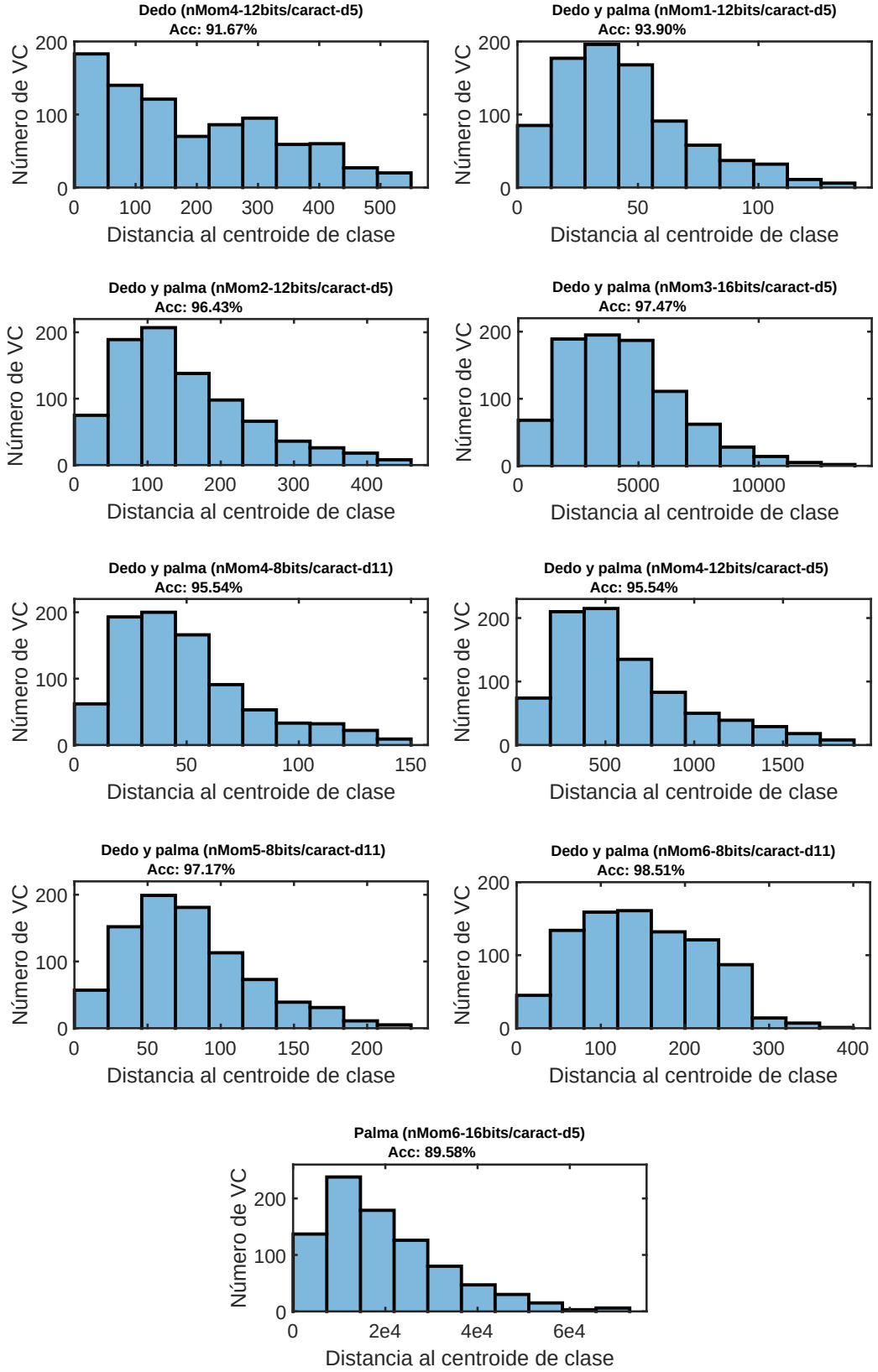


Figura 4.16. Histogramas de distancias con el correspondiente porcentaje de clasificación para los nueve casos de la tabla 4.4. En la figura, VC significa "vectores de características".

UNIVERSIDAD DE MÁLAGA

En este punto, para comparar diferentes opciones, proponemos el uso de la figura de mérito definida en (4.4):

$$f(\alpha, d, nBits, r) = \frac{\alpha}{d \cdot nBits \cdot r} \tag{4.4}$$

donde α se corresponde con el porcentaje de clasificación, d es el número total de características a la entrada del clasificador, $nBits$ es el número de bits por característica y r es el número total de recursos de hardware utilizados, como se define en (4.5):

$$r = \frac{1}{6} \cdot \left[\frac{\text{LUTRAM}}{\max(\text{LUTRAM})} + \frac{\text{BRAMs}}{\max(\text{BRAMs})} + \frac{\text{Pares de FF}}{\max(\text{Pares de FF})} + \frac{\text{DSPs}}{\max(\text{DSPs})} + \right. \\ \left. + \frac{\text{LUT Logic}}{\max(\text{LUT Logic})} + \frac{\text{F7-MUXES}}{\max(\text{F7-MUXES})} \right] \tag{4.5}$$

donde max(LUTRAM), max(BRAMs), max(Pares de FF), max(DSPs), max(LUT Logic), max(F7-MUXES) son el máximo número de LUTRAM, BRAMs, pares de FF, DSPs, LUT Logic y F7-MUXES que contiene el dispositivo Zynq™-7000 System on Chip (SoC) XC7Z020-CLG484-1, respectivamente.

La figura 4.17 presenta los valores de la figura de mérito (4.4) para cada entrada de la tabla 4.4. Las etiquetas en la figura indican el caso correspondiente junto con su porcentaje de clasificación. Entre todas las implementaciones, la que ofrece el mejor rendimiento en términos de recursos hardware y porcentaje de clasificación es 'Dedo y palma (nMom1-12bits/caract-d5)'. Esta implementación requiere menos recursos en comparación con las demás, que utilizan un mayor número de momentos de la imagen por sensor (parte derecha de la figura 4.17). Aunque estas implementaciones logran altos porcentajes de clasificación, también consumen más recursos hardware. Por otro lado, la implementación para 'Dedo y palma (nMom1-12bits/caract-d5)' sólo utiliza los primeros momentos de la imagen táctil para el dedo y la palma, la característica d5 (una combinación de las características individuales AR y MAX en la tabla 4.2), y 12 bits por característica; logrando una clasificación cercana al $94\,\%$.

La matriz de confusión para el mejor caso, 'Dedo y palma (nMom1-12bits/caract-d5)', de acuerdo a la figura 4.17, se muestra en la figura 4.18. Esta matriz de confusión muestra que con esta implementación se clasificaron de forma precisa las 42 clases de objetos de la figura 4.5, alcanzando una precisión de $94.5\,\%$ en los experimentos realizados. A pesar del tamaño de los datos del conjunto de test (665 vectores de características), solamente unos pocos casos se clasificaron incorrectamente.

UNIVERSIDAD DE MÁLAGA

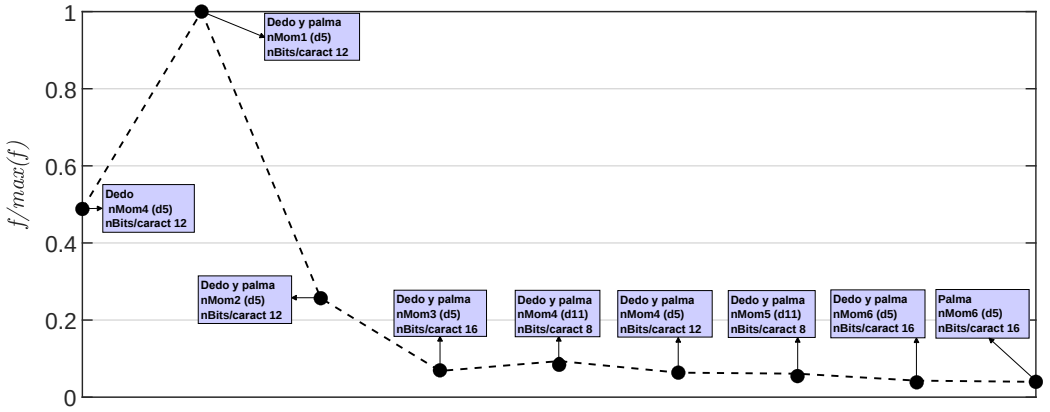


Figura 4.17. Resultado de la figura de mérito en (4.4) normalizada respecto al mejor caso, cuando no se aplica PCA.

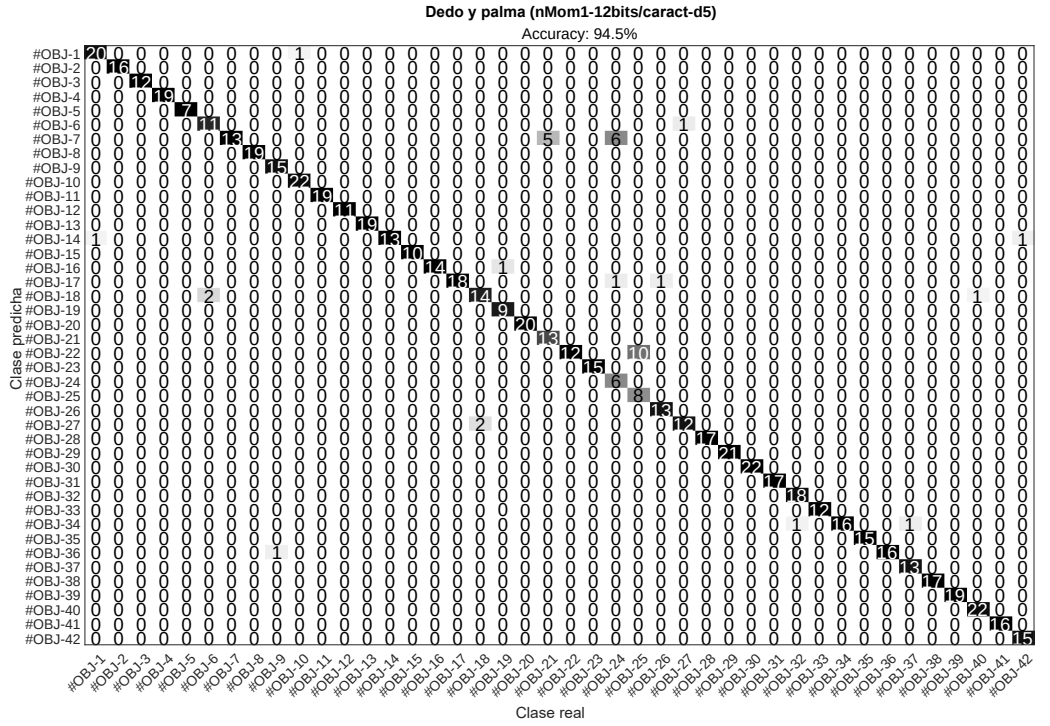


Figura 4.18. Matriz de confusión para la implementación óptima de la figura 4.17. Este caso 'Dedo y palma (nMom1-12bits/caract-d5)' (segunda fila de la tabla 4.4) utiliza el primer momento de la imagen $M_{0,0}$ para ambos sensores (dedo y palma), la característica d5 (ver tabla 4.2) y 12 bits por característica.

A partir de la matriz de confusión de la figura 4.18, se pueden calcular la sensibilidad y especificidad para cada clase, expresiones (3.32) y (3.33) en la sección 3.9 del capítulo 3. Una vez calculadas, se pueden representar las 42 clases en el espacio ROC de la figura 4.19.

De la figura 4.19 se pueden extraer algunas conclusiones respecto al procedimiento de extracción de características:

UNIVERSIDAD DE MÁLAGA

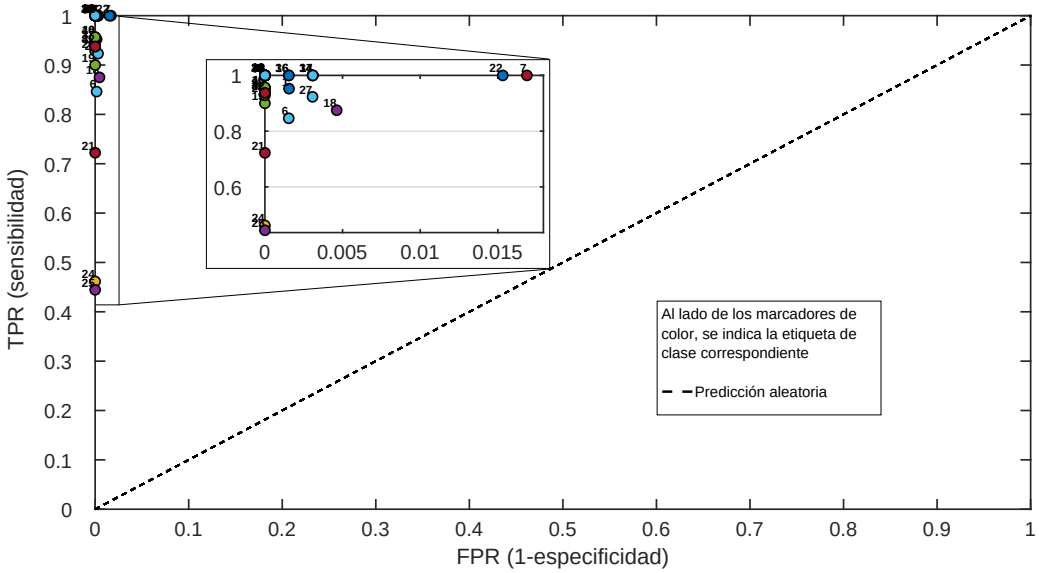


Figura 4.19. Representación en el espacio ROC para cada una de las 42 clases de objetos, cuando no se aplica PCA. En cada gráfico, el eje vertical representa la sensibilidad (la proporción de verdaderos positivos), y el eje horizontal muestra la tasa de falsos positivos (1−especificidad). Un modelo perfecto se ubicaría en la esquina superior izquierda del gráfico.

1. La sensibilidad es diferente para cada clase, con valores que van desde 0.4 hasta la unidad. Esto indica que el procedimiento varía en la identificación de los casos positivos de cada clase. Algunas, como la segunda, tercera, cuarta y quinta, muestran una sensibilidad perfecta de valor 1, lo que significa que se reconocen correctamente todos los casos positivos de estas clases.

2. Hay que prestar atención a aquellas clases en las que la sensibilidad es más baja, ya que pueden pasar por alto algunos casos positivos. Concretamente, las clases #OBJ-24 (correspondiente a 'Ovoide naranja para terapia de mano (horizontal)') y la #OBJ-25 (correspondiente a 'Ovoide verde para terapia de mano (boca arriba)') tienen los valores más bajos, de 0.46 y 0.44, respectivamente. Para estas clases, existe una correspondencia de mayor número de casos clasificados incorrectamente en la matriz de confusión de la figura 4.19.

3. Por otro lado, la mayoría de las clases tienen una especificidad cercana a la unidad, lo que indica que el procedimiento produce una baja tasa de falsos positivos.

4. En conclusión, la clasificación muestra un alto nivel de efectividad, ya que la mayoría de las clases tienen una sensibilidad y especificidad cercanas a 1 (esquina superior izquierda en la figura 4.19). Esto sugiere que el procedimiento es capaz de identificar correctamente los casos positivos y negativos la mayoría de las veces.

UNIVERSIDAD DE MÁLAGA

Con respecto al tiempo de computación, es el mismo para todos los supuestos de la tabla 4.4, tres ciclos de reloj después de que la última entrada de la secuencia de momentos se haya leído (ver sección 4.4), gracias a la implementación en paralelo ilustrada en la figura 4.12.

4.5.2. Resultados obtenidos con PCA

En la tabla 4.5 se presentan los resultados de clasificación para los mismos casos de la tabla 4.3, cuando se considera la aplicación de la técnica PCA.

Tabla 4.5. Porcentajes de clasificación con PCA. En la tabla, nCm es el número de componentes principales.

Sensor	nbits/ caract	Combinación de los momentos de orden $M_{p,q}$					
		$M_{0,0}$	$M_{0,0}$,$M_{0,1}$	$M_{0,0}$,$M_{0,1}$, $M_{1,0}$	$M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$	$M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$	$M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$,$M_{1,1}$
Dedo	8	$< 30\,\%$ (todos los casos)	$< 45\,\%$ (todos los casos)	$64.4\,\%$ (d15, nC5)	$81.4\,\%$ (d11, nC5)	$85.9\,\%$ (d11, nC5)	$89\,\%$ (d11, nC6)
	12	$64.1\,\%$ (d15, nC4)	$83\,\%$ (d5, nC3)	$89.3\,\%$ (d5, nC4)	**90.9 %** **(d5, nC3)**	$90.6\,\%$ (d5, nC6)	$91.1\,\%$ (d5, nC5)
	16	$76.2\,\%$ (d5, nC2)	$85.3\,\%$ (d5, nC4)	$90.5\,\%$ (d5, nC4)	$91.2\,\%$ (d5, nC6)	$90.5\,\%$ (d11, nC6)	$90.8\,\%$ (d5, nC4)
Palma	8	$< 41\,\%$ (todos los casos)	$75.2\,\%$ (d11, nC4)	$85.2\,\%$ (d5, nC6)	$87.2\,\%$ (d5, nC6)	$88\,\%$ (d5, nC6)	$89.4\,\%$ (d5, nC5)
	12	$84.5\,\%$ (d11, nC3)	$86.5\,\%$ (d5, nC4)	$88.5\,\%$ (d5, nC6)	**89 %** **(d5, nC5)**	$87.1\,\%$ (d5, nC5)	$89.1\,\%$ (d5, nC6)
	16	$82.7\,\%$ (d11, nC3)	$87.5\,\%$ (d5, nC3)	$88.2\,\%$ (d11, nC6)	$88.5\,\%$ (d5, nC5)	$87.8\,\%$ (d5, nC5)	$89\,\%$ (d5, nC5)
Dedo y palma	8	$64.7\,\%$ (d15, nC4)	$92\,\%$ (d11, nC5)	$95.8\,\%$ (d11, nC4)	**94.5 %** **(d11, nC5)**	**96.3 %** **(d11, nC5)**	**94.4 %** **(d11, nC3)**
	12	**94.6 %** **(d5, nC3)**	**97.3 %** **(d5, nC5)**	$97.3\,\%$ (d5, nC5)	**95.2 %** **(d5, nC5)**	$96.1\,\%$ (d5, nC5)	$95.5\,\%$ (d8, nC3)
	16	$95.1\,\%$ (d5, nC4)	$96.9\,\%$ (d5, nC5)	**97 %** **(d11, nC5)**	$95.1\,\%$ (d5, nC6)	$96.9\,\%$ (d5, nC6)	$97\,\%$ (d8, nC5)

Con el objetivo de visualizar la separación entre clases, la figura 4.20 muestra la distribución de las clases en un espacio bidimensional con dos componentes principales, cuando se consideran los sensores del dedo, palma, y dedo y palma, respectivamente. En todos los casos, se utilizan los seis primeros momentos de la imagen y, a modo de ejemplo, se aplica la característica d8 (tabla 4.2) con 16 bits por característica. Los conjuntos de entrenamiento para cada clase de la figura 4.20 se representan como distribuciones elípticas gaussianas. El centroide de cada clase se corresponde con un *array* de 2 valores y se determina a través del procedimiento de entrenamiento señalado en la subsección 4.3.4. Las clases de la figura 4.20(c) son más compactas, es decir, la distancia de los vectores de características al centroide correspondiente a sus respectivas clases es menor. Por lo tanto, las clases son más distinguibles que las de la figura 4.21(a),(b). En conclusión, el clasificador alcanza mayores porcentajes de clasificación al recibir información de múltiples fuentes (en este caso, dedo y palma conjuntamente), que de fuentes individuales (dedo o palma por separado).

UNIVERSIDAD DE MÁLAGA

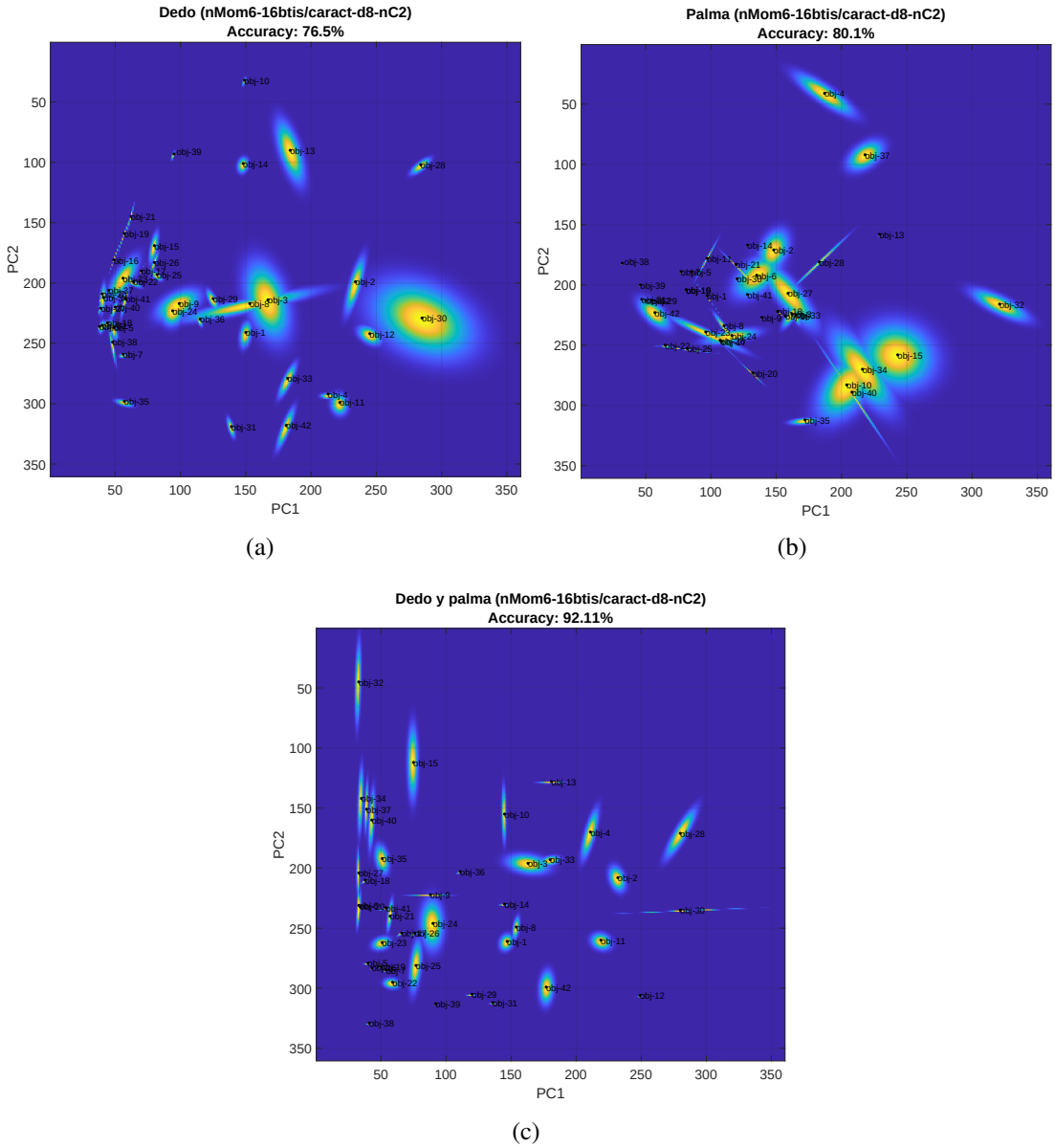


Figura 4.20. Comparación de la distribución de los datos en un espacio de características de 2 componentes principales para: (a) dedo, (b) palma, (c) dedo y palma. En todos los casos, el número de momentos es 6, la característica utilizada es d8 de la tabla 4.2, y el número de bits por característica es 16. En la figura, PC1 representa la componente principal 1 y PC2 representa la componente principal 2. La clase correspondiente a cada centroide se incluye al lado del mismo.

Por otro lado, la figura 4.21 muestra la distribución de las clases en un espacio tridimensional con tres componentes principales, cuando se consideran los sensores del dedo, palma, y dedo y palma, respectivamente. En todos los casos, se utilizan los seis momentos de la imagen y, también a modo de ejemplo, se aplica la característica d5 (tabla 4.2) con 8 bits por característica. Los conjuntos de entrenamiento para cada clase de la figura 4.21 se representan como distribuciones elipsoidales gaussianas. En esta ocasión, el centroide de cada clase se corresponde con un *array* de 3 valores que se determina a través del procedimiento de entrenamiento señalado en la subsección 4.3.4. Para este escenario,

UNIVERSIDAD DE MÁLAGA

ocurre lo mismo que para el caso bidimensional. Las clases en la figura 4.21(c) son más distinguibles que las de la figura 4.21(a),(b), en las que la distancia de cada vector de características a su centroide correspondiente es mayor. Por tanto, el clasificador obtiene un mayor porcentaje de clasificación, al tener información de más de un sensor. Este comportamiento se observa tanto en los escenarios en los que no se aplica PCA como en aquellos en los que sí se aplica, como se puede verificar en las tablas 4.3 y 4.5.

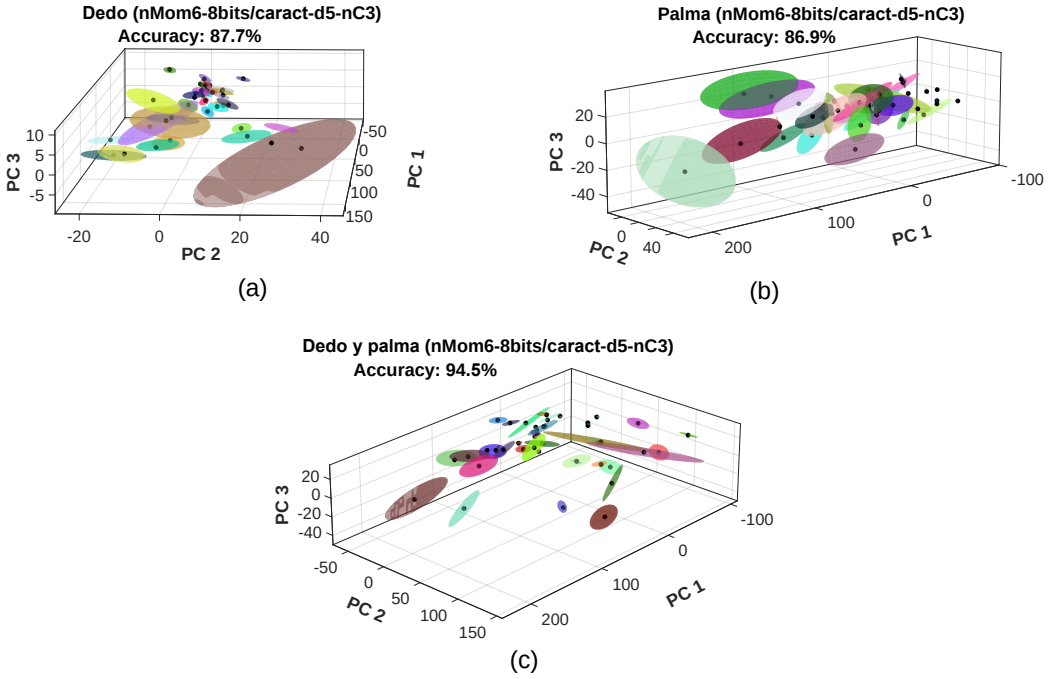


Figura 4.21. Comparación de la distribución de los datos en un espacio de características de 3 componentes principales para: (a) dedo, (b) palma, (c) dedo y palma. En todos los casos, el número de momentos es 6, la característica utilizada es d5 de la tabla 4.2, y el número de bits por característica es 8. En la figura, PC1 representa la componente principal 1, PC2 representa la componente principal 2, y PC3 representa la componente principal 3.

De la misma forma que cuando no se aplicó PCA, se seleccionaron aquellos casos que ofrecieron mejor rendimiento en términos de precisión (destacados en negrita en la tabla 4.5). Los resultados del consumo de recursos hardware se presenta en la tabla 4.6. En esta ocasión, el número de ciclos que lleva la aplicación de PCA depende del número de componentes principales (ver última columna de la tabla 4.6). El consumo de recursos de memoria es similar en todos los casos, lo que significa que incrementar la dimensión del vector de características no implica un aumento significativo de recursos.

La figura 4.22 muestra una comparación de las implementaciones presentadas en la tabla 4.6, a partir de la figura de mérito definida en (4.4). La mejor combinación corresponde a 'Dedo y palma, nMom1 (d5), nBits/caract 12, nC3'. La matriz de confusión correspondiente se muestra en la figura 4.23.

UNIVERSIDAD DE MÁLAGA

Tabla 4.6. Datos de rendimiento para la implementación de la extracción de características con PCA (módulo 'VHDL Features Computing Module' más 'HLS Principal Component Analysis (PCA) Computing Module' en la figura 4.8), tal y como se proporciona de los informes de utilización de Vivado IDE®. Las columnas de la tabla son las mismas que la de la tabla 4.4.

Mejor caso con PCA	LUTRAM	BRAM	Pares de FF	LUT Logic	DSP	F7 MUXES	t_{fe} (μs)	Consumo de potencia (mW)
Dedo: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$ (d5, 12 bits/caract, nC3)	40	2	441	1076	7	6	2.75	34
Dedo y palma: $M_{0,0}$ (d5, 12 bits/caract, nC3)	36	2	443	1078	7	6	1.43	32
Dedo y palma: $M_{0,0}$,$M_{0,1}$ (d5, 12 bits/caract, nC5)	36	3	454	1110	7	6	4.55	37
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$ (d5, 16 bits/caract, nC4)	48	3	516	1219	7	6	5.41	39
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$ (d11, 8 bits/caract, nC5)	36	3	478	1185	7	6	13.35	46
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$ (d5, 12 bits/caract, nC5)	36	3	518	1198	7	6	8.95	41
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$ (d11, 8 bits/caract, nC5)	36	3	507	1235	7	6	16.65	49
Dedo y palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$,$M_{1,1}$ (d11, 8 bits/caract, nC5)	24	4	495	1310	7	6	19.95	54
Palma: $M_{0,0}$,$M_{0,1}$, $M_{1,0}$,$M_{0,2}$, $M_{2,0}$,$M_{1,1}$ (d5, 16 bits/caract, nC5)	48	3	506	1202	7	6	13.35	40

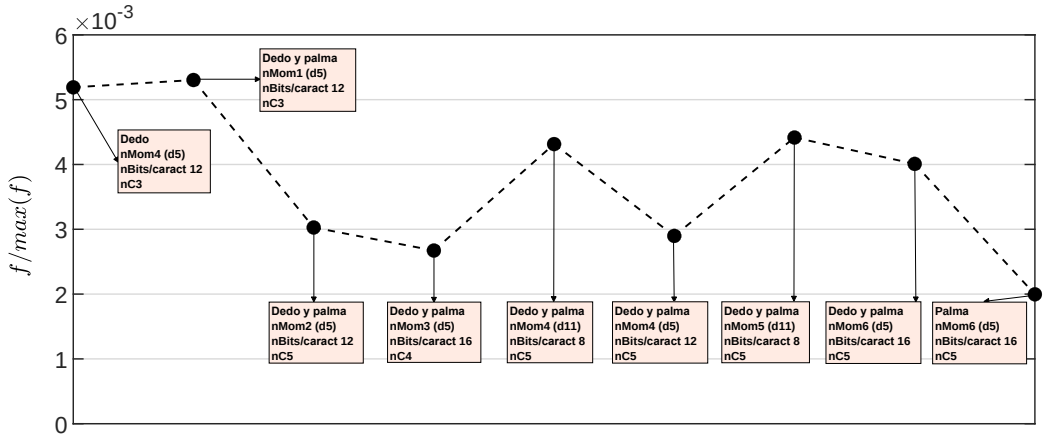


Figura 4.22. Resultado de la figura de mérito en (4.4) normalizado con respecto al mejor caso en la figura 4.17, cuando se aplica PCA.

Al igual que cuando no se aplica PCA, a partir de la matriz de confusión de la figura 4.23, se pueden calcular la sensibilidad y especificidad para cada clase. Una vez calculadas, se pueden representar las 42 clases en el espacio ROC de la figura 4.24.

UNIVERSIDAD DE MÁLAGA

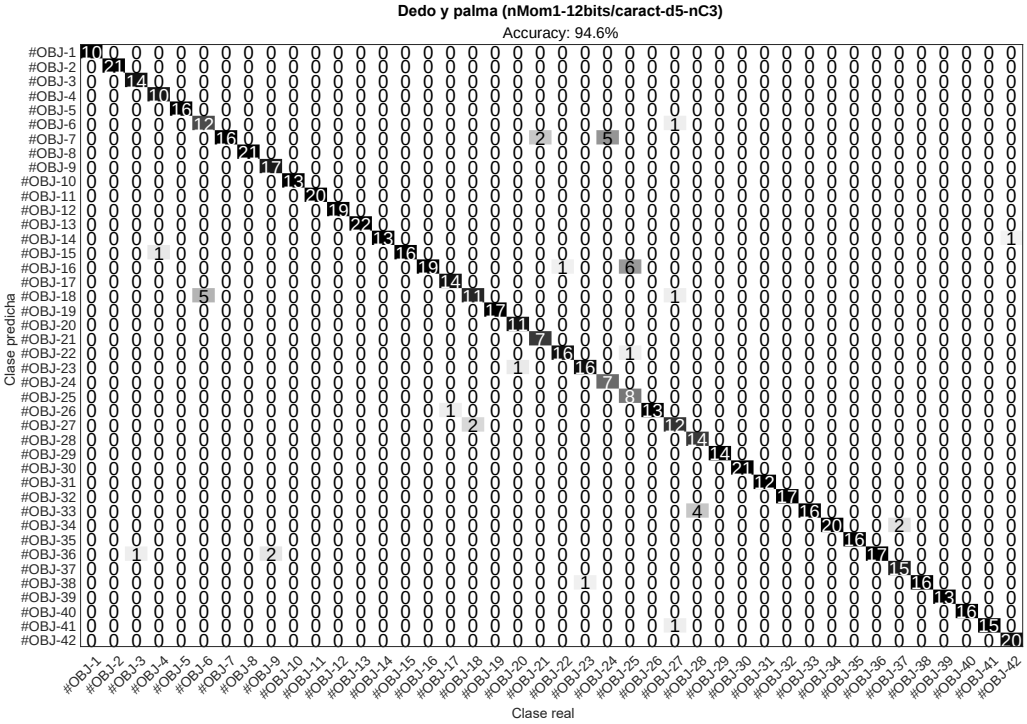


Figura 4.23. Matriz de confusión para la implementación óptima de la figura 4.22. Este caso 'Dedo y palma, nMom1 (d5), nBits/caract 12, nC3' (segunda fila de la tabla 4.6) utiliza el primer momento de la imagen $M_{0,0}$ para ambos sensores (dedo y palma), la característica d5 (ver tabla 4.2), 12 bits por característica y 3 componentes principales.

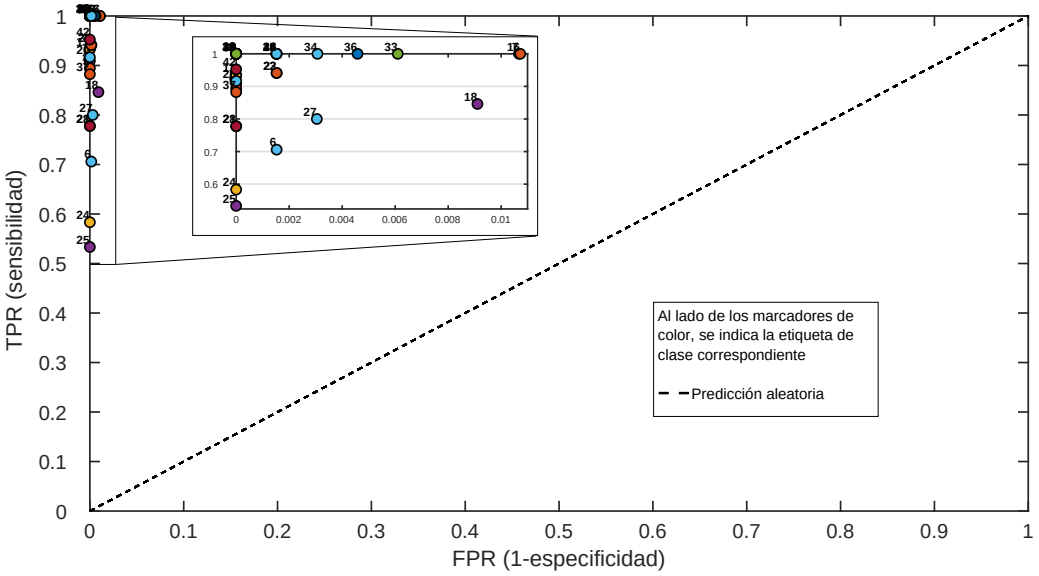


Figura 4.24. Representación en el espacio ROC para cada una de las 42 clases de objetos, cuando se aplica la técnica de PCA. En cada gráfico, el eje vertical representa la sensibilidad (la proporción de verdaderos positivos), y el eje horizontal muestra la tasa de falsos positivos (1−especificidad). Un modelo perfecto se ubicaría en la esquina superior izquierda del gráfico.

De la figura 4.24 se pueden extraer algunas conclusiones para la aplicación de PCA:

1. La sensibilidad varía entre las clases, con valores que oscilan entre 0.53 hasta la

UNIVERSIDAD DE MÁLAGA

unidad. Algunas clases, como la primera, la segunda y la quinta, muestran una sensibilidad perfecta de valor 1, lo que significa que el procedimiento identifica correctamente todos los casos positivos de estas clases. Sin embargo, otras clases, como la séptima, la decimosexta y la vigésimo quinta, tienen una sensibilidad inferior, lo que indica que el procedimiento puede pasar por alto algunos casos positivos en estas clases.

2. Al igual que ocurría cuando no se aplicaba PCA, las clases #OBJ-24 (con sensibilidad de 0.58) y #OBJ-25 (con sensibilidad de 0.53), correspondientes a 'Ovoide naranja para terapia de mano' en posición horizontal, y 'Ovoide verde para terapia de mano' en posición boca arriba, respectivamente, producen más falsos positivos. Esto también se puede comprobar en la matriz de confusión de la figura 4.23, donde se clasificaron más casos como incorrectos, respecto a las demás clases.

3. Por otra parte, la mayoría de las clases presentan una especificidad cercana a la unidad, lo que significa que el procedimiento no produce falsos positivos para la mayoría de las clases.

4. En cualquier caso, la clasificación demuestra de nuevo un buen rendimiento, al igual que en el caso sin PCA, con una sensibilidad y especificidad generalmente altas. Por lo tanto, el clasificador identifica correctamente los casos positivos y negativos para la mayoría de las clases.

La figura 4.25 muestra el resultado de agregar el retardo de entrada-salida de la extracción de características (ver tabla 4.4 y tabla 4.6) al retardo del clasificador. Este último se midió con un osciloscopio MDO4104B-6 Mixed Domain Tektronix® de 6 GHz. La dimensión del vector de características (número de características como componentes) es la variable en el eje de abscisas de la figura 4.25. Se puede observar que cuanto mayor es el número de características, más tiempo tarda el clasificador en proporcionar una salida. Dado que el número de características es menor en las implementaciones con el módulo PCA, estas realizaciones tienen un mejor rendimiento en este aspecto. La figura 4.26 muestra la medida en osciloscopio para dos de los valores de tiempo en la figura 4.25.

4.6. Conclusiones

A partir del análisis de resultados, se puede concluir que los mejores resultados en términos de porcentaje de clasificación se logran cuando ambos sensores (dedo y palma) se usan en conjunto, obteniendo una precisión por encima del $94\,\%$. La implementación de

UNIVERSIDAD DE MÁLAGA

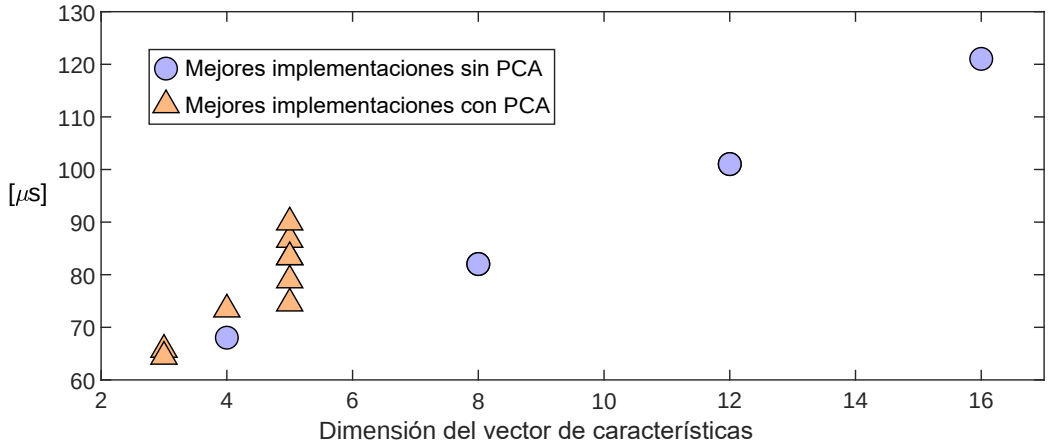


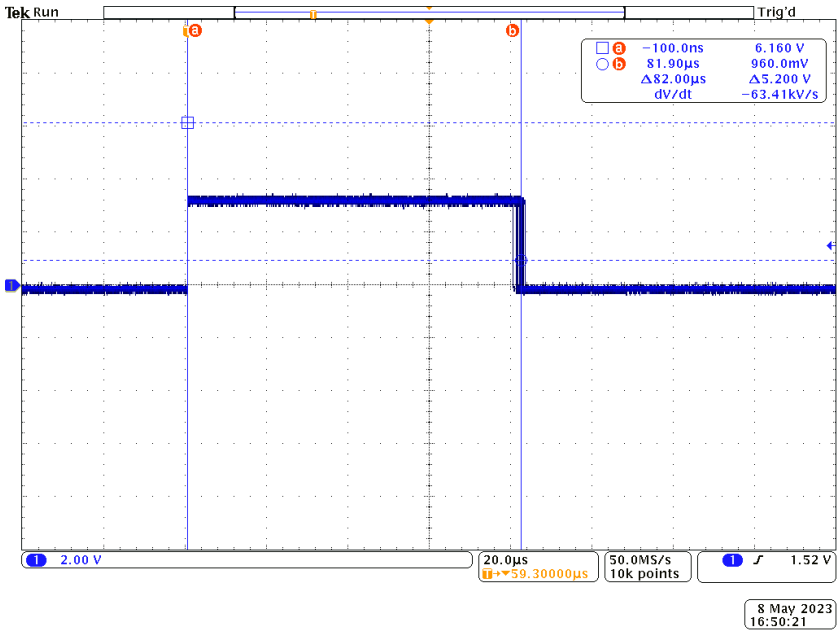
Figura 4.25. Tiempo correspondiente al retardo de extracción de características más el retardo de clasificación, frente a la dimensión del vector de características de entrada del clasificador. Los resultados corresponden a los casos en la tabla 4.4 y 4.6, para las implementaciones sin PCA y con PCA, respectivamente.

la extracción de características en la FPGA cuando no se aplica PCA es más eficiente en términos de recursos (ver tablas 4.4 y 4.6) y consumo de potencia (2 mW en contra de los 32 mW cuando se aplica PCA).

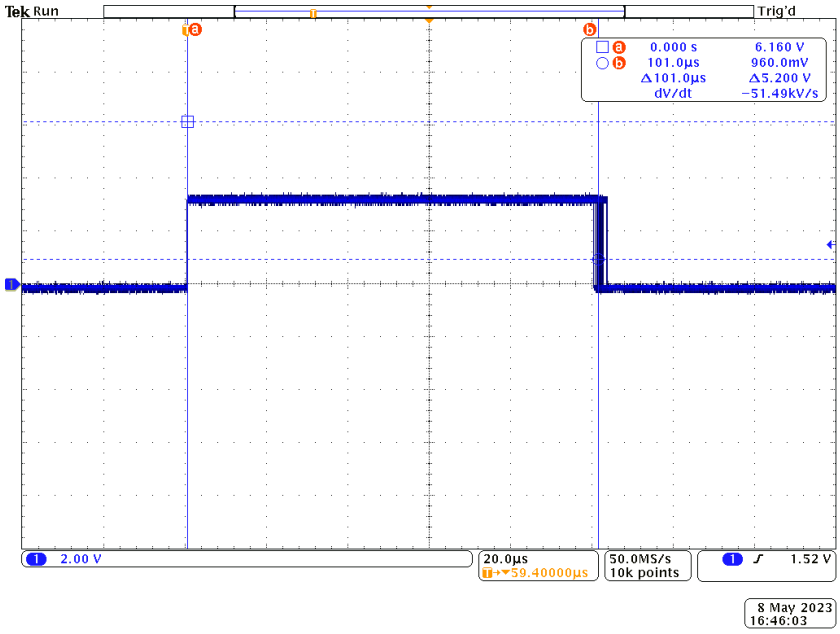
La aplicación de PCA puede resultar ventajosa para reducir el retardo de entrada-salida del clasificador y aumentar la separación entre clases. Además, es interesante considerar el caso en el que se utiliza solamente el sensor correspondiente al dedo, lo que simplifica el sistema. En esta situación, si los primeros cuatro momentos de la imagen táctil se registran, se alcanza un porcentaje de clasificación del $91.7\,\%$ sin PCA y del $90.9\,\%$ con PCA y 3 componentes principales. Por tanto, vemos que los porcentajes de clasificación son similares en ambas situaciones, aunque la utilización de recursos en la extracción de características es mucho mayor cuando se aplica PCA, y el consumo de potencia de esta parte resultó en 34 mW en contra de los 4 mW sin PCA. Sin embargo, el tiempo de extracción de características más el retardo de entrada-salida del clasificador fue de 65.7 μs con PCA y de 82 μs sin PCA.

En conclusión, con el *setup* experimental y el conjunto de clases utilizados en este estudio, el procedimiento que no aplica PCA es mejor, puesto que la estrategia propuesta, basada en el cálculo sencillo de cuatro características a partir de las secuencias de momentos táctiles, resulta en una realización muy eficiente. Una mejora en el retardo de entrada-salida se observa con PCA, siendo incluso más significativa si únicamente se considera el dedo como sensor de entrada. La elección entre ambas opciones también depende de la complejidad de otras tareas que tengan que llevarse a cabo en tiempo real.

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura 4.26. Medida con osciloscopio de tiempos de retardo de extracción de características más el retardo de clasificación, frente a la dimensión del vector de características de entrada del clasificador, para dos casos de la figura 4.25: (a) 81.90 μs y (b) 101 μs.

4.7. Resumen

En este capítulo se propone la implementación de una estrategia para el reconocimiento y clasificación de objetos de diferente rigidez, a partir de experimentos en los que se utilizan sensores táctiles que cuentan con electrónica embebida. A continuación, se hace

UNIVERSIDAD DE MÁLAGA

un resumen de los contenidos del capítulo:

- En la sección 4.1, se destaca la importancia de la detección de rigidez de objetos en diversas aplicaciones, como la robótica, la medicina y la industria alimentaria. Se destaca que los sensores táctiles inteligentes pueden proporcionar información valiosa sobre las propiedades mecánicas de objetos, lo que puede ser útil en tareas de clasificación y reconocimiento. Se establece como objetivo principal la implementación de una propuesta de algoritmo para la detección de rigidez de objetos utilizando sensores táctiles inteligentes.

- En la sección 4.2 se comentan inicialmente los resultados preliminares sobre 9 objetos de geometría básica, presentados en el congreso ISIE [43]. A continuación, se propone el uso de cuatro características simples, extraídas de las curvas de momentos de la imagen. Además, se considera la aplicación de técnicas PCA para reducir la dimensionalidad de los datos.

- Para obtener las curvas temporales de momentos de la imagen se parte del *setup* experimental y sensores táctiles descritos en el capítulo 2. En la sección 4.3, con el objetivo de aumentar el estudio a más objetos de diferentes propiedades y en distintas posiciones, se presenta el conjunto de 42 clases de objetos considerado en esta parte de la tesis. Además, se explica el procedimiento experimental de palpación llevado a cabo. Por último, se propone el uso del algoritmo de entrenamiento y clasificación *k-means*, que también se consideró para la detección de texturas.

- La propuesta de extracción de características se implementa en el chip SoC Zynq^{TM}-XC7Z020-CLG484-1. En la sección 4.4 se describe la arquitectura global de implementación, así como los módulos específicos de extracción de características, aplicación de la técnica PCA y sincronización de señales, implementados en la parte PL del SoC. La parte PS se encarga de la clasificación, de modo similar a como se hacía en el capítulo 3 para las texturas.

- En la sección 4.5 de resultados, se discuten los resultados obtenidos cuando se aplica la técnica de PCA, y cuando no se aplica PCA. Se seleccionan aquellos casos que ofrecen mejores prestaciones en términos de consumo de recursos y potencia en la implementación FPGA, y se presentan las matrices de confusión y representaciones ROC para los mismos.

- Finalmente, en la sección 4.6, se concluye que aunque la aplicación de PCA logra una mejora en el retardo de entrada-salida, el procedimiento que no aplica PCA es

UNIVERSIDAD DE MÁLAGA

más eficiente en términos de consumo de recursos y potencia en una implementación en tiempo real en FPGA. En la sección 4.7 se resumen los contenidos de este capítulo.

UNIVERSIDAD DE MÁLAGA

5

Conclusiones y líneas futuras

En este último capítulo de la tesis, se exponen las principales conclusiones del trabajo de investigación (sección 5.1) y se proponen posibles líneas de trabajo futuras (sección 5.2).

5.1. Conclusiones

En esta tesis, se ha abordado la implementación de primitivas de preprocesamiento en FPGA para la detección de texturas y reconocimiento de rigidez. Estas propiedades son de gran interés en sistemas robóticos que posean manos o pinzas artificiales equipadas con sensores táctiles. A través del tacto activo, se han explorado superficies con distintas frecuencias espaciales y se han reconocido objetos por su rigidez. Los sensores táctiles inteligentes se colocan en los dedos y palma de la mano robótica. A partir de la información táctil de entrada a estos sensores, se ha considerado la implementación de algoritmos de extracción de características, ejecutados en la electrónica local del sensor basada en FPGAs. De esta forma, se consigue descargar al procesador central de tareas que requieran una alta computación y consumo de recursos. Para evaluar el rendimiento de los algoritmos de pre-procesado, se utiliza un dispositivo SoC. Este tipo de dispositivos contienen una parte hardware basada en FPGA (PL), y una parte de sistema con un microprocesador (PS). En esta tesis, se decide la implementación de los algoritmos de

UNIVERSIDAD DE MÁLAGA

pre-procesado en la parte PL del SoC, dejando para la parte PS la ejecución en tiempo real de un clasificador, en el núcleo ARM® del SoC.

Las principales aportaciones del trabajo de investigación llevado a cabo son las siguientes:

1. Se ha propuesto e implementado un algoritmo de barrido para la extracción de características frecuenciales, en la electrónica local (basada en FGPA) de un sensor táctil inteligente. La aplicación en tiempo real del algoritmo de barrido planteado demuestra que su implementación es exitosa para la distinción de un conjunto de 12 texturas diferentes.

2. En la identificación de texturas, se midieron el consumo de recursos lógicos, de memoria, de potencia y latencia de la implementación hardware del algoritmo de barrido para diferentes tamaños del vector de salida, diferentes tamaños de ventana en la aplicación del filtro de media móvil y diferentes factores de diezmado. Estos resultados se compararon con los de la implementación de una FFT.

3. Se llevó a cabo un estudio con el conjunto de 12 clases de texturas utilizando la implementación óptima del algoritmo de barrido. Modificando el tamaño del vector de características mediante el factor de diezmado, se obtuvieron curvas de resultados de clasificación en tiempo real para este algoritmo. Además, se comparó con la aplicación de una FFT, demostrando que la propuesta del barrido es robusta en comparación con los resultados de la FFT.

4. Para el mismo conjunto de 12 texturas, se llevó a cabo un análisis de los resultados de aplicación de la mejor implementación para el algoritmo de Goertzel. De igual forma que para el barrido, se comparó su rendimiento con el de la FFT, resultando su implementación más ventajosa en cuanto al consumo de recursos hardware y de potencia.

5. Para el reconocimiento de un conjunto de 42 clases de objetos de diferente rigidez, se ha implementado en FPGA un procedimiento, en el que se propone el uso de cuatro características obtenidas de los momentos de la imagen táctil. A partir de estos momentos, se obtiene información sobre el área de contacto y geometría del objeto. La combinación de un número determinado de estos momentos junto a las cuatro características propuestas proporciona buenos resultados en la identificación de estas clases.

6. En situaciones en las que se utilizan los sensores del dedo y la palma conjuntamente con los seis momentos de la imagen y las cuatro características propuestas,

UNIVERSIDAD DE MÁLAGA

el tamaño del vector de características es muy grande. Por tanto, se propuso la implementación de la técnica de reducción de la dimensionalidad de datos PCA. Se realizó un análisis comparativo de la aplicación de las cuatro características por separado y junto a PCA, tanto en términos de resultados de clasificación como en consumo de recursos lógicos, memoria, potencia y latencia.

En la detección de texturas, el algoritmo de barrido propuesto identifica las frecuencias espaciales de la superficie variando un parámetro de retardo. A partir del análisis de los resultados de clasificación e implementación, se comprobó que la estrategia óptima en términos de consumo y resultados de clasificación, se corresponde con el envío de las posiciones de los mínimos, sin filtro de media móvil y aplicando diezmado. Las principales ventajas del algoritmo propuesto residen en la simplicidad de su implementación, ya que con pocas características se puede deducir el contenido armónico de la textura. Por otra parte, esta simplificación puede limitar su capacidad. En el vector de características sólo se considera la localización de los mínimos, pero no su potencia. Otra aproximación que se ha propuesto, el algoritmo de Goertzel, sí transmite información de la potencia, pero para un número escogido de frecuencias en el espectro. El rendimiento de este algoritmo para las texturas usadas en los experimentos es bueno, aunque queda pendiente una investigación más exhaustiva que evalúe su dependencia del conjunto de frecuencias calculadas. En ambos casos, únicamente se usa la información de un canal. Sin embargo, esta limitación se supera escalando los algoritmos a un mayor número de sensores (tácteles), ya que son fácilmente replicables en hardware. Además, la implementación de los algoritmos en FPGA permite realizar en paralelo la adquisición y el pre-procesado de la imagen táctil. Por otro lado, aunque se asume que la velocidad de exploración es conocida y constante, esto no constituye una limitación fuerte, puesto que en sistemas robóticos se puede aplicar un control de velocidad para mejorar el rendimiento [95]. En cualquier caso, hay que tener en cuenta que la evaluación de la eficacia de los algoritmos se ve condicionada por diversos factores. En primer lugar, el rendimiento está estrechamente vinculado a la tecnología del sensor utilizado, dado que los datos registrados se generan a partir de la interacción entre el sensor y la superficie mediante la exploración por tacto activo. A su vez, el tamaño y la naturaleza del conjunto de texturas utilizadas para evaluar el rendimiento, también influyen en los resultados obtenidos.

En el reconocimiento de la rigidez de objetos, se ha considerado una propuesta de implementación hardware basada en la extracción de cuatro características simples, a partir de las secuencias temporales de momentos de la imagen táctil. Estas curvas de momentos se obtuvieron a través de la exploración de un conjunto de objetos de diferentes rigideces. Para reducir la dimensionalidad de los datos, se implementó la técnica de PCA. Aunque esta técnica reduce el tamaño del vector de características y el retardo de

UNIVERSIDAD DE MÁLAGA

entrada-salida en la etapa de clasificación, la implementación sin PCA resulta más favorable en términos de consumo de recursos lógicos y de memoria en la FPGA. Además, el consumo de potencia y tiempo de extracción es menor cuando no se utiliza PCA.

5.2. Líneas futuras de trabajo

Los resultados obtenidos en esta tesis han abierto nuevas oportunidades para explorar futuras líneas de investigación, con el objetivo de dar continuidad al trabajo realizado y mejorar el rendimiento de los algoritmos propuestos para la extracción de características. A continuación, se proponen algunas:

- Escalar los algoritmos propuestos a un mayor número de canales de información táctil, y en concreto, a cada uno de los tácteles en el sensor de la matriz táctil del dedo o palma.
- Estudiar el rendimiento del algoritmo de Goertzel en función de los índices calculados y del conjunto de texturas escogido.
- En el caso del reconocimiento de objetos por rigidez, estudiar la influencia del tamaño del objeto y la resolución espacial de los sensores táctiles en la etapa de clasificación.
- Integrar la capacidad del sistema para reconocer la deformación de los objetos con otras funciones, como la detección de texturas o la identificación del fenómeno del *slippage*.
- Implementar otros algoritmos de aprendizaje y clasificación, y comparar sus resultados con los obtenidos por el clasificador *k-means*.
- Desarrollar prototipos que incorporen los algoritmos de extracción de características en los sensores táctiles de manos o pinzas artificiales, para su aplicación en robótica e ingeniería biomédica.

UNIVERSIDAD DE MÁLAGA

Apéndice A Figuras complementarias

En este apéndice se presentan figuras adicionales que complementan los capítulos de la tesis.



Figura A.1. Imagen de la tarjeta de adquisición 6009 de National Instruments® (capítulo 2).

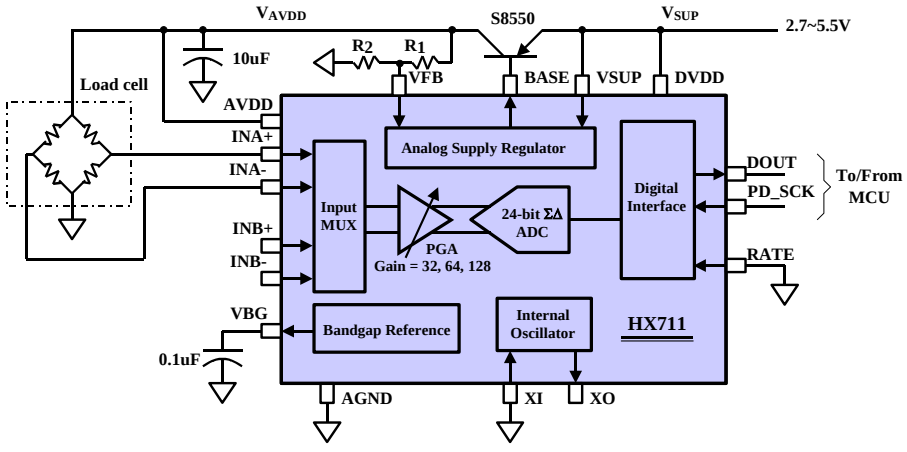
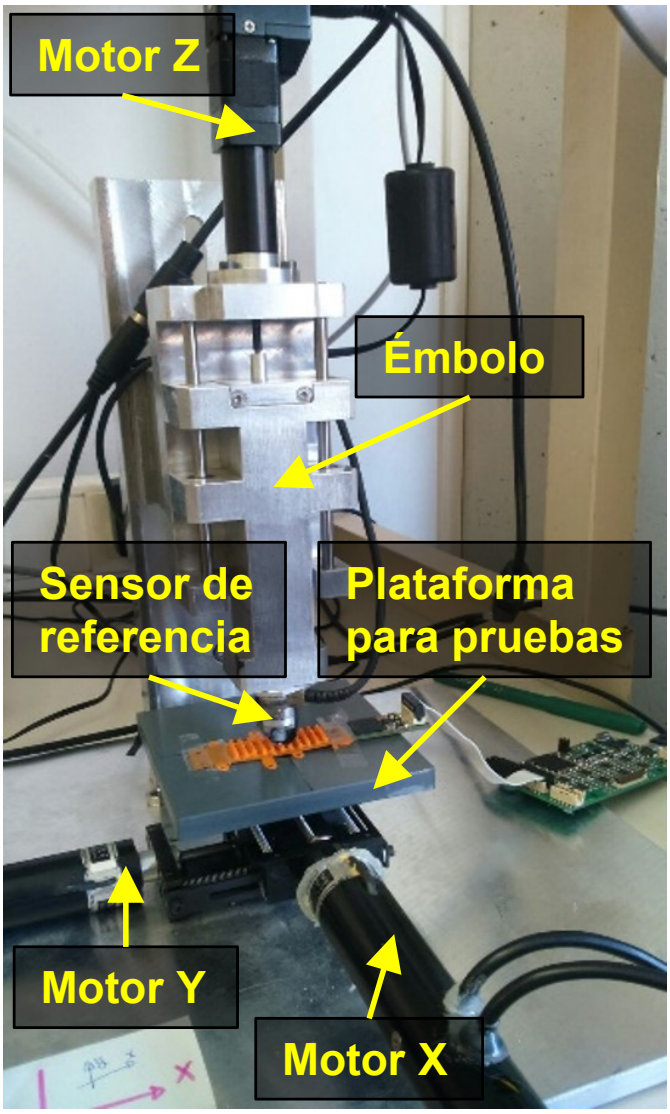
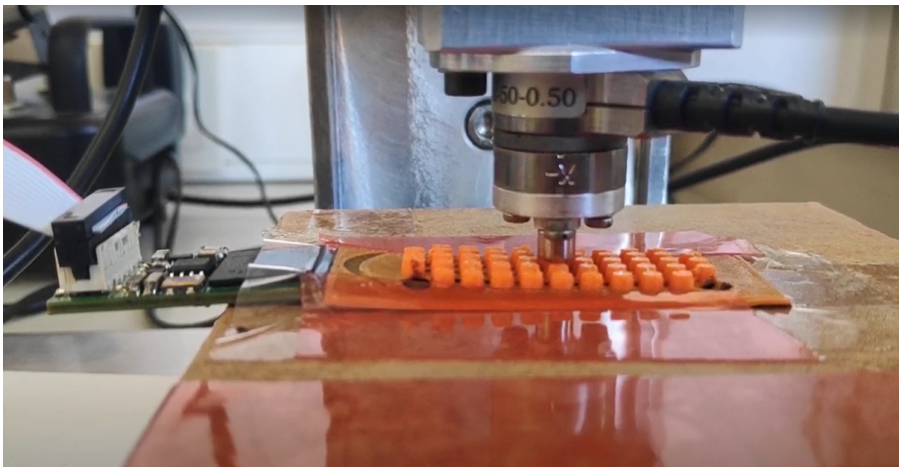


Figura A.2. Esquema de conexión entre la célula de carga y el conversor HX711 [68] (capítulo 2).

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura A.3. (a) Fotografía del calibrador triaxial utilizado. Se pueden ver los motores (T-NA08A50 Zaber, Vancouver, BC, Canadá) para los tres ejes de movimiento. El motor Z mueve el émbolo para que el sensor de referencia presione el táctel sobre la plataforma de experimentación. (b) Aplicación de fuerza sobre un táctel del sensor de dedo (capítulo 2).

UNIVERSIDAD DE MÁLAGA

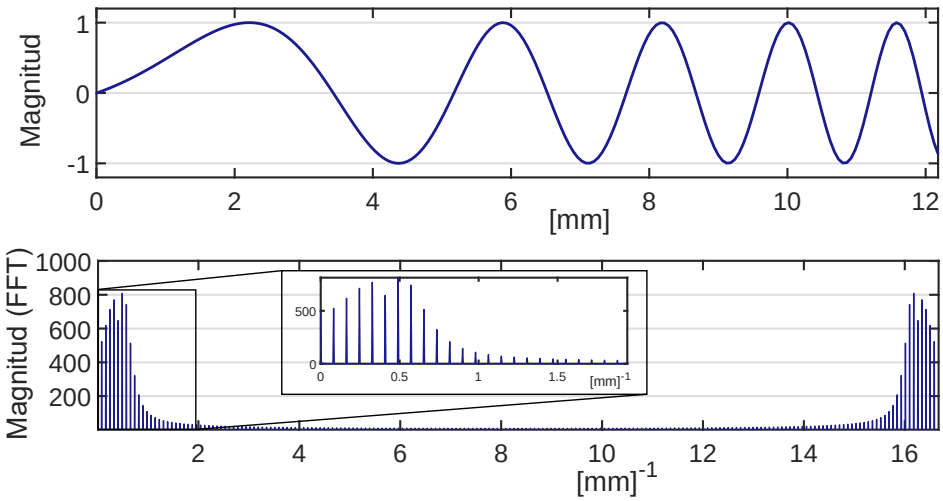
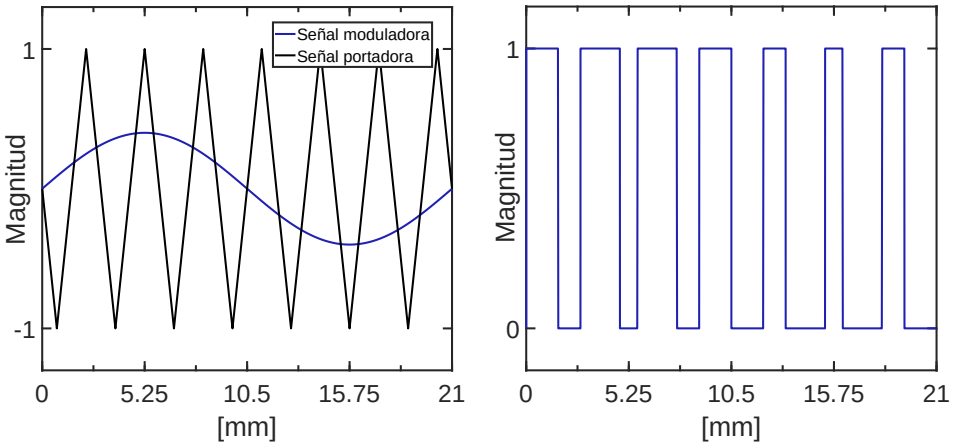
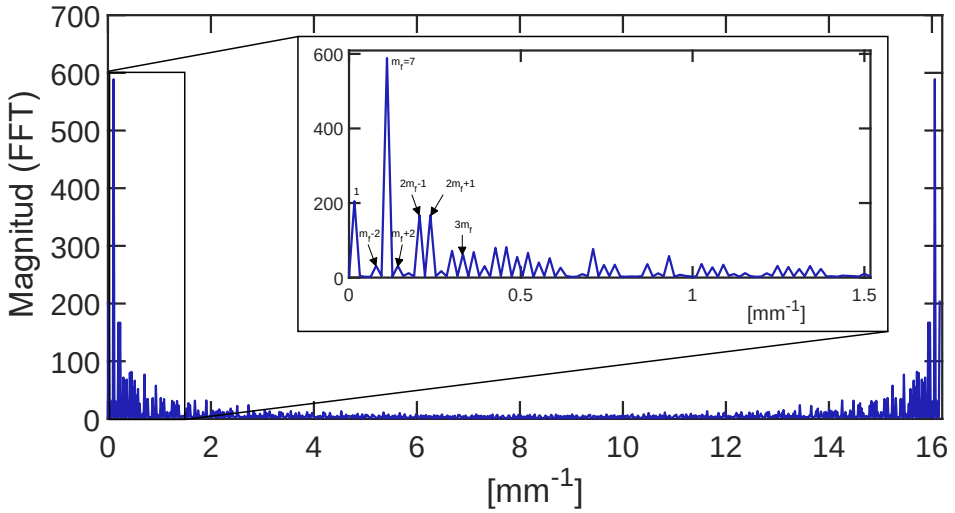


Figura A.4. Representación de un período espacial para la textura #TEX-1 de la tabla 3.5 (parte superior de la figura) y espectro en frecuencia con un *zoom* para los armónicos principales (parte inferior de la figura), (capítulo 3).

UNIVERSIDAD DE MÁLAGA



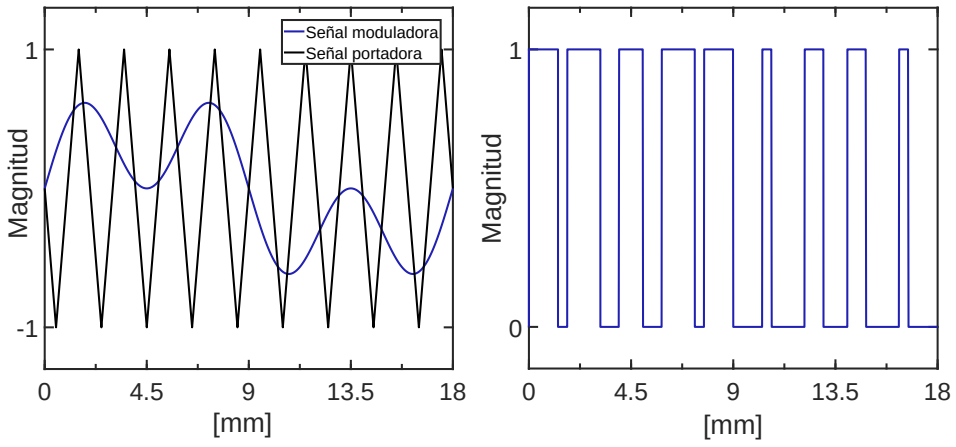
(a)



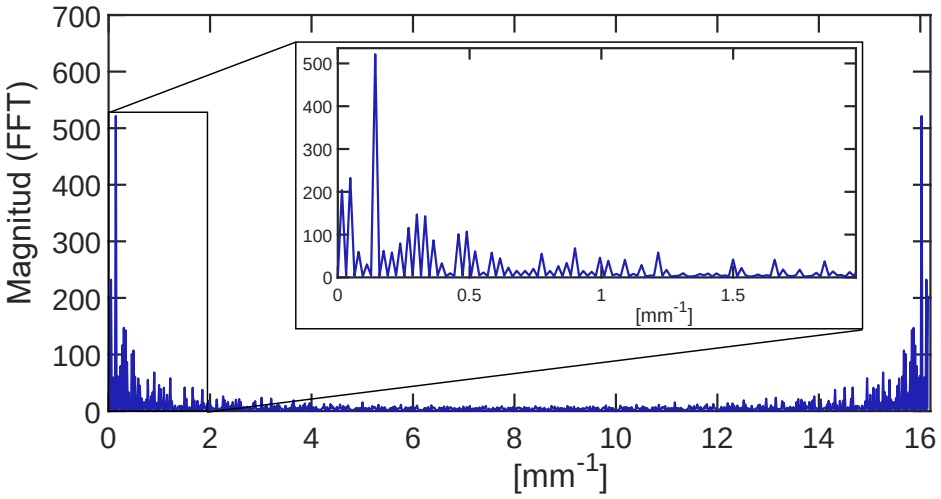
(b)

Figura A.5. (a) Modulación lineal en frecuencia para la textura #TEX-5 de la tabla 3.5. En la figura . Se representa un período espacial de la textura. (b) Espectro en frecuencia para dicha modulación con un *zoom* para los armónicos principales, que se muestran como múltiplos de $m_f = 7$ (capítulo 3).

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura A.6. (a) Modulación lineal en frecuencia para la textura #TEX-6 de la tabla 3.5. Se representa un período espacial de la textura. (b) Espectro en frecuencia para dicha modulación con un *zoom* para los armónicos principales (capítulo 3).

UNIVERSIDAD DE MÁLAGA

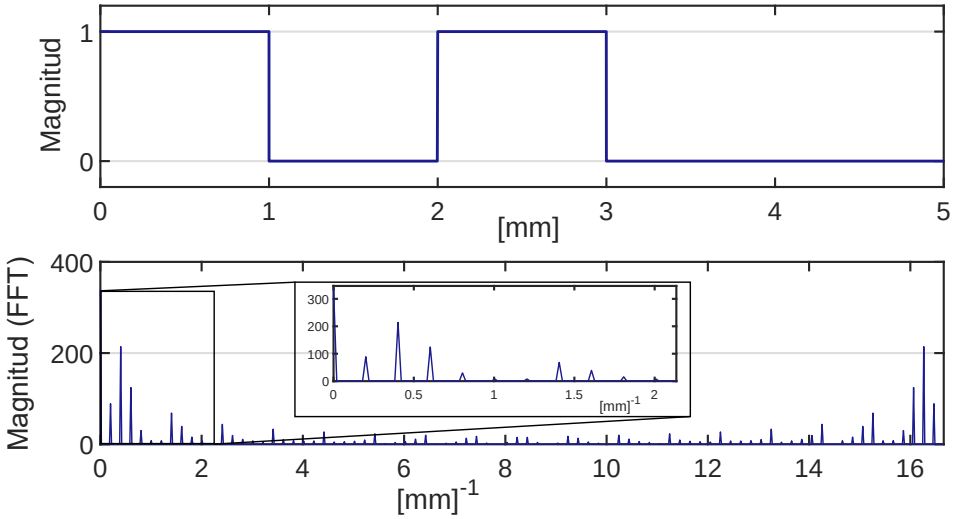


Figura A.7. Representación de un período espacial para la textura #TEX-8 de la tabla 3.5 (parte superior de la figura) y espectro en frecuencia con un *zoom* para los armónicos principales (parte inferior de la figura), (capítulo 3).

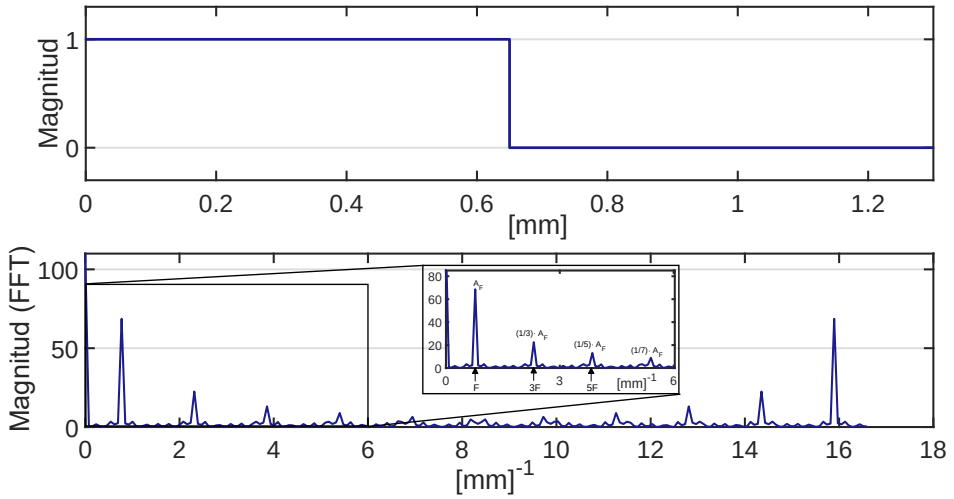


Figura A.8. Representación de un período espacial para la textura #TEX-9 de la tabla 3.5 (parte superior de la figura) y espectro en frecuencia con un *zoom* para los armónicos principales, típicos de una onda cuadrada (parte inferior de la figura). En la imagen, F indica armónico fundamental y A_F amplitud del armónico fundamental (capítulo 3).

UNIVERSIDAD DE MÁLAGA

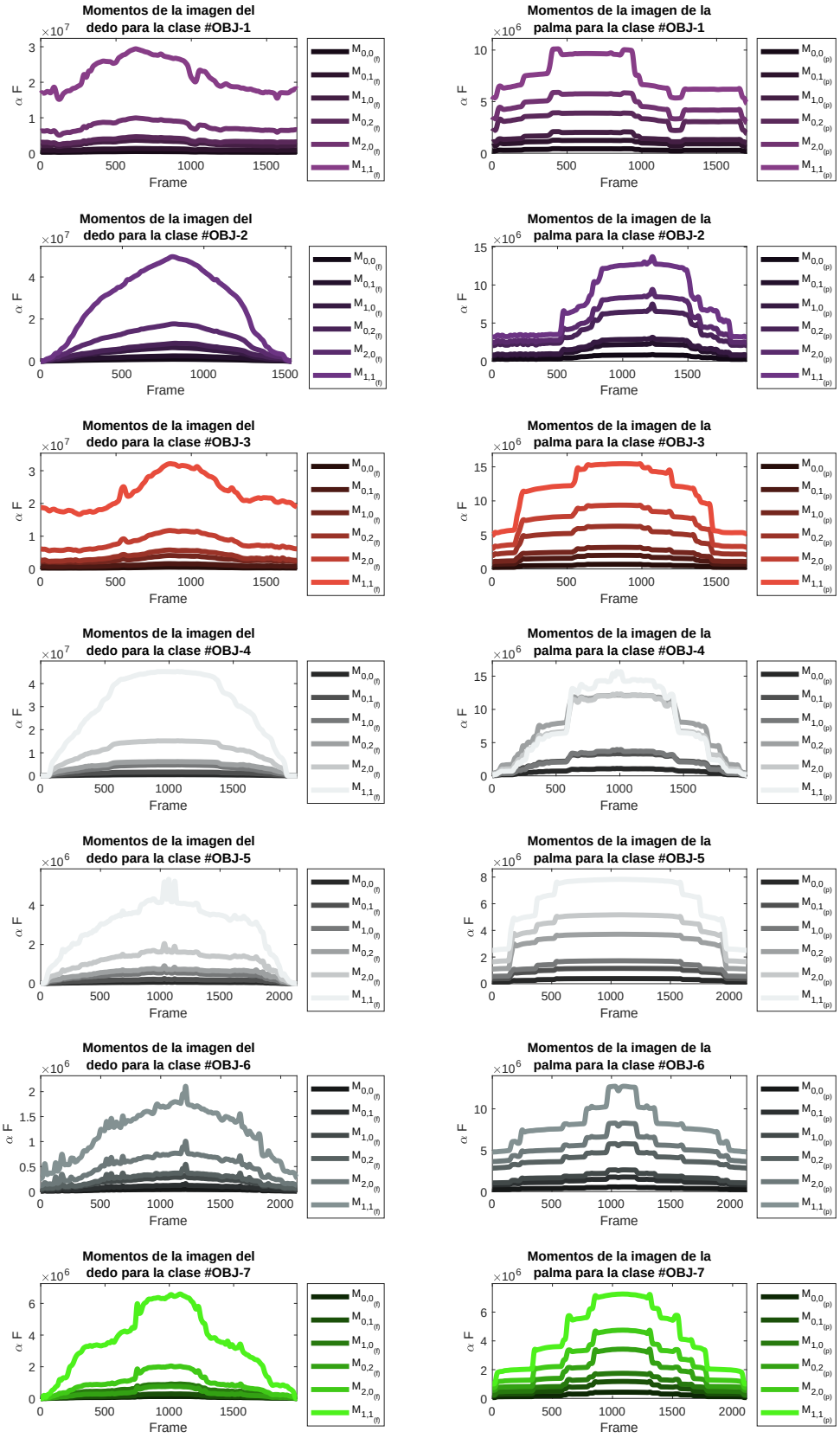


Figura A.9. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-1, #OBJ-2, #OBJ-3, #OBJ-4, #OBJ-5, #OBJ-6 y #OBJ-7 (capítulo 4).

UNIVERSIDAD DE MÁLAGA

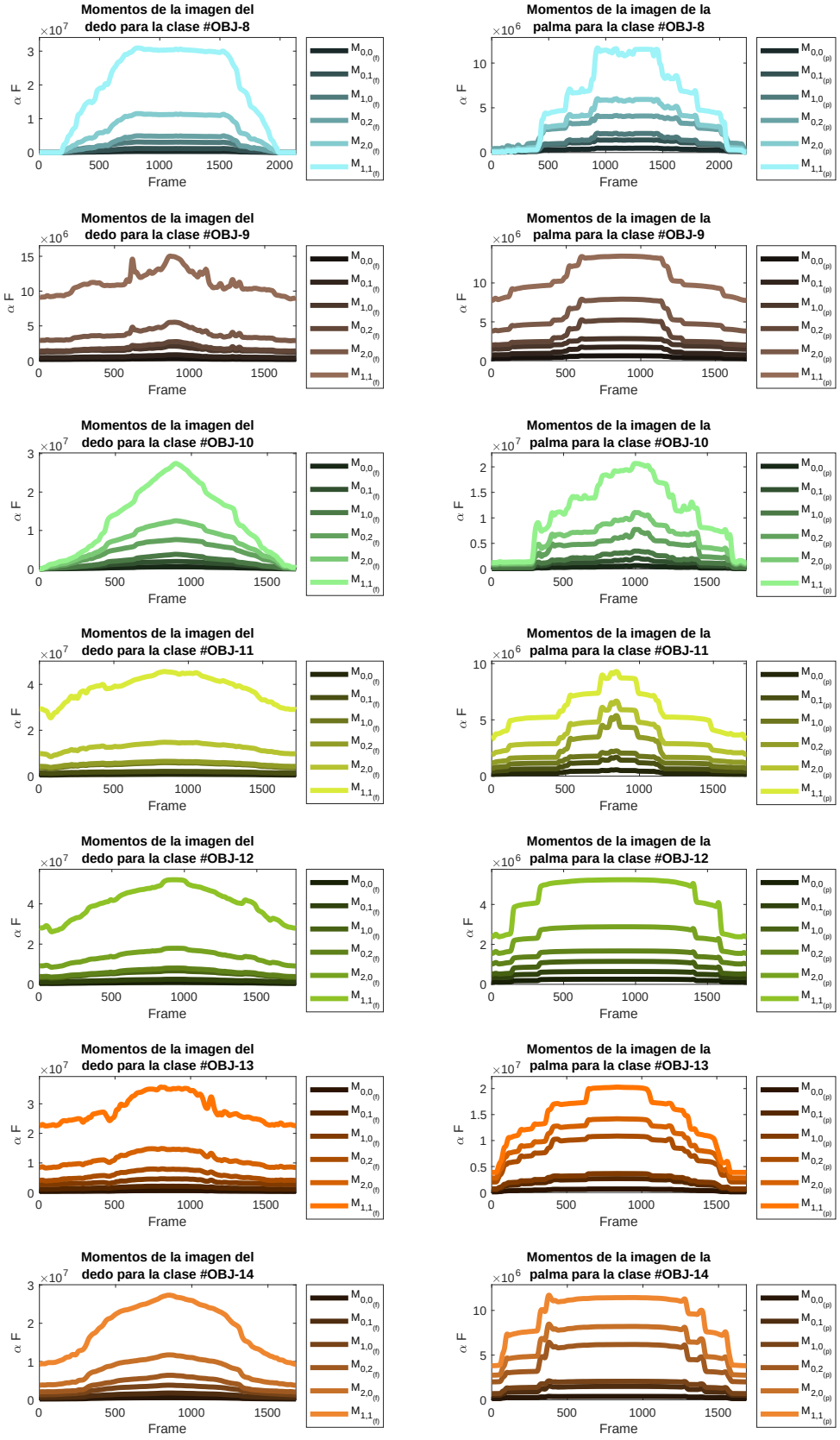


Figura A.10. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-8, #OBJ-9, #OBJ-10, #OBJ-11, #OBJ-12, #OBJ-13 y #OBJ-14 (capítulo 4).

UNIVERSIDAD DE MÁLAGA

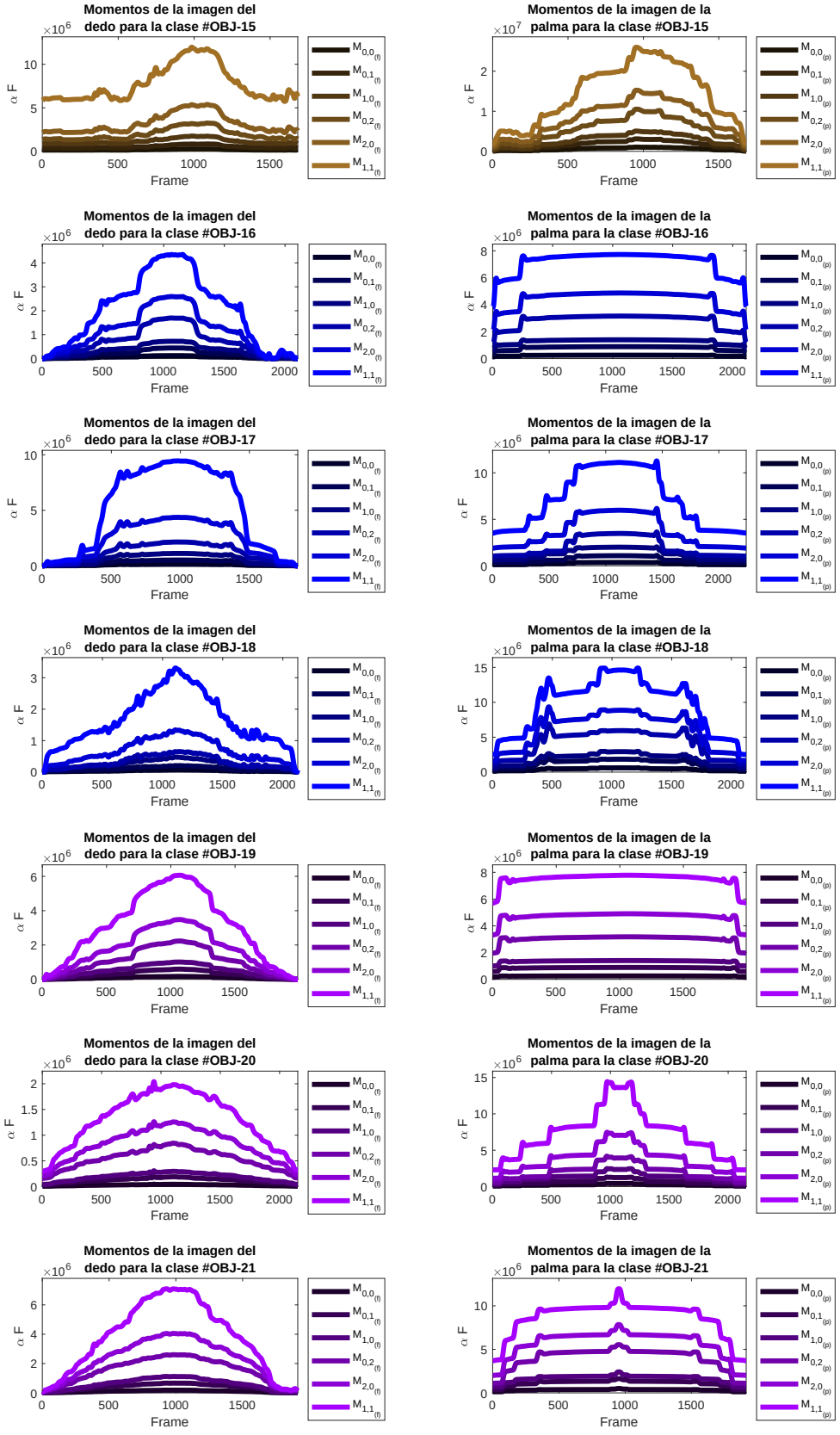


Figura A.11. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-15, #OBJ-16, #OBJ-17, #OBJ-18, #OBJ-19, #OBJ-20 y #OBJ-21 (capítulo 4).

UNIVERSIDAD DE MÁLAGA

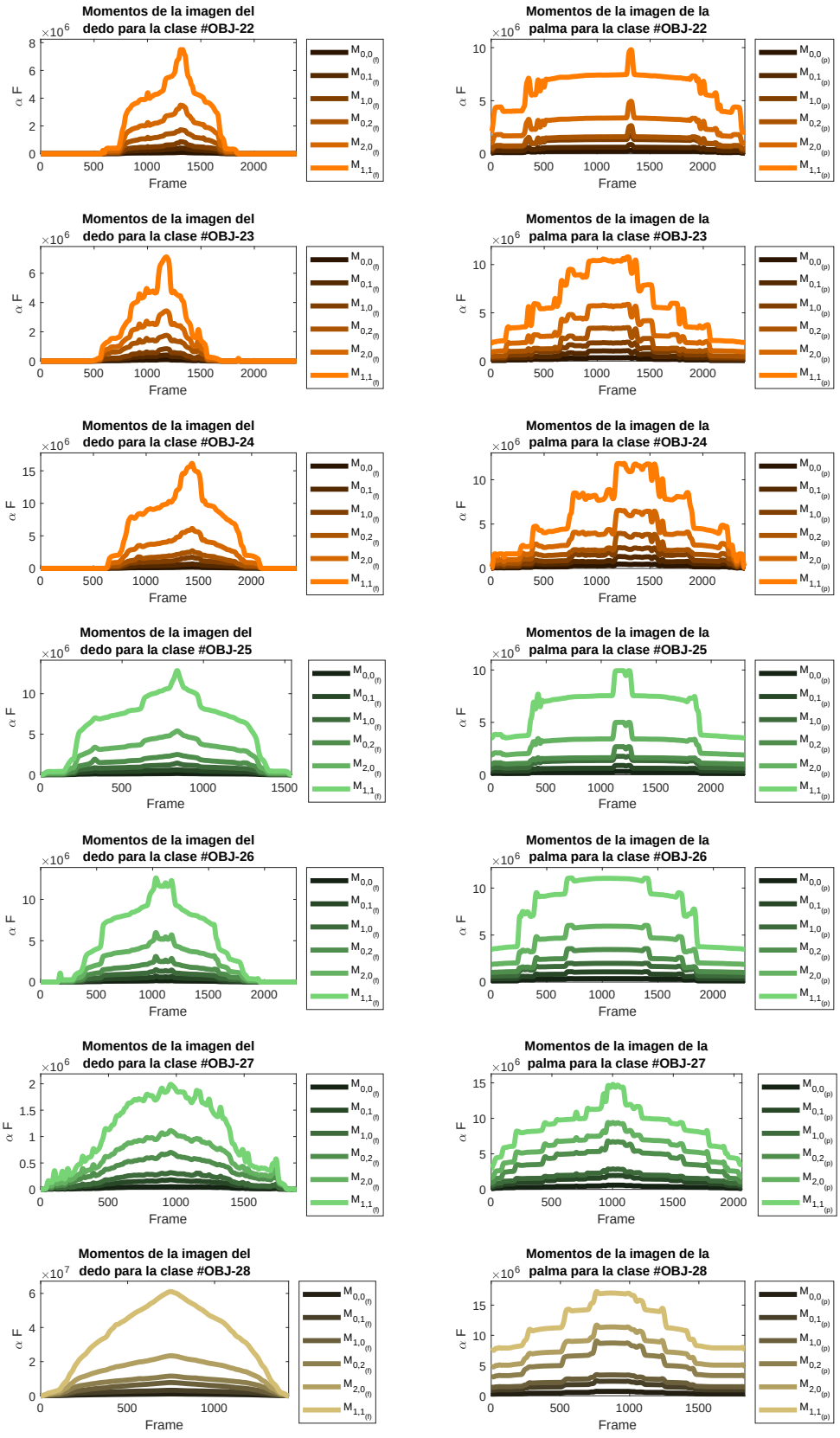


Figura A.12. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-22, #OBJ-23, #OBJ-24, #OBJ-25, #OBJ-26, #OBJ-27 y #OBJ-28 (capítulo 4).

UNIVERSIDAD DE MÁLAGA

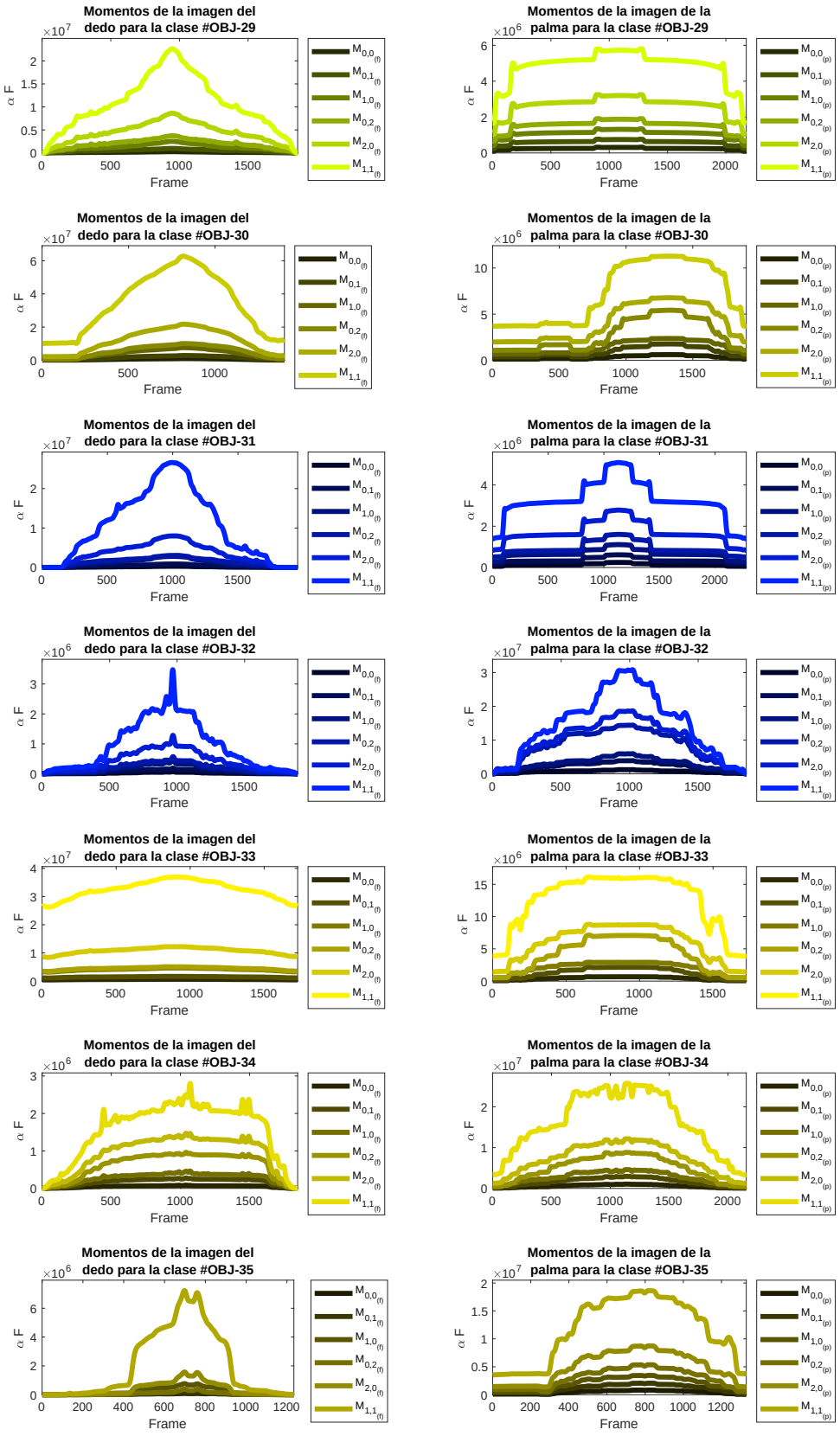


Figura A.13. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-29, #OBJ-30, #OBJ-31, #OBJ-32, #OBJ-33, #OBJ-34 y #OBJ-35 (capítulo 4).

UNIVERSIDAD DE MÁLAGA

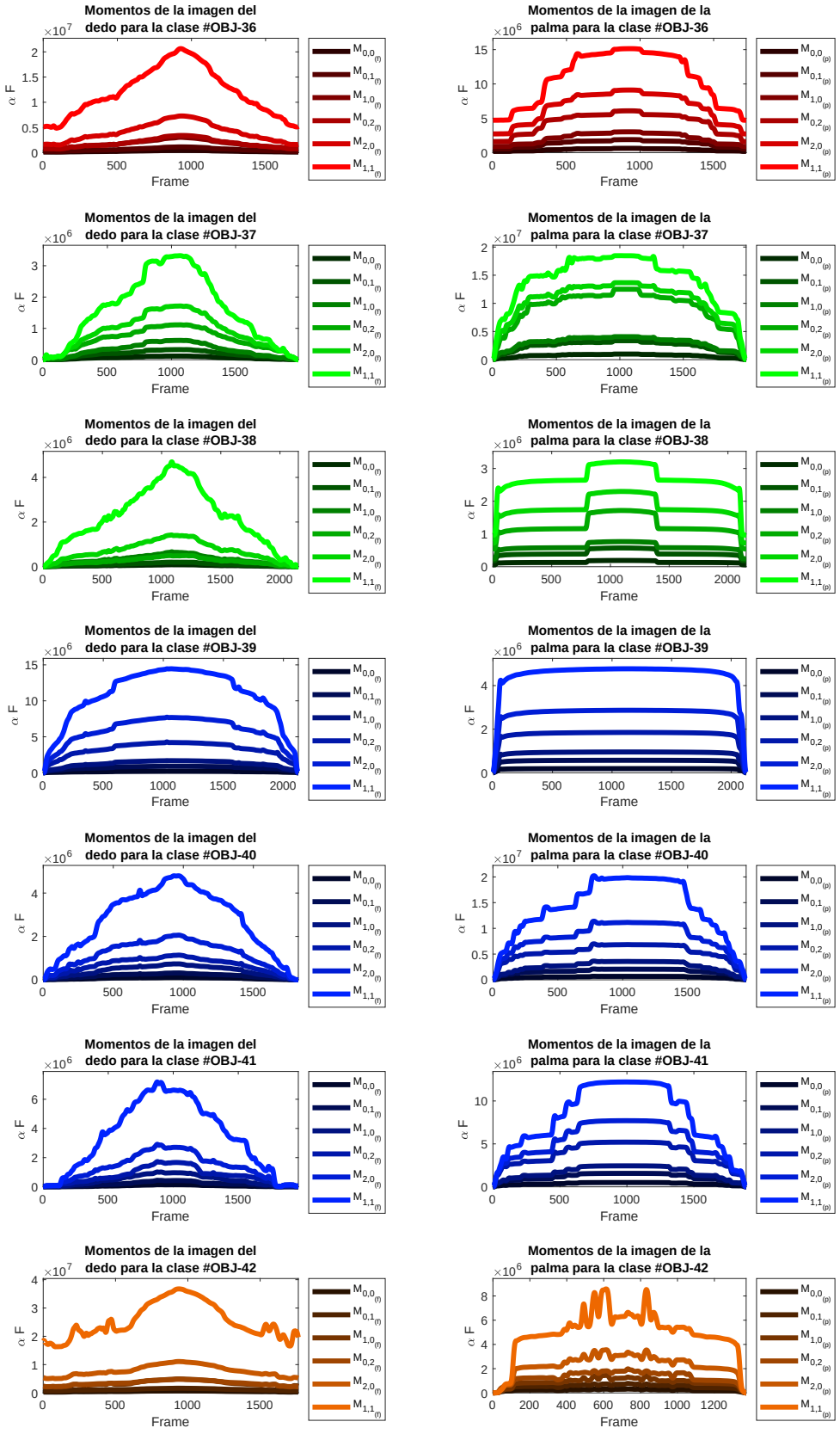


Figura A.14. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-36, #OBJ-37, #OBJ-38, #OBJ-39, #OBJ-40, #OBJ-41 y #OBJ-42 (capítulo 4).

UNIVERSIDAD DE MÁLAGA

Apéndice B Tablas complementarias

En este apéndice se presentan tablas adicionales que complementan los capítulos de la tesis.

UNIVERSIDAD DE MÁLAGA

Tabla B.1. Principales características para las partes PS y PL del dispositivo SoC Zynq™ XC7Z020-CLG484-1 [75] (capítulo 2).

Sistema de procesamiento (PS)	
Procesador	Doble núcleo ARM® Cortex-A9™
Extensiones del procesador	NEON™ & Simple/Doble precisión en punto flotante (por núcleo)
Máxima frecuencia	667 MHz
Caché L1	32 kB (instrucciones), 32 kB (datos) por núcleo
Caché L2	512 kB
Memoria On-Chip	256 kB (SRAM)
Soporte de memoria externa	DDR3, DDR2, LPDDR2
Soporte de memoria estática	2x QSPI-SPI, NAND, NOR
Canales de DMA	8 (4 dedicados a la parte PL)
Principales periféricos	2x USB 2.0 (OTG) w/DMA, 2x Tri-mode Gigabit Ethernet w/DMA, 2x SD/SDIO w/DMA, 2x UART (2), 2x CAN2.0B, 2x I2C, 2x SPI, 4x 32b GPIO
Seguridad	AES y SHA 256b para inicio o *boot* seguro
Puertos para interfaz entre PS y PL	2x AXI 32b Master, 2x AXI 32b Slave, 4x AXI 64b/32b Memory AXI 64b ACP, 16 Interrupts
Lógica programable (PL)	
Logic Cells	85k
LUTs	53200
FFs	106400
SLICEs	13300
BRAMs (bloques de 36 Kb)	140, (4.9) Mb
DSPs (18x25 MACs)	220
Peak DSP Performance (Symmetric FIR)	158 GMACS
Agile Mixed Signal (XADC)	2x 12bit 1Msps A/D Converter
Multi Standards 3.3V I/O *buffers*	200

UNIVERSIDAD DE MÁLAGA

Tabla B.2. Valores correspondientes al período espacial mínimo sin tomar en cuenta el filtro antialiasing λ_{min}, expresión (3.9) del capítulo 3, y considerándolo $\lambda_{minFilter}$, expresión (3.23) del capítulo 3. Estos valores se calculan para diferentes velocidades v (capítulo 3).

Velocidad (mm/s)	λ_{min} (mm)	$\lambda_{minFilter}$ (mm)
2	0.008	0.012
5	0.02	0.03
10	0.04	0.06
20	0.08	0.12
30	0.12	0.19
50	0.2	0.3
75	0.3	0.5
100	0.4	0.6
150	0.6	0.9
200	0.8	1.2

Tabla B.3. Período espacial máximo λ_{max} en mm, que se obtiene para diferentes valores de M y de la velocidad v, de acuerdo a (3.8), sección 3.2, (capítulo 3).

	M							
Velocidad (mm/s)	8	16	32	64	128	256	512	1024
2	0.03	0.07	0.13	0.26	0.5	1.1	2.1	4.2
5	0.08	0.16	0.3	0.66	1.3	2.6	5.3	10.6
10	0.16	0.3	0.7	1.3	2.6	5.3	10.6	21.1
20	0.3	0.7	1.3	2.6	5.3	10.6	21.1	42.2
30	0.5	1	2	4	7.9	15.8	31.7	63.3
50	0.8	1.7	3.3	6.6	13.2	26.4	52.8	105.6
75	1.2	2.5	5	9.9	19.8	39.6	79.2	158.4
100	1.7	3.3	6.6	13.2	26.4	52.8	105.6	211.1
150	2.5	5	9.9	19.8	39.6	79.2	158.4	316.7
200	3.3	6.6	13.2	26.4	52.8	105.6	211.1	422.3

UNIVERSIDAD DE MÁLAGA

UNIVERSIDAD
DE MÁLAGA

Apéndice C Consideraciones adicionales para el motor NEMA 17 y *driver* Pololu A4988

En este apéndice se muestran algunas consideraciones adicionales para los motores paso a paso NEMA 17 utilizados para el control de trayectorias de experimentos (sección C.1) y el *driver* Pololu Shield A4988 (sección C.2). Estos elementos se introducen en el capítulo 2.

C.1. Resolución de los motores NEMA 17 y frecuencia de resonancia

En la imagen de la figura C.1 se presenta una imagen adicional de las dimensiones mecánicas del motor NEMA 17 utilizado.

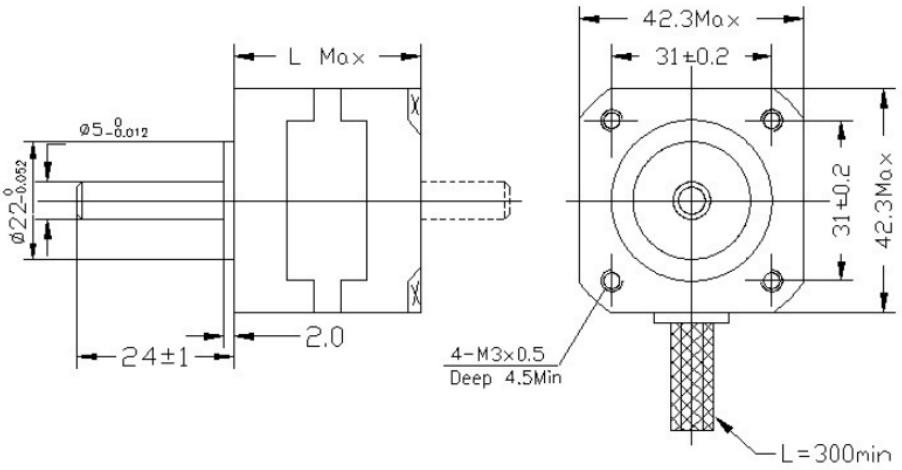


Figura C.1. Dimensiones mecánicas del motor NEMA 17 utilizado [66]. La longitud nominal máxima (L Max) del motor es 40 mm. Los datos de longitud están en mm (capítulo 2).

UNIVERSIDAD DE MÁLAGA

Los motores NEMA 17 tienen una resolución de 200 pasos por revolución. Dado un desplazamiento d a lo largo de un eje en mm, el número de pasos N_{steps} correspondiente que tiene que llevar a cabo el motor paso a paso viene dado por la expresión C.1.

$$N_{steps} = \frac{200 \text{ (pasos/revolución)} \cdot f_{\mu step}}{2\pi R \text{ (mm/revolución)}} \cdot d \tag{C.1}$$

donde $f_{\mu step}$ es un factor de *microstepping* (por ejemplo, para modo de $1/16$ pasos, $f_{\mu step} = 16$, ver siguiente apartado), y $R = 6.3$ mm es el radio estimado de la polea sobre la que se mueve la correa que hace desplazar a la pieza de sujeción de cada motor, ver detalle del motor en la impresora en la figura C.2. Los motores NEMA 17 pueden ejercer un torque total de 4 kg·cm, lo que se ha tenido en cuenta en la realización de los experimentos, para evitar así la pérdida de pasos en el desplazamiento de los motores.

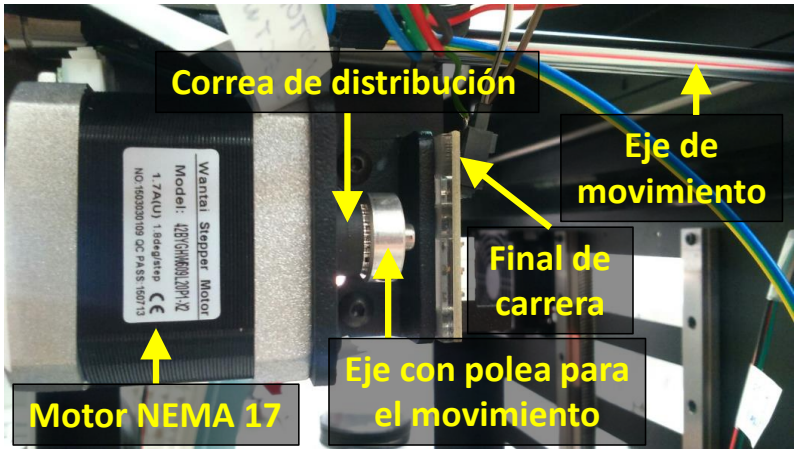


Figura C.2. Detalle del motor NEMA 17 colocado en la impresora 3D que se utilizó como banco de pruebas.

Finalmente, hay que destacar que todos los motores paso a paso tienen una frecuencia natural o de resonancia, que puede hacer que el motor pierda pasos o introduzca vibraciones espúreas y ruidos durante la experimentación. De acuerdo a [127], la frecuencia de resonancia de un motor se puede modelar con la expresión (C.2).

$$\omega \, \alpha \, \sqrt{\frac{K}{J}} \tag{C.2}$$

donde K es el torque del motor, y J es la inercia. Para el motor NEMA 17 utilizado en este proyecto, según los datos de la tabla 2.1 del capítulo 2, la frecuencia natural es proporcional a $\sqrt{\frac{0.4\text{Nm}}{5.4{\cdot}10^{-6}\text{kg cm}^2}} = 272$ rad/s (43.3 Hz). Cabe mencionar que factores como la inercia de la carga que soporte el motor también influyen en el espectro de frecuencias. Los ejes de cada motor de la impresora 3D mueven una correa de distribución (ver figura 2.25 del capítulo 2 y figura C.2), lo que a su vez hace mover la pieza de sujeción del

UNIVERSIDAD DE MÁLAGA

soporte del dedo. En este sentido, la velocidad configurada en las trayectorias también influye en la frecuencia de resonancia del motor [128].

El fenómeno de la resonancia se puede mitigar utilizando la técnica de *microstepping* para obtener pasos inferiores al nominal, reduciendo la velocidad de exploración, o variando los párametros K y J en la expresión (C.2). En el trabajo relativo a esta tesis, la velocidad de exploración de texturas se ha configurado en 30 mm/s y la de palpación de objetos en 10 mm/s. Los motores se configuran para trabajar con la técnica de *microstepping* a $1/16$ pasos. En cuanto al parámetro K, se puede enviar a los motores una corriente menor para desplazar la frecuencia natural, o bien modificar la inercia J del motor teniendo en cuenta que es la suma de la inercia del rotor (ver tabla 2.1 del capítulo 2) más la carga que soporte.

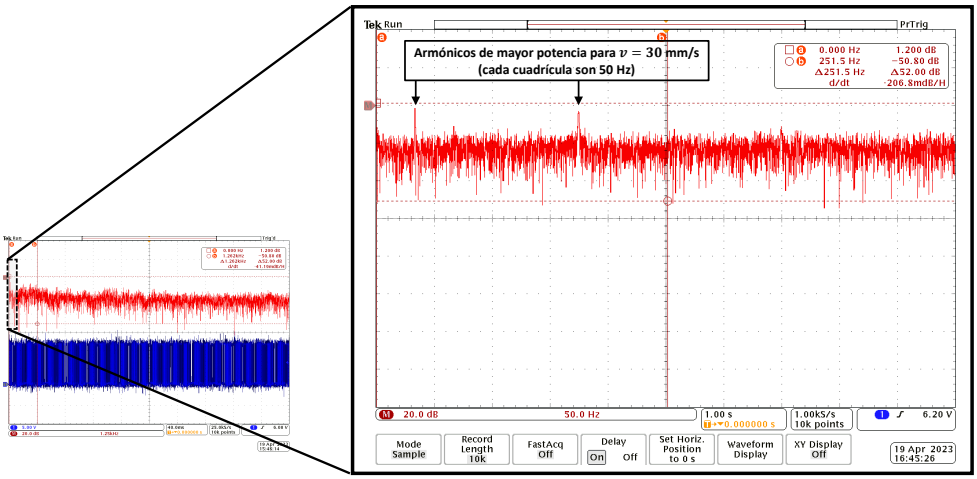
En las figuras C.3(a) y C.3(b), se presentan las capturas de mediciones con osciloscopio, que muestran el espectro en frecuencias de la tensión que llega a una de las dos bobinas del motor paso a paso, cuando se recorre una distancia de 220 mm a 30 mm/s y 200 mm/s, respectivamente. En ambos casos, se pueden observar armónicos en posiciones distintas, pero a mayor velocidad se aprecia un mayor ruido.

C.2. Electrónica interna del *driver* Pololu A4988 y técnica de *microstepping*

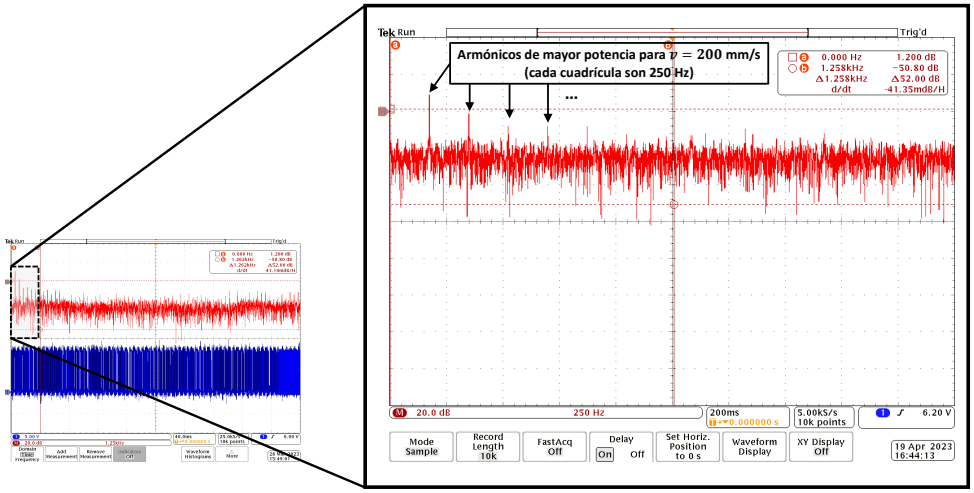
La electrónica de este *driver* es más compleja en comparación con otros controladores más simples como el L298N o el TB6612FNG. En la figura C.4 se puede ver una imagen donde se destacan en rojo el regulador de corriente y los pines de entrada para *microstepping*, que se comentan a continuación. Sin embargo, sólo se necesitan dos puertos digitales del microcontrolador, lo que deja más puertos libres para otras funciones. Además, el Pololu A4988 incorpora funciones especialmente diseñadas para el control de motores paso a paso.

El *microstepping* es una técnica que permite obtener pasos inferiores al paso nominal del motor que vamos a controlar. Esto se logra haciendo variar la corriente para cada bobina, de forma que se emula un valor analógico, aplicando a cada fase una función senoidal discretizada. En comparación con el modo *full-step*, en el que el controlador envía corriente sólo a una bobina en un momento dado, el *microstepping* reduce el ruido y la vibración del motor. Sin embargo, el diseño tanto en hardware como en software es más complejo. El *driver* Pololu A4988 utilizado en este proyecto permite configurar los modos de *microstepping* a través de los pines MS1, MS2 y MS3, como se muestra en

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura C.3. (a) Medidas de $10k$ puntos del espectro en frecuencia, para un desplazamiento a 30mm/s y *zoom* para una división por cuadrícula de 50 Hz ($1kS/s$), y (b) para un desplazamiento a 200mm/s con *zoom* para una división por cuadrícula de 250 Hz ($5kS/s$). Las medidas se realizaron con un osciloscopio MDO4104B-6 Mixed Domain Tektronix® de 6 GHz.

la figura C.4 del apéndice A de figuras. La tabla C.1 presenta los modos disponibles y la resolución de paso correspondiente para este *driver*. En la figura C.5 se puede ver el esquema para el conexionado de este *driver*.

Por otra parte, la inclusión del regulador de intensidad es necesaria, ya que los motores NEMA 17 requieren una intensidad nominal de 1.7 A y una resistencia por fase de 2 Ω, ver tabla 2.1 del capítulo 2. Esto significa que, según la ley de Ohm, es necesario aplicar 3.4 V a cada bobina. Sin embargo, para un correcto funcionamiento del motor, se debe aplicar una tensión superior. La fuente de alimentación utilizada es de 12 V nominales. Gracias al regulador de corriente, se puede garantizar que el valor promedio

UNIVERSIDAD DE MÁLAGA

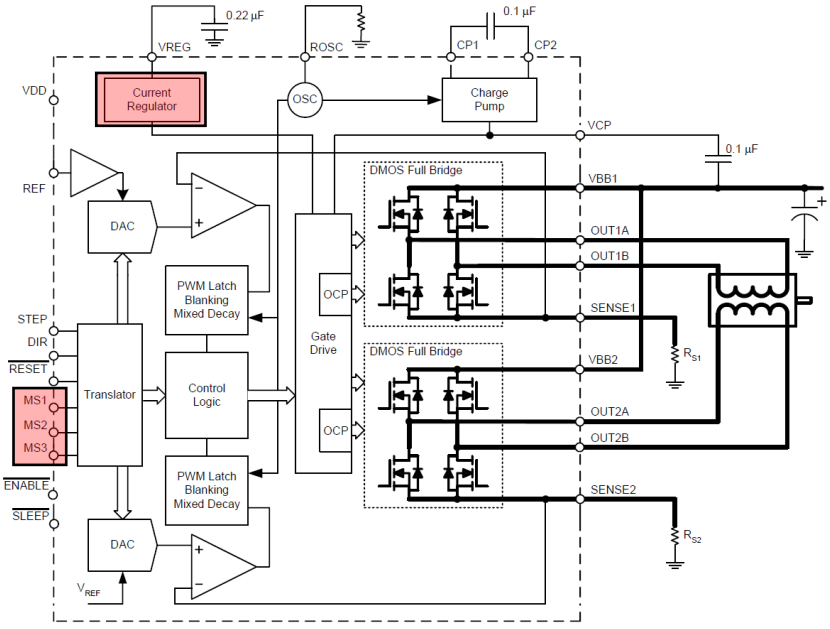


Figura C.4. Electrónica del *driver* Pololu A4988 de acuerdo a su datasheet [65]. En la figura, se pueden ver destacados un regulador de corriente y los pines de entrada MS1, MS2 y MS3 para la funcionalidad de *microsteeping* (capítulo 2).

Tabla C.1. Valores lógicos en los pines para los modos de configuración de *microsteeping*, para el *driver* Pololu A4988.

Resolución	Pin MS1	Pin MS2	Pin MS3
Full step	LOW	LOW	LOW
1/2 step	HIGH	LOW	LOW
1/4 step	LOW	HIGH	LOW
1/8 step	HIGH	HIGH	LOW
1/16 step	HIGH	HIGH	HIGH

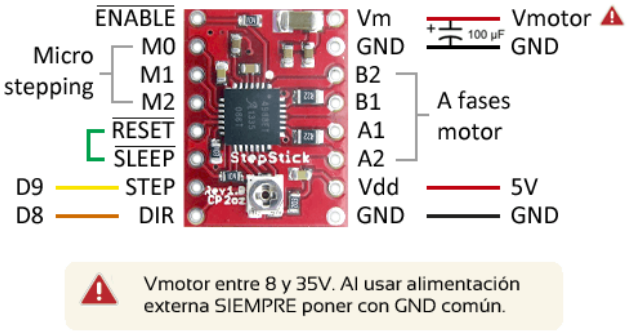


Figura C.5. Conexiones de los pines del *driver* [129].

de intensidad que atraviesa cada bobina es la intensidad del motor[1]. La corriente máxima

[1]El regulador de intensidad proporciona una señal modulada por ancho de pulso (*Pulse Width Modulation, PWM*). Durante cada período de la señal PWM, esta técnica denominada *chopping*, mantiene el

UNIVERSIDAD DE MÁLAGA

I_{max} que llega a cada motor se calcula con la fórmula C.3.

$$I_{max} = V_{ref}/(8 \cdot Rs) \tag{C.3}$$

donde V_{ref} es la tensión de referencia del *driver*, y Rs el de la resistencia externa de sensado de corriente, que varía con el fabricante, típicamente $Rs = 50\,\Omega$ para el *driver* Pololu A4988 [129].

UNIVERSIDAD DE MÁLAGA

motor activo durante un período de tiempo menor al tiempo que está desactivado [65]. Este valor se ajusta mediante el potenciómetro que incorpora el limitador

Apéndice D Discretización de la lámina piezoresistiva mediante la impresora láser

En este apéndice se explica el proceso de discretización de la lámina de piezoresistivo que se coloca junto al *cover* estructurado con el fin de obtener la huella del dedo artificial. La discretización de la lámina permite el aislamiento eléctrico de los electrodos de la matriz táctil, lo que contribuye a reducir la diafonía o *crosstalk* mencionado en el capítulo 2. Para llevar a cabo este proceso, se utiliza una máquina (o impresora) de grabado y corte por láser de 40 W de potencia máxima. En los siguientes apartados se detallan las diferentes partes del *setup* utilizado y el procedimiento seguido.

D.1. Descripción del *setup* utilizado

En la imagen de la figura D.1 se pueden observar las diferentes partes del *setup* utilizado para llevar a cabo la discretización de la lámina piezoresistiva. A continuación se hace hincapié en las partes más importantes.

En la figura D.1 cabe destacar la máquina de grabado y corte por láser. Se trata del modelo 9225A2-HS-001 de la marca XINKETAI ELECTRONIC CO, LTD. Las principales características de esta máquina se resumen en la tabla D.1.

El sistema de refrigeración del láser utiliza un circuito de agua que se encarga de absorber la mayor parte del calor emitido. Para garantizar el flujo adecuado de agua, se emplea una bomba de agua. En la figura D.2 se muestra un detalle de esta pieza. Se trata de una bomba con un caudal máximo de 2000 l/h y una capacidad máxima de elevación del agua de 2 m. Es importante asegurarse de que el nivel del agua en el recipiente que se muestra en la imagen cubra por completo la bomba de agua.

UNIVERSIDAD DE MÁLAGA

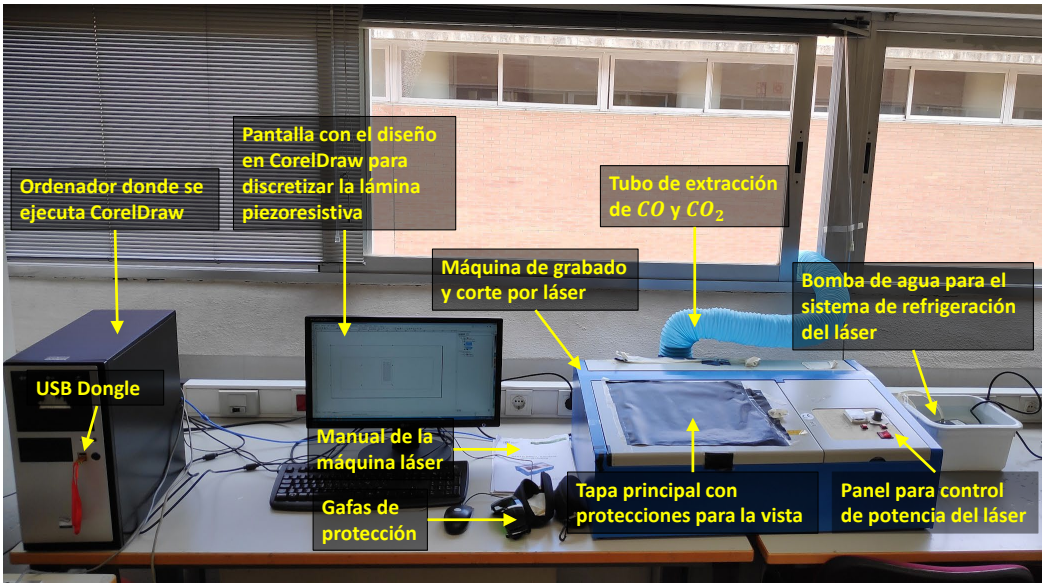


Figura D.1. Diferentes partes del *setup* utilizado para la discretización de la lámina piezoresistiva mediante la máquina de grabado y corte por láser.

Tabla D.1. Principales características de la máquina de grabado y corte por láser.

Propiedad	Valor
Potencia máxima del láser	40 W
Área de grabado	300 x 200 mm
Velocidad libre	$1 - 200$ mm/s
Velocidad de grabado	$1 - 300$ mm/s
Velocidad de corte	$1 - 10$ mm/s
Precisión de posicionamiento	0.01 mm
Temperatura de funcionamiento	$5 - 50^{o}C$gcm^2

En la figura D.2, se puede apreciar un tubo para la salida de agua, que se encuentra en la parte superior de la bomba de agua. Este tubo tiene la función de hacer circular el agua bombeada alrededor del láser, con el objetivo de disipar el calor que éste genera. Específicamente, se trata de un láser de CO_2 de 40 W, ubicado dentro de la carcasa trasera de la máquina de corte por láser. En la figura D.3, se muestra una fotografía. Es crucial verificar la correcta circulación del agua la primera vez que se pone en funcionamiento la máquina, ya que esto asegura la refrigeración adecuada del láser.

Durante el funcionamiento del láser, se emiten moléculas de CO_2. Al atacar con el láser superficies de diferentes materiales, pueden generarse monóxido de carbono (CO) u otros gases dañinos para la salud en la combustión. Por lo tanto, es crucial verificar que el ventilador y el tubo de extracción de estos gases funcionen correctamente. Además, se recomienda encarecidamente el uso de una mascarilla con filtro durante el tiempo de uso.

UNIVERSIDAD DE MÁLAGA



Figura D.2. Detalle de la bomba de agua para el sistema de refrigeración del láser. En la fotografía, la bomba se encontraba en funcionamiento.



Figura D.3. Fotografía del láser de CO_2 de 40 W que utiliza la máquina de corte por láser, en el interior de la carcasa trasera de la misma.

En la imagen de la figura D.4, se muestra la entrada del tubo de extracción conectado al ventilador, ubicado en la parte trasera de la máquina.

En la figura D.5, se presenta el panel de control que permite regular manualmente la potencia del láser. Este panel consta de varios elementos, incluyendo un indicador de intensidad de corriente en mA, dos pulsadores para controlar el láser, un regulador de potencia y un interruptor de encendido y apagado para la máquina. En todos los procedimientos de grabado y corte realizados en este trabajo de tesis, se utilizó una potencia

UNIVERSIDAD DE MÁLAGA



Figura D.4. Ventilador y tubo de extracción.

de 10 W y una velocidad de grabado y corte de 30 mm/s. Durante el funcionamiento de la máquina, el pulsador “Laser Switch” debe estar activado.

En la figura D.6, se presenta la máquina láser con la tapa principal levantada, mostrando el área de impresión. En esta imagen, se pueden apreciar los ejes de movimiento, así como los espejos utilizados para reflejar el haz del láser hasta alcanzar el material de grabado. Antes de utilizar el láser, es necesario realizar la calibración de los espejos. Esto se logra ajustando los tornillos ubicados junto al espejo, lo que permite modificar el ángulo de incidencia del haz del láser y asegurar que llegue correctamente al área de trabajo. En cuanto al pulsador ”Test Switch”mostrado en la figura D.5, su función es activar el láser durante el tiempo que se mantiene presionado. Esta funcionalidad puede ayudar a llevar adecuadamente a cabo el proceso de calibración.

UNIVERSIDAD DE MÁLAGA

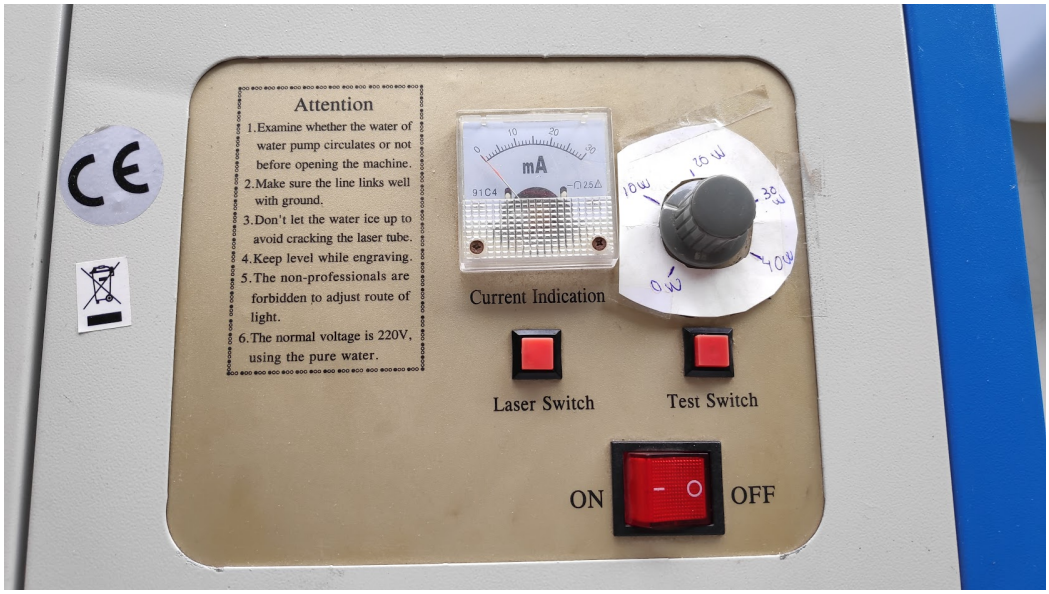


Figura D.5. Panel de control manual de la impresora láser.

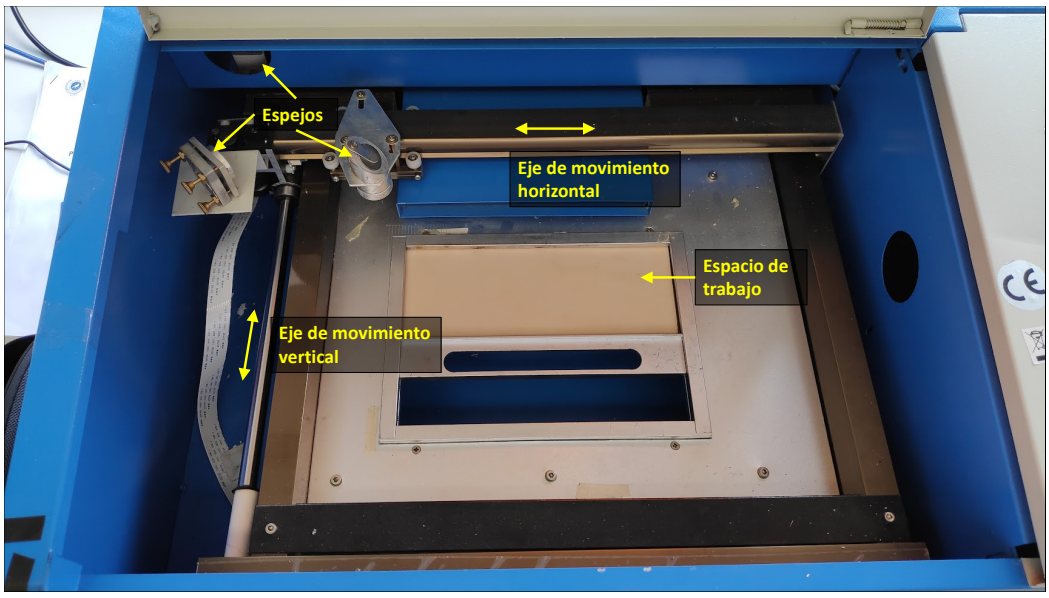


Figura D.6. Espacio de trabajo de la máquina de corte por láser.

El software utilizado para el grabado y corte por láser es CorelDRAW®. Se trata de un programa de diseño gráfico y vectorial ampliamente utilizado para crear ilustraciones, diseños de impresión y otros elementos visuales. A través de este software, es posible editar las formas geométricas necesarias para el grabado por láser de la lámina piezoresistiva. En la figura D.7, se muestra el área de diseño y edición de este software. En la parte derecha de la figura, se encuentran varios elementos agrupados bajo la denominación "Administrador de objetos". Esta sección contiene los elementos geométricos del diseño para el grabado y corte con el láser. Además, en la parte superior de la interfaz, se encuentran una serie de iconos que permiten la configuración de los parámetros de la

UNIVERSIDAD DE MÁLAGA

impresora.

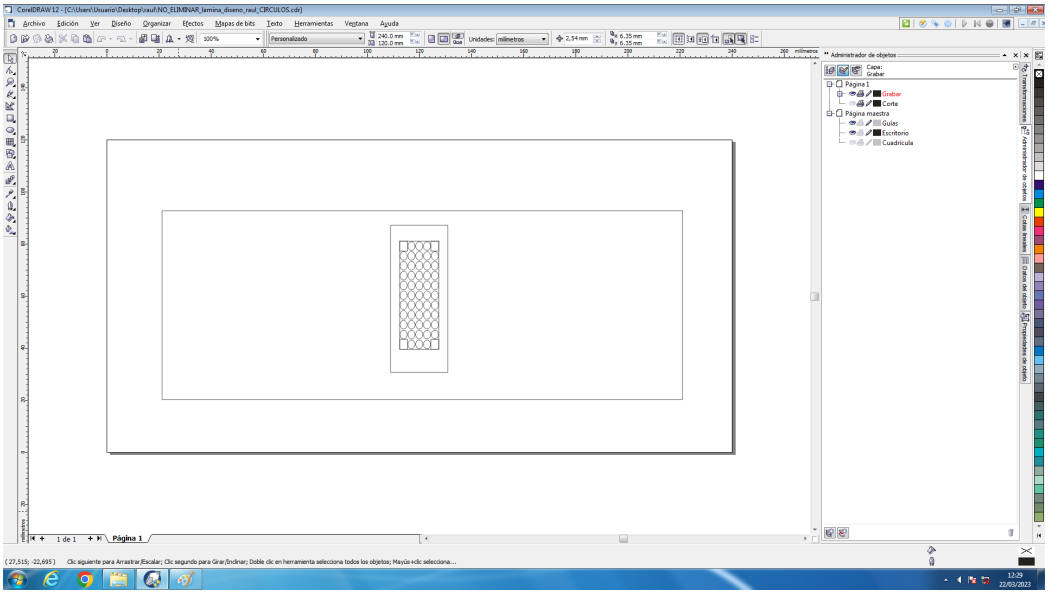


Figura D.7. Área de trabajo y edición para CorelDRAW®. En este caso se muestra el diseño que se utilizará para grabado sobre la lámina piezoresistiva.

Para garantizar el correcto funcionamiento de la máquina, es imprescindible utilizar la llave USB o *dongle* mostrada en el *setup* de la figura D.1. Esta llave USB tiene la función de autenticar la licencia y proteger el software contra copias ilegales o no autorizadas. En la figura D.8, se muestra un detalle del dispositivo. Es esencial tener esta llave USB conectada durante el uso de la máquina para asegurar la operación adecuada del software y cumplir con los términos de licencia.



Figura D.8. Dispositivo USB para validar la licencia de uso de la máquina de corte por láser.

Es fundamental enfatizar nuevamente la importancia de la seguridad del usuario. En el *setup* mostrado en la figura D.1, se puede observar la tapa principal de la máquina láser, la cual cuenta con protecciones para salvaguardar la vista del operador. Estas protecciones complementan la cubierta de plástico presente en la tapa principal. Dicha cubierta constituye un elemento de seguridad diseñado para filtrar y bloquear la radiación láser generada durante el proceso de grabado y corte.

UNIVERSIDAD DE MÁLAGA

La cubierta está fabricada de policarbonato, que se trata de un material que absorbe y dispersa la radiación láser. De esta manera, se protege al operador y se evita que la radiación se propague al entorno circundante. Esta cubierta de plástico actúa como una barrera transparente y segura, permitiendo la visibilidad del proceso de corte mientras protege los ojos y la piel de la exposición directa a la radiación láser. La figura D.9 muestra dicha cubierta en la tapa principal de la máquina láser.

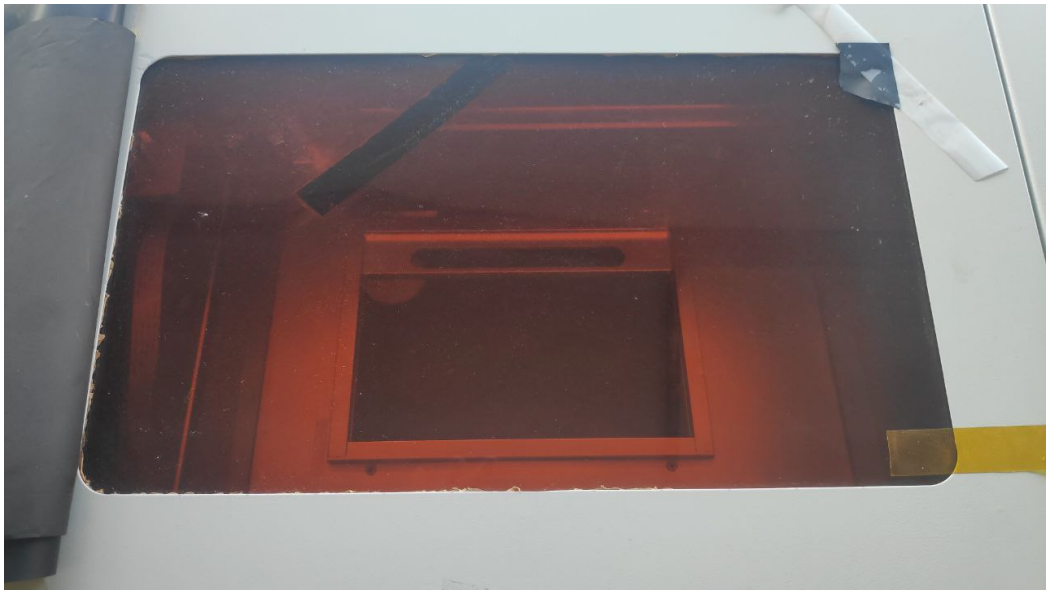


Figura D.9. Cubierta protectora, para dispersar la radiación emitida por el láser.

Además de la cubierta y las protecciones oscuras instaladas, es necesario utilizar gafas de protección antes de operar la máquina láser. Estas gafas se pueden ver también en la figura D.1. Las gafas están específicamente diseñadas para resguardar los ojos de la radiación láser emitida durante el funcionamiento de la máquina láser. Las longitudes de onda de esta radiación pueden ser perjudiciales y causar lesiones graves, incluyendo daño ocular permanente. Por tanto, es imprescindible usar las gafas de protección como una medida adicional para garantizar la seguridad ocular.

D.2. Procedimiento de discretización por láser de la lámina piezoresistiva

Para llevar a cabo la discretización de la lámina piezoresistiva se utiliza el *setup* de la figura D.1, siguiendo una serie de pasos, que se describen a continuación:

1. Calibrado correcto de la impresora láser. Para ello, se puede modificar el ángulo de incidencia de los espejos (ver figura D.10) y testear en dos puntos diagonalmente

UNIVERSIDAD DE MÁLAGA

opuestos del área de impresión con el pulsador de “Test Switch”. Así, se asegura que el láser actuará en el espacio destinado para ello.

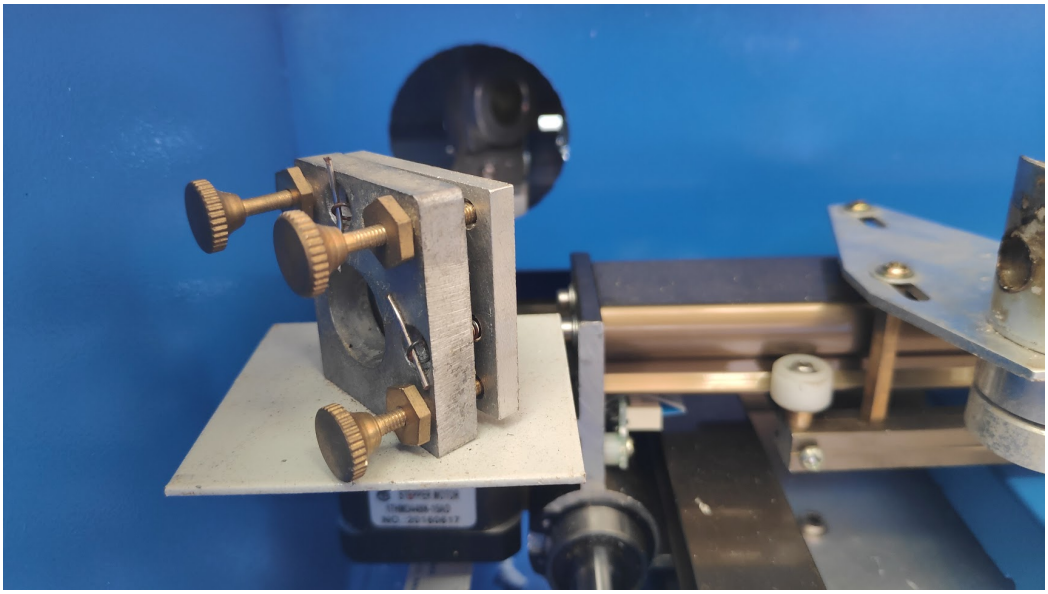


Figura D.10. Detalle del espejo con tornillos que permiten modificar el ángulo de incidencia del láser.

2. Para realizar el grabado de forma precisa en la lámina piezoresistiva, se pueden imprimir líneas de referencia (que forman rectángulos) sobre una pieza de madera, como se muestra en la figura D.11. Estas líneas de referencia sirven como guía para colocar la huella correspondiente al conjunto del *cover* y la lámina piezoresistiva. Las dimensiones de los rectángulos de referencia se corresponden con las del diseño del *cover* estructurado.



Figura D.11. Referencias para el grabado de la lámina piezoresistiva.

UNIVERSIDAD DE MÁLAGA

3. A continuación, se coloca cinta de doble cara sobre el *cover* (ver figura D.12 A), después se adhiere un trozo de material piezoresistivo (figura D.12 B) y, finalmente se coloca la huella en el área de impresión (figura D.12 C).

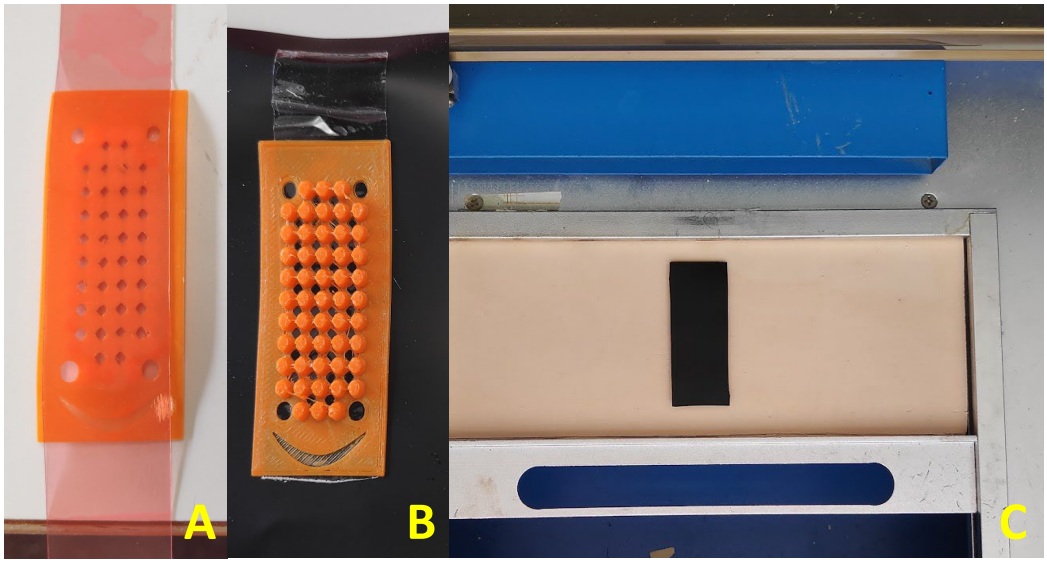


Figura D.12. Preparación de la huella.

4. El diseño para la discretización de la lámina piezoresistiva se muestra en la figura D.13. En este diseño, se emplean elipses con las mismas dimensiones que las de los electrodos de la matriz táctil de 5 filas y 11 columnas. Con el fin de favorecer la colocación manual de la huella, de manera precisa sobre la lámina de electrodos, se sacrifican los cuatro tácteles de las esquinas. Estas áreas se perforarán utilizando una broca de precisión adecuada con una herramienta Dremel®. Los agujeros resultantes se utilizarán como puntos de referencia visual o fiduciales durante el montaje del sensor de dedo. De esta manera, se facilita la alineación y colocación precisa de la huella sobre los electrodos.

5. En la figura D.13 también se incluye una referencia del tamaño de un píxel en la esquina superior izquierda de la plantilla de diseño. Esta referencia es necesaria para el movimiento adecuado de los cabezales de los espejos. Por otro lado, se señalan los iconos para configuración y control del grabado y corte por láser. Los ajustes generales y la configuración específica para grabado y corte utilizada se presentan en la figura D.14(a) y D.14(b), respectivamente.

6. Antes de poner en funcionamiento la impresora, es importante utilizar todos los elementos de protección personal (gafas, mascarilla). También hay que asegurarse de que la tapa principal de la impresora cierre por completo el espacio destinado a la impresión, y comprobar que el ventilador y la bomba de agua estén funcionando

UNIVERSIDAD DE MÁLAGA

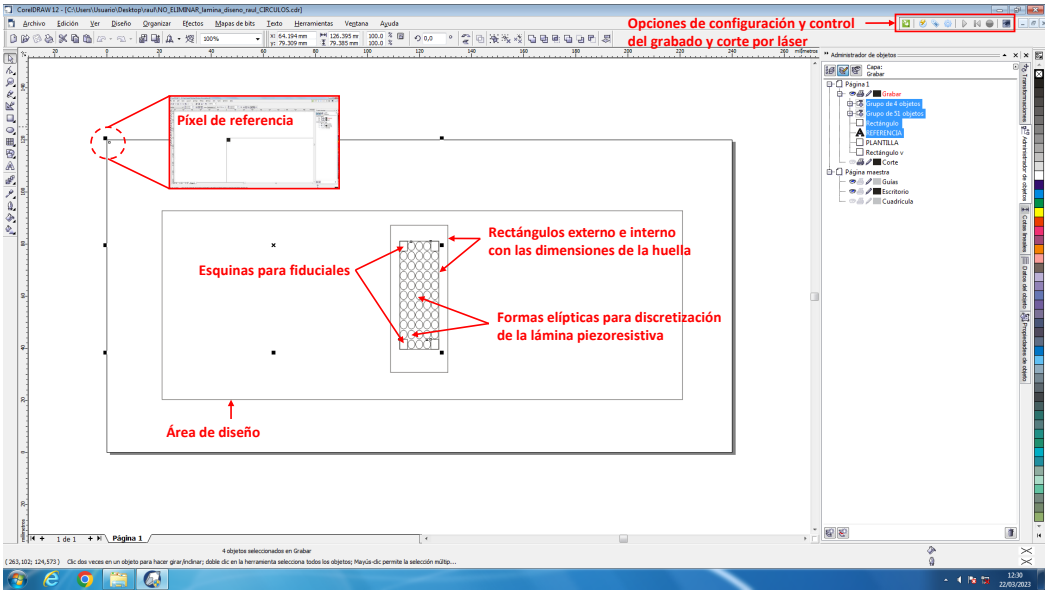


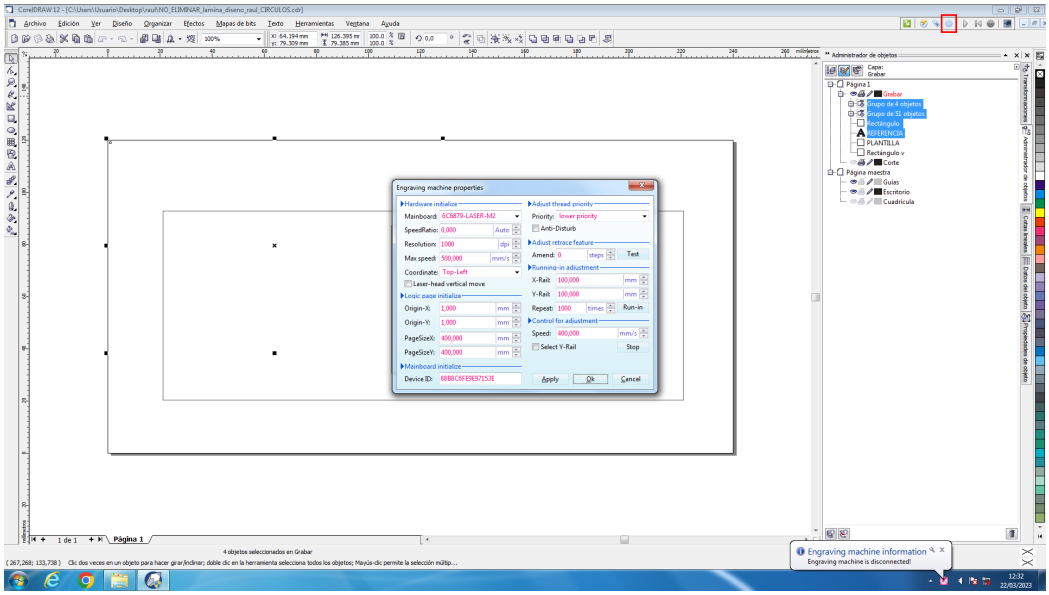
Figura D.13. Área de diseño en CorelDRAW® donde se ve el diseño para la discretización láser de la lámina piezoresistiva.

correctamente. Una vez hecho esto, se puede iniciar el proceso de grabado y corte por láser, ajustando la potencia del láser a 10 W (ver figura D.5). En particular, se realiza un proceso de grabado de 5 iteraciones, a una velocidad de 30 mm/s.

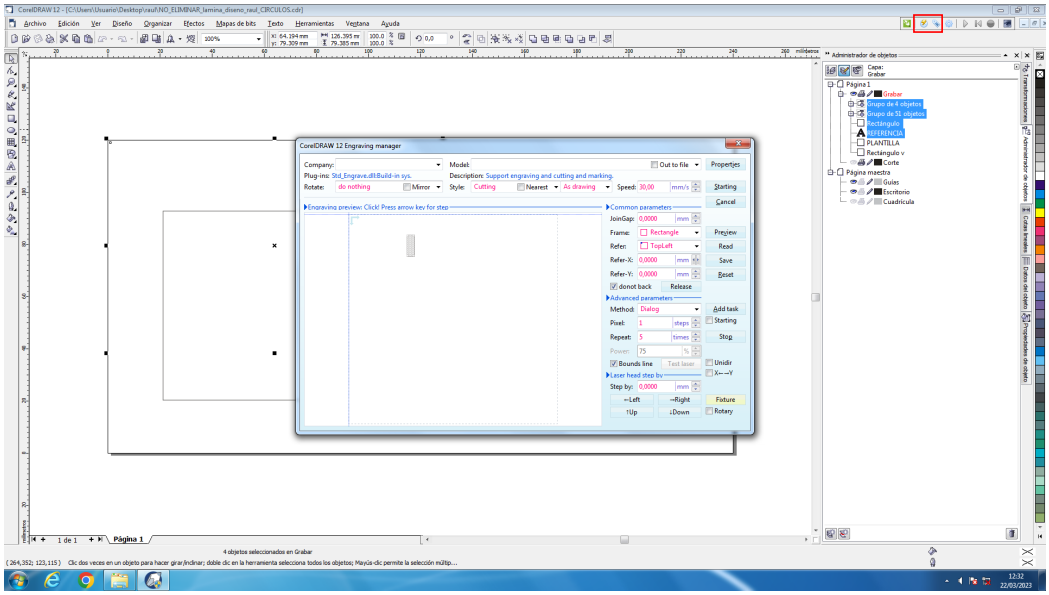
7. Cuando el proceso ha terminado, la interfaz de CorelDRAW® avisa al usuario. Los resultados se muestran en la figura D.15.

8. Por último, se taladran los fiduciales comentados previamente, y se coloca la huella encima de la lámina de electrodos. En la figura D.16(a) se muestra una imagen de la parte anterior del *cover*, antes y después de la adhesión de la lámina piezoresistiva discretizada, y en la figura D.16(b) se muestra la misma comparativa pero para la parte posterior de la huella. Por otro lado, en la figura D.17(a) y D.17(b) se muestran fotografías del montaje final en el sensor de dedo artificial.

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura D.14. (a) Ajustes generales de la impresora láser. (b) Ajustes específicos para el grabado y corte por láser.

UNIVERSIDAD DE MÁLAGA

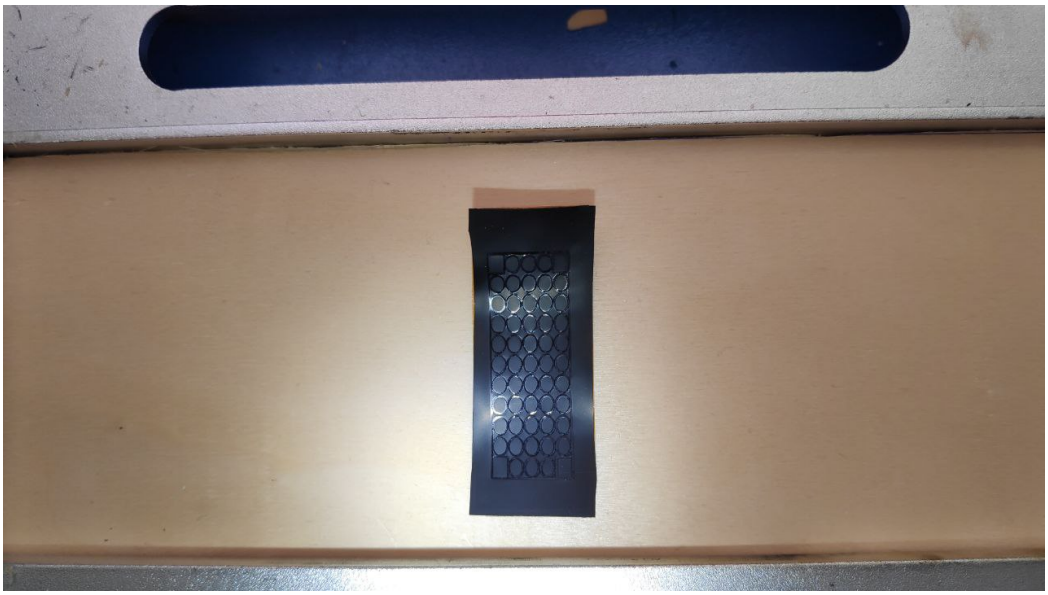
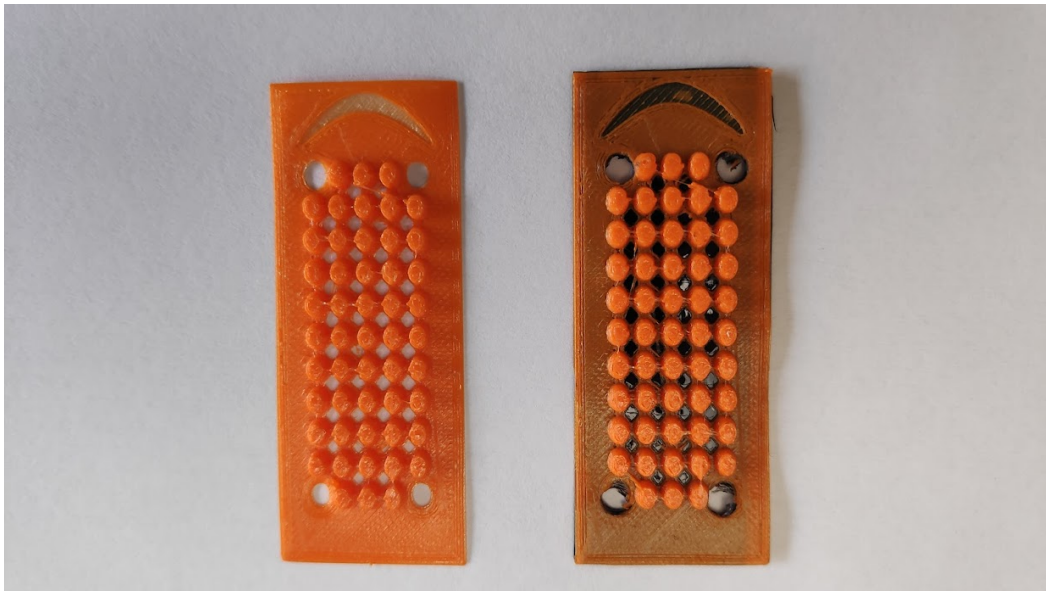
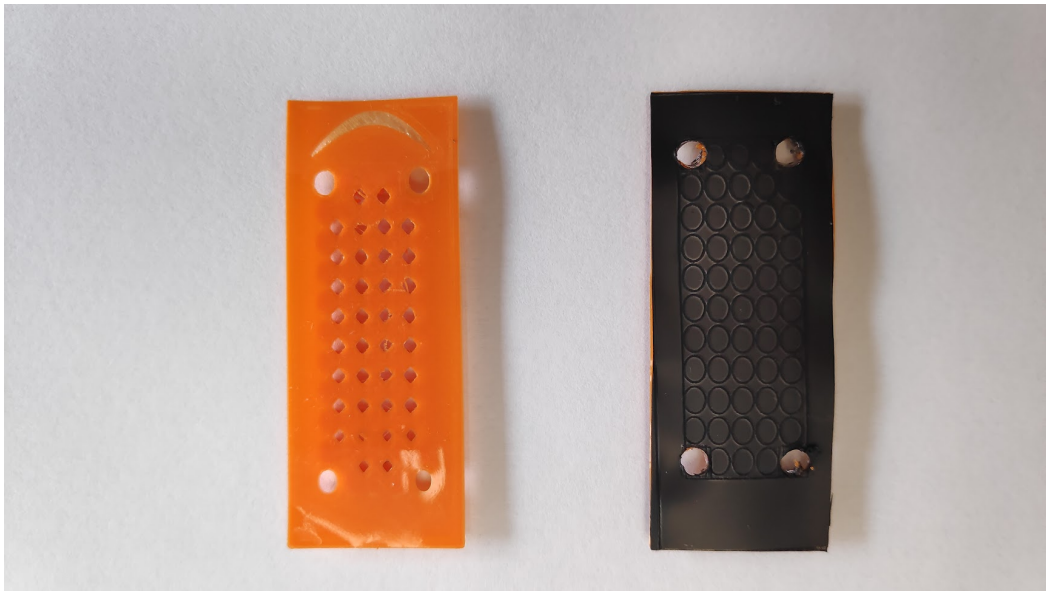


Figura D.15. Resultados de discretización por láser de la lámina piezoresistiva.

UNIVERSIDAD DE MÁLAGA



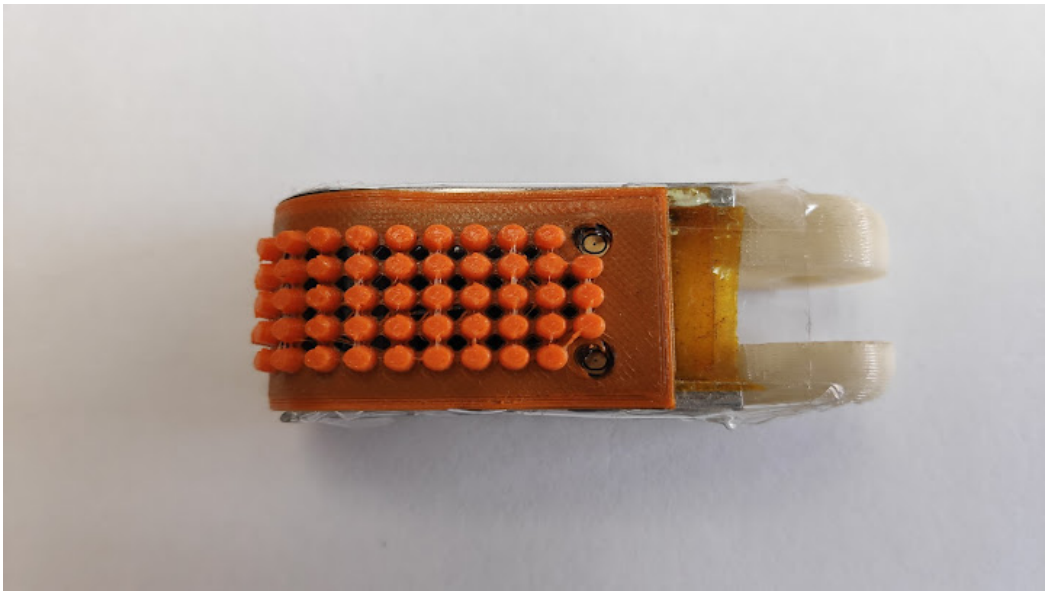
(a)



(b)

Figura D.16. (a) Comparativa de la parte anterior del *cover* estructurado, antes y después de adherir la lámina piezoresistiva discretizada por láser. (b) Comparativa de la parte posterior del *cover* estructurado, antes y después de adherir la lámina piezoresistiva discretizada por láser.

UNIVERSIDAD DE MÁLAGA



(a)



(b)

Figura D.17. (a) Fotografía superior del montaje final en el sensor de dedo artificial. (b) Fotografía lateral del montaje final en el sensor de dedo artificial.

UNIVERSIDAD DE MÁLAGA

Apéndice E Diagramas de la arquitectura en Vivado® y configuración en SDK

En este apéndice se presentan el diagrama general de la arquitectura implementada en Vivado®, para el algoritmo de barrido (sección E.1) y para el reconocimiento de rigidez de objetos (E.2). En cada caso, se diferencia entre sus distintas partes. Por otro lado, se muestran las principales opciones de configuración en SDK, comunes en ambas partes, para el correcto funcionamiento del sistema (sección E.3).

E.1. Diagramas de bloques para la detección de texturas

En la figura E.1 se puede ver el diagrama general de la arquitectura del sistema en Vivado®, para el algoritmo de barrido. Este diagrama es equivalente al presentando en la figura 3.38, en la sección 3.8 del capítulo 3. En la figura, el módulo de 'fir_compiler_0' se corresponde con el 'Antialiasing Filter'. El módulo 'control_interface_0' se corresponde con el bloque de 'Block RAM + AXISTREAM Interface'. El módulo 'controlador_two_sides_0' es el bloque 'SPI Controller'. Por otra parte, el módulo 'axi_gpio_0' se utiliza para enviar por un puerto de propósito general el valor de la cola FIFO a la salida de la arquitectura. El módulo 'processing_system7_0' se corresponde con las opciones de configuración para el ARM® y controla las entradas y salidas del *hardware* hacia el ARM® o del ARM® al *hardware*. También hay dos grupos de jerarquías 'hier_1' y 'hier_2' que se comentan a continuación.

Grupo de jerarquía 1

Vivado® permite agrupar bloques en jerarquías para hacer más cómodo el manejo del diseño. El primer grupo 'hier_1' se presenta en la figura E.2. Se corresponde con el

UNIVERSIDAD DE MÁLAGA

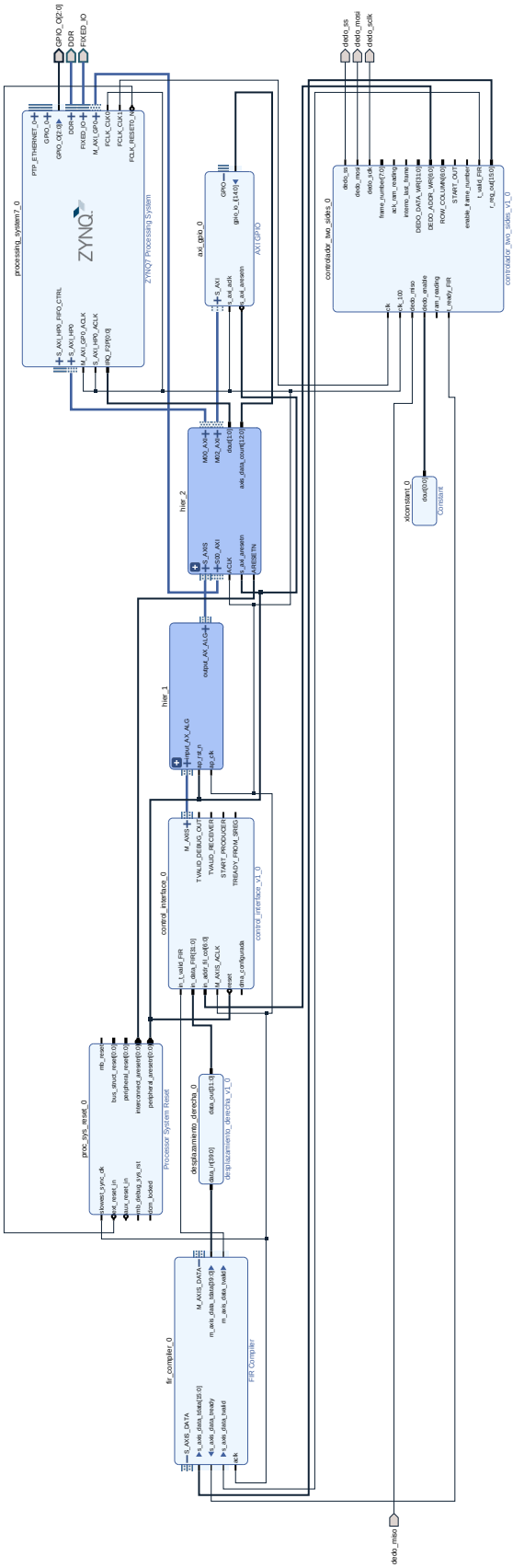


Figura E.1. Diagrama general de la arquitectura del sistema en Vivado®para el algoritmo de barrido.

UNIVERSIDAD DE MÁLAGA

módulo de algoritmo de barrido diseñado con la herramienta HLS de Vivado 'HLS-Vivado Sweep Algorithm'.

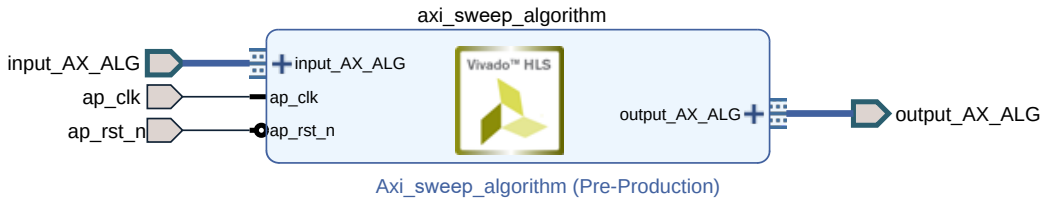


Figura E.2. Grupo de jerarquía que contiene al algoritmo de barrido en HLS.

En el caso de utilizar la implementación de la FFT, el primer grupo de jerarquía se correspondería con la figura E.3.

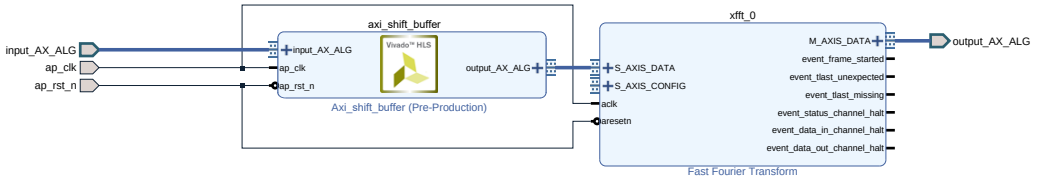


Figura E.3. Grupo de jerarquía que contiene el *buffer* de desplazamiento y el IP Core para la FFT.

El IP Core de Vivado® para el cálculo de los coeficientes de la FFT 'xfft_0' se ha diseñado con las siguientes opciones de configuración [130]:

- Target Clock Frequency: 100 MHz
- Target Data Throughput: 50 Msps
- Architecture Choice: Pipelined, Streaming I/O. Con esta arquitectura el módulo puede simultáneamente procesar el *frame* actual de datos, cargar los datos para el próximo *frame*, y enviar los resultados obtenidos para el *frame* previo. Esto lo logra a costa de aumentar el consumo de recursos respecto a otras posibles configuraciones, que no pueden trabajar de forma simultánea.
- Data Format: Fixed Point. De esta forma, el módulo trabaja con datos de entrada de 16 bits. La salida del módulo se proporciona con 16 bits para parte real y 16 bits para la parte imaginaria o factor de fase, ambas en formato 'Fix16-frac15' (con 15 bits para la parte fraccional).
- Scaling Options: Scaled. El módulo opera mediante un esquema de flujo que replica pequeñas unidades 'en mariposa' para obtener la salida de forma óptima. Con esta opción evitamos que ocurra un *overflow* en las sucesivas operaciones de este procedimiento.

UNIVERSIDAD DE MÁLAGA

- Rounding Modes: Truncation. Con esta opción conseguimos reducir el ancho de la parte fraccional en las consecutivas operaciones para calcular la salida de la FFT.
- Output Ordering: Es importante tener en cuenta el formato de salida de la FFT. El IP Core de Xilinx™ ofrece la opción de obtener los resultados en formato natural o en formato de orden inverso de bits (*bit-reversed order*). Por defecto, la FFT proporciona los resultados en formato *bit-reversed order*[1]. Aunque es posible configurar el IP Core para que proporcione los resultados en formato natural, esto conlleva un coste en memoria y tiempo de ejecución. Por lo tanto, el uso del formato *bit-reversed order* en el IP Core ofrece un diseño más eficiente en términos de latencia y recursos [130] [131]. En esta tesis, el IP Core en *hardware* está configurado con el formato *bit-reversed order*, debiendo incluirse una pequeña rutina en la parte PS para reorganizar los resultados de la FFT.

Por otra parte, si se utiliza la implementación del algoritmo de Goertzel 'HLS-Vivado Goertzel Algorithm', diseñado con la herramienta HLS de Vivado, el primer grupo de jerarquía se correspondería con la figura E.4.



Figura E.4. Grupo de jerarquía que contiene el módulo en HLS para el algoritmo de Goertzel.

Grupo de jerarquía 2

El segundo grupo 'hier_2' se presenta en la figura E.5. Se corresponde con la configuración y control del módulo DMA y la implementación de la FIFO de salida.

E.2. Diagramas de bloques para la distinción de la rigidez de objetos

En esta sección, en la figura E.6 se puede ver el diagrama general de la arquitectura del sistema en Vivado®, para el reconocimiento de la rigidez de objetos. Este diagrama es equivalente al presentando en la figura 4.8, en la sección 4.4 del capítulo 4. En

[1]El formato *bit-reversed order* se calcula tomando el índice correspondiente a cada valor de entrada, escrito en binario, e invirtiendo el orden de los dígitos. Por ejemplo, para una secuencia de hasta 8 valores codificados en 3 bits, el índice binario '000' se convierte en '000', '001' se convierte en '100', '010' se mantiene como '010', '011' se convierte en '110', y así sucesivamente [130].

UNIVERSIDAD DE MÁLAGA

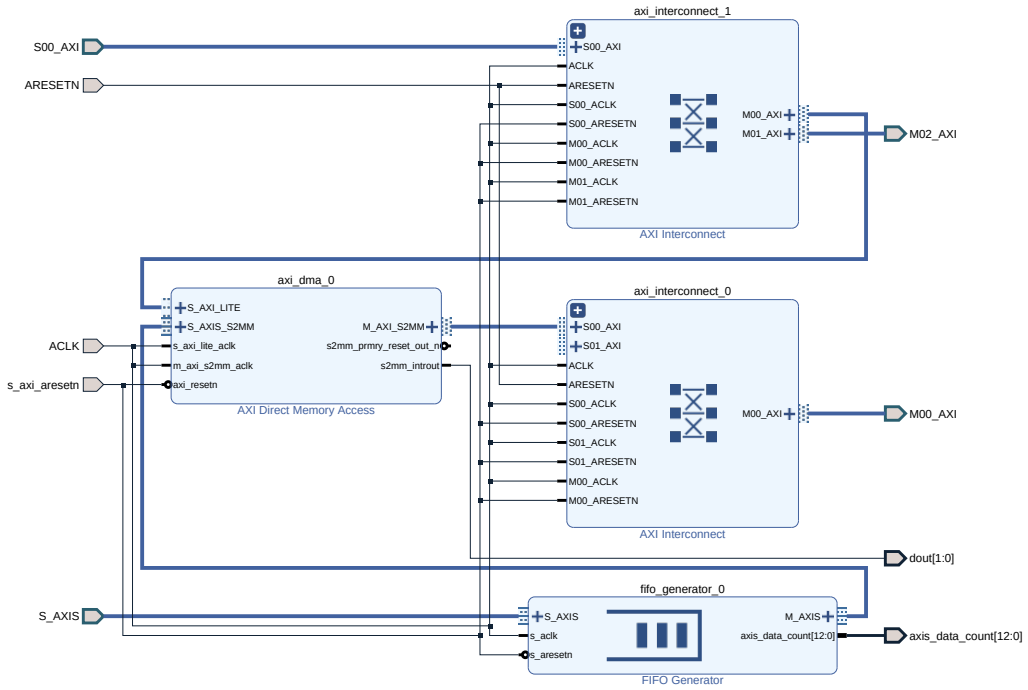


Figura E.5. Grupo de jerarquía que contiene el control DMA y la FIFO de salida.

concreto, el diagrama se corresponde a la inclusión de los sensores de dedo y palma, cálculo de los dos primeros momentos de la imagen táctil $M_{0,0}$ y $M_{0,1}$, para dedo y palma, respectivamente, así como el cálculo de las características AR y MAX comentadas en el capítulo 4, aunque la arquitectura es la misma si se incluyen más características. Además, se incluye el módulo de cálculo PCA para 3 componentes principales, aunque el procedimiento sería similar para un número diferente de componentes principales. En la figura, el módulo 'axi_gpio_0' se utiliza para enviar por un puerto de propósito general el valor de la cola FIFO a la salida de la arquitectura. El módulo 'processing_system7_0' se corresponde con las opciones de configuración para el ARM® y controla las entradas y salidas del *hardware* hacia el ARM® o del ARM® al *hardware*. El módulo 'wrapped_pca_mmult_hw_0' se corresponde con la multiplicación *hardware* de matrices para el cálculo PCA. Si no se utliza PCA, este módulo debe omitirse. Por último, el módulo 'axi_gpio_1' y los puertos de salida se utilizan para depuración por los puertos PMODs de la ZedBoard™. En este diseño existen hasta siete grupos de jerarquía 'hier_dedo', 'hier_palma', 'hier_serial2parallel', 'hier_parallel2serial', 'hier_t_ready_in', 'hier_t_validRig' y 'hier_dma' (el contenido de este último se ha presentado en la figura E.5). El resto de grupos de jerarquía se exponen a continuación.

Grupos de jerarquía para dedo y palma

La figura E.7 muestra los módulos del cálculo de características para el sensor de dedo. Estos módulos se corresponden con los bloques 'VHDL Features Computing Module

UNIVERSIDAD DE MÁLAGA

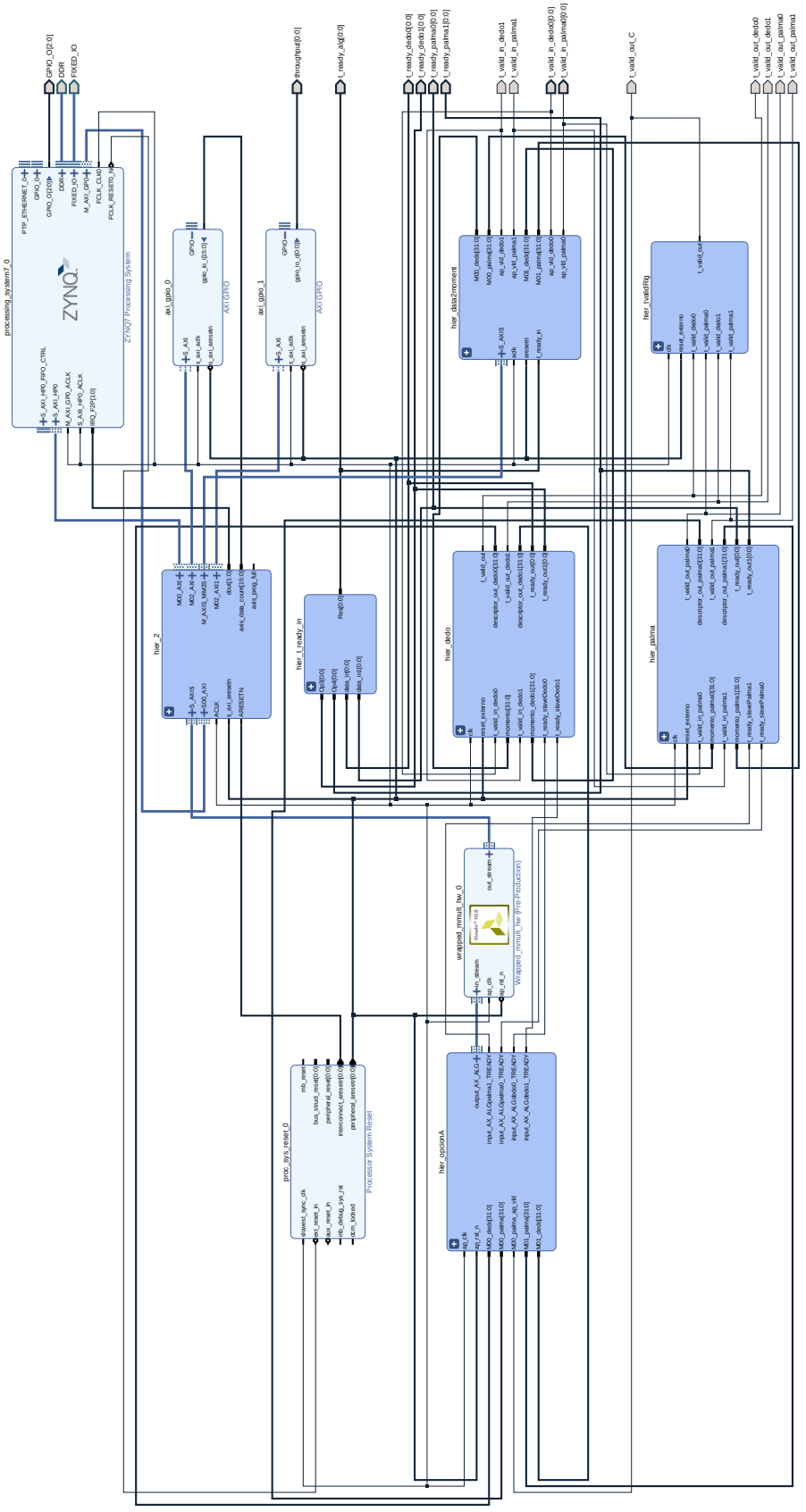


Figura E.6. Diagrama general de la arquitectura del sistema en Vivado® para el reconocimiento de objetos por rigidez.

UNIVERSIDAD DE MÁLAGA

(Finger)' de la figura 4.8 del capítulo 4. La figura E.8 por su parte, muestra los módulos del cálculo de características para el sensor de palma. Estos módulos se corresponden con los bloques 'VHDL Features Computing Module (Palm)' de la figura 4.8 del capítulo 4.

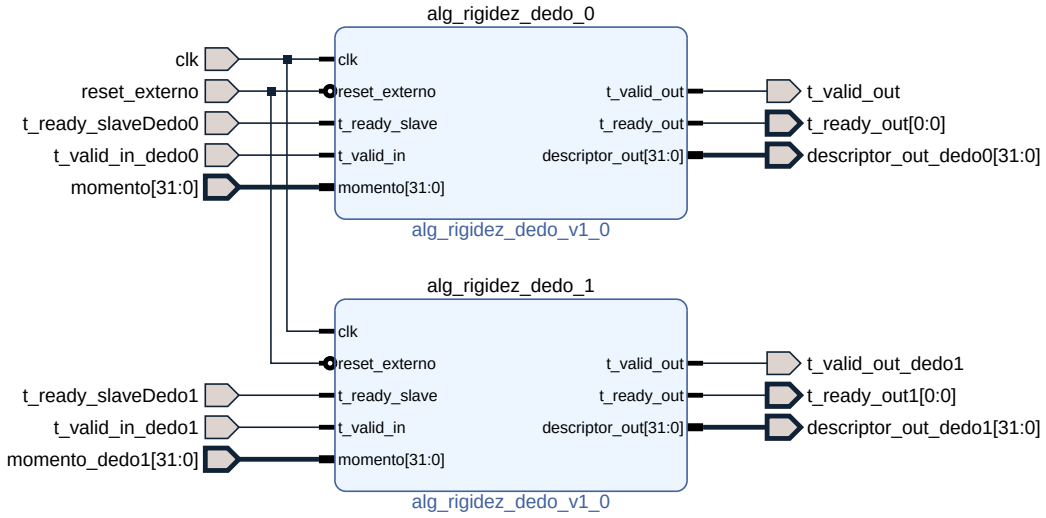


Figura E.7. Grupo de jerarquía que contiene el cálculo de características para el dedo, en el reconocimiento de la rigidez de objetos.

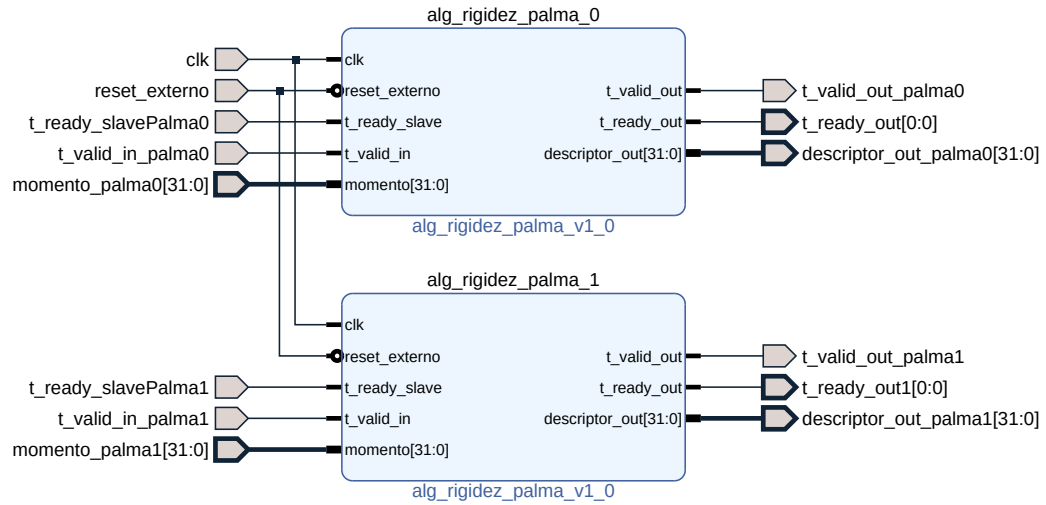


Figura E.8. Grupo de jerarquía que contiene el cálculo de características para la palma, en el reconocimiento de la rigidez de objetos.

Grupos de jerarquía para las interfaces serie a paralelo y paralelo a serie

El módulo 'AXISTREAM Serial to Parallel Interface' de la figura 4.8 del capítulo 4, se corresponde con el diagrama en Vivado® que se muestra a continuación en la figura E.9. Por otro lado, en la figura E.10 se muestra el diagrama equivalente al módulo

UNIVERSIDAD DE MÁLAGA

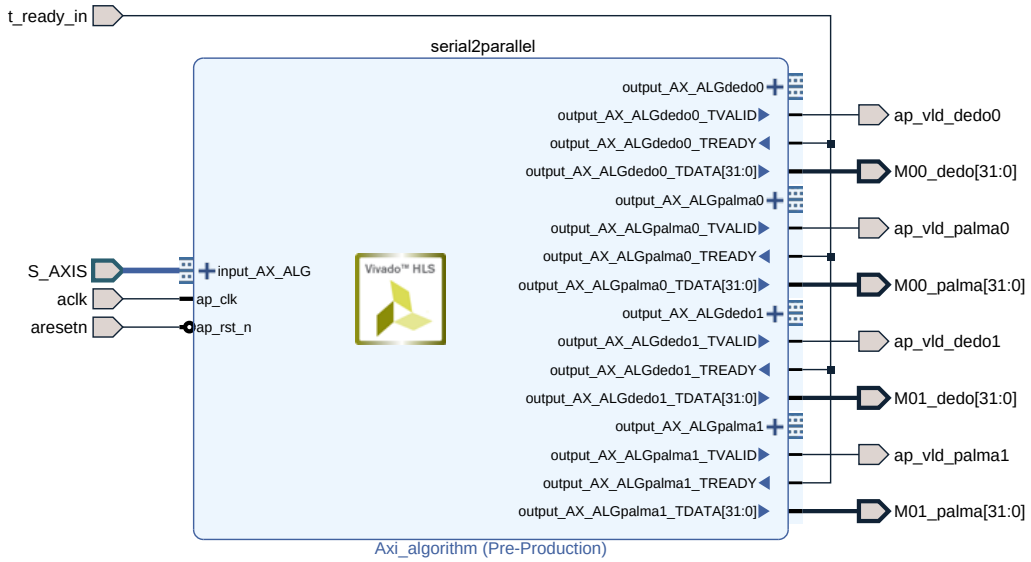


Figura E.9. Grupo de jerarquía para la interfaz serie a paralelo.

'AXISTREAM Parallel to Serial Interface' de la figura 4.8 del capítulo 4.

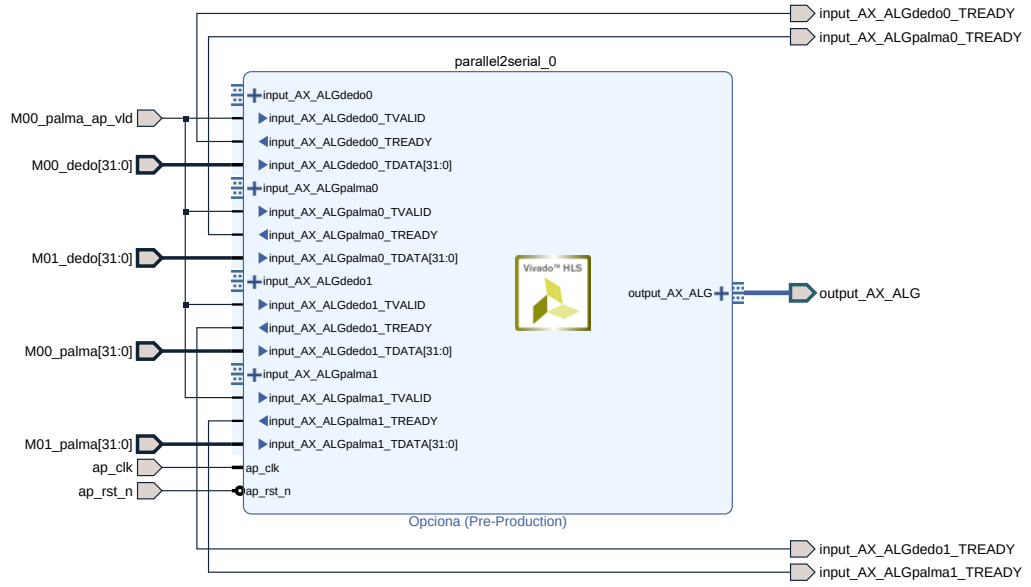


Figura E.10. Grupo de jerarquía para la interfaz paralelo a serie.

Grupos de jerarquía para la sincronización de las señales de *t_ready* y *t_valid*

Los módulos 'VHDL Synchronization Modules' de la figura 4.8 del capítulo 4, se incluyen para la sincronización de las señales de *t_ready* y *t_valid*. La figura E.11 muestra el diagrama equivalente (en este caso sólo consiste en puertas lógicas AND) para la sincronización de las señales *t_ready*. Por su parte, la figura E.12 muestra el diagrama

UNIVERSIDAD DE MÁLAGA

equivalente (ver máquina de estados de la figura 4.9 del capítulo 4) para la sincronización de las señales de *t_valid*.

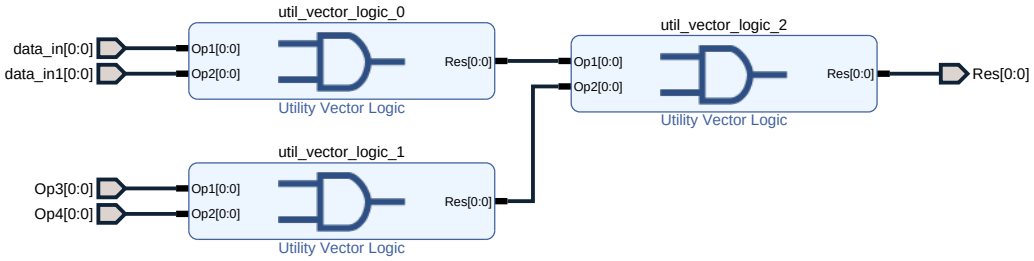


Figura E.11. Sincronización de las señales de *t_ready*.

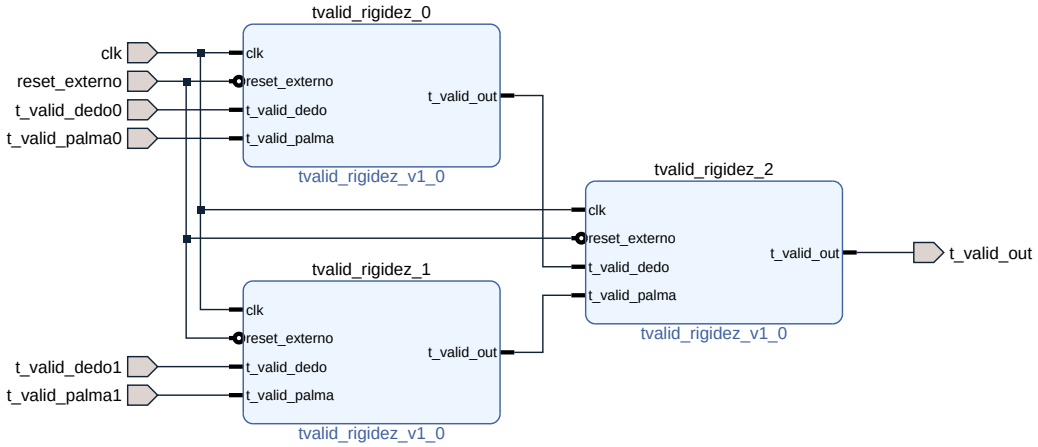


Figura E.12. Sincronización de las señales de *t_valid*.

E.3. Configuración SDK

Una vez se crea una nueva aplicación de proyecto en el entorno de programación SDK de Vivado®, es necesario configurar una serie de opciones para compilación, librerías, configuración del sistema operativo en tiempo real FreeRTOS® y pila lwIP. En esta sección se muestran las principales opciones consideradas para el sistema desarrollado.

En primer lugar, cabe mencionar que se añade la librería *math.h* para llevar a cabo algunas operaciones aritméticas necesarias. Esto se logra mediante el comando $-m$ de la pestaña 'C Libraries'. Por otro lado, se establecen las opciones de configuración del *linker script*. Con estas opciones se especifica cómo se deben combinar y organizar los archivos objeto obtenidos durante el proceso de compilación para generar un ejecutable final. En particular, se establece un tamaño de ≈ 95.37 MB para los tamaños de pila 'Heap Size' y 'Stack Size', respectivamente.

UNIVERSIDAD DE MÁLAGA

Por otro lado, como se indicó en el capítulo 3, en el proyecto desarrollado se han incluido FreeRTOS® junto con la pila ligera, de TCP/IP, lwIP. El uso de FreeRTOS®+lwIP requiere de una serie de parámetros de configuración. La mayoría de estos parámetros se dejan al valor por defecto. En la tabla E.1 y tabla E.2 se muestran aquellos parámetros que se modificaron para FreeRTOS® y lwIP, respectivamente.

Tabla E.1. Opciones de configuración *Kernel Behavior* para FreeRTOS®.

Kernel Behavior			Hook functions
Minimal Stack Size (bytes)	Tick Rate (ticks/s)	Total Heap Size (bytes)	
200000	100	26614400	TRUE

Tabla E.2. Opciones de configuración para la pila lwIP.

api mode	dhcp mode	lwip_mem_size (heap memory in bytes)	temac_adapter_options (physical link speed in Mbps)
SOCKET API	false	26614400	100

UNIVERSIDAD DE MÁLAGA

Apéndice F Simulación en Vivado®

En este apéndice, se muestra primero en la sección F.1 un ejemplo de simulación para el cálculo de V, expresión (3.3) del capítulo 3. Después, en la sección F.2 también se muestra otra simulación para el cálculo de las cuatro características propuestas a partir de la secuencia de momentos de la imagen táctil, explicadas en el capítulo 4.

F.1. Ejemplo de simulación para el barrido

Esta simulación permite verificar que para unos datos fijos guardados en una memoria BRAM de doble puerto, la salida de V es correcta. En concreto, se configura el algoritmo para que la longitud del vector de salida sea $M = 16$, y la frecuencia de muestreo de $F_s = 300$ Hz. Vivado® incluye la herramienta de simulación XSIM desarrollada por Xilinx®, y que se ha utilizado en este trabajo de tesis.

En la figura F.1 se presenta el diagrama de bloques conceptual que se considera para esta simulación. En este diseño se ha instanciado un bloque que genera una señal de 'START' (que se activa al inicio) y una señal de 'CURRENT_FRAME' que alterna su valor lógico entre '1' y '0' con la frecuencia de muestreo. Por otro lado, se instancia una memoria BRAM de doble puerto. La memoria contiene dos rangos de direcciones (en cada uno hay 32 datos de 32 bits cada uno). De esta forma, para cada *frame* de datos, se lee de un rango de direcciones distinto. Como la señal 'CURRENT_FRAME' toma el valor '1' y '0' de forma consecutiva, se puede depurar el diseño para verificar que la información que se obtiene es coherente, para dos *frames* consecutivos de datos de entrada diferentes. Por otro lado, el bloque de interfaz se encarga de direccionar a la memoria de doble puerto y de leer los datos que pasará al bus AXISTREAM. Estos datos serán la entrada del módulo de algoritmo que calcula V. El *script* de simulación

UNIVERSIDAD DE MÁLAGA

permite configurar el reloj de 100 MHz, activar inicialmente a '1' una señal que indica que el módulo DMA está configurado y generar la señal de *reset*.

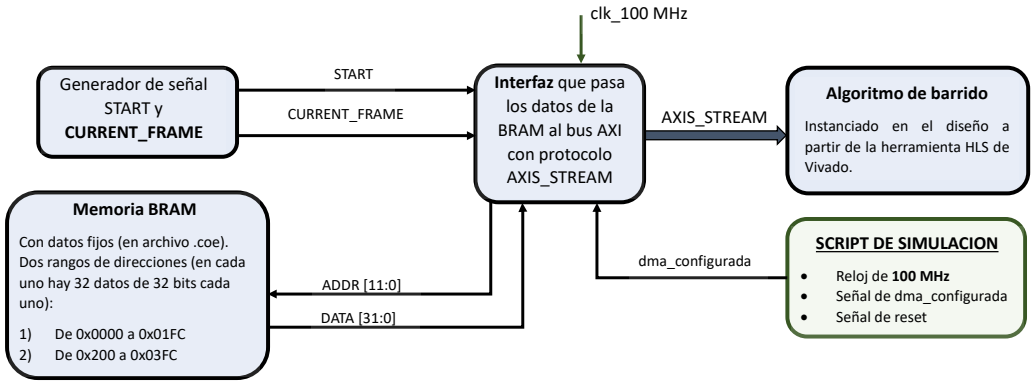


Figura F.1. Diagrama conceptual para la simulación.

Una captura de 60 ms para la simulación se presenta en la imagen de la figura F.2. En la captura se observan los cambios de la señal lógica 'CURRENT_FRAME' (en magenta) para cada período de muestreo simulado. Por otra parte, las señales 'output_AX_ALG_tvalid' y 'output_AX_ALG_tdata[31:0]' (también en magenta) son las señales de salida del algoritmo para datos válidos y datos de salida, respectivamente. La señal en amarillo 'input_AX_ALG_TREADY' se activa cuando el algoritmo de barrido está listo para recibir nuevos datos a la entrada. La señal 'm00_axis_tvalid' (también en amarillo) se activa para indicar que los datos que se envían hacia el algoritmo son válidos. La señal en amarillo 'input_AX_ALG_TDATA[31:0]' contiene los datos de entrada al algoritmo. Cabe destacar que, en esta simulación, el tiempo que tarda el algoritmo en procesar los datos una vez los recibe es de 4515 ns. Este tiempo es muy pequeño en comparación con el período de muestreo simulado de 3.3 ms. Esto es ventajoso para implementaciones que requieran un mayor tamaño del vector de salida M, ya que la frecuencia de muestreo del sistema real no va a suponer una limitación.

A continuación, en la figura F.3 se muestra un *zoom* de la figura F.2 para los datos de entrada al algoritmo cuando 'CURRENT_FRAME=0'. Cuando se realizó esta simulación, se comprobó que los datos que se veían son los mismos que se habían grabado en la correspondiente zona de memoria que se estaba direccionando. Aunque no se incluye captura de la simulación, se comprobó que los datos obtenidos de la zona de memoria correspondiente a 'CURRENT_FRAME=1' también eran correctos.

Por otro lado, en la figura F.4 se muestra un *zoom* para los datos de salida del algoritmo de barrido con 'CURRENT_FRAME=0'. Cuando se realizó esta simulación, se comprobó que la salida del algoritmo se corresponde con los 16 datos de salida que el cálculo de V para el barrido, expresión (3.3) del capítulo 3, obtendría para la secuencia

UNIVERSIDAD DE MÁLAGA

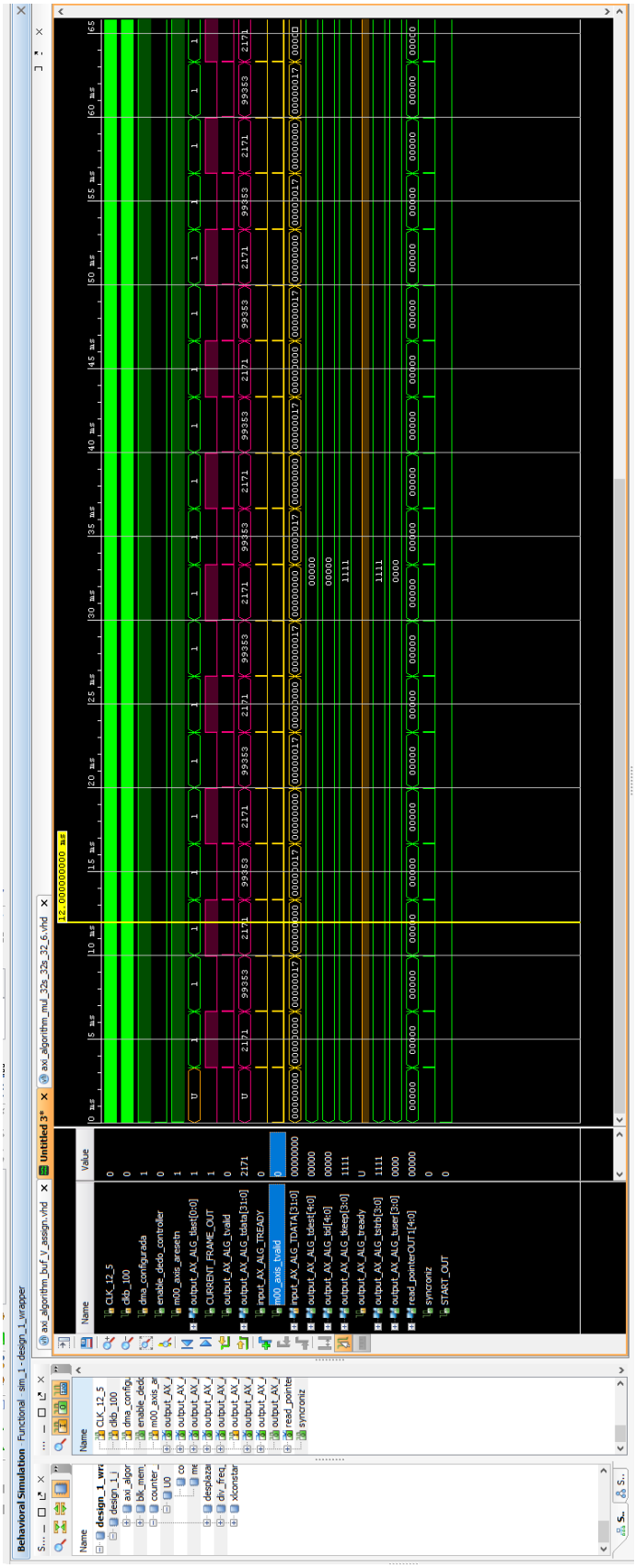


Figura F.2. Captura de la simulación durante 60 ms.

UNIVERSIDAD DE MÁLAGA

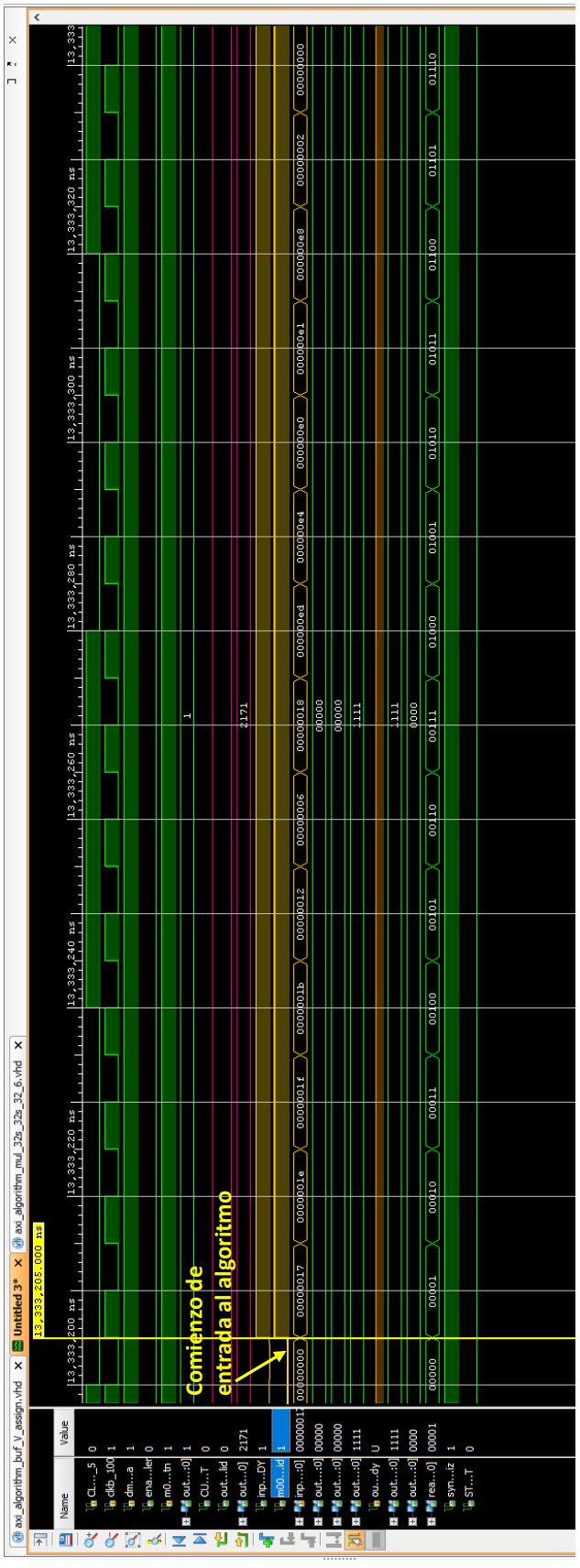


Figura F.3. *Zoom* en la parte de los datos de entrada al algoritmo, cuando la señal 'CURRENT_FRAME=0'.

UNIVERSIDAD DE MÁLAGA

de entrada que se simula en la figura F.3. Como se puede comprobar, la señal de 't_valid' permanece al valor lógico '1' para los 16 datos de salida. Aunque no se incluye captura de la simulación, también se comprobó que los datos de salida del algoritmo cuando 'CURRENT_FRAME=1' eran correctos.

F.2. Ejemplo de simulación para la detección de rigidez de objetos

A continuación se muestra una simulación realizada para el módulo 'VHDL Features Computing Module' de la figura 4.8 del capítulo 4. En la figura F.5 se puede ver la simulación completa de $20\mu s$. En esta simulación se dio como entrada una secuencia de seis palabras de 32 bits, incluyendo en la parte más significativa los bits indicativos de las características $H1$, MAX, $H2$ y de final de secuencia, END (ver subsección 4.4.2 del capítulo 4).

En la simulación de la figura F.5 el cálculo de estas características se simula varias veces para los mismos datos de entrada. Ahora, en la figura F.6 se hace un *zoom* en el momento en el que se proporciona la salida del algoritmo. Se puede comprobar como transcurren 3 ciclos de reloj desde la llegada del último dato válido a la entrada, hasta que se proporciona la salida.

En resumen, la simulación en Vivado® nos permite comprobar que el funcionamiento de los algoritmos es correcto antes de pasar a la fase de síntesis lógica RTL e implementación del diseño. Los archivos .tcl de códigos de simulación utilizados se proporcionan en el repositorio público en *https://github.com/raulLkRb/thesis.git*.

UNIVERSIDAD DE MÁLAGA

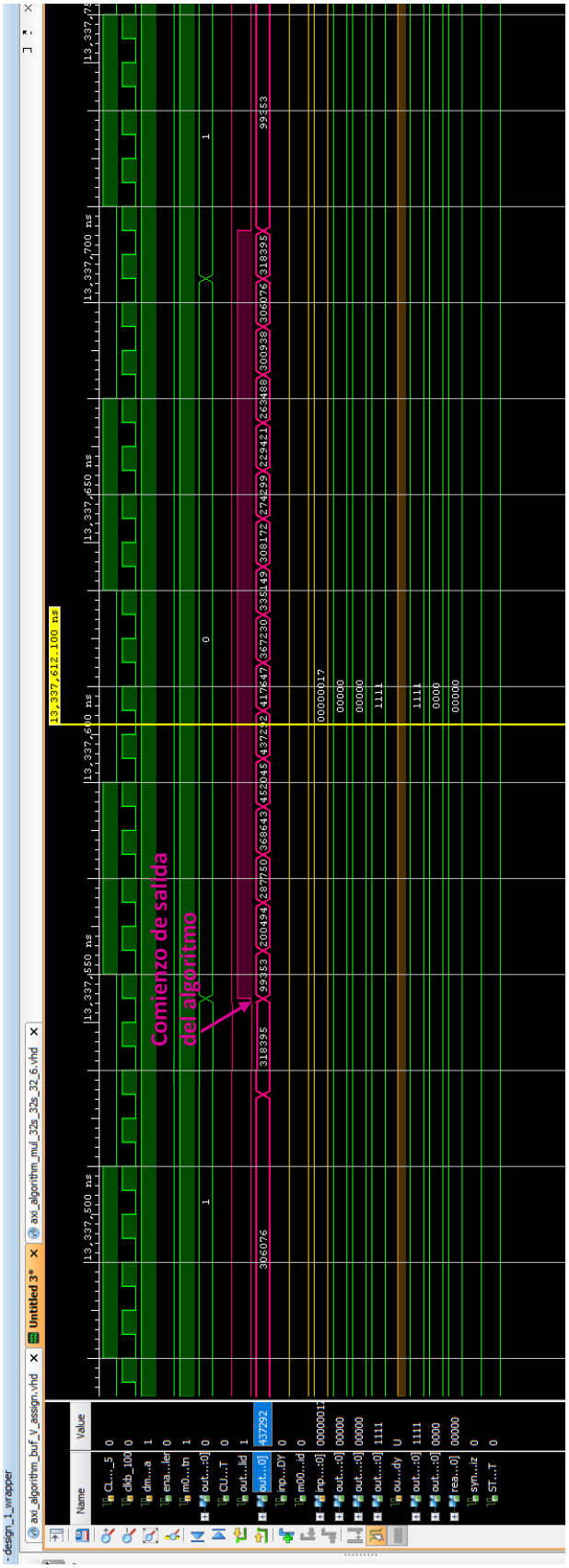


Figura F.4. *Zoom* en la parte de salida del algoritmo de barrido, cuando la señal 'CURRENT_FRAME=0'.

UNIVERSIDAD DE MÁLAGA

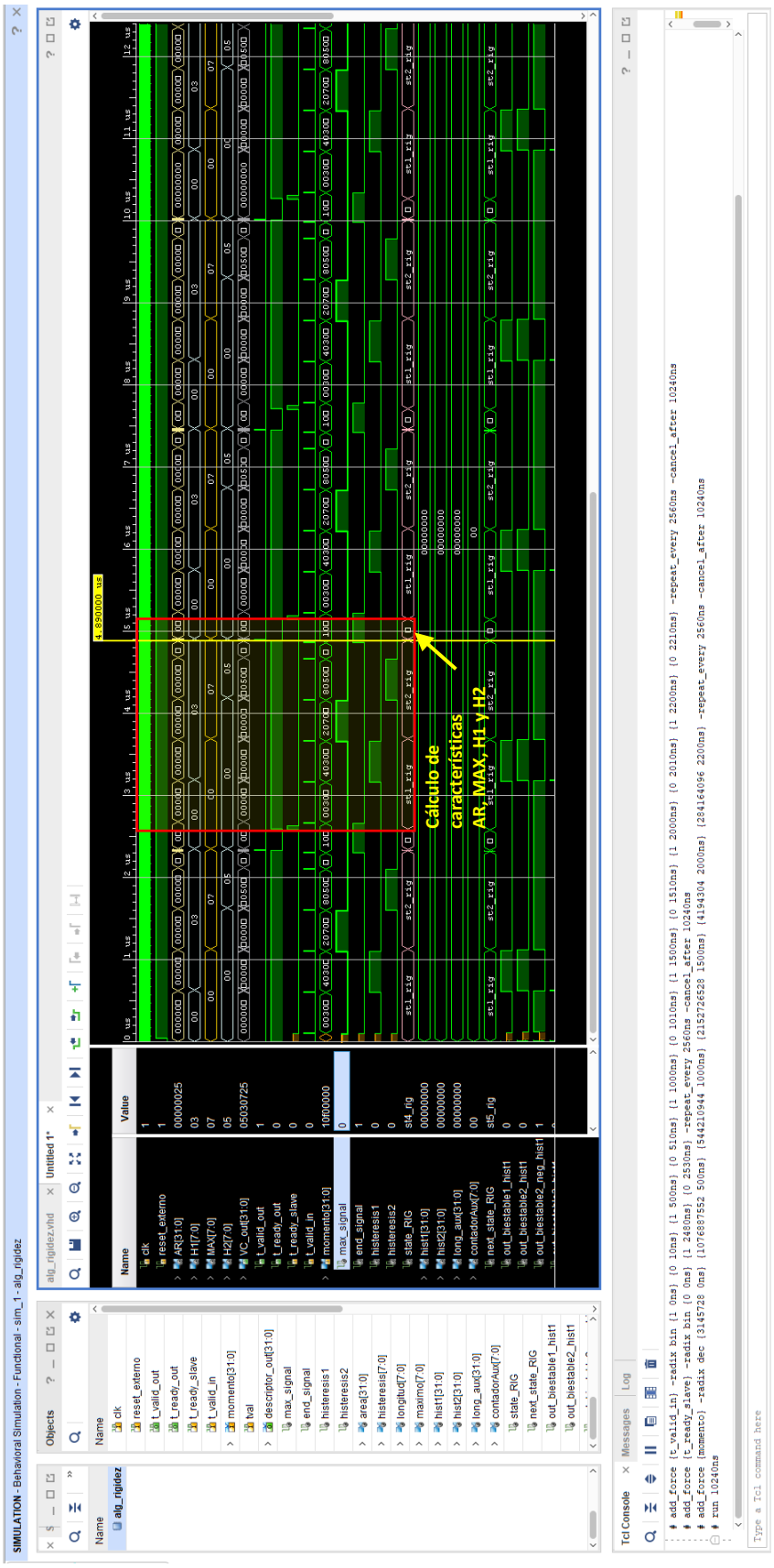


Figura F.5. Simulación completa de $20\mu s$ para el cálculo de las características AR, MAX, $H1$ y $H2$ propuestas en el capítulo 4. En la figura, VC_out[31:0] se corresponde con el vector de características a la salida del algoritmo de extracción de características.

UNIVERSIDAD DE MÁLAGA

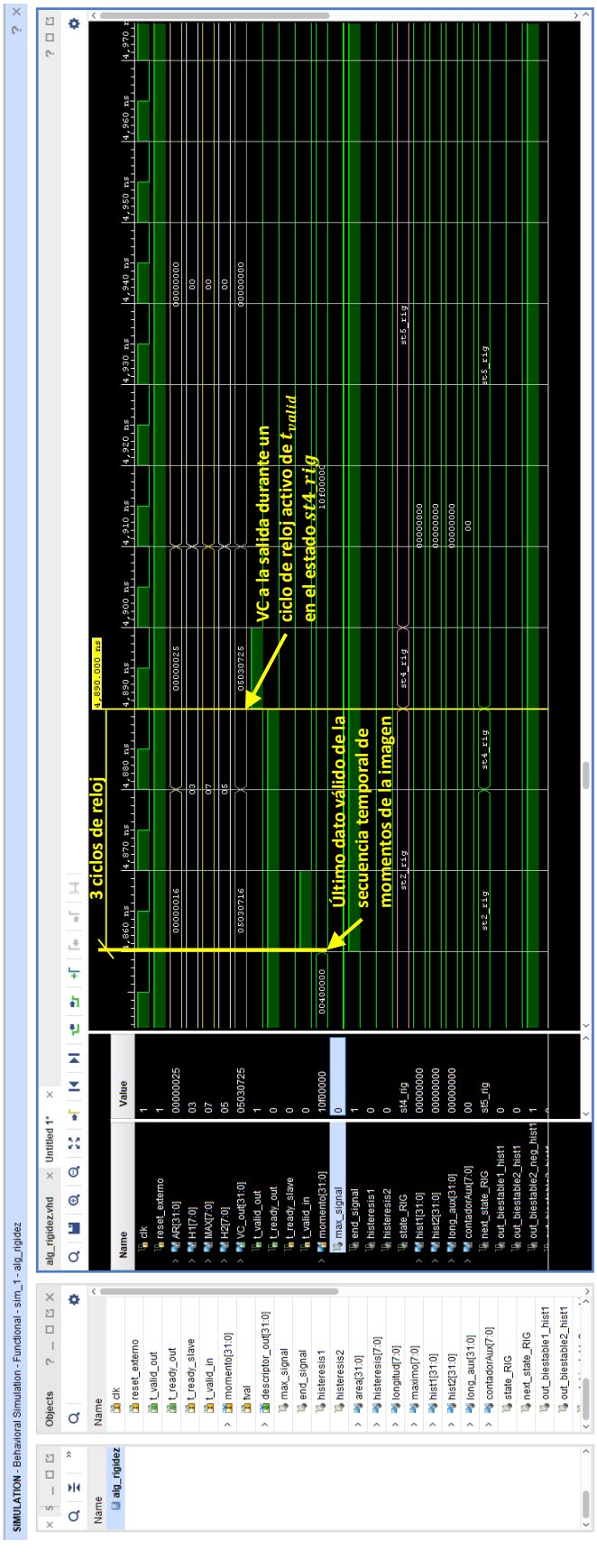


Figura F.6. *Zoom* en el instante en el que se proporciona la salida del vector de características, como una palabra de 32 bits, durante el ciclo de reloj en el que la señal de *t_valid_out* se encuentra activa.

UNIVERSIDAD DE MÁLAGA

Índice de figuras

1.1. El desarrollo de sensores táctiles en manos robóticas se puede utilizar para aplicaciones en prostética [12]. 3

1.2. Prototipo de mano de Barrett [38], con los sensores correspondientes a los dedos y palma artificiales de los que se parte en esta tesis y la PCB con la electrónica de adquisición y comunicación para estos sensores [36]. 6

1.3. (a) Aplicación del algoritmo de pre-procesado para identificación de texturas de diferente resolución espacial [39]. (b) Experimentos para el reconocimiento de la rigidez de objetos, mediante su palpación experimental [40]. 8

2.1. Sensor correspondiente al dedo artificial, consistente en una placa de circuito impreso o *Printed-Circuit Board* (PCB) semirígida con el *cover* estructurado que ayuda a concentrar la fuerza en los diferentes tácteles de la matriz de electrodos. Por la parte de atrás del *cover* se monta una lámina piezoresistiva discretizada. El sensor se puede conectar mediante el protocolo bus serie *Serial Peripheral Interface* (SPI) a una placa basada en FPGA con la electrónica de adquisición y comunicación propuesta en [36]. Las figuras no están a escala. 16

2.2. Simulaciones de elementos finitos en Comsol®: (a) dimensiones del *cover* estructurado, (b) distribución de la presión a lo largo del área de contacto para diferentes fuerzas tangenciales, (c) diferentes alturas para la estructura en cada táctel del sensor, y (d) anchura del sustrato. Estos resultados se presentaron en el congreso CIBEC 2018 [60]. 18

UNIVERSIDAD DE MÁLAGA

2.3. Diseño del *cover* estructurado para el dedo. Las cotas de distancia están dadas en mm. 19

2.4. Perspectiva isométrica de otros diseños para el *cover* con *zoom* del detalle de la superficie estructurada. Los diseños (a) y (b) están basados en saliente en forma de tronco de cono más esbeltos que el de la figura 2.3. Los diseños (c) y (d) son cilíndricos con los extremos redondeados. El diseño (e) tiene un acabado semiesférico. 19

2.5. Detalle de la electrónica local al dedo propuesta en [37]. 20

2.6. Propuesta de conexión directa entre sensor y FPGA tomada de la tesis [36]. La matriz del sensor tiene 5 filas y 11 columnas. Se incluyen dos filas de resistencias de calibración. 22

2.7. Forma de la señal para la carga/descarga del condensador con o sin resistencias en paralelo. 23

2.8. (a) Carga y, (b) descarga del condensador utilizando resistencias de calibración R_c en paralelo con R_x. 24

2.9. (a) Curva obtenida sin resistencia de linealización para un elástomero continuo hecho de Filaflex®. (b) Curva obtenida con una resistencia de linealización de $6.8\ k\Omega$ para el *cover* estructurado. El tiempo de descarga del condensador t_{dC} está expresado en ns, teniendo en cuenta que la frecuencia de reloj en la FPGA Spartan-6® es de 50 MHz (20 ns). 25

2.10. (a) Imagen de la capa sensible de piezoresistivo discretizada mediante impresora láser. (b) *Cover* estructurado. (c) Montaje completo del dedo. 26

2.11. Secuencia de aplicación progresiva de fuerza para un *cover* continuo no estructurado, en algunos instantes de tiempo. En este test el rango de aplicación va de 0 N a 10 N. La primera fila se corresponde con la aplicación de una fuerza creciente, y la segunda con una fuerza decreciente. 27

2.12. Secuencia de aplicación progresiva de fuerza para un *cover* estructurado, en algunos instantes de tiempo. En este test el rango de aplicación va de 0 N a 10 N. La primera fila se corresponde con la aplicación de una fuerza creciente, y la segunda con una fuerza decreciente. 28

UNIVERSIDAD DE MÁLAGA

2.13. Salida de la matriz del sensor (a) con el *cover* continuo aplicando una fuerza con el calibrador triaxial, sobre un táctel, de 27.94 N, y (b) con el *cover* estructurado con una fuerza de 27.87 N. 28

2.14. Aplicación de fuerzas sobre pieza en forma de puente con el calibrador triaxial. 29

2.15. Secuencia de aplicación de fuerza progresiva de 0 N a 10 N en la pieza puente de la figura 2.14. Se utiliza una lámina piezoresistiva continua. La primera fila muestra fuerza creciente y la segunda, fuerza decreciente. 29

2.16. Secuencia de aplicación de fuerza progresiva de 0 N a 10 N en la pieza puente de la figura 2.14. Se emplea una lámina piezoresistiva discretizada por láser. La primera fila muestra fuerza creciente y la segunda, fuerza decreciente. 29

2.17. Salida de la matriz del sensor al realizar un experimento con la pieza-puente de la figura 2.14; (a) con la capa de piezoresistivo continuo aplicando una fuerza con el calibrador triaxial de 23.75 N, y (b) con el piezoresistivo discretizado por láser con una fuerza de 23.66 N. 30

2.18. Aplicación de fuerzas sobre pieza en forma de prisma con el calibrador triaxial. 30

2.19. Secuencia de aplicación de fuerza progresiva de 0 N a 10 N en la pieza en forma de prisma de la figura 2.18. Se emplea una lámina piezoresistiva continua. La primera fila muestra fuerza creciente y la segunda, fuerza decreciente. 30

2.20. Secuencia de aplicación de fuerza progresiva de 0 N a 10 N en la pieza en forma de prisma de la figura 2.18. Se emplea una lámina piezoresistiva discretizada por láser. La primera fila muestra fuerza creciente y la segunda, fuerza decreciente. 31

2.21. Salida de la matriz del sensor al realizar un experimento con la pieza-prisma de la figura 2.18; (a) con la capa de piezoresistivo continuo aplicando una fuerza con el calibrador triaxial de 28.54 N, y (b) con el piezoresistivo discretizado por láser con una fuerza de 28.55 N. 31

2.22. *Setup* preliminar para un test de estímulos laterales. 32

UNIVERSIDAD DE MÁLAGA

2.23. Comparación entre la magnitud del espectro de la salida de un táctel del sensor para el *cover* estructurado y el *cover* continuo, cuando un estímulo lateral se aplica, con dos velocidades de desplazamiento diferentes: (a) 3 mm/s y (b) 6.2 mm/s. 32

2.24. Palma artificial consistente en una PCB semirígida, con un elastómero continuo consistente en una goma, con módulo de Young $0.68 \cdot 10^6$ [61]. Las figuras no están a escala. 33

2.25. Disposición general del *setup* experimental adaptado de una impresora 3D. 34

2.26. Esquema de conexión de todos los elementos a la electrónica de control para el *setup* de la figura 2.25. 35

2.27. Diagrama de flujo de la ejecución en Arduino® y Processing® para el control de los motores del *setup*, y salvado de datos en fichero a través de LabVIEW®. 37

2.28. *Setup* para la exploración de texturas. 38

2.29. *Setup* para la palpación de objetos. 39

2.30. Placa de desarrollo ZedBoard™ con sus principales módulos diferenciados. 41

2.31. Negociación básica utilizando el protocolo de comunicación AXI-Stream [74]. 42

2.32. Esquema interno de una CLB, que muestra a su vez el interior de un SLICE. 43

3.1. (a) Representación de la detección de una textura por un dedo humano y su realización bionspirada [41], y (b) su propuesta de realización simplificada [42]. 51

3.2. Diagrama temporal para el cálculo de V en (3.3), para $1 \leq m \leq M$. En el diagrama, t es el tiempo. 53

UNIVERSIDAD DE MÁLAGA

3.3. Cálculo de V (3.3) normalizado con los extremos señalados para las texturas ideales: (a) $S_1(x) = sin(\frac{2\pi}{\lambda_1} \cdot x)$ y (b) $S_2(x) = sin(\frac{2\pi}{\lambda_1} \cdot x) + sin(\frac{2\pi}{\lambda_2} \cdot x) + sin(\frac{2\pi}{\lambda_3} \cdot x)$. Para $S_1(x)$, el primer mínimo indica el período espacial de la textura: $\lambda_1 = v \cdot D_{min} = 10mm/s\cdot 0.1$s $= 1$mm. Para $S_2(x)$, los tres primeros mínimos indican el período espacial correspondiente a las tres frecuencias espaciales que definen la textura: $\lambda_1 = v \cdot D_{min1} = 10mm/s\cdot 0.0725$ s $= 0.72$ mm, $\lambda_2 = v \cdot D_{min2} = 10mm/s\cdot 0.135$ s $= 1.35$ mm y $\lambda_3 = v \cdot D_{min3} = 10$mm/s $\cdot\ 0.215$ s $= 2.15$ mm. 55

3.4. (a) Cálculo de V (3.3) normalizado con singularidades proporcionadas por el algoritmo de gradiente descendente, de acuerdo con la expresión en (3.11), para una textura con un solo armónico en $\frac{1}{\lambda_1}$, donde $\lambda_1 = v \cdot D_{min} = 10\frac{mm}{s} \cdot 0.1$ s $= 1$ mm. (b) Actualización de D y gradiente de la varianza de acuerdo con la expresión en (3.10). 56

3.5. (a) Cálculo de V (3.3) normalizado con singularidades proporcionadas por el algoritmo de gradiente descendente, de acuerdo con la expresión en (3.11), para una textura con tres armónicos en $\frac{1}{\lambda_1}$, $\frac{1}{\lambda_2}$ y $\frac{1}{\lambda_3}$, donde $\lambda_1 = v \cdot D_{min1} = 10\frac{mm}{s} \cdot 0.0725$ s $= 0.72$ mm, $\lambda_2 = v \cdot D_{min2} = 10\frac{mm}{s} \cdot 0.135$ s $= 1.35$ mm y $\lambda_3 = v \cdot D_{min3} = 10\frac{mm}{s} \cdot 0.215$ s $= 2.15$ mm. (b) Actualización de D y gradiente de la varianza de acuerdo con la expresión en (3.10). 56

3.6. Muestras de texturas periódicas con período espacial λ de: (a) 13 mm, (b) 6.5 mm, (c) 3.25 mm y (d) 1.67 mm, [42]. La longitud de cada textura es de 150 mm. En la imagen, las unidades están en mm. 57

3.7. Para cada una de las texturas de la figura 3.6, se calcula V (3.3), y se representa de forma normalizada. El período espacial para cada una de ellas es de: (a) 13 mm, (b) 6.5 mm, (c) 3.25 mm y (d) 1.67 mm. Las texturas (a), (b) y (c) se exploran a una velocidad de $v = 10$ mm/s y la textura (d) se explora a una velocidad de $v = 3$ mm/s. En cada una de ellas se representa el valor para el parámetro D en la ecuación (3.3), a partir del cual se puede calcular la frecuencia espacial principal de cada una de ellas. 58

3.8. Gráfico para el flujo de señales correspondiente al sistema de segundo orden dado por la función de transferencia en (3.13), extraído de [112]. Las variables de estado se indican en la figura. 60

UNIVERSIDAD DE MÁLAGA

3.9. Respuesta al impulso del filtro *antialiasing* y coeficientes del filtro con precisión 'double' en Matlab®. 63

3.10. La figura muestra la superposición de la respuesta en magnitud del filtro *antialiasing* diseñado con precisión 'double' (Matlab®) y con precisión 'Fix16-frac15 bits', así como su respuesta en fase en radianes. Se han añadido los ejes equivalentes en π rad/muestras y frecuencia espacial mm^{-1}. 63

3.11. (a) Respuesta del filtro en frecuencia, y (b) en tiempo, para una muestra de exploración de una textura con frecuencia espacial $1/1.3$ mm^{-1}. . . . 64

3.12. (a) Respuesta del filtro IIR de Butterworth diseñado para introducir deformación en fase. (b) Respuesta en fase para el mismo filtro. 67

3.13. (a) Textura $\#5$ en la tabla 3.3, donde A es la amplitud máxima posible. (b) Textura $\#5$ una vez se le ha aplicado la transformación dada en (3.22), donde (0,A') es el rango de amplitud después de la transformación. (c) Textura $\#5$ final, una vez se le han introducido las tres componentes de ruido; de fase, ruido blanco y modulación en amplitud. 68

3.14. (a) Matriz de confusión obtenida para el caso del barrido. (b) Matriz de confusión obtenida para el caso de la FFT. 69

3.15. (a) Ejemplo de modulación progresiva en frecuencia con SNR$= 15$ dB. La longitud de onda de la señal portadora es $\lambda_c = 2$ mm para todos los casos y $m_a = 0.4$. Las longitudes de onda de la señal moduladora están indicadas en cada caso en la figura. (b) Resultado de la modulación, obtenida una vez se le añade el ruido. En todos los casos se simula una distancia de 63.3 mm con $M = 1024$, teniendo en cuenta que la velocidad de exploración se establece en $vel = 30$ mm/s y la frecuencia de muestreo es $F_s = 485$ Hz. En la figura se muestran los primeros 20 mm de la simulación. 70

3.16. (a) Matriz de confusión obtenida para barrido con 18 texturas y SNR=1 dB. (b) Matriz de confusión obtenida para FFT con 18 texturas y SNR=1 dB. (c) Matriz de confusión obtenida para barrido con 24 texturas y SNR=15 dB. (d) Matriz de confusión obtenida para FFT con 24 texturas y SNR=15 dB. 72

UNIVERSIDAD DE MÁLAGA

3.17. Fotografías de la parte superior de las 12 texturas utilizadas en los experimentos de exploración de texturas de esta tesis doctoral. 74

3.18. Texturas #TEX-2, #TEX-3 y #TEX-4 de la tabla 3.5, se representa un período de la textura #TEX-6 para las tres (parte superior de la figura) y espectro en frecuencia para con un *zoom* para los armónicos principales (parte inferior de la figura). 75

3.19. (a) Modulación lineal en frecuencia para la textura #TEX-7 de la tabla 3.5. Se representa un período espacial de la textura. (b) Espectro en frecuencia para dicha modulación con un *zoom* para los armónicos principales, que se muestran como múltiplos de $m_f = 13$. 76

3.20. Fotografía del *setup* experimental con las diferentes partes usadas para explorar las texturas. La placa de desarrollo ZedBoard™ (descrita en la sección 3.8) con sus puertos de entrada/salida y el núcleo basado en un SoC Zynq-7000® también se indican en la imagen. 77

3.21. Secuencia de exploración. (a) Posición inicial. (b) Movimiento de desplazamiento lateral a $v = 30$ mm/s. (c) Posición final. 78

3.22. Salida digital del sensor táctil para cada una de las texturas del conjunto de la figura 3.17, cuando se exploran a una velocidad de $v = 30$ mm/s. . 79

3.23. Salida de V (3.3) para $1 \leq m \leq M$ ($D = m \cdot T_s$) para el conjunto de texturas de la figura 3.17. 80

3.24. (a) Salida digital del sensor táctil tomada a $v = 30$ mm/s para la superficie de control en metacrilato liso. (b) Salida de V (3.3) con $1 \leq m \leq M$ ($D = m \cdot T_s$) para esta superficie. 81

3.25. Salida de V (3.3) para $1 \leq m \leq M$ ($D = m \cdot T_s$), para la textura #TEX-1 del conjunto de texturas de la figura 3.17. Adicionalmente, una representación del vector de la localización de los mínimos se muestra en la parte superior de la figura. 81

3.26. Salida de la aplicación de la FFT obtenida para el conjunto de texturas de la figura 3.17. Se ha hecho un *zoom* en los primeros valores del espectro. 82

UNIVERSIDAD DE MÁLAGA

3.27. Representación del valor de los índices de Goertzel, para $K = 16$, considerando el cálculo de los mismos de forma: (a) equiespaciada, (b) con $p = 1$, (c) con $p = 10$, (d) con $p = 25$, (e) con $p = 50$ y (f) con $p = 100$. 85

3.28. Procedimiento para la obtención de los conjuntos de entrenamiento y test, para los algoritmos de *sweep* y FFT, respectivamente. 86

3.29. Ejemplo de inspección visual de tramas para la textura #TEX-9 en la figura 3.17. En la imagen, las exploraciones 27, 147 y 151 se descartaron. 88

3.30. Representación para dimensión $n = 3$ de la distancia d entre el centroide del conjunto $\{C\}$ y el vector de referencia unitario $\boldsymbol{u_{ref}}$, y de las distancias d_1, d_2 y d_3 entre los *outliers* p_1, p_2 y p_3 y el vector $\boldsymbol{u_{ref}}$. Los ejes están representados como γ_1, γ_2 y γ_3. 89

3.31. Cálculo de la distancia euclídea entre cada entrada i de la matriz de características **MC** y un vector de referencia unitario $\boldsymbol{u_{ref}}$, que da como resultado el vector de distancias $\boldsymbol{v_d}$. 89

3.32. Diagrama de tipo *rain plot* que muestra la distribución (para el conjunto de entrenamiento) de todos los valores de $\boldsymbol{v_d}$ para cada clase, junto a su media y desviación típica. Las distribuciones se muestran para el algoritmo de *sweep*. 90

3.33. Diagrama de flujo simplificado para el proceso de entrenamiento y obtención de centroides óptimos con *k-means*. En la figura, los índices i y j recorren bucles hasta alcanzar el número de iteraciones total ($nIter$) del procedimiento, y el número de épocas de entrenamiento ($nEpocas$). 91

3.34. Pseudocódigo para el algoritmo de clasificación *k-means* en tiempo real ante nuevas entradas de características. 92

3.35. Representación de las distancia entre todos los valores del conjunto de entrenamiento de cada clase con respecto al centroide de la clase para la textura: (a) #TEX-1, (b) #TEX-2, (c) #TEX-3, (d) #TEX-4, (e) #TEX-5, (f) #TEX-6, (g) #TEX-7, (h) #TEX-8, (i) #TEX-9, (j) #TEX-10, (k) #TEX-11 y (l) #TEX-12. 93

UNIVERSIDAD DE MÁLAGA

3.36. Detalle del cálculo de V (3.3) para la textura #TEX-4 de la figura 3.17. Se muestra el rango de $D = 0$ s a $D = 1$ s. Como se puede ver en el *zoom* aplicado, aparecen algunos picos de baja amplitud debidos a la interacción entre el sensor y la superficie. 96

3.37. Diezmado sobre el cálculo de V (3.3) mostrado en la figura 3.36 para la textura #TEX-4. Se aplica un factor de diezmado de: (a) $d = 2$, (b) $d = 4$, (c) $d = 8$. Se puede ver que la localización del mínimo mejora conforme el diezmado es mayor. 96

3.38. Diagrama general de la arquitectura en la Zynq-7000® XC7Z020 para el algoritmo de barrido *sweep*, FFT, o algoritmo de Goertzel. Cada módulo está representado mediante líneas discontinuas y equivale al IP Core diseñado o utilizado para cada algoritmo. 101

3.39. Esquema básico de comunicación mediante el protocolo bus serie SPI. . 102

3.40. Pseudocódigo para el algoritmo de barrido o *sweep*, que se aplica para cada nueva muestra de entrada. 106

3.41. Modelado hardware aproximado para el cálculo de V (3.3) en tiempo real. 107

3.42. (a) Pseudocódigo para la opción de aplicación de un filtro de media móvil al cálculo de V, expresión (3.3). (b) Pseudocódigo para el cálculo de las posición de mínimos en V. 109

3.43. Comparación para el número de LUTs utilizadas como lógica o LUT Logic entre seis configuraciones del algoritmo de barrido, y la FFT. . . 111

3.44. Comparación para el número de elementos lógicos de acarreo (CARRY Logic) entre seis configuraciones del algoritmo de barrido, y la FFT. . . 111

3.45. Comparación para el número de Flip-Flops o registros utilizados entre seis configuraciones del algoritmo de barrido, y la FFT. 111

3.46. Comparación para la memoria distribuida o LUTRAM entre seis configuraciones del algoritmo de barrido, y la FFT. 112

3.47. Comparación para los bloques BRAM empotrados entre seis configuraciones del algoritmo de barrido, y la FFT. 112

UNIVERSIDAD DE MÁLAGA

3.48. Comparación para la suma de memoria distribuida (LUTRAM) y bloques empotrados BRAM en kb, entre seis configuraciones del algoritmo de barrido, y la FFT. 112

3.49. Comparación para los bloques DSP empotrados entre seis configuraciones del algoritmo de barrido, y la FFT. 113

3.50. Comparación del consumo de potencia en mW entre seis configuraciones del algoritmo de barrido, y la FFT. 113

3.51. Comparación del tiempo de extracción de características t_{fe} (o latencia) entre seis configuraciones del algoritmo de barrido, y la FFT. 113

3.52. Pseudocódigo correspondiente al algoritmo de Goertzel. 115

3.53. Consumo global 'Post-Implementation' para la implementación del *sweep* en SoC, en porcentajes de utilización respecto al número total de recursos que proporciona el dispositivo Zynq-7020®. (a) Consumo global de recursos hardware. (b) Consumo de potencia en hardware para las diferentes partes del diseño. 118

3.54. Diagrama de flujo resumido del código en C implementado en el ARM® a partir del entorno de desarrollo SDK. En el diagrama se muestra la ejecución de los tres hilos o *threads* utilizados y la ISR correspondiente al módulo DMA. 121

3.55. Porcentajes de clasificación para el algoritmo de *sweep* y FFT, dependiendo del número de componentes del vector de características que se envían al clasificador, y tamaño en kB del vector de características para ambos casos [39]. La longitud del vector de salida es $M = 1024$. 123

3.56. Matrices de confusión con los resultados del algoritmo de *sweep* para las texturas mostradas en la figura 3.17 y cada punto del gráfico 3.55, con $M = 1024$. Estos resultados se han extraído directamente a partir de un test en tiempo real en el chip Zynq-7020®. 124

3.57. Matrices de confusión con los resultados de la FFT para las texturas mostradas en la figura 3.17 y cada punto del gráfico 3.55, con $M = 1024$. Estos resultados se han extraído a partir de las funciones pre-compiladas MEX de Matlab®. 125

UNIVERSIDAD DE MÁLAGA

3.58. Representación en el espacio ROC para cada una de las 12 texturas, cuando se aplica el algoritmo de *sweep*. En cada gráfico, el eje vertical representa la sensibilidad (la proporción de verdaderos positivos), y el eje horizontal muestra la tasa de falsos positivos (1-especificidad). Un modelo perfecto se ubicaría en la esquina superior izquierda del gráfico. 127

3.59. Matriz de confusión cuando se aplica el algoritmo de Goertzel con $N = 1024$, $K = 16$ y $p = 10$ sobre las 12 texturas de la figura 3.17. 129

4.1. Sección del diseño de las nueve piezas preliminares [43]: (a) cubo de densidad baja, (b) cubo de densidad media, (c) cubo de densidad alta, (d) cilindro de densidad baja, (e) cilindro de densidad media, (f) cilindro de densidad alta, (g) esfera de densidad baja, (h) esfera de densidad media y (i) esfera de densidad alta. 138

4.2. Características propuestas para el momento $M_{0,0}$ y el objeto #OBJ-2 (berenjena): (a) área bajo la curva (AR), (b) valor máximo de la curva (MAX), (c) valor en magnitud a $2/3$ del tiempo para alcanzar el máximo en la parte ascendente de la curva ($H1$) y (d) valor en magnitud a $1/3$ del mismo tiempo después de que el máximo sea alcanzado ($H2$). . . . 140

4.3. Diagramas de cajas y bigotes para las diferentes características propuestas, para: (a) el sensor correspondiente al dedo artificial y (b) el sensor correspondiente a la palma. En ambos casos, la representación está hecha para el momento $M_{0,0}$. 141

4.4. Fotografía del *setup* experimental para la exploración de objetos con distinta rigidez. 144

4.5. Conjunto de las 42 clases de objetos utilizadas para llevar a cabo las exploraciones de palpación, ver también tabla 4.1. 145

4.6. Ilustración de la secuencia de exploración para la clase $\#OBJ$-17. . . . 146

4.7. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (a) y (c), y con la palma (b) y (d), calculadas a partir de la exploración de las clases de objetos #OBJ-17 y #OBJ-38. 146

UNIVERSIDAD DE MÁLAGA

4.8. Arquitectura general de la implementación en el SoC Zynq-7000 XC7Z020®. La parte con línea discontinua en color anaranjado sólo se implementa si se aplica PCA, [40]. 148

4.9. Máquina de estados para el módulo 'VHDL Synchronization Modules' de la figura 4.8. En cada transición de estados, el formato indicado es: $t_{validOut1}$,$t_{validOut2}$/$t_{validOut}$, donde $t_{validOut1}$ y $t_{validOut2}$ son las entradas al módulo, que se corresponden con las señales de dato válido del módulo 'VHDL Synchronization Modules', y $t_{validOut}$ es la señal de salida. . 149

4.10. Ejemplo de sincronización de las señales de dato válido $t_{validOutDedo0}$, $t_{validOutDedo1}$, $t_{validOutPalma0}$. Las señales se midieron con el osciloscopio MDO4104B-6 Mixed Domain Tektronix®. Los cursores de medida están colocados en los umbrales de tensión ascendente o descendente, respectivamente. 150

4.11. Configuración en cascada para el módulo 'VHDL Synchronization Modules' de la figura 4.8, para la sincronización de N señales de entrada. . 151

4.12. Propuesta de implementación para el módulo 'VHDL Features Computing Module' de la figura 4.8, [40]. 152

4.13. Máquina de estados para el módulo 'VHDL Features Computing Module' en la figura 4.8. 153

4.14. Medida con el osciloscopio MDO4104B-6 Mixed Domain Tektronix® del conjunto de los siguientes conjuntos de señales: (a) $t_{validOut}$, $t_{validOutDedo0}$, $t_{validInDedo0}$, $t_{validInPalma0}$ y (b) $t_{validOut}$, t_{ready}, $t_{validOutDedo0}$, $t_{validInPalma0}$. En cada caso, los cursores de medida están colocados en los umbrales de tensión ascendente o descendente, respectivamente. 154

4.15. Pseudocódigo para el módulo *HLS Principal Component Analysis (PCA) Computing Module* en la figura 4.8. 156

4.16. Histogramas de distancias con el correspondiente porcentaje de clasificación para los nueve casos de la tabla 4.4. En la figura, VC significa "vectores de características". 159

4.17. Resultado de la figura de mérito en (4.4) normalizada respecto al mejor caso, cuando no se aplica PCA. 161

UNIVERSIDAD DE MÁLAGA

4.18. Matriz de confusión para la implementación óptima de la figura 4.17. Este caso 'Dedo y palma (nMom1-12bits/caract-d5)' (segunda fila de la tabla 4.4) utiliza el primer momento de la imagen $M_{0,0}$ para ambos sensores (dedo y palma), la característica d5 (ver tabla 4.2) y 12 bits por característica. 161

4.19. Representación en el espacio ROC para cada una de las 42 clases de objetos, cuando no se aplica PCA. En cada gráfico, el eje vertical representa la sensibilidad (la proporción de verdaderos positivos), y el eje horizontal muestra la tasa de falsos positivos ($1-$especificidad). Un modelo perfecto se ubicaría en la esquina superior izquierda del gráfico. . . 162

4.20. Comparación de la distribución de los datos en un espacio de características de 2 componentes principales para: (a) dedo, (b) palma, (c) dedo y palma. En todos los casos, el número de momentos es 6, la característica utilizada es d8 de la tabla 4.2, y el número de bits por característica es 16. En la figura, PC1 representa la componente principal 1 y PC2 representa la componente principal 2. La clase correspondiente a cada centroide se incluye al lado del mismo. 164

4.21. Comparación de la distribución de los datos en un espacio de características de 3 componentes principales para: (a) dedo, (b) palma, (c) dedo y palma. En todos los casos, el número de momentos es 6, la característica utilizada es d5 de la tabla 4.2, y el número de bits por característica es 8. En la figura, PC1 representa la componente principal 1, PC2 representa la componente principal 2, y PC3 representa la componente principal 3. 165

4.22. Resultado de la figura de mérito en (4.4) normalizado con respecto al mejor caso en la figura 4.17, cuando se aplica PCA. 166

4.23. Matriz de confusión para la implementación óptima de la figura 4.22. Este caso 'Dedo y palma, nMom1 (d5), nBits/caract 12, nC3' (segunda fila de la tabla 4.6) utiliza el primer momento de la imagen $M_{0,0}$ para ambos sensores (dedo y palma), la característica d5 (ver tabla 4.2), 12 bits por característica y 3 componentes principales. 167

UNIVERSIDAD DE MÁLAGA

4.24. Representación en el espacio ROC para cada una de las 42 clases de objetos, cuando se aplica la técnica de PCA. En cada gráfico, el eje vertical representa la sensibilidad (la proporción de verdaderos positivos), y el eje horizontal muestra la tasa de falsos positivos ($1-$especificidad). Un modelo perfecto se ubicaría en la esquina superior izquierda del gráfico. 167

4.25. Tiempo correspondiente al retardo de extracción de características más el retardo de clasificación, frente a la dimensión del vector de características de entrada del clasificador. Los resultados corresponden a los casos en la tabla 4.4 y 4.6, para las implementaciones sin PCA y con PCA, respectivamente. 169

4.26. Medida con osciloscopio de tiempos de retardo de extracción de características más el retardo de clasificación, frente a la dimensión del vector de características de entrada del clasificador, para dos casos de la figura 4.25: (a) 81.90 μs y (b) 101 μs. 170

A.1. Imagen de la tarjeta de adquisición 6009 de National Instruments® (capítulo 2). 177

A.2. Esquema de conexión entre la célula de carga y el conversor HX711 [68] (capítulo 2). 177

A.3. (a) Fotografía del calibrador triaxial utilizado. Se pueden ver los motores (T-NA08A50 Zaber, Vancouver, BC, Canadá) para los tres ejes de movimiento. El motor Z mueve el émbolo para que el sensor de referencia presione el táctel sobre la plataforma de experimentación. (b) Aplicación de fuerza sobre un táctel del sensor de dedo (capítulo 2). 178

A.4. Representación de un período espacial para la textura #TEX-1 de la tabla 3.5 (parte superior de la figura) y espectro en frecuencia con un *zoom* para los armónicos principales (parte inferior de la figura), (capítulo 3). 179

A.5. (a) Modulación lineal en frecuencia para la textura #TEX-5 de la tabla 3.5. En la figura . Se representa un período espacial de la textura. (b) Espectro en frecuencia para dicha modulación con un *zoom* para los armónicos principales, que se muestran como múltiplos de $m_f = 7$ (capítulo 3). 180

UNIVERSIDAD DE MÁLAGA

A.6. (a) Modulación lineal en frecuencia para la textura #TEX-6 de la tabla 3.5. Se representa un período espacial de la textura. (b) Espectro en frecuencia para dicha modulación con un *zoom* para los armónicos principales (capítulo 3). 181

A.7. Representación de un período espacial para la textura #TEX-8 de la tabla 3.5 (parte superior de la figura) y espectro en frecuencia con un *zoom* para los armónicos principales (parte inferior de la figura), (capítulo 3). 182

A.8. Representación de un período espacial para la textura #TEX-9 de la tabla 3.5 (parte superior de la figura) y espectro en frecuencia con un *zoom* para los armónicos principales, típicos de una onda cuadrada (parte inferior de la figura). En la imagen, F indica armónico fundamental y A_F amplitud del armónico fundamental (capítulo 3). 182

A.9. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-1, #OBJ-2, #OBJ-3, #OBJ-4, #OBJ-5, #OBJ-6 y #OBJ-7 (capítulo 4). 183

A.10. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-8, #OBJ-9, #OBJ-10, #OBJ-11, #OBJ-12, #OBJ-13 y #OBJ-14 (capítulo 4). 184

A.11. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-15, #OBJ-16, #OBJ-17, #OBJ-18, #OBJ-19, #OBJ-20 y #OBJ-21 (capítulo 4). 185

A.12. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-22, #OBJ-23, #OBJ-24, #OBJ-25, #OBJ-26, #OBJ-27 y #OBJ-28 (capítulo 4). 186

UNIVERSIDAD DE MÁLAGA

A.13. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-29, #OBJ-30, #OBJ-31, #OBJ-32, #OBJ-33, #OBJ-34 y #OBJ-35 (capítulo 4). 187

A.14. Secuencias temporales obtenidas en una determinada iteración, para los seis primeros momentos de la imagen táctil ($M_{p,q}$), con el dedo (primera columna), y con la palma (segunda columna), calculadas a partir de la exploración de las clases de objetos #OBJ-36, #OBJ-37, #OBJ-38, #OBJ-39, #OBJ-40, #OBJ-41 y #OBJ-42 (capítulo 4). 188

C.1. Dimensiones mecánicas del motor NEMA 17 utilizado [66]. La longitud nominal máxima (L Max) del motor es 40 mm. Los datos de longitud están en mm (capítulo 2). 193

C.2. Detalle del motor NEMA 17 colocado en la impresora 3D que se utilizó como banco de pruebas. 194

C.3. (a) Medidas de $10k$ puntos del espectro en frecuencia, para un desplazamiento a 30mm/s y *zoom* para una división por cuadrícula de 50 Hz ($1kS/s$), y (b) para un desplazamiento a 200mm/s con *zoom* para una división por cuadrícula de 250 Hz ($5kS/s$). Las medidas se realizaron con un osciloscopio MDO4104B-6 Mixed Domain Tektronix® de 6 GHz. 196

C.4. Electrónica del *driver* Pololu A4988 de acuerdo a su datasheet [65]. En la figura, se pueden ver destacados un regulador de corriente y los pines de entrada MS1, MS2 y MS3 para la funcionalidad de *microsteeping* (capítulo 2). 197

C.5. Conexiones de los pines del *driver* [129]. 197

D.1. Diferentes partes del *setup* utilizado para la discretización de la lámina piezoresistiva mediante la máquina de grabado y corte por láser. 200

D.2. Detalle de la bomba de agua para el sistema de refrigeración del láser. En la fotografía, la bomba se encontraba en funcionamiento. 201

D.3. Fotografía del láser de CO_2 de 40 W que utiliza la máquina de corte por láser, en el interior de la carcasa trasera de la misma. 201

UNIVERSIDAD DE MÁLAGA

D.4. Ventilador y tubo de extracción. 202

D.5. Panel de control manual de la impresora láser. 203

D.6. Espacio de trabajo de la máquina de corte por láser. 203

D.7. Área de trabajo y edición para CorelDRAW®. En este caso se muestra el diseño que se utilizará para grabado sobre la lámina piezoresistiva. . . . 204

D.8. Dispositivo USB para validar la licencia de uso de la máquina de corte por láser. 204

D.9. Cubierta protectora, para dispersar la radiación emitida por el láser. . . . 205

D.10. Detalle del espejo con tornillos que permiten modificar el ángulo de incidencia del láser. 206

D.11. Referencias para el grabado de la lámina piezoresistiva. 206

D.12. Preparación de la huella. 207

D.13. Área de diseño en CorelDRAW® donde se ve el diseño para la discretización láser de la lámina piezoresistiva. 208

D.14. (a) Ajustes generales de la impresora láser. (b) Ajustes específicos para el grabado y corte por láser. 209

D.15. Resultados de discretización por láser de la lámina piezoresistiva. 210

D.16. (a) Comparativa de la parte anterior del *cover* estructurado, antes y después de adherir la lámina piezoresistiva discretizada por láser. (b) Comparativa de la parte posterior del *cover* estructurado, antes y después de adherir la lámina piezoresistiva discretizada por láser. 211

D.17. (a) Fotografía superior del montaje final en el sensor de dedo artificial. (b) Fotografía lateral del montaje final en el sensor de dedo artificial. . . 212

E.1. Diagrama general de la arquitectura del sistema en Vivado®para el algoritmo de barrido. 214

E.2. Grupo de jerarquía que contiene al algoritmo de barrido en HLS. 215

UNIVERSIDAD DE MÁLAGA

E.3. Grupo de jerarquía que contiene el *buffer* de desplazamiento y el IP Core para la FFT. 215

E.4. Grupo de jerarquía que contiene el módulo en HLS para el algoritmo de Goertzel. 216

E.5. Grupo de jerarquía que contiene el control DMA y la FIFO de salida. . . 217

E.6. Diagrama general de la arquitectura del sistema en Vivado® para el reconocimiento de objetos por rigidez. 218

E.7. Grupo de jerarquía que contiene el cálculo de características para el dedo, en el reconocimiento de la rigidez de objetos. 219

E.8. Grupo de jerarquía que contiene el cálculo de características para la palma, en el reconocimiento de la rigidez de objetos. 219

E.9. Grupo de jerarquía para la interfaz serie a paralelo. 220

E.10. Grupo de jerarquía para la interfaz paralelo a serie. 220

E.11. Sincronización de las señales de *t_ready*. 221

E.12. Sincronización de las señales de *t_valid*. 221

F.1. Diagrama conceptual para la simulación. 224

F.2. Captura de la simulación durante 60 ms. 225

F.3. *Zoom* en la parte de los datos de entrada al algoritmo, cuando la señal 'CURRENT_FRAME=0'. 226

F.4. *Zoom* en la parte de salida del algoritmo de barrido, cuando la señal 'CURRENT_FRAME=0'. 228

F.5. Simulación completa de $20\mu s$ para el cálculo de las características AR, MAX, $H1$ y $H2$ propuestas en el capítulo 4. En la figura, VC_out[31:0] se corresponde con el vector de características a la salida del algoritmo de extracción de características. 229

F.6. *Zoom* en el instante en el que se proporciona la salida del vector de características, como una palabra de 32 bits, durante el ciclo de reloj en el que la señal de *t_valid_out* se encuentra activa. 230

UNIVERSIDAD DE MÁLAGA

Índice de tablas

2.1. Principales valores nominales del motor NEMA 17 utilizado [66]. . . . 35

3.1. Especificaciones en frecuencia del filtro *antialiasing*. 62

3.2. Especificaciones de ganancia en dB del filtro *antialiasing*. 62

3.3. Conjunto preliminar de texturas ideales para llevar a cabo la simulación inicial 65

3.4. Resultados de la simulación con modulación en frecuencia progresiva cuando se suma ruido blanco (se indica mediante la relación SNR), que indican el porcentaje de clasificación correcta en cada caso. 71

3.5. Principales propiedades de las doce texturas empleadas en este trabajo de tesis. 74

3.6. Porcentajes de clasificación con *k-means* en Matlab® para el envío al clasificador de las posiciones de los mínimos, máximos, o mínimos y máximos conjuntamente. En todos los casos se proporciona la desviación típica entre los mejores casos de clasificación. Los datos proporcionados son para $M = 1024$ e incluyen la clasificación de *outliers* en el conjunto de test (ver sección 3.6). 95

UNIVERSIDAD DE MÁLAGA

3.7. Porcentajes de clasificación obtenidos utilizando el algoritmo *k-means* en Matlab® al enviar las posiciones de los mínimos, máximos o de los mínimos y máximos conjuntamente como características de entrada al clasificador. Además, se utilizan distintos factores de diezmado d, indicando entre paréntesis el número de componentes del vector de características correspondiente (comp. VC). En todos los casos se proporciona la desviación típica de los mejores resultados de clasificación. Los datos proporcionados incluyen la clasificación de *outliers*. 97

3.8. Porcentajes de clasificación obtenidos utilizando el algoritmo *k-means* en Matlab® al enviar los coeficientes de la FFT como características de entrada al clasificador. En todos los casos se proporciona la desviación típica de los mejores resultados de clasificación, indicando entre paréntesis el número de componentes del vector de características correspondiente (comp. VC). Los datos proporcionados incluyen la clasificación de *outliers*. 98

3.9. Porcentajes de clasificación con *k-means* cuando se considera el algoritmo de Goertzel propuesto para $N = 1024$, para diferentes valores del número de índices K y del parámetro de control p en (3.26). En todos los casos se proporciona el valor máximo de clasificación encontrado. Los datos proporcionados incluyen la clasificación de *outliers* en el conjunto de test (consultar sección 3.6). 99

3.10. Estimación de recursos hardware y latencia para diferentes combinaciones de las directivas de optimización DATAFLOW, PIPELINE y UNROLL. 108

3.11. Datos del consumo real de recursos lógicos y de memoria en hardware, así como el consumo de potencia y latencia, para la implementación del algoritmo de Goertzel con $N = 1024$, con $K = 4$, $K = 8$ y $K = 16$, y un factor de control $p = 10$. Para poder comparar con barrido y FFT también se incluye el consumo de ambos para $M = 1024$, así como los porcentajes de clasificación para cada caso. Estos datos se recogen tal y como los proporciona el informe de utilización de Vivado IDE®. 117

3.12. Datos del consumo real de recursos lógicos y de memoria para la implementación de los diferentes módulos hardware en el diseño global de la figura 3.38, tal y como los proporciona el informe de utilización o 'Utilization Report' de Vivado IDE®. 117

UNIVERSIDAD DE MÁLAGA

4.1. Tabla con las 42 clases empleadas en este trabajo, ver también figura 4.5. 143

4.2. Etiquetas para la combinación de características: área bajo la curva (AR), valor máximo (MAX), valor de magnitud en la parte ascendente a los $2/3$ del tiempo hasta alcanzar el máximo ($H1$) y valor de magnitud a $1/3$ de ese mismo tiempo después de alcanzar el máximo ($H2$). 157

4.3. Porcentajes de clasificación sin PCA. 157

4.4. Datos de rendimiento para la implementación de la extracción de características sin PCA (módulo 'VHDL Features Computing Module' en la figura 4.8), tal y como se proporciona de los informes de utilización de Vivado IDE®. Las dos últimas columnas indican el retardo de entrada-salida del módulo (en ns), y su consumo de potencia (en mW). 158

4.5. Porcentajes de clasificación con PCA. En la tabla, nCm es el número de componentes principales. 163

4.6. Datos de rendimiento para la implementación de la extracción de características con PCA (módulo 'VHDL Features Computing Module' más 'HLS Principal Component Analysis (PCA) Computing Module' en la figura 4.8), tal y como se proporciona de los informes de utilización de Vivado IDE®. Las columnas de la tabla son las mismas que la de la tabla 4.4. 166

B.1. Principales características para las partes PS y PL del dispositivo SoC Zynq™ XC7Z020-CLG484-1 [75] (capítulo 2). 190

B.2. Valores correspondientes al período espacial mínimo sin tomar en cuenta el filtro antialiasing λ_{min}, expresión (3.9) del capítulo 3, y considerándolo $\lambda_{minFilter}$, expresión (3.23) del capítulo 3. Estos valores se calculan para diferentes velocidades v (capítulo 3). 191

B.3. Período espacial máximo λ_{max} en mm, que se obtiene para diferentes valores de M y de la velocidad v, de acuerdo a (3.8), sección 3.2, (capítulo 3). 191

C.1. Valores lógicos en los pines para los modos de configuración de *microsteeping*, para el *driver* Pololu A4988. 197

UNIVERSIDAD DE MÁLAGA

D.1. Principales características de la máquina de grabado y corte por láser. . 200

E.1. Opciones de configuración *Kernel Behavior* para FreeRTOS®. 222

E.2. Opciones de configuración para la pila lwIP. 222

UNIVERSIDAD DE MÁLAGA



Bibliografía

[1] R. S. Dahiya and M. Valle, *Robotic Tactile Sensing - Technologies and System*. Springer, 2013, ISBN: 978-94-007-0578-4. [Online]. Available: https://link.springer.com/book/10.1007/978-94-007-0579-1

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

[3] Y. Al-Handarish, O. M. Omisore, T. Igbe, S. Han, H. Li, W. Du, J. Zhang, and L. Wang, "A Survey of Tactile-Sensing Systems and Their Applications in Biomedical Engineering," *Advances in Materials Science and Engineering*, vol. 2020, 2020, ISSN: 16878442. [Online]. Available: http://dx.doi.org/10.1155/2020/4047937

[4] P. Peng and R. Rajamani, "Flexible microtactile sensor for normal and shear elasticity measurements," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 4907–4913, 2012. [Online]. Available: https://doi.org/10.1109/TIE.2011.2172175

[5] S. Pyo, J. Lee, K. Bae, S. Sim, and J. Kim, "Recent Progress in Flexible Tactile Sensors for Human-Interactive Systems: From Sensors to Advanced Applications," *Advanced Materials*, vol. 33, no. 47, pp. 1–26, 2021, ISSN: 15214095. [Online]. Available: https://doi.org/10.1002/adma.202005902

[6] R. S. Dahiya, G. Metta, M. Valle, and G. Sandini, "Tactile Sensing — From Humans to Humanoids," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 1–20, 2010. [Online]. Available: https://doi.org/10.1109/TRO.2009.2033627

UNIVERSIDAD DE MÁLAGA

[7] M. A. Abd, R. Paul, A. Aravelli, O. Bai, L. Lagos, M. Lin, and E. D. Engeberg, "Hierarchical tactile sensation integration from prosthetic fingertips enables multi-texture surface recognition†," *Sensors*, vol. 21, no. 13, 2021, ISSN: 14248220. [Online]. Available: https://doi.org/10.3390/s21134324

[8] I. N. de Estadística, "Tasa de población con discapacidad que tiene diagnosticadas determinadas enfermedades crónicas según la enfermedad por CCAA y sexo," Accedido el 23.04.2023. [Online]. Available: https://www.ine.es/jaxi/Tabla.htm?path=/t15/p418/a2008/hogares/p02/modulo1/l0/&file=04028.px&L=0

[9] M. S. Keszler, J. T. Heckman, G. E. Kaufman, and D. C. Morgenroth, "Advances in Prosthetics and Rehabilitation of Individuals with Limb Loss," *Physical Medicine and Rehabilitation Clinics of North America*, vol. 30, no. 2, pp. 423–437, 2019, ISSN: 15581381. [Online]. Available: https://doi.org/10.1016/j.pmr.2018.12.013

[10] 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," *Annals of Biomedical Engineering*, vol. 49, no. 1, pp. 57–74, 2021, ISSN: 15739686. [Online]. Available: http://dx.doi.org/10.1007/s10439-020-02678-8

[11] M. Hays, L. Osborn, R. Ghosh, M. Iskarous, C. Hunt, and N. V. Thakor, "Neuromorphic vision and tactile fusion for upper limb prosthesis control," *International IEEE/EMBS Conference on Neural Engineering, NER*, vol. 2019-March, pp. 981–984, 2019, ISBN: 9781538679210, ISSN: 19483554. [Online]. Available: https://doi.org/10.1109/NER.2019.8716890

[12] M. Rodríguez, "Científicos de FAU avanzan en prótesis con dedos que sienten las texturas," Accedido el 23.04.2023. [Online]. Available: https://elsolnewsmedia.com/cientificos-de-fau-avanzan-en-protesis-con-dedos-que-sienten-las-texturas/

[13] Z. Xia, Z. Deng, B. Fang, Y. Yang, and F. Sun, "A review on sensory perception for dexterous robotic manipulation," *International Journal of Advanced Robotic Systems*, vol. 19, no. 2, pp. 1–18, 2022, DOI: 10.1177/17298806221095974. [Online]. Available: https://doi.org/10.1177/17298806221095974

[14] H. Kawasaki, T. Komatsu, and K. Uchiyama, "Dexterous anthropomorphic robot hand with distributed tactile sensor: Gifu hand II," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 3, pp. 296–303, 2002, ISBN: 0780357310, ISSN: 10834435. [Online]. Available: https://doi.org/10.1109/TMECH.2002.802720

UNIVERSIDAD DE MÁLAGA

[15] Y. Zhang, Y. Liu, M. Jin, and H. Liu, "Design of a Finger-Tip Flexible Tactile Sensor for an Anthropomorphic Robot Hand," in *Intelligent Robotics and Applications*, ser. Lecture Notes in Computer Science, H. Liu, H. Ding, Z. Xiong, and X. Zhu, Eds., vol. 6424. Berlin, Heidelberg: Springer, 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-16584-9_72

[16] K. Kaneko, K. Harada, and F. Kanehiro, "Development of multi-fingered hand for life-size humanoid robots," *Proceedings - IEEE International Conference on Robotics and Automation*, no. April, pp. 913–920, 2007, ISBN: 1424406021, ISSN: 10504729. [Online]. Available: https://doi.org/10.1109/ROBOT.2007.363102

[17] B. Fang, Y. Chen, F. Sun, D. Yang, X. Zhang, Z. Xia, and H. Liu, "A petal-array capacitive tactile sensor with micro-pin for robotic fingertip sensing," *2020 3rd IEEE International Conference on Soft Robotics, RoboSoft 2020*, pp. 452–457, 2020, ISBN: 9781728165707. [Online]. Available: https://doi.org/10.1109/RoboSoft48309.2020.9116047

[18] N. Jamali and C. Sammut, "Majority Voting: Material Classification by Tactile Sensing Using Surface Texture," *IEEE Transactions on Robotics*, vol. 27, no. 3, pp. 508–521, 2011, ISBN: 1552-3098, ISSN: 1552-3098. [Online]. Available: http://dx.doi.org/10.1109/TRO.2011.2127110

[19] M. Lambeta and et al., "DIGIT: A Novel Design for a Low-Cost Compact High-Resolution Tactile Sensor With Application to In-Hand Manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3838–3845, July 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2005.14679

[20] C. M. Boutry, M. Negre, M. Jorda, O. Vardoulis, A. Chortos, O. Khatib, and Z. Bao, "A hierarchically patterned , bioinspired e-skin able to detect the direction of applied pressure for robotics," *ScienceRobotics*, vol. 6914, no. November, pp. 1–10, 2018. [Online]. Available: https://doi.org/10.1126/scirobotics.aau6914

[21] B. Oh, Y.-G. Park, H. Jung, S. Ji, W. Hyung Cheong, J. Cheon, W. Lee, and P. Jang-Ung, "Untethered Soft Robotics with Fully Integrated Wireless Sensing and Actuating Systems for Somatosensory," *Soft Robotics*, vol. 00, no. 00, pp. 1–10, 2020. [Online]. Available: https://doi.org/10.1089/soro.2019.0066

[22] C. M. Oddo, L. Beccai, M. Felder, F. Giovacchini, and M. C. Carrozza, "Artificial roughness encoding with a bio-inspired MEMS- Based tactile sensor array," *Sensors*, vol. 9, no. 5, pp. 3161–3183, 2009, ISSN: 14248220. [Online]. Available: https://doi.org/10.3390/s90503161

UNIVERSIDAD DE MÁLAGA

[23] N. Wettels, V. J. Santos, R. S. Johansson, and E. Loeb, "Biomimetic Tactile Sensor Array," *Advanced Robotics*, no. December 2014, pp. 37–41, 2012. [Online]. Available: https://doi.org/10.1163/156855308X314533

[24] X. Hui, L. Hongbin, L. Shan, S. Lakmal D., and A. Kaspar, "Fiber Optics Tactile Array Probe for Tissue Palpation during Minimally Invasive Surgery," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan*, vol. 2013-November, 2013. [Online]. Available: https://doi.org/10.1109/IROS.2013.6696714

[25] R. Dahiya, P. Mittendorfer, M. Valle, G. Cheng, and V. Lumelsky, "Directions toward effective utilization of tactile skin: A review," *IEEE Sensors Journal*, vol. 13, no. 11, pp. 4121–4138, 2013. [Online]. Available: http://dx.doi.org/10.1109/JSEN.2013.2279056

[26] B. Ward-Cherrier, L. Cramphorn, and N. F. Lepora, "Tactile Manipulation With a TacThumb Integrated on the Open-Hand M2 Gripper," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 169–175, 2016, ISSN: 23773766. [Online]. Available: https://doi.org/10.1109/LRA.2016.2514420

[27] M. Kaboli and G. Cheng, "Robust Tactile Descriptors for Discriminating Objects from Textural Properties via Artificial Robotic Skin," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 985–1003, 2018. [Online]. Available: http://dx.doi.org/10.1109/TRO.2018.2830364

[28] R. Kageyama, S. Kagami, M. Inaba, and H. Inoue, "Development of Soft and Distributed Tactile Sensors and the Application to a Humanoid Robot," in *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, 1999. [Online]. Available: https://doi.org/10.1109/ICSMC.1999.825396

[29] P. P. Systems, "RoboTouchTM Tactile Sensors with Digital Output for the Barrett Hand." [Online]. Available: http://static1.squarespace.com/static/53836bf1e4b011aa8ac0ffb9/t/53c85b3ae4b052122aacf107/1405639482642/PPS+_RoboTouch_SpecSheet.pdf

[30] SCHUNK, "SCHUNK Modular and Mobile Gripping Systems Brochure." Accedido el 17.06.2023. [Online]. Available: https://schunk.com/us/en/gripping-systems/c/PUB_8293

[31] WEISS ROBOTICS, "SENSOR FINGER," Accedido el 17.06.2023. [Online]. Available: https://weiss-robotics.com/sensor-fingers/

UNIVERSIDAD DE MÁLAGA

[32] D. Goger, N. Gorges, and H. Worn, "Tactile sensing for an anthropomorphic robotic hand: Hardware and signal processing," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 895–901, 2009. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.2009.5152650

[33] T. Mukai, M. Onishi, T. Odashima, S. Hirano, and Z. Luo, "Development of the tactile sensor system of a human-interactive robot 'RI-MAN'," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 505–512, 2008, ISSN: 15523098. [Online]. Available: https://doi.org/10.1109/TRO.2008.917006

[34] R. Maldonado-López, F. Vidal-Verdú, G. Liñán, and Á. Rodriguez-Vazquez, "Integrated Circuitry to Detect Slippage Inspired by Human Skin and Artificial Retinas," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 8, pp. 1554–1565, 2009, ISSN: 15498328. [Online]. Available: https://doi.org/10.1109/TCSI.2008.2008290

[35] R. Romeo and L. Zollo, "Methods and Sensors for Slip Detection in Robotics: A Survey," *IEEE Access*, vol. 8, pp. 73 027–73 050, 2020. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2020.2987849

[36] Ó. Oballe Peinado, "Aportaciones al hardware para sensores táctiles inteligentes basados en FPGAs," Ph.D. dissertation, Universidad de Málaga, 2016. [Online]. Available: http://hdl.handle.net/10630/11964

[37] O. Oballe-Peinado, J. Hidalgo-Lopez, J. Castellanos-Ramos, J. Sanchez-Duran, R. Navas-Gonzalez, J. Herran, and F. Vidal-Verdu, "FPGA-Based Tactile Sensor Suite Electronics for Real-Time Embedded Processing," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 12, pp. 9657–9665, 2017. [Online]. Available: http://dx.doi.org/10.1109/TIE.2017.2714137

[38] Barrett ADVANCED ROBOTICS, "Multi-fingered programmable grasper," Accedido el 18.06.2023. [Online]. Available: https://advanced.barrett.com/barrettha nd

[39] R. Lora-Rivera, Ó. Oballe-Peinado, and F. Vidal-Verdú, "Texture Detection with Feature Extraction on Embedded FPGA," *IEEE Sensors Journal*, vol. PP, pp. 1–1, 2023, ISSN: 1530-437X. [Online]. Available: https: //doi.org/10.1109/JSEN.2023.3268794

[40] ——, "Proposal and Implementation of a Procedure for Compliance Recognition of Objects with Smart Tactile Sensors," *Sensors 2023, Vol. 23, Page 4120*,

UNIVERSIDAD DE MÁLAGA

vol. 23, no. 8, p. 4120, 2023, DOI: 10.3390/S23084120, ISSN: 1424-8220. [Online]. Available: https://doi.org/10.3390/s23084120

[41] M. Shimojo and M. Ishikawa, "Active touch sensing method using a spatial filtering tactile sensor," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 1, pp. 948–954, 1993. [Online]. Available: http://dx.doi.org/10.1109/ROBOT.1993.292098

[42] R. Lora-Rivera, A. Guzmán-Manzano, J. Luna-Cortés, O. Oballe-Peinado, and F. Vidal-Verdú, "Texture Detection by Hardware-Friendly Low-Level Preprocessing," *Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics*, vol. 2020-November, pp. 922–927, 2020. [Online]. Available: http://dx.doi.org/10.1109/BioRob49111.2020.9224388

[43] R. Lora-Rivera, J. A. Luna-Cortes, A. De Guzman-Manzano, P. Ruiz-Barroso, J. Castellanos-Ramos, O. Oballe-Peinado, and F. Vidal-Verdu, "Object stiffness recognition with descriptors given by an FPGA-based tactile sensor," in *IEEE International Symposium on Industrial Electronics*, vol. 2020-June, 2020, pp. 561–566, ISBN: 9781728156354. [Online]. Available: https://doi.org/10.1109/ISIE45063.2020.9152575

[44] A. De Guzman-Manzano, A. Trujillo-Leon, R. Lora-Rivera, and F. Vidal-Verdu, "Tactile Sensor on Cane Handle for Gait Phase Analysis," *Proceedings of IEEE Sensors*, vol. 2019-Octob, pp. 1–4, 2019, ISBN: 9781728116341, ISSN: 21689229. [Online]. Available: https://doi.org/10.1109/SENSORS43011.2019.8956868

[45] S. Chun, Y. Choi, D. I. Suh, G. Y. Bae, S. Hyun, and W. Park, "A tactile sensor using single layer graphene for surface texture recognition," *Nanoscale*, vol. 9, no. 29, pp. 10 248–10 255, 2017, DOI: 10.1039/c7nr03748a, ISSN: 20403372. [Online]. Available: https://doi.org/10.1039/C7NR03748A

[46] S. C. Mannsfeld, B. C. Tee, R. M. Stoltenberg, C. V. Chen, S. Barman, B. V. Muir, A. N. Sokolov, C. Reese, and Z. Bao, "Highly sensitive flexible pressure sensors with microstructured rubber dielectric layers," *Nature Materials*, vol. 9, no. 10, pp. 859–864, 2010, ISSN: 14764660. [Online]. Available: https://doi.org/10.1038/nmat2834

[47] S. Ding, Y. Pan, M. Tong, and X. Zhao, "Tactile perception of roughness and hardness to discriminate materials by friction-induced vibration," *Sensors*

UNIVERSIDAD DE MÁLAGA

(Switzerland), vol. 17, no. 12, 2017, ISSN: 14248220. [Online]. Available: http://dx.doi.org/10.3390/s17122748

[48] Y. Xin, H. Tian, C. Guo, X. Li, H. Sun, P. Wang, C. Qian, S. Wang, and C. Wang, "A biomimetic tactile sensing system based on polyvinylidene fluoride film," *Review of Scientific Instruments*, vol. 87, no. 2, 2016, ISSN: 10897623. [Online]. Available: https://doi.org/10.1063/1.4941736

[49] 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, 2018. [Online]. Available: http://dx.doi.org/10.1002/smll.201703902

[50] G. Palli and S. Pirozzi, "Mechatronics A miniaturized optical force sensor for tendon-driven mechatronic systems : Design and experimental evaluation," *Mechatronics*, vol. 22, no. 8, pp. 1097–1111, 2012, ISSN: 0957-4158. [Online]. Available: https://doi.org/10.1016/j.mechatronics.2012.09.005

[51] B. Fang, F. Sun, Y. Chen, and C. Zhu, "A Tendon-Driven Dexterous Hand Design with Tactile Sensor Array for Grasping and Manipulation," *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), Dali, China*, no. December, pp. 203–210, 2019, ISBN: 9781728163215. [Online]. Available: https://doi.org/10.1109/ROBIO49542.2019.8961671

[52] U. Jeong and K. J. Cho, "A novel low-cost, large curvature bend sensor based on a Bowden-cable," *Sensors (Switzerland)*, vol. 16, no. 7, 2016, ISSN: 14248220. [Online]. Available: https://doi.org/10.3390/s16070961

[53] D. e. Göger, "Tactile proximity sensors for robotic applications," *2013 IEEE International Conference on Industrial Technology (ICIT), Cape Town, South Africa*, pp. 978–983, 2013, ISBN: 9781467345699. [Online]. Available: https://doi.org/10.1109/ICIT.2013.6505804

[54] U. Pierre Claver and G. Zhao, "Recent Progress in Flexible Pressure Sensors Based Electronic Skin," *Advanced Engineering Materials*, vol. 23, no. 5, pp. 1–17, 2021, ISSN: 15272648. [Online]. Available: http://dx.doi.org/10.1002/adem.202001187

[55] A. T. León, "Haptic Interface Based on Tactile Sensors for Assistive Devices," Ph.D. dissertation, Universidad de Málaga, 2018. [Online]. Available: https://riuma.uma.es/xmlui/handle/10630/16687

UNIVERSIDAD DE MÁLAGA

[56] Z. Wang, Q. Li, Y. Chen, B. Cui, Y. Li, F. Besenbacher, and M. Dong, "The ambipolar transport behavior of WSe 2 transistors and its analogue circuits," *NPG Asia Materials*, pp. 1–10, 2018, ISBN: 4142701800, ISSN: 1884-4057. [Online]. Available: https://doi.org/10.1038/s41427-018-0062-1

[57] W. Yuan, S. Dong, and E. Adelson, "GelSight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors (Switzerland)*, vol. 17, no. 12, 2017. [Online]. Available: http://dx.doi.org/10.3390/s17122762

[58] RECREUS, "FILAFLEX 82A ORIGINAL, Technical Data Sheet," Accedido el 18.04.2023. [Online]. Available: https://drive.google.com/file/d/1aoB6tmYfSAZzNzXSSVivrn7VJ3XXMOEw/view

[59] M. Schöpfer, C. Schürmann, M. Pardowitz, and H. Ritter, "Using a Piezo-Resistive Tactile Sensor for Detection of Incipient Slippage," *Robotics ISR 2010 41st International Symposium on and 2010 6th German Conference on Robotics ROBOTIK*, pp. 14–20, 2010, ISBN: 9783800732739. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5756770

[60] R. Lora-Rivera, A. Fernandez-Ruiz, J. Cuesta-Ramirez, A. De Guzman-Manzano, J. Castellanos-Ramos, O. Oballe-Peinado, and F. Vidal-Verdu, "Tactile Sensor with a Structured 3D Printed Cover and Laser-Isolated Tactels," *2018 9th Cairo International Biomedical Engineering Conference, CIBEC 2018 - Proceedings*, pp. 98–101, 2019. [Online]. Available: http://dx.doi.org/10.1109/CIBEC.2018.8641757

[61] J. Castellanos Ramos, "Tesis. Caracterización , Modelado y Diseño de Sensores Táctiles Piezorresistivos," Ph.D. dissertation, Universidad de Málaga, 2015. [Online]. Available: https://riuma.uma.es/xmlui/handle/10630/11333

[62] capLINQ, Accedido el 23.04.2023. [Online]. Available: https://www.caplinq.com/

[63] Arduino, "Arduino Mega 2560 Rev3," Accedido el 24.04.2023. [Online]. Available: https://store.arduino.cc/products/arduino-mega-2560-rev3

[64] RepRap, "RAMPS 1.4/es," Accedido el 24.04.2023. [Online]. Available: https://reprap.org/wiki/RAMPS_1.4/es

[65] A. MicroSystems, "A4988 DMOS Microstepping Driver with Translator And Overcurrent Protection," pp. 1–22, 2014. [Online]. Available: https://www.allegromicro.com/en/products/motor-drivers/brush-dc-motor-drivers/a4988

UNIVERSIDAD DE MÁLAGA

[66] Promoco, "Hybrid Stepper Motor 42BYGH Series Hybrid Stepper Motor." [Online]. Available: https://www.promoco-motors.com/

[67] R. Condit, Microchip Technology, D. W. Jones, and University of Iowa, "Stepping Motors Fundamentals - AN907," pp. 1–22, 2004.

[68] A. Semiconductor, "24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales HX711," pp. 1–9, 2017. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

[69] N. Mechatronics, "TUTORIAL TRASMISOR DE CELDA DE CARGA HX711, BALANZA DIGITAL," Accedido el 23.04.2023. [Online]. Available: https://naylampmechatronics.com/blog/25_tutorial-trasmisor-de-celda-de-carga-hx711-balanza-digital.html

[70] M. M. Letras Luna, "Arquitectura Hardware en FPGA para el Minado de Conjuntos Frecuentes en Conjuntos de Datos Usando una Estrategia de Segmentacion," Ph.D. dissertation, Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Tonantzintla, Puebla (México), 2015.

[71] *Zedboard™*, Accedido el 17.06.2023. [Online]. Available: https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/zedboard/

[72] arm, "ARCHITECTURE: System Architectures. AMBA 4," Accedido el 23.05.2023. [Online]. Available: https://www.arm.com/architecture/system-architectures/amba/amba-4

[73] Xilinx, "Vivado Design Suite: AXI Reference Guide," 2017, Accedido el 17.06.2023. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ug1037-vivado-axi-reference-guide

[74] Arm, "AMBA AXI-Stream Protocol Specification," 2021, Accedido el 23.05.2023. [Online]. Available: https://developer.arm.com/documentation/ihi0051/latest/

[75] X. Inc., "Zynq-7000 SoC Data Sheet: Overview," pp. 1–21, 2018. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[76] "FreeRTOS™," accessed: 03.07.2023. [Online]. Available: https://www.freertos.org/

UNIVERSIDAD DE MÁLAGA

[77] A. Drimus, M. Borlum Petersen, and A. Bilberg, "Object texture recognition by dynamic tactile sensing using active exploration," *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, pp. 277–283, 2012. [Online]. Available: http://dx.doi.org/10.1109/ROMAN.2012.6343766

[78] J. Gandarias, A. Garcia-Cerezo, and J. Gomez-De-Gabriel, "CNN-Based Methods for Object Recognition with High-Resolution Tactile Sensors," *IEEE Sensors Journal*, vol. 19, no. 16, pp. 6872–6882, 2019. [Online]. Available: http://dx.doi.org/10.1109/JSEN.2019.2912968

[79] R. Li and E. H. Adelson, "Sensing and recognizing surface textures using a gelsight sensor," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1241–1247, 2013, ISSN: 10636919. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2013.164

[80] M. Alameh, Y. Abbass, A. Ibrahim, and M. Valle, "Smart tactile sensing systems based on embedded CNN implementations," *Micromachines*, vol. 11, no. 1, 2020. [Online]. Available: http://dx.doi.org/10.3390/mi11010103

[81] N. F. Lepora, K. Aquilina, L. Cramphorn, and S. Member, "Exploratory tactile servoing with active touch," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1–8, 2017. [Online]. Available: https://doi.org/10.1109/LRA.2017.2662071

[82] L. Cramphorn, B. Ward-Cherrier, and N. F. Lepora, "Tactile manipulation with biomimetic active touch," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, pp. 123–129, 2016, ISBN: 9781467380263, ISSN: 10504729. [Online]. Available: https://doi.org/10.1109/ICRA.2016.7487124

[83] T. Matsubara and K. Shibata, "Active tactile exploration with uncertainty and travel cost for fast shape estimation of unknown objects," *Robotics and Autonomous Systems*, vol. 91, pp. 314–326, 2017, ISBN: 9781509020652, ISSN: 09218890. [Online]. Available: https://doi.org/10.1016/j.robot.2017.01.014

[84] C. J. Miller, C. A. Zorman, and G. E. Wnek, "An improved tactile sensing device for material characterization via friction-induced vibrations," *Sensors and Actuators, A: Physical*, 2020. [Online]. Available: https://doi.org/10.1016/j.sna.2019.111824

[85] J. Scheibert, S. Leurent, A. Prevost, and G. Debregeas, "The Role of Fingerprints in the Coding of Tactile Information Probed with a Biomimetic Sensor,"

UNIVERSIDAD DE MÁLAGA

Science, vol. 323, no. March, pp. 1503–1506, 2009. [Online]. Available: http://dx.doi.org/10.1126/science.1166467

[86] R. Fagiani, F. Massi, E. Chatelet, Y. Berthier, and A. Akay, "Tactile perception by friction induced vibrations," *Tribology International*, vol. 44, no. 10, pp. 1100–1110, 2011, ISSN: 0301679X. [Online]. Available: http://dx.doi.org/10.1016/j.triboint.2011.03.019

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

[88] 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 Transactions on Robotics*, vol. 36, no. 6, pp. 1619–1634, 2020, ISSN: 19410468. [Online]. Available: http://dx.doi.org/10.1109/TRO.2020.3003230

[89] S. Chen, S. Ge, W. Tang, J. Zhang, and N. Chen, "Tactile perception of fabrics with an artificial finger compared to human sensing," *Textile Research Journal*, vol. 85, no. 20, pp. 2177–2187, 2015, ISSN: 17467748. [Online]. Available: http://dx.doi.org/10.1177/0040517515586164

[90] J. Fishel and G. Loeb, "Bayesian exploration for intelligent identification of textures," *Frontiers in Neurorobotics*, no. JUNE, pp. 1–20, 2012. [Online]. Available: http://dx.doi.org/10.3389/fnbot.2012.00004

[91] A. Gómez-Eguíluz, I. Raño, S. A. Coleman, and T. M. McGinnity, "Multimodal Material identification through recursive tactile sensing," *Robotics and Autonomous Systems*, vol. 106, pp. 130–139, 2018, ISSN: 09218890. [Online]. Available: https://doi.org/10.1016/j.robot.2018.05.003

[92] D. S. Chathuranga, Z. Wang, V. A. Ho, A. Mitani, and S. Hirai, "A biomimetic soft fingertip applicable to haptic feedback systems for texture identification," in *2013 IEEE International Symposium on Haptic Audio Visual Environments and Games (HAVE)*, Istanbul, Turkey, 2013, pp. 29–33. [Online]. Available: https://doi.org/10.1109/HAVE.2013.6679606

[93] V. Chu, I. McMahon, L. Riano, C. G. McDonald, Q. He, J. Martinez Perez-Tejada, M. Arrigo, T. Darrell, and K. J. Kuchenbecker, "Robotic learning of haptic adjectives through physical interaction," *Robotics and Autonomous*

UNIVERSIDAD DE MÁLAGA

Systems, vol. 63, no. P3, pp. 279–292, 2015, ISSN: 09218890. [Online]. Available: https://doi.org/10.1016/j.robot.2014.09.021

[94] T. E. Alves de Oliveira, A. M. Cretu, and E. M. Petriu, "Multimodal Bio-Inspired Tactile Sensing Module for Surface Characterization," *Sensors (Basel, Switzerland)*, vol. 17, no. 6, pp. 1–19, 2017, ISSN: 14248220. [Online]. Available: http://dx.doi.org/10.3390/s17061187

[95] S. Wang, A. Albini, P. Maiolino, F. Mastrogiovanni, and G. Cannata, "Fabric Classification Using a Finger-Shaped Tactile Sensor via Robotic Sliding," *Frontiers in Neurorobotics*, vol. 16, 2022. [Online]. Available: http://dx.doi.org/10.3389/fnbot.2022.808222

[96] O. Kursun and A. Patooghy, "An Embedded System for Collection and Real-Time Classification of a Tactile Dataset," *IEEE Access*, vol. 8, pp. 97 462–97 473, 2020, ISSN: 21693536. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2020.2996576

[97] A. Ibrahim, P. Gastaldo, H. Chible, and M. Valle, "Real-time digital signal processing based on FPGAs for electronic skin implementation," *Sensors (Switzerland)*, vol. 17, no. 3, 2017. [Online]. Available: http://dx.doi.org/10.3390/s17030558

[98] A. V. Oppenheim and R. W. Schafer, *Tratamiento de señales en tiempo discreto*. Prentice Hall, 2012, ISBN: 978-8483227183.

[99] M. Jabri and B. Flower, "Weight Perturbation: An Optimal Architecture and Learning Technique for Analog VLSI Feedforward and Recurrent Multilayer Networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, 1992. [Online]. Available: https://doi.org/10.1109/72.105429

[100] D. W. Hart, *"Power Electronics"*. McGraw Hill, 2010, ISBN: 9780073380674.

[101] A. P. Ruz, *"Convertidores conmutados de potencia. Test de Autoevaluación"*. MARCOMBOUNIVERSITARIA, 2011, ISBN: 978-8426724830.

[102] A. Khan, M. Khosravi, S. Denei, P. Maiolino, W. Kasprzak, F. Mastrogiovanni, and G. Cannata, "A tactile-based fabric learning and classification architecture," *2016 IEEE International Conference on Information and Automation for Sustainability: Interoperable Sustainable Smart Systems for Next Generation, ICIAfS 2016*, 2016. [Online]. Available: http://dx.doi.org/10.1109/ICIAFS.2016.7946535

UNIVERSIDAD DE MÁLAGA

[103] RECREUS, "PLA, Technical Data Sheet," Accedido el 18.04.2023. [Online]. Available: https://drive.google.com/file/d/1fpLy7KjPz1tw-hXgqQOb-JzvzXUX5bmF/view

[104] S. Kumar, "Understanding K-Means, K-Means++ and, K-Medoids Clustering Algorithms," Accedido el 17.06.2023. [Online]. Available: https://towardsdatascience.com/understanding-k-means-k-means-and-k-medoids-clustering-algorithms-ad9c9fbf47ca

[105] R. Lora-Rivera, "TEXTURE DETECTION WITH FEATURE EXTRACTION ON EMBEDDED FPGA - DATASET," Accedido el 22.05.2023. [Online]. Available: https://ieee-dataport.org/documents/texture-detection-feature-extraction-embedded-fpga-dataset

[106] P. Fränti and S. Sieranoja, "How much can k-means be improved by using better initialization and repeats?" *Pattern Recognition*, vol. 93, pp. 95–112, 2019, DOI: 10.1016/j.patcog.2019.04.014, ISSN: 00313203. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2019.04.014

[107] C. Yang, Z. He, and W. Yu, "Comparison of Public Peak Detection Algorithms for MALDI Mass Spectrometry Data Analysis," Accedido el 24.04.2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2631518/

[108] M. A. Karami, "Finding Maximums in Noisy Data," Accedido el 24.04.2023. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/finding-maximums-in-noisy-data/

[109] ARMdeveloper, *"AMBA 4 AXI4-Stream Protocol"*. Copyright ©ARM. All rights reserved., 2010, Accedido el 18.04.2023. [Online]. Available: https://developer.arm.com/documentation/ihi0051/a/

[110] J. Osborne, A. Patooghy, B. Sarsekeyev, and O. Kursun, "Eco-CMB: A Hardware-Accelerated Band-Power Feature Extractor for Tactile Embedded Systems," *Midwest Symposium on Circuits and Systems*, vol. 2021-Augus, pp. 198–203, 2021, ISBN: 9781665424615, ISSN: 15483746. [Online]. Available: http://dx.doi.org/10.1109/MWSCAS47672.2021.9531685

[111] R. Lora-Rivera, Ó. Oballe-Peinado, and F. Vidal-Verdú, "Texture Detection with Tactile Sensors based on the Goertzel Algorithm," in *IWANN International Work Conference on Artificial Neural Networks*, Azores (Portugal), 2023, pp. 7–8, ISBN: 978-84-19214-58-4.

UNIVERSIDAD DE MÁLAGA

[112] P. Sysel and P. Rajmic, "Goertzel algorithm generalized to non-integer multiples of fundamental frequency," *Eurasip Journal on Advances in Signal Processing*, vol. 2012, no. 1, pp. 1–8, 2012, ISSN: 16876172. [Online]. Available: https://doi.org/10.1186/1687-6180-2012-56

[113] M. LABORATORIES™, "Detecting A Single Frequency Efficiently," Accedido el 17.07.2023. [Online]. Available: https://www.mstarlabs.com/dsp/goertzel/goertzel.html

[114] S. Gao, Y. Dai, and A. Nathan, "Tactile and Vision Perception for Intelligent Humanoids," *Advanced Intelligent Systems*, vol. 4, no. 2, p. 2100074, 2022, ISSN: 2640-4567. [Online]. Available: https://doi.org/10.1002/aisy.202100074

[115] Y. Chen, Z. Gao, F. Zhang, Z. Wen, and X. Sun, "Recent progress in self-powered multifunctional e-skin for advanced applications," *Exploration*, vol. 2, no. 1, p. 20210112, 2022. [Online]. Available: https://doi.org/10.1002/EXP.20210112

[116] A. B. Balasubramanian, D. P. Magee, and D. G. Taylor, "Stiffness Estimation in Single Degree of Freedom Mechanisms using Regression," in *IECON Proceedings (Industrial Electronics Conference)*, vol. 2021-Octob. IEEE, 2021, ISBN: 9781665435543. [Online]. Available: https://doi.org/10.1109/IECON48115.2021.9590002

[117] L. Wang, Q. Li, J. Lam, and Z. Wang, "Tactual Recognition of Soft Objects from Deformation Cues," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 96–103, 2022, ISSN: 23773766. [Online]. Available: https://doi.org/10.1109/LRA.2021.3119393

[118] I. Bandyopadhyaya, D. Babu, A. Kumar, and J. Roychowdhury, "Tactile sensing based softness classification using machine learning," in *IEEE International Advance Computing Conference (IACC)*, 2014, ISBN: 9781479925728, ISSN: 1479683X. [Online]. Available: https://doi.org/10.1109/IAdCC.2014.6779503

[119] J. Windau and W. M. Shen, "An inertia-based surface identification system," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2330–2335, 2010, ISBN: 9781424450381, ISSN: 10504729. [Online]. Available: https://doi.org/10.1109/ROBOT.2010.5509310

[120] J. Thompson, D. Kasun Prasanga, and T. Murakami, "Identification of unknown object properties based on tactile motion sequence using 2-finger gripper robot," *Precision Engineering, Elsevier*, 2021, DOI: https://doi.org/10.1016/j.precisioneng.2021.12.009.

UNIVERSIDAD DE MÁLAGA

[121] W. Bottcher, P. MacHado, N. Lama, and T. M. McGinnity, "Object recognition for robotics from tactile time series data utilising different neural network architectures," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2021-July, 2021, ISBN: 9780738133669. [Online]. Available: https://doi.org/10.1109/IJCNN52387.2021.9533388

[122] T. Bhattacharjee, J. M. Rehg, and C. C. Kemp, "Inferring Object Properties with a Tactile-Sensing Array Given Varying Joint Stiffness and Velocity," *International Journal of Humanoid Robotics*, vol. 15, no. 1, 2018, ISSN: 02198436. [Online]. Available: https://doi.org/10.1142/S0219843617500244

[123] A. Drimus, G. Kootstra, A. Bilberg, and D. Kragic, "Classification of rigid and deformable objects using a novel tactile sensor," in *IEEE 15th International Conference on Advanced Robotics: New Boundaries for Robotics, ICAR 2011*. IEEE, 2011, pp. 427–434, ISBN: 9781457711589. [Online]. Available: https://doi.org/10.1109/ICAR.2011.6088622

[124] Z. Kappassov, D. Baimukashev, O. Adiyatov, S. Salakchinov, Y. Massalin, and H. A. Varol, "A Series Elastic Tactile Sensing Array for Tactile Exploration of Deformable and Rigid Objects," in *IEEE International Conference on Intelligent Robots and Systems*, 2018, pp. 520–525, ISBN: 9781538680940, ISSN: 21530866. [Online]. Available: https://doi.org/10.1109/IROS.2018.8593755

[125] L. Kaufman and P. J. Rousseuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Society, International Biometric, 1991, vol. 47, DOI:10.1002/9780470316801, ISBN: 9780471878766.

[126] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977. [Online]. Available: https://doi.org/10.1145/355744.355745

[127] D. Collins, "What is stepper motor resonance and how can it be avoided?" Accedido el 23.04.2023. [Online]. Available: https://www.linearmotiontips.com/what-is-stepper-motor-resonance-and-how-can-it-be-avoided/

[128] D. Muller, "Strategies To Avoid Stepper Motor Resonance Problems," Accedido el 23.04.2023. [Online]. Available: https://www.automation.com/en-us/articles/november-2021/strategies-avoid-stepper-motor-resonance-problems

[129] L. Llamas, "Motores paso a paso con Arduino y driver A4988 o DRV8825," Accedido el 23.04.2023. [Online]. Available: https://www.luisllamas.es/motores-paso-paso-arduino-driver-a4988-drv8825/

UNIVERSIDAD DE MÁLAGA

[130] D. S. Arar, “Intro Software Walkthrough: Fast Fourier Transforms and the Xilinx FFT IP Core,” Accedido el 03.05.2023. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/xilinx-fft-ip-core-fast-fourier-transform-software-walkthrough/

[131] Xilinx, “Fast Fourier Transform v7.1 (Product Specification),” 2011, Accedido el 18.06.2023. [Online]. Available: https://docs.xilinx.com/v/u/en-US/xfft_ds260

UNIVERSIDAD DE MÁLAGA