SISTEMA DE RECONOCIMIENTO DE GESTOS BASADO EN VISIÓN ESTÉREO E IMPLEMENTADO EN FPGA CON APLICACIÓN A LA INTERACCIÓN HOMBRE-ROBOT

JUAN DAVID GUERRERO BALAGUERA

UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA
SEDE SECCIONAL SOGAMOSO
MAESTRÍA EN INGENIERÍA
SOGAMOSO
2017

SISTEMA DE RECONOCIMIENTO DE GESTOS BASADO EN VISIÓN ESTÉREO E IMPLEMENTADO EN FPGA CON APLICACIÓN A LA INTERACCIÓN HOMBRE-ROBOT

JUAN DAVID GUERRERO BALAGUERA

Trabajo de grado para optar por el título de maestría en ingeniería con énfasis en ingeniería electrónica

Director de Tesis
Ing. WILSON JAVIER PÉREZ HOLGUÍN, PhD

UNIVERSIDAD PEDAGÓGICA Y TECNOLÓGICA DE COLOMBIA
SEDE SECCIONAL SOGAMOSO
MAESTRÍA EN INGENIERÍA
SOGAMOSO
2017

Nota de Aceptación

_____

_____

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Sogamoso, 13 de Febrero de 2017

Escribe aquí tu dedicatoria

Ejemplo: A Dios, mis padres

Agradecimientos

Expresa aquí tus agradecimientos (Puede ser un párrafo o dos pero no más de una página).

Resumen

El reconocimiento de gestos de las manos es utilizado en una gran variedad de aplicaciones de interacción hombre-computador (HCI). Dentro de más sobresalientes se encuentran: reconocimiento del lenguaje de señas, realidad virtual, control de aparatos de televisión y videojuegos, video vigilancia e interacción humano-robot (HRI), etc. La interacción de los humanos con los robots es un tema que ha sido abordado por la ciencia ficción y que ha trascendido a la realidad gracias al avance tecnológico de la ciencia y la tecnología durante los últimos años. La investigación en esta área busca mejorar la comunicación entre humanos y robots para que la interacción se desarrolle con mayor naturalidad. El reconocimiento de los gestos de las manos basado en visión artificial es la herramienta utilizada en la actualidad para establecer comunicación con los robots. Sin embargo, existe la necesidad de comunicar información cada vez más compleja empleando por tanto una mayor cantidad de gestos. Esto causa un incremento en el tiempo requerido para procesar la información y bajas tasas de reconocimiento, lo que es inaceptable en aplicaciones de reconocimiento de gestos en tiempo real.

Los arreglos lógicos programables en campo (FPGA) son una buena alternativa frente a los sistemas de cómputo convencionales para el desarrollo de aplicaciones de reconocimiento de gestos de las manos basado en visión, ya que permiten la implementación de arquitecturas de procesamiento paralelo de alto rendimiento y su consumo de potencia es significativamente menor al de otras plataformas tales como las GPU. En este trabajo se presenta el desarrollo de un sistema de reconociendo de gestos de las manos basado en visión estéreo e implementado en FPGA, diseñado para tele-operar un robot móvil mediante siete gestos de las manos. El sistema está compuesto por cuatro etapas principales: *i*) captura, *ii*) pre-procesamiento, *iii*) extracción de características y *iv*) reconocimiento. La captura de los gestos se realiza mediante un sistema estereoscópico de cámaras. En la etapa de pre-procesamiento se realiza la extracción de la imagen de la mano empleando un proceso de segmentación por color de piel y el mapa de disparidad. En la etapa de extracción de características se emplea la orientación de la trayectoria de la mano para describir cada gesto. En la etapa de reconocimiento se emplea un clasificador basado en Modelos Ocultos de Markov (HMM).

Las etapas de captura, pre-procesamiento y extracción de características fueron desarrolladas en VHDL y sintetizadas en un dispositivo FPGA-SoC Cyclone 5 SoC de Altera, mediante la tarjeta de desarrollo DE1-SoC. La etapa de reconocimiento se implementa en software empleando el HPS del dispositivo FPGA-SoC. EL sistema está en la capacidad de procesar imágenes estéreo a una frecuencia de pixel de 100MHz con una velocidad de procesamiento de 307fps aproximadamente. Vale la pena resaltar que el sistema diseñado tiene una tasa de reconocimiento de gestos mayor al 99%.

Palabras Clave: Visión Artificial, Reconocimiento de Gestos, Human-Computer Interaction (HCI), Human-Robot Interaction (HRI), VHDL, FPGA, HMM.

Contenido

Pág

Lista de Tablas ..... **ix**

Lista de Figuras ..... **xi**

Lista de Anexos ..... **xviii**

Lista de Abreviaturas ..... **xix**

1 Introducción **1**

2 Revisión de Literatura **3**
2.1 Captura de Gestos de las Manos 3
2.1.1 Cámaras estéreo 5
2.1.2 Cámaras ToF (Time of Flight) 6
2.1.3 Sensores de luz estructurada 7
2.2 Pre-procesamiento 7
2.3 Extracción de Características 9
2.4 Reconocimiento 10
2.5 Reconocimiento de Gestos de las Manos e Implementación en Hardware. 13

3 Conceptos Fundamentales **21**
3.1 Visión Estéreo 21
3.1.1 Transformada Census 23
3.1.2 Transformada Census Dispersa 24
3.1.3 Suma de Distancias de Hamming (SHD) 25
3.2 Procesamiento Digital de Imágenes 25
3.3 Procesamiento Embebido de Imágenes 27
3.3.1 Procesamiento de imágenes en tiempo real 27
3.3.2 Procesamiento paralelo de imágenes 27
3.3.3 Técnicas de mapeo de algoritmos en FPGA 29
3.4 Modelos Ocultos de Markov 32
3.4.1 Problemas básicos de un HMM 33

4 Descripción del Sistema de Reconocimiento de Gestos **37**
4.1 Etapas del Sistema de Reconocimiento de Gestos 37
4.2 Proceso de Diseño del sistema de Reconocimiento de Gestos. 39
4.2.1 Desarrollo de algoritmos 39
4.2.2 Selección de la arquitectura 40
4.2.3 Implementación 40

5 Captura de gestos **44**

5.1 Módulo de Configuración de la Cámara 45
5.2 Módulo de Captura de Imágenes....47
5.3 Módulo de Sincronización de Imágenes Estéreo....49
5.4 Módulo de Conversión de YCbCr422 a YCbCr444 51

6 Pre-procesamiento 53
6.1 Filtro de Mediana 53
6.1.1 Arquitectura en hardware 55
6.1.2 Generador de ventana móvil 56
6.1.3 Red de ordenamiento para el cálculo de la mediana....57
6.1.4 Generador de la señal de validez de los datos de salida 59
6.2 Correspondencia Estéreo 60
6.2.1 Arquitectura en hardware 65
6.2.2 Módulo de similitud SHD 67
6.2.3 Transformada Census Dispersa 72
6.2.4 Calculo de la disparidad 74
6.2.5 Comprobación de consistencia Izquierda-derecha (LRCC)....75
6.3 Segmentación 77
6.3.1 Arquitectura en hardware 79
6.3.2 Segmentación del color de piel 80
6.3.3 Segmentación del mapa de disparidad....81
6.4 Operación Morfológica de Apertura 82
6.4.1 Arquitectura en hardware 84
6.4.2 Erosión y dilatación 85

7 Extracción de características 89
7.1 Cálculo del Centroide de la Forma de la Mano....89
7.2 Suavizado del Gesto: Filtro de Media Móvil....92
7.3 Detección del Gesto, "*Gesture Spotting*" 94
7.4 Extracción de Características 97
7.4.1 Interfaz de conexión FPGA-HPS y extracción de características....99

8 Reconocimiento....103
8.1 Selección del Modelo de los HMMs 104
8.2 Inicialización de los Parámetros del HMM Con Topología LRB 105
8.3 Entrenamiento de los HMM.... 106
8.4 Validación Cruzada *K-fold* y Selección del Número de Estados de los HMMs 106
8.5 Implementación de la Etapa de Reconocimiento en el HPS 108
8.5.1 Control del robot Lego Mindstorms NXT.... 112

9 Resultados 114
9.1 Captura de Gestos 114
9.2 Etapa de Pre-procesamiento 115
9.3 Extracción de Características 122
9.4 Reconocimiento de Gestos 125
9.5 Implementación del Sistema de Reconocimiento de Gestos en FPGA-SoC 126

10 Conclusiones y Trabajos Futuros.....130
10.1 Trabajos Futuros..... 131

Bibliografía.....133

Lista de Tablas

Pág

Tabla 1. Comparación de los sensores de profundidad (L. Chen et al., 2013) 5

Tabla 2. Resumen de los trabajos más representativos en reconocimiento de gestos. 16

Tabla 3. Resumen de los trabajos más representativos en reconocimiento de gestos (Continuación). 17

Tabla 4. Resumen de los trabajos más representativos en reconocimiento de gestos (Continuación). 18

Tabla 5. Resumen de los trabajos más representativos en reconocimiento de gestos (Continuación). 19

Tabla 6. Resumen de los trabajos más representativos en reconocimiento de gestos (Continuación). 20

Tabla 7. Parámetros de configuración del sensor OV5642 45

Tabla 8. Descripción de los puertos de entrada-salida del módulo de configuración de la cámara. 47

Tabla 9. Descripción de los puertos de entrada-salida del módulo de captura de imágenes. 49

Tabla 10. Descripción de los puertos de entradas-salida del módulo de sincronización. 50

Tabla 11. Descripción de los puertos de entrada-salida del módulo de conversión se YCbCr 4:2:2 a 4:4:4. 52

Tabla 12. Mapa de registros del módulo de memoria FIFO. 101

Tabla 13. Descripción de los bits que conforman el registro de control del módulo FIFO. 102

Tabla 14. Descripción de las entradas de la FSM del sistema de reconocimiento. 111

Tabla 15. Lista de comandos empleados para el control del robot de acuerdo con el gesto realizado. 113

Tabla 16. Resultados del filtro de mediana implementado en Software y Hardware. 116

Tabla 17. Porcentaje de pixeles correctamente coincidentes del algoritmo de correspondencia estéreo implementado en Software y Hardware. 118

Tabla 18. Tiempo de ejecución de la correspondencia estéreo en software y en hardware. 118

Tabla 19. Parámetros utilizados para calcular el tiempo de ejecución en hardware. 119

Tabla 20. Tiempo de ejecución de la segmentación del mapa de disparidad en software y en hardware. 121

Tabla 21. Tiempo de ejecución de la operación morfológica de apertura en software y en hardware. 122

Tabla 22. Descripción de la funcionalidad de los botones, switches y leds de la tarjeta De1-SoC en el sistema de reconocimiento de gestos. 127

Tabla 23. Recursos del dispositivo FPGA-SoC utilizados por el sistema de reconocimiento de gestos. 127

Tabla 24. Matriz de confusión del sistema de reconocimiento implementado. 129

Lista de Figuras

Pág

Figura 1. Etapas que componen un sistema de reconocimiento de gestos de las manos. Fuente autor. 3

Figura 2. Sensores de profundidad. (a) Cámara estéreo (Pointgrey, 2016). (b) Sensor time-of-flight (ToF) (SwissRanger SR4000 MESA Imaging AG). (c) luz estructurada (Kinect). Fuente autor. 5

Figura 3. Sistema de imágenes estéreo simplificado. Adaptación a partir de (Mattoccia, 2013). 21

Figura 4. Imagen estéreo “Tsukuba”. (a) Imagen izquierda. (b) Imagen derecha. (c) mapa de disparidad ideal (*ground truth*). Fuente (Scharstein & Szeliski, 2002). 22

Figura 5. Método de correspondencia estéreo basado en el área. En la imagen de referencia la ventana se encuentra centrada en el punto x, mientras que en la imagen objetivo hay múltiples ventana centradas en [x,x-d_{max}]. Adaptación a partir de (Mattoccia, 2013). 22

Figura 6. Calculo de la transformada Census para un pixel con una vecindad de 3X3 pixeles. Fuente Autor. 24

Figura 7. Mascaras para el cálculo de la transformada Census. (a). Mascara 5X5 convencional. (b) Mascara 5X5 uniformemente distribuida, vecindad de transformada Census con 50% de dispersión. Fuente Autor. 24

Figura 8. Pirámide del procesamiento de imágenes. Adaptación a partir de (Bailey, 2011)....25

Figura 9. Filtro Local. La zona sombreada representa una vecindad de píxeles de entrada a un filtro, el cual genera un pixel nuevo en el centro de dicha vecindad. Adaptación a partir de (Bailey, 2011)....26

Figura 10. Paralelismo temporal empleando procesadores pipeline. Adaptación a partir de (Bailey, 2011)....28

Figura 11. Paralelismo espacial basado en partición de la imagen. (a) Partición por filas. (b) Partición por Columnas. (c) Partición por bloques. Adaptación a partir de (Bailey, 2011). 28

Figura 12. Procesamiento de flujo de datos. Conversión de paralelismo espacial en paralelismo temporal. Adaptación a partir de (Bailey, 2011). 29

Figura 13. Ejemplo de ejecucion *Pipeline*. (a) Realizar el calculo en un solo ciclo de reloj. (b) Realizar el calculo en dos ciclos de reloj (dos etapas pipeline). (c) Realizar el calculo en cuatro ciclos de reloj (cuatro etapas Pipeline). Adaptacion a partir de (Bailey, 2011). 30

Figura 14. Escaneo de la imagen mediante una ventana 3X3 empleando el metodo de cacheo *Row Buffering*. Adaptacion a partir de (Bailey, 2011). 32

Figura 15. Aquitectura *Row Buffering* para FPGA. Adaptacion a partir de (Bailey, 2011). 32

Figura 16. Gestos empleados en este trabajo para controlar a distancia un robot. Fuente autor. ...37

Figura 17. Esquema conceptual del sistema de reconocimiento de gestos de las manos. Fuente autor. 38

Figura 18. Gestos de la base de datos SKIG empleados para la primera fase del diseño del sistema de reconocimiento. Fuente (L. Liu & Shao, 2013). 40

Figura 19. Flujo de diseño empleado para la implementación en FPGA del sistema de reconocimiento. Adaptación a partir de (Bailey, 2011)....41

Figura 20. Diagrama general que expone la arquitectura del sistema de reconocimiento desarrollado. Fuente autor. 43

Figura 21. Sistema de cámaras estereoscópico desarrollado para la aplicación. Fuente autor. 44

Figura 22. Interfaz de conexión entre el sensor OV5642 y la FPGA CYCLONE V SoC. Fuente autor. 45

Figura 23. Secuencia de envío de datos mediante bus I2C para configuración de los registros del sensor OV5642. 46

Figura 24. Arquitectura del módulo de inicialización del sensor OV5642. Fuente autor. 47

Figura 25. Diagrama de tiempos de las señales provenientes del sensor OV5642 empleadas en la captura de imágenes. Fuente autor. 48

Figura 26. Captura de una fila de la imagen con submuestreo de crominancia 4:2:2. Fuente autor. 48

Figura 27. Arquitectura interna del módulo de captura de imágenes. Fuente autor. 49

Figura 28. Arquitectura interna del módulo de sincronización de imágenes estéreo. Fuente autor. 50

Figura 29. Arquitectura interna del módulo de conversión de YCbCr422 a YCbCr444. Fuente autor. 52

Figura 30. Interconexión de los componentes que conforman la etapa de pre-procesamiento. Fuente autor. 53

Figura 31. Algoritmo Correspondiente al filtro de mediana aplicado a una imagen de entrada I_i 54

Figura 32. Algoritmo Correspondiente al cálculo de la mediana aplicada a un vector de entrada V55

Figura 33. Modulo de hardware del filtro de mediana. Flujo de las señales de datos y control. Fuente autor. 56

Figura 34. Diagrama esquemático del modulo de filtro de Mediana. Fuente autor.....56

Figura 35. Aquitectura de cache empleando *Row Buffering*. Adaptacion a partir de (Bailey, 2011).57

Figura 36. Elemento de procesamiento de comparacion e intercambio CS. (a) arquitectura interna. (b) diagrama esquemático. Fuente autor. 58

Figura 37. Aquitectura full-pipeline.del filtro de mediana basada en una red de ordenamiento de transposicion impar-par. Fuente autor.....59

Figura 38. Comportamiento del filtro de mediana en el tiempo. Fuente autor. 60

Figura 39. Generador de la señal de validez de los pixeles de salida procesados. Fuente autor. ...60

Figura 40. Algoritmo para la implementacion Software de la transformada Census Dispersa. 62

Figura 41. Algoritmo para la implementacion Software del calculo de la disparidad. 63

Figura 42. Algoritmo para la implementacion Software de SHD. 64

Figura 43. Algoritmo para la implementacion Software de LRCC. 64

Figura 44. Aquitectura general de correlacion estéreo. Adaptacion a partir de (Fife, 2011; Fife & Archibald, 2013).....65

Figura 45. Aquitectura *Pipeline* de correlacion estéreo. Adaptacion a partir de (Fife, 2011; Fife & Archibald, 2013).....66

Figura 46. Diagrama esquemático del modulo de correspondencia estéreo. Fuente autor.....67

Figura 47. Modulo de similitud SHD. Flujo de las señales de datos y control. Fuente autor. 67

Figura 48. Calculo de la distancia de Hamming. (a) comparacion de los pixeles mediante la operación XOR. (b) Configuracion de sumadores en arbol para obtener la cantidad de 1's. Fuente autor.....69

Figura 49. Calculo de la distancia de Hamming. (a) procedimiento para sumar las columnas de la ventana. (b) procedimiento para calcular la suma total de la ventana a partir de la suma de columnas. Adaptacion a partir de (Fife, 2011) 70

Figura 50. Requerimientos de memoria del modulo de similitud empleando generador de ventana movil (Azul) y Optimización de Suma de Ventanas (Rojo).....71

Figura 51. Calculo de la suma de distancias de Hamming mediante la optimizacion de suma de ventana. Adaptacion a partir de (Fife & Archibald, 2013).....72

Figura 52.Modulo de la Transformada Census. Flujo de la señales de datos y de control. Fuente autor.....73

Figura 53.Modulo de la Transformada Census. Flujo de la señales de datos y de control. Fuente autor.....73

Figura 54.Calculo de la transformada de census dispersa. Fuente autor. 74

Figura 55.Modulo de selección del mejor valor de disparidad. (a) diagrama esquematico del módulo. (b) arquitectura interna del módulo. Fuente autor. 75

Figura 56.Calculo de la comprobacion de consistencia izquierda-derecha LRCC. Fuente autor. 76

Figura 57.Algoritmo para la implementacion en software de la segmentacion por color de piel. 78

Figura 58.Algoritmo para la implementacion en software de la segmentacion del mapa de disparidad. 79

Figura 59.Modulo de Segmentacion. Flujo de las señales de datos y control. Fuente autor. 80

Figura 60.diagrama esquematico del modulo de segmentacion en hardware. Fuente autor. 80

Figura 61.Calculo de la segmentacion por color de piel. Fuente autor. 81

Figura 62.Calculo de la segmentacion del mapa de disparidad. Fuente autor. 82

Figura 63.Algoritmo para la implementacion en software de la operación morfológica de Erosión..83

Figura 64.Algoritmo para la implementacion en software de la operación morfológica de Dilatación. 84

Figura 65. Modulo de la operación morfológica de apertura. Flujo de las señales de datos y control. Fuente autor.....84

Figura 66.Diagrama esquemático del módulo de la operación morfológica de apertura en hardware. Fuente autor. 85

Figura 67.Diagrama general de la implementacion en hardware de un filtro morfológico. Fuente autor.....86

Figura 68.Arquitectura del generador de ventana para el filtrado morfológico. La señal $D / \overline{E}$ permite seleccional la operación morfológica a emplear. Adaptacion a partir de (Bailey, 2011).....87

Figura 69.Arquitectura para las operaciones morfológicas básica Erosion y dilatacion. La señal $D / \overline{E}$ permite seleccional la operación morfológica a emplear. Fuente autor.....88

Figura 70.Trayectoria de centroide de la mano para el gesto “Rotar a la Derecha 90º”. Fuente autor.....89

Figura 71.Algoritmo para el calculo del centroide de la forma de la mano en una imagen binaria...90

Figura 72. Arquitectura diseñada para el calculo del centroide. Fuente autor. 91

Figura 73.Diagrama esquemático del módulo para calcular el centroide de la mano en hardware. Fuente autor. 92

Figura 74.Suavizado del gesto “Rotar a la Derecha 90º”mediante el filtro de media móvil de longitud 4. Fuente autor.....93

Figura 75.Arquitectura en hardware para el calculo del filtro de media movil de longitud N. Fuente autor.....93

Figura 76.Diagrama esquemático del módulo de suavizado de la trayectoria de gestos en hardware. Fuente autor. 94

Figura 77. Fases de la ejecucion del gesto “Rotar derecha 90º”. Fuente autor. 95

Figura 78. Algortimo que describe el proceso para la deteccion de un gesto. 96

Figura 79. Implementacion en hardware de la deteccion de gesto válido. Fuente autor. 97

Figura 80. Diagrama esquemático de modulo de deteccion de gestos implementado en hardware. Fuente autor. 97

Figura 81. Calculo de la orientacion. (a) orientacion entre dos puntos consecutivos del centroide. (b) codificacion discreta de 1 a 8 dividiendo la orientacion en 45º. Fuente autor. 98

Figura 82. Algoritmo que describe la extraccion de la orientacion y su valor de cuantizacion para dos valores de centroide consecutivos.....99

Figura 83. Interconexion entre el procesamiento en desarrollado en FPGA y el HPS a traves del *Lightweight HPS-to-FPGA Bridge*. Fuente autor. 101

Figura 84. Diagrama de bloques de la etapa de reconocimiento de gestos empleando HMM. Adaptacion a partir de (M. O. S. M. Elmezain, 2010; Rabiner, 1989). 103

Figura 85. Topología LRB (Left-Right Banded) de 6 estados para un HMM. Adaptacion a partir de (M. O. S. M. Elmezain, 2010; Rabiner, 1989) 104

Figura 86. Procedimiento de validacion cruzada 3-fold. Adaptacion a partir de (Refaeilzadeh et al., 2009) 107

Figura 87. Inicializacion del algoritmo de forward. 109

Figura 88. Evaluacion del algoritmo de forward para un simbolo observado en un instante de tiempo t. 109

Figura 89. Finalizacion del algoritmo de Forward para calcular la probabilidad total del HMM. 110

Figura 90. Diagrama de estados de la etapa de reconocimiento desarrollada en software. Fuente autor. 111

Figura 91. Comunicación entre el sistema de reconocimeinto y el Robot LEGO NXT. Fuente autor. 112

Figura 92. Protocolo de intercambio de mensajes via bluetooth de LEGO NXT. Fuente autor. 113

Figura 93. Imágenes estéreo obtenidas en la etapa de captura. (a) imagen izquierda. (b) imagen derecha. Fuente autor. 114

Figura 94. Resultados del filtro de medianta aplicado a la imagen *lenna*. a) imagen original. b) imagen con ruido sal y pimienta al 5%. c) imagen filtrada en FPGA. d) imagen filtrada en MATLAB. Fuente autor. 115

Figura 95. Resultados de ejecución del algoritmo de correspondencia estéreo implementado en Software y Hardware. a) imagen izquierda del par estéreo empleado: *cones, teddy* y *tsukuba*. b) *ground-truth* c) mapa de disparidad en MATLAB. d) mapa de disparidad en FPGA. Fuente autor. 117

Figura 96. Mapa de disparidad calculado para una imagen durante la realizacion del gesto. a) captura de la imagen izquierda. b) captura de la imagen derecha. c) mapa de disparidad calculado por el sistema desarrollado. Fuente autor. 119

Figura 97. Segmentacion por color de piel. (a) Imagen izquierda del par estéreo capturado. (b) segmentacion por color de piel en MATLAB. (c) Segmentacion por color de piel en FPGA. Fuente autor. 120

Figura 98. Segmentacion del mapa de disparidad. (a) mapa de disparidad calculado a partir de las imágenes estéreo capturadas. (b) segmentacion del mapa de disparidad en MATLAB. (c) segmentacion del mapa de disparidad en FPGA. Fuente autor. 120

Figura 99. Operación morfológica de apertura. (a) imagen segmentada. (b) operación morfológica de apertura en MATLAB. (c) operación morfológica de apertura en FPGA. Fuente autor. 121

Figura 100. Elementro estructurante empleado en la operación morfológica de apertura. Fuente autor. 121

Figura 101. Secuencia de simbolos discretos que representan el gesto “Adelante”. Fuente autor.123

Figura 102. Secuencia de simbolos discretos que representan el gesto “Atras”. Fuente autor. 123

Figura 103. Secuencia de simbolos discretos que representan el gesto “Derecha”. Fuente autor.123

Figura 104. Secuencia de simbolos discretos que representan el gesto “Izquierda”. Fuente autor.124

Figura 105. Secuencia de simbolos discretos que representan el gesto “Rotar Derecha 90º”. Fuente autor. 124

Figura 106. Secuencia de simbolos discretos que representan el gesto “Rotar Izquierda 90º”. Fuente autor. 124

Figura 107. Secuencia de simbolos discretos que representan el gesto “Stop”. Fuente autor. 125

Figura 108. Resultados de la validacion cruzada 10-fold para determinar la exactitud del clasificador basado en HMMs con diferente numero de estados ocultos. Fuente autor.125

Figura 109. Implementacion del sistema de reconocimiento de gestos. Fuente autor. 126

Lista de Anexos

	Pág
Anexo A. Documentación utilizada para el desarrollo del trabajo.	CD-ROM
Anexo B. Configuración de los registros del sensor OV5642	CD-ROM
Anexo C. Diagrama de estados configuración de la cámara	CD-ROM
Anexo D. Descripción en lenguaje VHDL de los módulos de hardware desarrollados	CD-ROM
Anexo F. Resultado de simulación en Modelsim-Altera.	CD-ROM
Anexo G. Conceptos de filtrado de imágenes y FPGAs	CD-ROM
Anexo H. Aplicación desarrollada para Linux embebido y programa del Robot Lego utilizado.	CD-ROM

Lista de Abreviaturas

HCI	Human-Computer Interaction
HRI	Human-Robot Interaction
ToF	Time of Flight
FPGA	Field Programmable Gate Array
SVM	Support Vector Machine
HMM	Hidden Markov Model
ANN	Artificial Neural Network
ANMM	Average Neighborhood Margin Maximization
LDCRFs	Latent-Dynamic Conditional Random Fields
KNN	K-Nearest Neighbors
FEMD	Finger-Earth Mover's Distance
ROI	Region of Interest
PCA	Principal Component Analisys
DTW	Dynamic Time Warping
P2-DHMMs	Pseudo two dimension hidden Markov models
PSO	Particle Swarm Optimization
DHMM	Discrete Hidden Markov Model
FSM	Finite State Machine
SW	Software
HW	Hardware
FPGA	Field Programmable Gate Array
VHDL	VHSIC (Very High Speed Integrated Circuit) Hardware Description Language

1 Introducción

Los gestos son movimientos corporales expresivos que implican movimientos físicos de los dedos, las manos, los brazos, la cabeza, la cara o el cuerpo, con la intención de transmitir información significativa o interactuar con el entorno (Mitra & Acharya, 2007). Formalmente un gesto puede definirse como un patrón espacio-temporal que puede ser estático, dinámico o ambos. Los gestos de las manos son un poderoso canal de comunicación entre personas, los cuales son una parte importante de la transferencia de información en nuestra vida diaria (M. O. S. M. Elmezain, 2010).

Desde el surgimiento de los computadores existe la necesidad de que las personas puedan interactuar con este tipo de máquinas de una manera más natural. Hasta ahora, los sistemas más comunes empleados para interactuar con los computadores son el teclado y el mouse. La interacción humano-computador (HCI *Human-Computer Interaction*) es un campo de investigación que busca desarrollar nuevas metodologías y técnicas que permitan mejorar la interacción entre humanos y computadores. En esta área de investigación, el reconocimiento de gestos de las manos juega un papel importante, ya que permite el desarrollo de sistemas de interacción más cómodos entre las personas y los computadores. Actualmente, el reconocimiento de gestos basado en visión es un campo de investigación muy activo en el ámbito de la HCI, con grandes retos en el desarrollo de sistemas que permitan una interacción entre humanos y computadores con más naturalidad (M. O. S. M. Elmezain, 2010).

El reconocimiento de gestos de las manos basado en visión tiene bastantes aplicaciones en el campo de investigación en HCI. Entre estas se cuentan: reconocimiento del lenguaje de señas, realidad virtual, control de aparatos de televisión y videojuegos, video vigilancia e interacción humano-robot (HRI). La HRI es un campo de investigación multidisciplinar bastante amplio cuyo objetivo es construir interfaces de comunicación para interactuar de manera fácil e intuitiva con un robot. El reconocimiento de gestos basado en visión, es utilizado con mayor frecuencia en aplicaciones de HRI, superando a otros métodos de comunicación como el sonido o el tacto. En el reconocimiento de gestos, los enfoques actuales tratan de emplear gestos más diversos para interactuar con los robots. Sin embargo, esto implica problemas de elevado de tiempo de procesamiento y baja tasa de reconocimiento (Nguyen-Duc-Thanh, Lee, & Kim, 2012).

El continuo avance tecnológico ha permitido el surgimiento de tecnologías para adquirir y procesar gestos en tiempo real. Sin embargo, la gran mayoría de sistemas de procesamiento se basan en arquitecturas de cómputo convencional, o en el mejor de los casos, se emplean dispositivos de procesamiento paralelo como las GPU. Aunque estos sistemas solucionan de forma apropiada el problema de tiempo de procesamiento mencionado anteriormente, tienen la desventaja de poseer un elevado consumo de

energía eléctrica y de requerir una conexión permanente con un computador. Esto restringe en gran medida su uso en aplicaciones de HRI (ya que la autonomía de los robots depende de la capacidad de sus baterías para almacenar energía), además de limitar considerablemente la portabilidad del sistema requerido. Por otro lado los arreglos de compuertas programables en campo o FPGA por sus siglas en inglés (*Field Programmable Gate Array*) han cobrado gran importancia durante los últimos años en el desarrollo de aplicaciones que requieren una elevada capacidad de procesamiento y bajo consumo de potencia, razón por la cual se han convertido en una excelente alternativa para el desarrollo de sistemas de reconocimiento gestos orientados a HRI.

En este trabajo se presenta el diseño e implementación en FPGA de un sistema de reconocimiento de gestos basado en visión estéreo, orientado a HRI. El sistema de reconocimiento está conformado por cuatro etapas principales: *i*) captura de los gestos, *ii*) pre-procesamiento, *iii*) extracción de características y *iv*) reconocimiento. El sistema realiza la captura y procesamiento de las imágenes estéreo en tiempo real gracias al diseño de una arquitectura de hardware que permite realizar procesamiento paralelo de alto rendimiento. El sistema de reconocimiento está diseñado para tele-operar un robot que puede realizar siete acciones básicas: avanzar, retroceder, girar a la izquierda, girar a la derecha, rotar 90º a la izquierda, rotar 90º a la derecha y detenerse. El sistema propuesto fue sintetizado en el dispositivo FPGA-SoC Cyclone V 5CSEMA5F31C6 de Altera, el cual está disponible en la tarjeta de desarrollo DE1-SoC.

El resto de este documento está organizada de la siguiente manera: En el capítulo 2 se presenta la revisión de literatura, en la que se discuten cada una de las etapas que conforman un sistema de reconocimiento de gestos típico. El capítulo 3 presenta algunos conceptos fundamentales sobre procesamiento de imágenes, visión estéreo, paralelismo y mapeo de algoritmos en FPGAs. En el capítulo 4 se hace una descripción detallada del sistema de reconocimiento desarrollado. El capítulo 5 describe el desarrollo de hardware propuesto para capturar y sincronizar las imágenes estereoscópicas. En el capítulo 6 se muestra el diseño e implementación en hardware de la etapa de pre-procesamiento del sistema de reconocimiento propuesto. En el capítulo 7 se exponen las características empleadas para describir un gesto, así como la implementación en hardware de dichas características. En el capítulo 8 se presenta el diseño del clasificador empleado en la etapa de reconocimiento. En el capítulo 9 se analizan los resultados obtenidos y en el capítulo 10 se detallan las conclusiones y los trabajos futuros.

2 Revisión de Literatura

El análisis de movimiento humano ha sido un área de investigación muy activa en la visión por computador, cuyo objetivo principal es segmentar, capturar y reconocer el movimiento humano automáticamente y en tiempo real, e incluso predecir las acciones humanas siguientes. Este tema de investigación puede ser aplicado ampliamente a varias áreas, como la vigilancia y seguridad de los espacios públicos, tales como centros comerciales, parques, estaciones de transporte público terrestre y aeropuertos. Por otra parte, el análisis automático de movimiento humano se puede utilizar en aplicaciones de Interacción Humana-Computador e Interacción Humana-Robot (HCI / HRI), realidad virtual, videojuegos, medicina entro otros (L. Chen, Wei, & Ferryman, 2013; Premaratne, 2014).

El reconocimiento de los gestos de las manos es un tema de investigación amplio, en el cual se han desarrollado diferentes técnicas de análisis para capturar, segmentar y reconocer automáticamente dichos gestos. Existe una cantidad considerable de trabajos desarrollados sobre este tema, los cuales presentan diferentes técnicas y métodos para su modelado, análisis y reconocimiento (Mitra & Acharya, 2007; Premaratne, 2014).

En los trabajos sobre reconocimiento de gestos de las manos presentados por (Moni & Shawkat Ali, 2009), (Suarez & Murphy, 2012) y (L. Chen et al., 2013), se exponen las etapas que componen un sistema de reconocimiento de gestos, tales como: la adquisición de la imagen, el pre-procesamiento, la extracción de características y el reconocimiento (L. Chen et al., 2013; Moni & Shawkat Ali, 2009; Premaratne, 2014; Suarez & Murphy, 2012). La Figura 1 muestra un diagrama de bloques que representa el flujo de la información a través de cada una de las etapas.

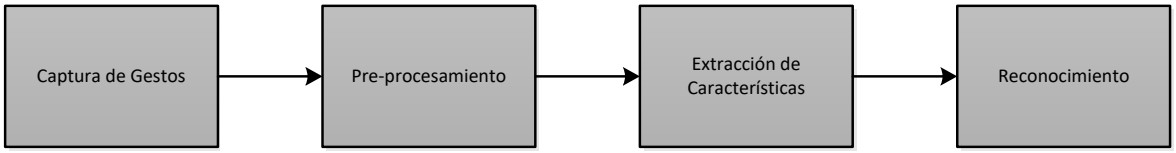


Figura 1. Etapas que componen un sistema de reconocimiento de gestos de las manos. Fuente autor.

2.1 Captura de Gestos de las Manos

La primera etapa del reconocimiento de gestos es la etapa de captura. En ésta se adquieren los gestos que van a ser objeto de análisis para una determinada aplicación. En la literatura existen múltiples métodos de captura de gestos entre los que se destaca el uso de guantes instrumentados (L. Chen et al., 2013). El empleo de guantes para la captura de gestos se remonta a la década de los años 70 con el desarrollo del primer prototipo llamado *Sayre Glove* (Premaratne, 2014). Durante los últimos 20 años los avances tecnológicos han permitido el desarrollo de este tipo de instrumentos para la

captura de gestos cuyas aplicaciones van desde videojuegos hasta animación de películas en 3D. Algunos de estos instrumentos comerciales desarrollados son: *Z*^{TM}*Glove*, *MIT Data Glove*, *CyberGlove* II, *CyberGlove* III, 5DT *Data Globe Ultra*, X-IST *Data Glove* y P5 *Glove* (Premaratne, 2014).

Aunque los guantes instrumentados son herramientas con gran potencial en el desarrollo de aplicaciones de reconocimiento de gestos de las manos, estos tienden a sufrir desgaste mecánico haciendo que con el tiempo las mediciones se vean afectadas. Además, el uso de guantes no permite que el usuario interactúe de forma natural con la aplicación final debido a que es un aditamento que debe llevarse sobre el cuerpo (Mitra & Acharya, 2007). En contraposición, los métodos basados en visión permiten al usuario interactuar con mayor naturalidad sin la necesidad de accesorios adicionales. Sin embargo, este método de captura implica otro tipo de exigencias como la detección de la manos, seguimiento de trayectorias, oclusiones de partes del cuerpo y principalmente la necesidad de hacer mayor calculo computacional (Mitra & Acharya, 2007).

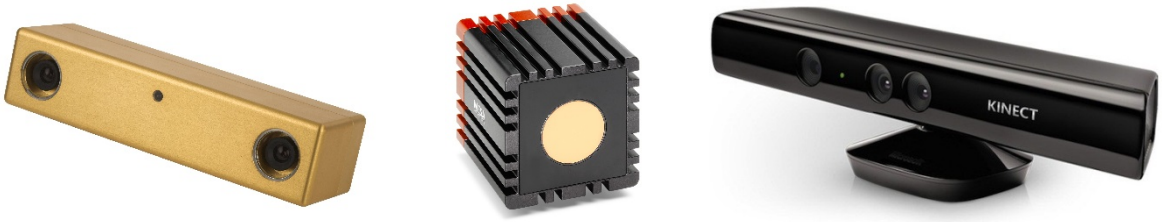
Para capturar los gestos empleando técnicas de visión, se emplean imágenes digitales las cuales contienen bastante información relacionada con el entorno. Sin embargo, dichas imágenes son sensibles a cambios en la iluminación y las tareas que incluyen la sustracción del fondo son difíciles de realizar robustamente (Aggarwal & Ryoo, 2011; L. Chen et al., 2013; Ji & Liu, 2010; Moeslund, Hilton, & Krüger, 2006; Poppe, 2010; Weinland, Ronfard, & Boyer, 2011). Por otra parte, las imágenes de profundidad tienen mayores ventajas sobre las imágenes de intensidad. En primer lugar, estas imágenes tienen muy buena invariabilidad frente a cambios en la iluminación. Y en segundo lugar, proveen una estructura tridimensional de las escenas, lo cual facilita las tareas de sustracción de fondo, segmentación y estimación de movimiento (L. Chen et al., 2013).

La captura de gestos empleando imágenes de profundidad se realiza mediante sensores de profundidad. En la actualidad existen tres tecnologías principales para realizar este tipo de sensado: cámaras estéreo, cámaras *ToF* (*Time of Flight*) y luz estructurada (L. Chen et al., 2013; Moni & Shawkat Ali, 2009; Poppe, 2010; Suarez & Murphy, 2012; Weinland et al., 2011). En la Figura 2 se presentan los tres tipos de sensores de profundidad. A la izquierda se observa el sistema de visión estéreo *bumblebee*, en el centro la cámara de *Mesa Imaging* basada en el sensor *ToF Swiss Ranger* SR4000, y a la derecha, se presenta el sistema Kinect de Microsoft. En la Tabla 1 se hace la comparación de las características principales de los tres métodos de sensado de profundidad. Cada uno de estos métodos de captura presenta dificultades. Por ejemplo, los sistemas de luz estructural solo funcionan bien en ambientes cerrados y no son adecuados para el desarrollo de aplicaciones móviles; los sistemas basados en *ToF* son costosos y las imágenes capturadas son de bajas resoluciones; las cámaras estereoscópicas dependen de la iluminación ambiente requiriendo un esfuerzo en el pre-procesamiento de las

imágenes. Sin embargo, este modelo puede usarse en aplicaciones móviles utilizando cámaras convencionales (L. Chen et al., 2013).

Tabla 1. Comparación de los sensores de profundidad (L. Chen et al., 2013)

Tipo de sensor	Cámaras estéreo	Time of Flight	Luz estructurada
Resolución	Alta. 640X480 o mas	Baja. 64X48 hasta 200X200	640X480
Velocidad	Depende de la velocidad de las cámaras.	Rápido	Rápido
Rango	Limitado por la línea base	5m a 10m	0.8m a 3.5m (espacios cerrados)
Resolución de profundidad	Depende de la línea base y resolución de las cámaras	Menor a 5mm	Menor a 1cm
Campo de visión	No limitado. Depende de los lentes de las cámaras	43ºvertical, 69º horizontal	43ºvertical, 57º horizontal
Huecos en el mapa de disparidad	Si	No	Si
Precio	Barato	Costoso	Barato



(a) (b) (c)

Figura 2. Sensores de profundidad. (a) Cámara estéreo (Pointgrey, 2016). (b) Sensor Time-of-Flight (ToF) (SwissRanger SR4000 MESA Imaging AG). (c) luz estructurada (Kinect). Fuente autor.

2.1.1 Cámaras estéreo

La visión estereoscópica está inspirada en la visión humana. Este tipo de visión obtiene la información tridimensional de la escena a partir de dos o más imágenes tomadas desde distintos puntos de vista, de la misma forma como lo hace la visión estéreo humana (L. Chen et al., 2013). La obtención del mapa de disparidad a partir de imágenes estéreo es un tema de investigación que se ha desarrollado desde los años 60 y ha permitido el desarrollo de productos para investigación tales como el sistema *bumblebee* de la compañía *PointGrey* (Pointgrey, 2016).

El cálculo del mapa de disparidad a partir de imágenes estereoscópicas, aún sigue siendo un desafío debido a la complejidad que involucra la geometría estéreo. Además, las imágenes estéreo son sensibles a cambios en la iluminación lo que incrementa la complejidad en la obtención de la correspondencia estéreo (Hartley & Zisserman, 2004). Por este tipo de dificultades, la reconstrucción del mapa de disparidad en tiempo real y en aplicaciones de la vida real aun es un reto para los investigadores en el área (L. Chen et al., 2013). En la actualidad, el uso de tecnologías de hardware reconfigurable como los FPGA brindan una alternativa a la solución a los inconvenientes que se tienen con este tipo de sistemas de visión (Greisen, Heinzle, Gross, & Burg, 2011).

El procesamiento en tiempo real de los sistemas de visión estéreo es una actividad difícil de satisfacer cuando se utilizan sistemas de cómputo convencionales. Por esta razón, la comunidad científica ha afrontado esta tarea mediante el uso de sistemas de lógica reconfigurable tales como las FPGAs y los ASICs, así como mediante unidades de procesamiento grafico tales como las GPUs (Greisen et al., 2011).

Dentro de los trabajos más relevantes en esta área se encuentran por una parte, el desarrollo de técnicas de procesamiento para FPGAS tales como la implementación de algoritmos basados en arquitecturas pipeline y arreglos sistólicos que emplean matrices de elementos de básicos de procesamiento, y por otra parte, los métodos para el cálculo de los mapas de disparidad tales como SSD (*sum of squared differences*), SAD (*sum of Absolute diferences*), NCC (*normalized cross correlation*), entre otros. (Banz, Hesselbarth, Flatt, Blume, & Pirsch, 2010; Bendaoudi & Khouas, 2013; Ding, Du, Wang, & Liu, 2010; Gudis, Wal, Kuthirummal, & Chai, 2012; Ibarra-Manzano, Almanza-Ojeda, Devy, Boizard, & Fourniols, 2009; Isakova, Başak, & Sönmez, 2012; Ohmura, Mitamura, Takauji, & Kaneko, 2010; Sungchan Park & Jeong, 2007; Woodfill, Gordon, Jurasek, Brown, & Buck, 2006).

En (Banz et al., 2010) se plantea al desarrollo de un sistema para generar mapas de disparidad de imágenes VGA de 640X480 pixeles, con un rango de 128 pixeles de disparidad, operando bajo condiciones reales a una taza de 30 fps (*frames per second*) y una frecuencia de 39MHz, e implementado en un dispositivo FPGA Virtex 5. En (Ding et al., 2010) se presenta la implementación de un algoritmo basado en el método NCC para el cálculo de disparidad de imágenes estéreo de 512x512 pixeles a 70 fps, implementado en un dispositivo FPGA. En (Ibarra-Manzano et al., 2009) se presenta la implementación en hardware reconfigurable de un algoritmo de visión esterero denso basado en la transformada Census. En (Shan et al., 2012) se presenta el método SAD mediante el que se procesan imágenes de 1280x1024 a 23 fps estéreo para obtener el mapa de disparidad en tiempo real en una FPGA. En el trabajo realizado por (Fife & Archibald, 2013) se presenta la implementación en FPGA de una arquitectura con uso óptimo de recursos, que permite hacer el cálculo del mapa de disparidad en tiempo real de imágenes estéreo. Para el desarrollo de dicha arquitectura se utilizó la transformada Census dispersa. El sistema implementado tiene la capacidad de procesar los algoritmos de correspondencia estéreo a una velocidad de 500MHz.

2.1.2 Cámaras ToF (*Time of Flight*)

Aunque la visión estéreo en los humanos funciona muy bien, hacer que un sistema artificial funcione de la misma manera requiere un esfuerzo enorme en el desarrollo de técnicas y algoritmos adecuados para tal fin. Por esta razón, se han realizado avances tecnológicos más robustos tales como el desarrollo de la cámaras denominadas *ToF* (*Time of flight*). A diferencia de las cámaras estéreo, las cámaras *ToF* emplean una sola

cámara y una fuente de luz que envía pulsos a las superficies que se encuentren en la escena. Estas cámaras miden el tiempo que tardan los pulsos en reflejarse sobre las superficies. Para calcular la distancia a la que se encuentran los objetos en una escena se emplea el tiempo medido por la cámara y la velocidad de la luz (L. Chen et al., 2013).

En comparación con otros dispositivos de escaneo láser 3D, las cámaras TOF son más económicas y pequeñas. La mayoría de los dispositivos comerciales actuales utilizan una señal de luz infrarroja modulada sinusoidalmente, y la distancia se calcula utilizando el desfase de la señal reflejada en un detector estándar CMOS o CCD (Kolb, Barth, & Koch, 2008). Las principales ventajas de las cámaras *ToF* son su alta velocidad y la posibilidad de obtener un mapa de disparidad denso que cubre cada pixel. El principal inconveniente práctico es su elevado precio, aunque son todavía más económicas que otros dispositivos de escaneado 3D. El inconveniente técnico más importante es su baja resolución (L. Chen et al., 2013; Premaratne, 2014).

2.1.3 Sensores de luz estructurada

En el año 2010 Microsoft lanzo Kinect, un dispositivo de sensado de gestos basado en visión, orientado al uso doméstico únicamente (L. Chen et al., 2013). Kinect está compuesto por una cámara RGB convencional y un sensor infrarrojo de profundidad (L. Chen et al., 2013; Premaratne, 2014). La cámara y el sensor en conjunto tienen una resolución de 640X480 pixeles con una velocidad de 30 fps, y su rango de medición esta entre 0.8m a 3.5m con una resolución inferior a 1cm (L. Chen et al., 2013; Premaratne, 2014). Sin embargo, la percepción de profundidad solo alcanza una resolución máxima de 320X240 pixeles (Premaratne, 2014).

Kinect calcula el mapa de disparidad empleando la técnica de luz estructurada. Su funcionamiento es similar a la visión estéreo, pero en este caso una de las cámaras es reemplazada por una fuente de luz, la cual genera un patrón conocido. La ventaja de los sistemas que emplean luz estructurada sobre el sistema *ToF* es su bajo precio, por lo que son ideales para el desarrollo de aplicaciones de la vida cotidiana (L. Chen et al., 2013)

2.2 Pre-procesamiento

Los sistemas de captura de gestos basados en visión permiten obtener muchísima información del entorno, tales como color, textura y dimensiones de los objetos presentes en la escena, etc. Cuando se captura una escena empleando un sistema de visión, se tiene como resultado tanto información deseada como información no deseada. Por ejemplo si el sistema de visión está enfocado en las manos de una persona la cual puede realizar un gesto, el sistema de visión también observa los demás objetos que están presentes en la escena los cuales son información no deseada. Ya que el sistema de visión está desarrollado para responder a gestos, éste debe extraer únicamente la información que necesita y descartar aquella información que no representa utilidad. Este proceso se conoce como pre-procesamiento (Premaratne, 2014).

Enfocarse únicamente en la información deseada resulta un reto bastante grande para los sistemas de visión. En el caso de los gestos de las manos, identificar las diferentes posiciones de la mano, en diferentes gestos, con tonos de piel variados y con diferentes condiciones de iluminación, representa un problema bastante difícil (Premaratne, 2014).

Uno de los desafíos más grandes en el reconocimiento de gestos es la extracción solamente del gesto en una escena compleja. Uno de los métodos de extracción de gestos ampliamente utilizado es la segmentación por color de piel. La dificultad del reconocimiento se incrementa cuando el fondo contiene regiones con tonos similares a la piel. Para solucionar este problema se emplea el mapa de disparidad junto con la segmentación de color de piel, de esta forma aquellos objetos que se encuentran alejados de las manos pueden eliminarse más fácilmente (Premaratne, 2014).

Uno de los métodos de pre-procesamiento más utilizado en la literatura es la eliminación del fondo en imágenes de profundidad. Este método consiste en segmentar el área de la zona de interés del resto de la escena mediante la restricción de la distancia. Por ejemplo, para el reconocimiento de gestos de la mano un sencillo umbral permite segmentar las manos. Varios sistemas utilizan la información de profundidad para extraer el primer plano de la escena (Bellmore, Ptucha, & Savakis, 2011; Benko & Wilson, 2009; Bergh et al., 2011; Biswas & Basu, 2011; Breuer, Eckes, & Müller, 2007; C. P. Chen, Chen, Lee, Tsai, & Lei, 2011; M Elmezain, Al-Hamadi, Appenrodt, & Michaelis, 2008; Fujimura & Liu, 2006; Grammalidis, Goussis, Troufakos, & Strintzis, 2001; Guomundsson, Larsen, Aanaes, Pardas, & Casas, 2008; Kollorz, Penne, Hornegger, & Barke, 2008; Kurakin, Zhang, & Liu, 2012; X. L. X. Liu & Fujimura, 2004; Mo & Neumann, 2006; Muñoz-Salinas, Medina-Carnicer, Madrid-Cuevas, & Carmona-Poyato, 2008; Nickel, Scemann, & Stiefelhagen, 2004; Oikonomidis, Kyriazis, & Argyros, 2011; C. B. Park & Lee, 2011; Ren, Meng, & Yuan, 2011; L A Schwarz, Mkhitaryan, Mateus, & Navab, 2011; Loren Arthur Schwarz, Mkhitaryan, Mateus, & Navab, 2012; Suryanarayan, Subramanian, & Mandalapu, 2010; Zhang & Tian, 2013).

En (Benko & Wilson, 2009) se describe la construcción de un sistema de superficie interactiva de mano alzada llamada *DepthTouch* para realizar un seguimiento gestos de las manos utilizando una cámara de profundidad *ZSense* en el que se eligió una región de interés 3D para segmentar las manos del usuario. En los trabajos realizados por (Bergh & Gool, 2011; M Elmezain, Al-Hamadi, Appenrodt, et al., 2008; M Elmezain, Al-Hamadi, Pathan, & Michaelis, 2009; Mahmoud Elmezain & Al-Hamadi, 2012; Nickel et al., 2004; Nickel & Stiefelhagen, 2007; Oikonomidis et al., 2011; Yin & Xie, 2003) se extraen las manos de las imágenes combinando la información de color de la piel y la información de profundidad.

En otros trabajos se describe la construcción un modelo de fondo empleando únicamente las imágenes del fondo, las cuales son utilizadas para extraer la forma del cuerpo humano. (Loren Arthur Schwarz et al., 2012) realiza la resta sencilla de un fondo estático a la escena. (Jansen, Temmermans, & Deklerck, 2007) modelan el fondo mediante el promedio de varios *frames* consecutivos. (Guomundsson et al., 2008) modelan el fondo mediante un modelo probabilístico gaussiano.

Debido a que las imágenes en color y la información del mapa de disparidad pueden ser muy ruidosas, es necesario el uso de técnicas de reducción de ruido. Algunos métodos empleados son el filtro de mediana y las operaciones morfológicas (Breuer et al., 2007; Nickel et al., 2004; Nickel & Stiefelhagen, 2007; S Park, Yu, Kim, Kim, & Lee, 2012).

2.3 Extracción de Características

La etapa de extracción de características tiene como finalidad extraer información que permita clasificar los gestos de las manos adecuadamente. Por ejemplo, si se tiene una porción de la imagen en la cual para un observador humano se está realizando el gesto de deténgase o “Stop”, entonces es importante extraer la información que permite que dicho gesto sea único comparado con otros gestos. El éxito de cualquier sistema de clasificación radica en el desarrollo de características únicas y robustas. Sin embargo, los gestos de las manos pueden tener cierto grado de variabilidad. Esto hace que el desarrollo de sistemas artificiales que permitan interpretar adecuadamente dichas variaciones no sea una tarea trivial. Por lo tanto, se debe desarrollar un conjunto sólido de características que describa un gesto de forma única con el fin de obtener un reconocimiento confiable (Premaratne, 2014).

Existen diferentes enfoques para extraer características que permiten una clasificación adecuada, disminuyendo la cantidad de falsos positivos y negativos. Dentro de los métodos de extracción de características podemos encontrar: extracción de características basado en histogramas de orientación del gradiente, extracción de características mediante momentos invariantes, extracción de características basadas en análisis de componentes principales entre otros (Premaratne, 2014).

La trayectoria de un gesto realizado con la mano es un patrón espacio-temporal formado con los centroides de la mano en una secuencia consecutiva de imágenes. La trayectoria del gesto puede modelarse a partir de características como la posición, velocidad y orientación de la mano (C. P. Chen et al., 2011; M Elmezain, Al-Hamadi, Appenrodt, et al., 2008; M Elmezain et al., 2009; Mahmoud Elmezain & Al-Hamadi, 2012). En (M Elmezain, Al-Hamadi, Appenrodt, et al., 2008) se presenta un método de extracción de características que consiste en extraer la orientación del gesto calculando dicho parámetro a partir de dos puntos consecutivos pertenecientes a la trayectoria de dicho gesto. (M Elmezain et al., 2009; Mahmoud Elmezain & Al-Hamadi, 2012) presentan la extracción de características basada en la localización, orientación y velocidad de los

puntos de la trayectoria del gesto. En el mencionado trabajo se extraen dos tipos de características de localización; la primera mide la distancia entre el centro del gesto y cada uno de los puntos que lo conforman; la segunda, calcula la distancia entre el primer punto del gesto y cada uno de los puntos del gesto. La orientación es calculada mediante los vectores de desplazamiento entre cada uno de los puntos del gesto y el centro del mismo. La velocidad es calculada como la distancia euclidea entre dos puntos consecutivos. Estas características son calculadas en coordenadas rectangulares y en coordenadas polares.

Un método de extracción de características bastante conocido es el método basado en Descriptores de Fourier (FD). Este método ha sido empleado principalmente para describir formas; sin embargo, también brinda la posibilidad de describir gestos de las manos (Premaratne, 2014). Algunos trabajos presentan métodos de extracción de características basados en los FD. (Lin & Hwang, 1987) presentan una representación alternativa de las series de Fourier empleando características elípticas de Fourier. En su enfoque, una forma puede ser representada como un conjunto de elipses las cuales son invariantes a la rotación y traslación. (Lin & Jungthirapanich, 1990) muestran el desarrollo de descriptores elípticos de Fourier 3D a partir de descriptores elípticos de Fourier 2D.

En la literatura se describen otros métodos para la extracción de características. (C. B. Park & Lee, 2011) obtienen la información temporal del gesto junto con una función de distribución de probabilidad (PDF) del mapeo de la posición de la mano. Cuando una persona apunta en una dirección, las posiciones de la mano en el espacio tridimensional son modeladas como una mezcla finita de distribuciones gaussianas. (Yang, Jang, Beh, Han, & Ko, 2012) emplearon un vector de características tridimensional que contiene coordenadas consecutivas empleando el sistema de coordenadas esféricas con el fin de obtener la trayectoria de la mano compuesta por líneas y trazos curvos. (Bellmore et al., 2011) utilizaron un cuadro limitador para el área facial en RGB, la cual es convertida a una imagen de luminancia y transformada en una forma canónica de 50X60 pixeles. Los parámetros de normalización empleados son $\mu=128$ y $\sigma=100$. (Zhang & Tian, 2013) realizan la extracción características mediante el método de histograma de la orientación del gradiente (HOG), aplicando posteriormente el método de pirámide temporal dinámica a las características extraídas. (Muñoz-Salinas et al., 2008) presentan el análisis de componentes principales aplicado a las siluetas de un gesto.

2.4 Reconocimiento

La meta del reconocimiento de gestos de las manos se cumple cuando se acoplan una efectiva extracción de características con una clasificación altamente eficiente. En el reconocimiento de los gestos de las manos, separar los gestos en clases pre-asignadas convierte el problema de clasificación en un problema de clasificación multi-clase. Para que el proceso de clasificación sea efectivo, la extracción de características debe

realizarse bajo condiciones adecuadas de forma que las clases se puedan diferenciar una de otra (Premaratne, 2014).

Diferenciar un gesto de otro no es un problema grave cuando los gestos se diferencian bastante en su ejecución. Sin embargo, cuando el usuario realiza gestos que son muy similares, pero su significado es diferente, se presentan dificultades para su correcto reconocimiento, de tal forma que la cantidad de gestos que el sistema puede reconocer con exactitud es muy limitada. Para afrontar este inconveniente, se han desarrollado varios enfoques que permiten resolver el problema de clasificación de forma eficiente. Dichos métodos de reconocimiento se pueden clasificar en dos grupos: Clasificadores Lineales y Clasificadores no Lineales (Premaratne, 2014).

Los clasificadores lineales permiten diferenciar dos o más clases empleando rectas que se usan para establecer una separación entre las características de cada clase. Los clasificadores lineales más comunes son: el Perceptron, las máquinas de soporte vectorial lineales (SVM), el discriminante lineal de Fisher (LDA) y el clasificador Naive Bayes (Premaratne, 2014). Los clasificadores no lineales pueden realizar una mejor generalización cuando los datos de entrada tienen ruido. Estos clasificadores son ideales en presencia de datos o características multidimensionales. Algunos métodos de clasificación no lineales son: Redes Neuronales Artificiales multicapa (ANN), Maquinas de soporte vectorial no lineales (SVM), Modelos Ocultos de Markov (HMM) y el método de los K-vecinos más cercanos (KNN) (Mitra & Acharya, 2007; Premaratne, 2014; Suarez & Murphy, 2012).

Uno de los métodos más ampliamente utilizado en el reconocimiento de gestos es el de los Modelos Ocultos de Markov (HMM). Aunque dicho método fue empleado en sus inicios en el reconocimiento de voz (Rabiner, 1989), debido a la similitud que existe entre el reconocimiento de voz y el reconocimiento de gestos, este método se ha empleado también de forma efectiva para la solución de problemas de reconocimiento de gestos. Se puede definir un HMM como una colección de estados finitos conectados por transiciones, en el que cada estado se caracteriza por tener dos conjuntos de probabilidades: las probabilidades de transición y las probabilidades (continuas o discretas) de emitir un símbolo de salida de un alfabeto finito predefinido (Premaratne, 2014; Rabiner, 1989). Dentro de los trabajos que presentan los HMM como método de reconocimiento de gestos de las manos se encuentran: El trabajo presentado por (M Elmezain, Al-Hamadi, Appenrodt, et al., 2008) expone el uso HMMs para el reconocimiento de los números arábigos del 0 al 9. Los autores emplearon 10 HMMs, uno por cada uno de los gestos a reconocer. Las características extraídas se ingresan a los 10 modelos empleados para evaluar el sistema de reconocimiento. El modelo que obtiene la mayor probabilidad, es el que corresponde al gesto realizado. El sistema desarrollado tiene una tasa de reconocimiento de 98.94%. Asimismo, (M Elmezain et al., 2009) presenta el uso de HMM para el reconocimiento de las letras A-Z y los números 0-9. El procedimiento es similar al

de la descripción anterior. El sistema desarrollado tiene una tasa de reconocimiento de 98.33%. Otros trabajos con desarrollos similares son (Al-Rousan, Assaleh, & Tala'a, 2009; Bansal, Saxena, Desale, & Jadhav, 2011; F.-S. Chen, Fu, & Huang, 2003; M. O. S. M. Elmezain, 2010; Fahn & Chu, 2011; H. K. Lee & Kim, 1999; X. L. X. Liu & Fujimura, 2004; Muñoz-Salinas et al., 2008; Nickel & Stiefelhagen, 2007; C. B. Park & Lee, 2011; Wilson & Bobick, 1999; Yang et al., 2012).

El reconocimiento de gestos involucra, en algunos casos, dos o más métodos de reconocimiento para brindar una mayor confiabilidad del sistema. (Nickel & Stiefelhagen, 2007) presentan el desarrollo de un sistema de reconocimiento de gestos con aplicación a la Interacción Hombre-Robot (HRI), empleando Redes Neuronales Artificiales (ANN) y Modelos Ocultos de Markov (HMM). Las ANN realizan el reconocimiento de la orientación de la cabeza para estimar lo que la persona quiere enfocar. Los HMM son utilizados para modelar un gesto de las manos mediante tres fases: inicio, sostenimiento, y finalización. El sistema de reconocimiento desarrollado posee una tasa de reconocimiento superior al 90%. Por su parte (C. B. Park & Lee, 2011) exponen el desarrollo de un sistema de reconocimiento de gestos orientado a HRI empleando dos HMM en cascada. La primera etapa tiene un HMM cuya función es estimar la posición de la mano, asignando una posición más precisa mediante el modelado de las características cinemáticas de la acción. Las coordenadas 3D resultantes se utilizan como entrada en la segunda etapa de HMM, la cual discrimina los gestos de "apuntar con el dedo" de otros gestos. El sistema desarrollado tiene una tasa de reconocimiento superior al 89 %.

Otro método de clasificación empleado en el reconocimiento de los gestos de las manos es el método *Support Vector Machine* (SVM). Un SVM es un método de aprendizaje supervisado el cual también es un clasificador lineal que maximiza la distancia entre las líneas de decisión. La principal ventaja de los SVM es que empelan *kernels* para transformar datos no lineales a clústeres de datos separables dentro de un nuevo espacio de características (Premaratne, 2014). Dentro de los trabajos de reconocimiento de gestos que emplean este clasificador podemos encontrar a (Biswas & Basu, 2011; Muñoz-Salinas et al., 2008; Tang, 2011; Zhang & Tian, 2013). (Biswas & Basu, 2011) emplea un clasificador SVM multi-clase, el cual está en la capacidad de clasificar correctamente 8 gestos. (Rahman & Afrin, 2013) proponen un sistema de reconocimiento de gestos de las manos con aplicación a HCI empleando SVM. El sistema emplea un vector de características obtenido mediante la Transformada Wavelet Biortogonal. Dichas características ingresan a un SVM multiclase el cual está en la capacidad de clasificar los gestos de las manos en 10 clases diferentes con una tasa de reconocimiento del 92%.

Adicionalmente, en la literatura se reportan otros métodos de reconocimiento como los descritos a continuación. En (Mahmoud Elmezain & Al-Hamadi, 2012) se presenta el método *Latent-Dynamic Conditional Random Fields* (LDCRFs). Los LDCRFs son modelos gráficos no dirigidos que se han desarrollado para el etiquetado de datos secuenciales, lo

que permite emplearlos en reconocimiento de gestos dinámicos de las manos. En (Reyes, Domínguez, & Escalera, 2011) se representa un modelo del cuerpo humano mediante un vector de características definido por 15 articulaciones correspondientes a un modelo de esqueleto humano 3D empleando Kinect. Los autores usan el metodo *Dynamic Time Warping* (DTW) con la función de ponderado automático en cada conjunto para lograr el reconocimiento de la acción en tiempo real. Del mismo modo, (Sempena, Maulidevi, & Aryan, 2011) también usan Kinect para obtener el modelo del esqueleto del cuerpo humano, y DTW como método de reconocimiento. En este caso, las articulaciones del esqueleto son representadas usando cuaterniones para formar un vector de características de 60 elementos para las 15 articulaciones. En (Kollorz et al., 2008) se presenta el desarrollo de un sistema de reconocimiento de gestos estáticos de las manos mediante el método de los K-vecinos cercanos. En (Fujimura & Liu, 2006; Mo & Neumann, 2006) se presenta el desarrollo de sistemas de reconocimiento de gestos de las manos basados en reglas difusas.

2.5 Reconocimiento de Gestos de las Manos e Implementación en Hardware.

La mayor parte de los sistemas de reconocimiento de gestos de las manos reportados en la literatura son desarrollados principalmente en software y ejecutados en computadores convencionales, lo cual restringe su uso esencialmente al campo de las aplicaciones HCI. Los computadores convencionales no son la plataforma más adecuados para realizar procesamiento intensivo de imágenes en alta resolución y con un alto *framerate* (Shi & Tsui, 2007). Por otra parte, existen algunos trabajos en la literatura que describen el desarrollo de sistemas de reconocimiento de gestos de las manos empleando hardware reconfigurable como las FPGAs. Un algoritmo de procesamiento de imágenes implementado en una FPGA funciona mejor que uno similar ejecutado en una arquitectura de computo convencional debido principalmente al paralelismo que ofrece la arquitectura de las FPGAs, la cual hace posible inclusive el procesamiento de imágenes en tiempo real (S. H. Lee, SNC, & Siew, 2011).

Además del paralelismo que ofrecen los dispositivos FPGA para aplicaciones de procesamiento de imágenes en tiempo real, estos dispositivos son ideales para el desarrollo de aplicaciones de procesamiento embebido de imágenes principalmente por su bajo consumo de potencia, facilidad de uso, bajo costo, y tamaño comparadas con los sistemas de cómputo convencionales (GPU,CPU,DSP) (Mattoccia, 2013). Esto permite que los dispositivos FPGA sean ideales para el desarrollo de sistemas portátiles de reconocimiento de gestos basados en visión; navegación autónoma de robots; aplicaciones de apoyo para personas invidentes; aplicaciones para personas con discapacidad auditiva, etc.

A pesar de las ventajas de las FPGAs, el proceso de implementación de algoritmos, especialmente aquellos de visión por computador, no es una tarea sencilla como puede ser en una CPU con lenguajes tradicionales de alto nivel (Mattoccia, 2013). Esto se debe a que los algoritmos están desarrollados para ejecutarse de manera secuencial en el computador, por lo cual se deben desarrollar una serie de etapas que permitan transformar dichos algoritmos a su forma paralela de ejecución, para convertirlos en circuitos digitales compuestos por bloques funcionales, como memorias, flip-flops, registros etc. (Berkeley Design Technology, 2005). Por esta razón el desarrollo de aplicaciones de reconocimiento de gestos de las manos basadas en visión, e implementadas en FPGAs es un gran desafío principalmente cuando existen requerimientos de tiempo real.

En las Tablas 2 hasta la 6 se presenta un resumen de los trabajos más representativos de reconocimiento de gestos de las manos. En éstas se exponen las técnicas empleadas por los autores en cada una de las etapas que conforman un sistema de reconocimiento de gestos, junto con sus aplicaciones e implementación. Dentro de las implementaciones en hardware se puede apreciar que la mayoría está orientada al reconocimiento de gestos estáticos empleando únicamente una cámara digital.

Entre los trabajos de reconocimiento de gestos estáticos de las manos implementados en hardware se encuentran:(Bonato et al., 2004; Guerrero-Balaguera & Perez-Holguin, 2015; Jena, Majhi, Gupta, Sehrawat, & Khosla, 2012; S. H. Lee et al., 2011; Shi, Taib, & Lichman, 2006). En (Bonato et al., 2004) se presenta el desarrollo de un sistema de reconocimiento de gestos aplicados a HRI. El sistema emplea una cámara digital para captura de los gestos. En la etapa de pre-procesamiento se aplica la segmentación por color de piel para extraer la forma de la mano. En la etapa de extracción de características se recorta y comprime la imagen para describir la forma de la mano mediante un vector compuesto por 768 bits. En la etapa de reconocimiento se emplea una ANN. El sistema funciona en tiempo real con un *framerate* de 30fps. En (Guerrero-Balaguera & Perez-Holguin, 2015) se presenta un sistema de reconocimiento de gestos aplicado al reconocimiento de la Lengua de Señas Colombiana. El sistema emplea una cámara digital para la captura de los gestos. En la etapa de pre-procesamiento se aplica segmentación basada en el umbral de la componente de color rojo y la operación morfológica de apertura. Las características empleadas consisten en un vector que contiene la proyección de la forma de la mano sobre los ejes vertical y horizontal de la imagen. En la etapa de reconocimiento se empleó una ANN. En (Jena et al., 2012) se presenta el desarrollo de un sistema de reconocimiento de gestos de las manos con aplicación a HCI. El sistema emplea segmentación por color de piel para extraer la forma de la mano del fondo. Las características empleadas para describir los gestos son: área de la forma de la mano, perímetro de la forma de la mano, y la dirección del pulgar. Como método de reconocimiento emplea la distancia euclidea.

En cuanto al reconocimiento de gestos dinámicos de las manos implementados en hardware podemos encontrar los trabajos de (Cho, Li, & Chen, 2012; Shi & Tsui, 2007). En (Cho et al., 2012) se presenta el desarrollo de un sistema de reconocimiento de gestos con aplicación a HCI. Dicho sistema de reconocimiento emplea segmentación por color de piel para extraer la forma de la mano del fondo. Las características empleadas se basan en 16 posiciones posibles para describir la trayectoria de un gesto. En la etapa de reconocimiento se emplean HMMs, uno por cada gesto. El sistema fue implementado en FPGA con capacidad de reconocimiento en tiempo real con un *framerate* de 30fps. (Shi & Tsui, 2007) presentan el desarrollo de un sistema de reconocimiento de gestos con aplicación a HCI. En este trabajo se empleó la segmentación por color de piel para extraer la forma de la mano del fondo. En la etapa de extracción de características se empleó la trayectoria del gesto a partir del centro de gravedad de la mano. Como método de reconocimiento se empleó una ANN. El sistema fue implementado en FPGA y su funcionamiento se realiza en tiempo real con un *framerate* de 15fps

De acuerdo con la revisión de literatura realizada se puede observar que un sistema de reconocimiento de gestos de las manos basado en visión artificial consta de 4 etapas principales: captura, pre-procesamiento, y reconocimiento. Para extraer de forma eficaz el gesto del fondo se emplea la información de profundidad junto con la segmentación de color de piel. Esto permite determinar con mayor facilidad el gesto que se está realizando incluso en escenas con fondos complejos. En las etapas de extracción de características y reconocimiento se emplea una gran cantidad de algoritmos y metodologías que buscan mejorar la clasificación de un gesto realizado. Las características más representativas son los descriptores de Fourier, momentos invariantes de Hu; la orientación, la localización y la velocidad de la trayectoria de un gesto; PCA; y los descriptores de la forma de la mano. Como métodos de reconocimiento predominan los HMM, SVM y ANN.

En el presente trabajo se propone el diseño e implementación hardware de un sistema con la capacidad de reconocer 6 gestos dinámicos de las manos con aplicación a HRI, empleando el mapa de disparidad, la segmentación de color de piel y la operación morfológica de apertura en la etapa de pre-procesamiento. Para le etapa de extracción de características se emplea la orientación de la trayectoria del gesto. En la etapa de reconocimiento se plantea el uso de HMMs. Para obtención del mapa de disparidad se emplea un sistema de visión estereoscópico ideal para implementación en hardware debido al costo computacional que este requiere. La implementación del sistema se realiza en FPGA SoC con la capacidad de procesamiento en tiempo real con un *framerate* superior a 30fps.

Tabla 2. Resumen de los trabajos más representativos en reconocimiento de gestos.

Paper	Aplicación	Tipo de Gesto	No de Gestos	Método de Captura	Pre-procesamiento	Extracción de Características	Método de Reconocimiento	Implementación	Tiempo Real	fps
(Bergh et al., 2011)	HRI	Estático	-	Kinect	Depth Segm	Orientación	ANMM	SW	SI	-
(Bergh & Gool, 2011)	HCI	Dinámico	6	ToF	Skin Segm Depth Segm	Apariencia de la mano	ANMM	SW	SI	-
(Biswas & Basu, 2011)	HCI	Dinámico	8	Kinect	Sustracción del fondo	ROI	SVM	SW	N	-
(Breuer et al., 2007)	HCI	Dinámico	-	ToF	Reducción de ruido Depth Segm	PCA	Hausdorff distance	SW	SI	3
(Elmezain, Al-Hamadi, Appenrodt, et al., 2008)	-	Dinámico	10	Stereo	Skin Segm Depth Segm	Hand tracking Orientación	HMM	SW	SI	15
(M Elmezain et al., 2009)	HCI	Dinámico	36	Stereo	Skin Segm Depth Segm	Hand tracking Orientación Localización Velocidad	HMM	SW	SI	15
(Mahmoud Elmezain & Al-Hamadi, 2012)	HCI	Dinámico	36	Stereo	Skin Segm Depth Segm	Hand tracking Orientación Localización Velocidad	LDCRFs	SW	SI	15
(Kollorz et al., 2008)	HCI	Estático	12	ToF	Crecimiento de regiones	Proyección de la mano en los ejes x e y	KNN	SW	N	-
(Kurakin et al., 2012)	HCI	Dinámico	26	Kinect	Depth Segm	Características de ocupación de celda Características de siluetas	HMM	SW	SI	10
(X. L. X. Liu & Fujimura, 2004)	HRI	Dinámico	10	ToF	Depth Segm	Forma, Localización, Trayectoria, Orientación, Velocidad	HMM	SW	N	-
(Muñoz-Salinas et al., 2008)	HCI	Dinámico	10	Stereo	Depth Segm	Características de siluetas de profundidad	HMM SVM	SW	N	-

Tabla 3. Resumen de los trabajos más representativos en reconocimiento de gestos (Continuación).

Paper	Aplicación	Tipo de Gesto	No de Gestos	Método de Captura	Pre-procesamiento	Extracción de Características	Método de Reconocimiento	Implementación	Tiempo Real	fps
(Nickel et al., 2004)	HRI	Dinámico	8	Stereo	Skin Segm Morfologia	Vector de posiciones de la mano	HMM	SW	NO	-
(Nickel & Stiefelhagen, 2007)	HRI	Dinámico	-	Stereo	Skin Segm Morfologia	Vector de posiciones de la mano Orientación de la cabeza	ANN HMM	SW	NO	-
(Oikonomidis et al., 2011)	HCI	Dinámico	-	Kinect	Skin Segm Depth Segm	Parámetros de modelo de la mano	PSO	SW	SI	15
(C. B. Park & Lee, 2011)	HRI	Dinámico	-	Stereo	Connected component labeling	Velocidad y posición	HMM	SW	SI	10
(Rahman & Afrin, 2013)	HCI	Estático	10	Mono	Reducción de ruido Segmentación color de piel Detección de bordes	Biorthogonal Wavelet Transform	SVM	SW	NO	-
(Ren et al., 2011)	HCI	Estático	10	Kinect	Skin Segm Depth Segm	Forma de la mano	FEMD	SW	NO	-
(Reyes et al., 2011)	HCI	Dinámico	5	Kinect	Modelo esqueleto del cuerpo Humano	Descriptor de 14 articulaciones	DTW	SW	NO	-
(Sempena et al., 2011)	HCI	Dinámico	7	Kinect	Modelo esqueleto del cuerpo Humano	Descriptor de orientación de articulaciones	DTW	SW	NO	-
(Suryanarayan et al., 2010)	HCI	Estático	6	ToF	Umbralizacion de Otsu	PCA	SVM	SW	SI	16
(Yang et al., 2012)	HCI	Dinámico	8	Kinect	CAMSHIFT	Vector de coordenadas tridimensionales empleando el sistema de coordenadas esféricas	HMM	SW	SI	14

Tabla 4. Resumen de los trabajos más representativos en reconocimiento de gestos (Continuación).

Paper	Aplicación	Tipo de Gesto	No de Gestos	Método de Captura	Pre-procesamiento	Extracción de Características	Método de Reconocimiento	Implementación	Tiempo Real	fps
(Zhang & Tian, 2013)	HCI	Dinámico	12	Kinect	Enhanced Depth Motion Map	Método de Pirámide temporal dinámica	SVM	SW	NO	-
(Al-Rousan et al., 2009)	ASL	Estático	30	Mono	background removal,	Transformada dicreta del coseno (DCT)	HMM	SW	NO	-
(Althoff, Lindl, & Walchshäusl, 2005)	HCI	Dinámico	17	Mono	Adaptive threshold segmentation Operaciones morfológicas	Trayectoria del gesto	HMM	SW	SI	-
(Bansal et al., 2011)	HCI	Dinámico	8	Mono	Skin Segm	adjacency matrix	HMM	SW	NO	-
(Binh, Shuichi, & Ejima, 2005)	ASL	Dinámico	36	Mono	Skin Segm	Kalman filter Descriptores de movimiento y forma de la mano	P2-DHMMs	SW	SI	25
(Bonato et al., 2004)	HRI	Estático	7	Mono	Skin Segm Compresion de la imagen Centrado de la imagen	Un vector de 768 bits, describe la forma de la mano	ANN	HW	SI	30
(F.-S. Chen et al., 2003)	TSL	Dinámico	20	Mono	Background subtraction Otsu threshold Skin Segm Edge detection	Fourier descriptor (FD) Motion analisys	HMM	SW	SI	30
(Z.-H. Chen, Kim, Liang, Zhang, & Yuan, 2014)	HCI	Estático	13	Mono	background subtraction	Finger detection	Rule classifier	SW	NO	-
(Cho et al., 2012)	HCI	Dinámico	6	Mono	Median Filter Skin Segm	Características basadas en 16 posiciones del gesto	HMM	HW	SI	30

Tabla 5. Resumen de los trabajos más representativos en reconocimiento de gestos (Continuación).

Paper	Aplicación	Tipo de Gesto	No de Gestos	Método de Captura	Pre-procesamiento	Extracción de Características	Método de Reconocimiento	Implementación	Tiempo Real	fps
(Coogan, Awad, Han, & Sutherland, 2006)	HCI	Estático / Dinámico	28 / 17	Mono	Skin Segm	color, motion and position	DHMM	SW	SI	10
(Fahn & Chu, 2011)	HRI	Dinámico	8	Mono	Skin Segm Hand tracking	Localización, Orientación y Velocidad de la trayectoria del gesto	HMM	SW	SI	7
(Ghotkar & Kharate, 2012)	HCI	Estático	7	Mono	Hand segmentation Operaciones morfológicas	Centro de la mano	-	SW	NO	-
(Guerrero-Balaguera & Perez-Holguin, 2015)	CSL	Estático	23	Mono	Hand segmentation using simple treshold Operaciones morfológicas	Proyección de la forma a los ejes x e y	ANN	HW	NO	-
(Jena et al., 2012)	HCI	Estático	10	Mono	Skin Color Segmentation	Area Perímetro Dirección del pulgar	Distancia euclidea	HW	NO	-
(Keskin, Erkan, & Akarun, 2003)	HCI	Dinámico	8	Stereo	Hand segmentation	Secuencias de vectores de velocidad	HMM	SW	SI	-
(H. K. Lee & Kim, 1999)	HCI	Dinámico	10	Mono	Hand segmentation	Vector de características con dirección de la mano	HMM	SW	NO	5
(S. H. Lee et al., 2011)	HRI	Estático	4	Mono	Segmentación de la mano mediante la componente roja	Relación de aspecto Posición tamaño	Distancia euclidea	HW	SI	-
(Malima, Özgür, & Çetin, 2006)	HRI	Estático	5	Mono	Skin Color Segmentation	Vector binario contando la cantidad de dedos extendidos	-	SW	NO	-

Tabla 6. Resumen de los trabajos más representativos en reconocimiento de gestos (Continuación).

Paper	Aplicación	Tipo de Gesto	No de Gestos	Método de Captura	Pre-procesamiento	Extracción de Características	Método de Reconocimiento	Implementación	Tiempo Real	fps
(Manresa, Varona, Mas, & Perales, 2005)	HCI	Estático	8	Mono	Skin Color Segmentation	Características basadas en elipses	FSM	SW	SI	30
(Nguyen-Duc-Thanh et al., 2012)	HRI	Dinámico	10	Kinect	Detección del esqueleto humano	Características basadas en la diferencia relativa entre la posición actual y el inicio del gesto	HMM	SW	SI	-
(Shi et al., 2006)	HCI	Estático	4	Mono	Skin Color Segmentation	Características basadas en momentos	HMM	HW	SI	15
(Shi & Tsui, 2007)	HCI	Dinámico	8	Mono	Skin Color Segmentation Filtrado Detección de contornos	Calculo de la trayectoria del centro de gravedad de la mano	ANN	HW	SI	15
(Wang, Xia, Cai, Gao, & Cattani, 2012)	HCI	Dinámico	7	Mono	Detección de la mano empleando histograma del gradiente HoG	Orientación de la trayectoria de la mano	HMM	SW	NO	-
(Xu & Lee, 2012)	HCI	Dinámico	6	Kinect	Algoritmo de Camshift usando el mapa de disparidad	Orientación de la trayectoria de la mano	HMM	SW	SI	15
Éste trabajo	HRI	Dinámico	7	Stereo	Filtro Medina Skin Segm Depth Segm Oper Morfolog	Hand tracking Orientación	HMM	HW	SI	>30

3 Conceptos Fundamentales

3.1 Visión Estéreo

La visión estéreo tiene la intención de reconstruir la información 3D de una escena a partir de dos imágenes diferentes. Estas imágenes son capturadas por dos cámaras que se encuentran separadas por una distancia establecida previamente. La visión estéreo funciona bajo el mismo principio en que los ojos humanos y permite ver objetos con percepción de la profundidad a la que estos se encuentran (Brown, Burschka, & Hager, 2003; Colodro, Toledo, Martínez, Garrigós, & Ferrández, 2012; Greisen et al., 2011). En la Figura 3 se presenta un sistema de imágenes estéreo simplificado.

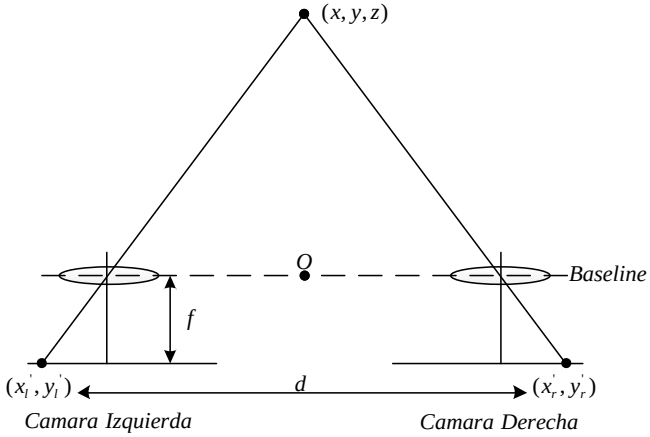


Figura 3. Sistema de imágenes estéreo simplificado. Figura adaptada por el autor a partir de (Mattoccia, 2013).

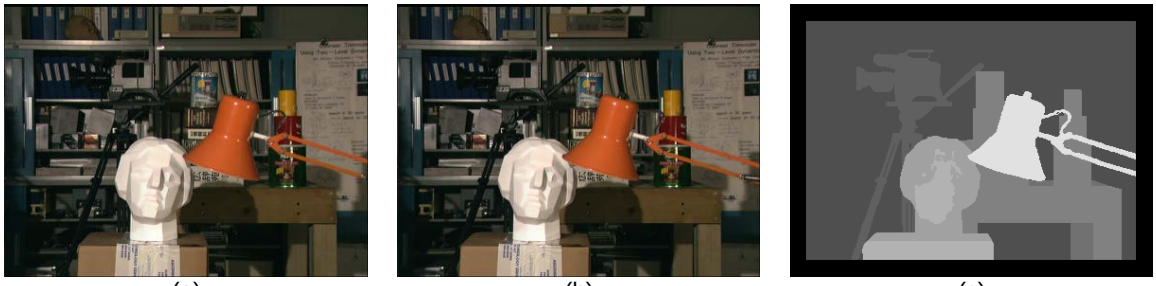
El sistema de imágenes estéreo de la Figura 3 tiene las siguientes características:

- 2 cámaras con sus ejes ópticos en paralelo y separados por una distancia *d*.
- La longitud focal de ambas cámaras es *f*.
- La línea que conecta los centros de los lentes de las cámaras se denomina línea base (*baseline*).
- El origen O se encuentra en la mitad de la línea base.
- El eje x del sistema de coordenadas 3D se encuentra paralelo a la línea base.
- Un objeto cualquiera en el espacio tridimensional que se encuentra en las coordenadas (x,y,z) se proyecta en coordenadas de imagen (x_l', y_l') y (x_r', y_r') en los planos de las imágenes izquierda y derecha de las respectivas cámaras.

En Figura 4 se presenta un ejemplo de imágenes estéreo, en la que se observa un ligero desplazamiento de la imagen izquierda (cámara izquierda) con respecto a la derecha (cámara derecha). La distancia entre un píxel y su correspondiente píxel de la otra imagen (cuando se superponen las dos imágenes) se denomina disparidad de píxel. Para

encontrar la correspondencia de las imágenes izquierda y derecha de la imagen estereoscópica se utiliza un algoritmo de correspondencia estéreo (M. O. S. M. Elmezain, 2010). El resultado es un mapa de disparidad que tiene el mismo tamaño que el par de imágenes estéreo con la disparidad para cada uno de los pixeles. Ver Figura 4 (c).

Los algoritmos de correspondencia estéreo se clasifican en dos categorías: métodos basados en área y métodos basados en características. Para los métodos basados en área, los elementos a emparejar son ventanas de la imagen de tamaño fijo y el criterio de semejanza es una cantidad medida de la correlación entre las ventanas en las dos imágenes. Para los métodos basados en características, la búsqueda de correspondencias está dirigida a un conjunto de características dispersas en vez de ventanas. La primera categoría permite calcular mapas de profundidad más densos, mientras que la segunda requiere información a priori para determinar las características óptimas a ser usadas (Colodro et al., 2012; M. O. S. M. Elmezain, 2010).



(a) (b) (c)

Figura 4. Imagen estéreo "Tsukuba". (a) imagen izquierda. (b) imagen derecha. (c) mapa de disparidad ideal (*ground truth*). Fuente (Scharstein & Szeliski, 2002).

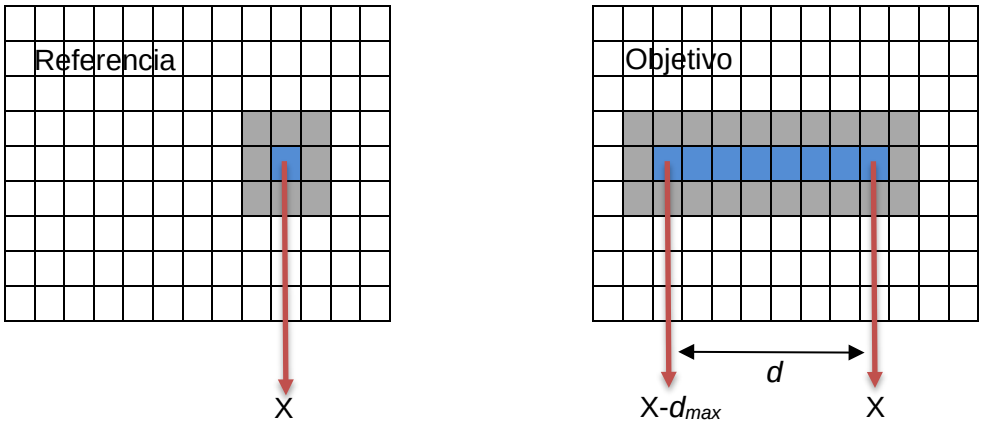


Figura 5. Método de correspondencia estéreo basado en el área. En la imagen de referencia la ventana se encuentra centrada en el punto x, mientras que en la imagen objetivo hay múltiples ventana centradas en $[x,x-d_{max}]$. Figura adaptada por autor a partir de (Mattoccia, 2013).

En la Figura 5 se muestra gráficamente el método de correspondencia estéreo basado en el área. En este método se toma una ventana en la imagen de referencia y se compara

con *d* ventanas de la imagen objetivo. Para esto se emplea una función de costo que permite determinar la similitud entre la ventana de referencia y cada una de las ventanas objetivo. La disparidad se calcula teniendo en cuenta la ventana objetivo con el mayor grado de similitud (Mattoccia, 2013).

Los métodos de correspondencia basados en área, determinan la mejor disparidad para un pixel y su vecindad, mediante la aplicación de una estrategia denominada WTA (*winner-takes-all*). Dicha estrategia selecciona la ventana candidata con el mejor valor de semejanza (Humenberger, Zinner, Weber, Kubinger, & Vincze, 2010; Mattoccia, 2013).

3.1.1 Transformada Census

La transformada Census es un algoritmo de correspondencia estéreo basado en el área que es robusto frente a cambios de iluminación. Este algoritmo transforma las imágenes antes de calcular la correspondencia para cada nivel de disparidad. La transformada está basada en una función de comparación ξ, la cual es usada para comparar el valor de dos pixeles P_1 y P_2. En la ecuación (1) se presenta la función comparación ξ (Ambrosch, Zinner, & Kubinger, 2009; Colodro et al., 2012; Zabih & Woodfill, 1994).

$$\xi(P_1,P_2)=\begin{cases}0, P_1 \le P_2\\ 1, P_1 > P_2\end{cases} \tag{1}$$

La Transformada Census considera una ventana $M\ X\ N$ centrada en un pixel $\mathrm{I}(u,v)$ de la imagen y codifica la información de la ventana en una cadena de $MN-1$ bits en la que asigna valor ‘1’ al bit asociado a cada pixel con valor mayor que el central y valor ‘0’ en otro caso (Ambrosch, Kubinger, Humenberger, & Steininger, 2009; Ambrosch, Zinner, et al., 2009; Colodro et al., 2012; Zabih & Woodfill, 1994). En la ecuación (2) se muestra una representación matemática de dicha transformada, donde $\otimes$ representa el operador de concatenación y $m=\frac{M-1}{2}$ y $n=\frac{N-1}{2}$.

$$I_{Census}(u,v)=\bigotimes_{i=-m}^{m}\ \bigotimes_{j=-n}^{n}\xi(I(u,v),I(\mathrm{u+i,v+j})) \tag{2}$$

En la Figura 6 se presenta un ejemplo para calcular la transformada Census para un pixel de una imagen cuya vecindad fue definida por una ventana de 3X3 pixeles. Teniendo en cuenta los valores arbitrarios seleccionados, se realiza la comparación del pixel central, cuyo valor es 20, con cada uno de los pixeles definidos en su vecindad. Este proceso da como resultado una cadena de bits como se muestra en la parte inferior derecha de la Figura 6. Este procedimiento se debe repetir en toda la imagen manteniendo siempre el mismo tamaño de vecindad seleccionado. El resultado final de la transformada Census será una matriz del tamaño de la imagen original, donde cada una de las posiciones de dicha matriz corresponderá a una cadena de bits.

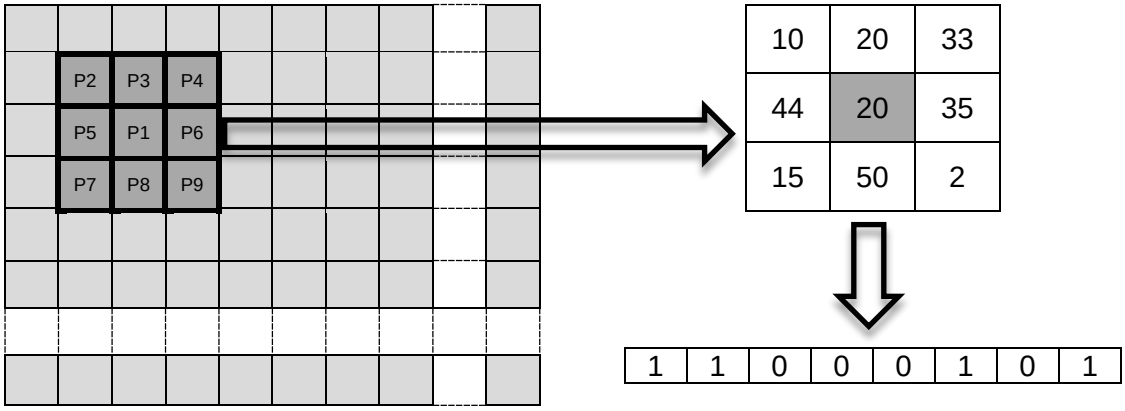


Figura 6. Calculo de la transformada Census para un pixel con una vecindad de 3X3 pixeles. Fuente Autor.

3.1.2 Transformada Census Dispersa

La principal desventaja de la transformada Census es el gran tamaño del vector resultante. Este vector corresponde al número de comparaciones de pixeles que se realizan, y tiene un gran impacto en los recursos de hardware necesarios para su implementación, especialmente en sistemas de visión estéreo. Por esta razón, se han planteado diferentes métodos para mitigar este inconveniente (Fife, 2011). Uno de ellos consiste en aplicar una máscara con un patrón uniforme de submuetreo, el cual determina los pixeles dentro de la ventana que son tenidos en cuenta para aplicar la transformación. Este método disminuye la cantidad de comparaciones necesarias sin deteriorar el resultado de la transformada Census (Fife, 2011; Humenberger et al., 2010).

En la Figura 7 se presentan dos ventanas cada una con un patrón diferente. Las zonas de color gris representan los pixeles que son comparados con el pixel central. En la ventana de la izquierda (a) se puede observar que todos los pixeles son tenidos en cuenta. En este caso, la cadena de bits resultante tendrá un total de 24 bits. En la ventana de la derecha (b) solo se tiene en cuenta el 50% de los pixeles de la vecindad, por lo cual la cadena de bits de la transformada Census tendrá solo 12 bits. Esta reducción del número de bits permite optimizar el uso de recursos de hardware necesarios (memoria) para calcular la disparidad en imágenes estéreo, sin afectar el resultado final (Fife, 2011).


(a) (b)

Figura 7. Mascaras para el cálculo de la transformada Census. (a). Mascara 5X5 convencional. (b) Mascara 5X5 uniformemente distribuida, vecindad de transformada Census con 50% de dispersión. Fuente Autor.

3.1.3 Suma de Distancias de Hamming (SHD)

La trasformada Census es únicamente el primer paso en el método de correspondencia de Census para imágenes estéreo. El siguiente paso es sumar las distancias de Hamming (SHD) de los vectores de la transformación de Census de cada una de las ventanas candidatas. El cálculo de la distancia de Hamming se hace comparando dos cadenas de bits, y el resultado es el número de bits en que difieren dichas cadenas (Ambrosch, Kubinger, et al., 2009; Ambrosch, Zinner, et al., 2009; Fife, 2011; Humenberger et al., 2010; Zabih & Woodfill, 1994). La similitud SHD puede ser medida como se muestra en la ecuación (3), donde $I_{LCensus}$ y $I_{RCensus}$ son las imágenes Izquierda y Derecha en su transformación de Census, y *d* corresponde a la disparidad. Ver Figura 5.

$$SHD: \sum_{i=-m}^{m} \sum_{j=-n}^{n} \text{Hamming}(I_{LCensus}(u+i, v+j), I_{RCensus}(u+i, v+j+d)) \quad \forall d \in \{0,1,2,....,d_{\max}\} \qquad (3)$$

3.2 Procesamiento Digital de Imágenes

El procesamiento digital de imágenes es un conjunto de operaciones matemáticas aplicadas a imágenes digitales. El objetivo del procesamiento de imágenes es transformar una imagen en otra con características diferentes, tales como eliminación de ruido, realce de contornos, detección de objetos o patrones etc (Gonzalez & Woods, 2002).

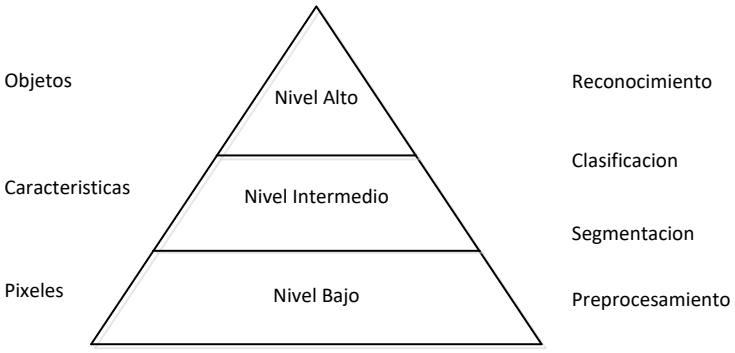


Figura 8. Pirámide del procesamiento de imágenes. Figura adaptada por el autor a partir de (Bailey, 2011).

Las operaciones de procesamiento de imágenes se agrupan en categorías de acuerdo con el tipo de datos que trabajan. Esta agrupación se conoce comúnmente como la pirámide del procesamiento de imágenes, ver Figura 8. En el nivel más bajo de la pirámide se encuentran las operaciones de pre-procesamiento. Estas son operaciones de bajo nivel orientadas a pixeles o a vecindades de pixeles, cuyo objetivo se centra en resaltar la información importante dentro de la imagen y eliminar aquella información que no representa utilidad. Ejemplos de operaciones de pre-procesamiento son: detección de bordes, filtrado para reducción de ruido, y realce de brillo y contraste. En el nivel intermedio se encuentran las operaciones de segmentación y clasificación. Las operaciones de segmentación transforman una imagen en regiones. Ejemplos de operaciones de segmentación son: Umbralización, detección de color, crecimiento de

regiones y etiquetado de componentes conexas. Las operaciones de clasificación transforman la información de regiones a características, y se emplean para identificar o clasificar objetos dentro de varias categorías. En el nivel más alto se encuentran las operaciones de reconocimiento, que permiten hacer una descripción o interpretación de la escena (Bailey, 2011).

En las operaciones de procesamiento de imágenes de bajo nivel se encuentran dos categorías: las operaciones a nivel de pixel, y operaciones orientadas a vecindades. La primera categoría es el tipo de operaciones más sencilla ya que el pixel de salida se obtiene a partir de una función f aplicada al correspondiente pixel de entrada (Bailey, 2011). Este tipo de operaciones se definen como se muestra en la ecuación (4), donde $I[x, y]$ representa un pixel ubicado en las coordenadas (x, y) de la imagen I y $Q[\mathrm{x,y}]$ representa el valor de un pixel obtenido mediante la función f aplicada al pixel de entrada $I[x, y]$. Las operaciones orientadas a pixel más comunes son: ajuste de brillo y contraste, negativo de la imagen, umbralización global, umbralización multinivel, segmentación por color y transformaciones entre espacios de color.

$$Q[x, y] = f(\mathrm{I[x, y]}) \tag{4}$$

Las operaciones orientadas a vecindades son una extensión de las operaciones orientadas a pixel, con la diferencia que el valor de salida depende de una función aplicada a una vecindad local (o ventana) (Bailey, 2011). Estas operaciones pueden definirse matemáticamente mediante la ecuación (5), donde W es una vecindad local centrada en $I[x, y]$, como se muestra en la Figura 9. Ejemplos de operaciones orientadas a vecindades son: filtro de Sobel, filtro de Prewit, filtro de Mediana, filtro de Media, erosión, dilatación, etc.

$$Q[x, y] = f(I[x, y], ..., I[x + \Delta x, y + \Delta y]), \qquad (\Delta x, \Delta y) \in W \tag{5}$$

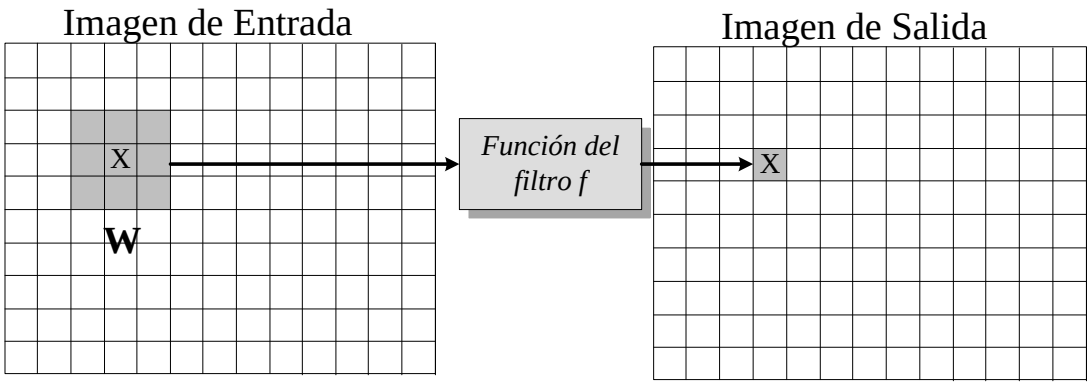


Figura 9. Filtro Local. La zona sombreada representa una vecindad de píxeles de entrada a un filtro, el cual genera un pixel nuevo en el centro de dicha vecindad. Figura adaptada por el autor a partir de (Bailey, 2011).

3.3 Procesamiento Embebido de Imágenes

Un sistema embebido es un sistema basado en computador que está integrado o empotrado dentro de un producto o componente. Un sistema embebido es usualmente diseñado para cumplir con una tarea en específica, o una pequeña cantidad de tareas específicas, comúnmente con restricciones de tiempo-real. Un ejemplo básico de un sistema de procesamiento embebido de imágenes es una cámara digital. Dichos sistemas permiten capturar, almacenar, retoca, visualizar, comprimir y descomprimir imágenes digitales (Bailey, 2011).

Los sistemas embebidos de visión son sumamente útiles en aplicaciones de cámaras "inteligentes", ya que no solo sirven para tomar imágenes, sino que analizan y extraen información importante para aplicaciones específicas, como es el caso de la vigilancia, inspección y control de calidad en la industria, visión en robots etc.

3.3.1 Procesamiento de imágenes en tiempo real

Un sistema de tiempo-real es aquel en el que la respuesta a un evento debe ocurrir dentro un tiempo específico, de lo contrario se considera que el sistema ha fallado. Un sistema de procesamiento de imágenes en tiempo real es aquel que captura imágenes con regularidad, analiza dichas imágenes para obtener información y utiliza la información resultante para controlar alguna actividad. Todo el procesamiento de imágenes debe ocurrir dentro de un tiempo establecido, usualmente (pero no siempre) éste tiempo se establece por la velocidad de *frames* de captura (Bailey, 2011).

3.3.2 Procesamiento paralelo de imágenes

Tradicionalmente las plataformas para procesamiento de imágenes están basadas en arquitecturas de computadores secuenciales. La ejecución de tareas se realiza como una secuencia de instrucciones ordenadas a través de una Unidad Lógico Aritmética (ALU). Sin embargo, un algoritmo, especialmente en procesamiento de imágenes, puede ser implementado mediante elementos de procesamiento separados dando como resultado una ejecución completamente paralela. Esto es especialmente útil para algoritmos de nivel bajo e intermedio en la pirámide de procesamiento de imágenes (Bailey, 2011).

Todos los algoritmos de procesamiento de imágenes se componen de una secuencia de operaciones de procesamiento de imágenes. Esto se traduce como una forma de *paralelismo temporal*. Dicha estructura sugiere utilizar un procesador separado para cada operación, como se observa en la Figura 10. A esto se le conoce como una arquitectura pipeline. El funcionamiento de esta arquitectura es muy similar a una línea de producción, en la que los datos pasan a través de cada una de las etapas a medida que se procesan. Cada procesador aplica su operación y el resultado es enviado a la siguiente etapa.

En procesamiento de imágenes es importante considerar el tiempo que tarda cada una de las etapas de procesamiento en recibir un dato, procesarlo y generar su respectiva respuesta. Dicho tiempo se conoce como latencia. La latencia puede ser pequeña si la operación solo depende de un pixel de entrada, o de una pequeña vecindad de pixeles. Cuando una operación requiere tener toda la imagen completa para obtener una salida, en este caso la latencia será muy alta. Las operaciones pipeline pueden brindar un aumento significativo en el rendimiento cuando todas las operaciones tienen baja latencia, ya que el procesador de la siguiente operación puede iniciar a trabajar aunque que el procesador anterior aún no haya terminado de hacerlo (Bailey, 2011).

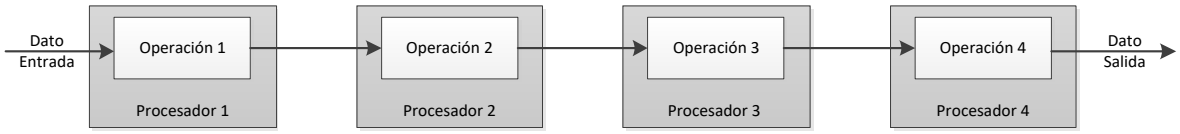


Figura 10. Paralelismo temporal empleando procesadores pipeline. Adaptación a partir de (Bailey, 2011).

Otro tipo de paralelismo que puede utilizarse en el procesamiento de imágenes es el paralelismo espacial. Este tipo de paralelismo consiste en particionar la imagen en bloques y utilizar un procesador independiente para implementar la operación en cada partición. Los esquemas de partición más usuales son dividir la imagen en bloques de filas, bloques de columnas o bloques rectangulares como se muestra en la Figura 11. Una consideración importante cuando se particiona la imagen es minimizar la comunicación entre procesadores, lo que implica minimizar la comunicación entre las diferentes particiones. Por consiguiente cada procesador debe tener una memoria local para reducir cualquier retardo causado por los accesos a memoria global (Bailey, 2011).

El paralelismo lógico reutiliza un bloque funcional muchas veces dentro de una operación. Este tipo de paralelismo comúnmente corresponde a bucles dentro de un algoritmo, para los cuales siempre se repite la misma operación. En este caso, el paralelismo lógico se implementa desdoblando el bucle presente en dicho algoritmo mediante la ejecución en paralelo de la operación (Bailey, 2011).

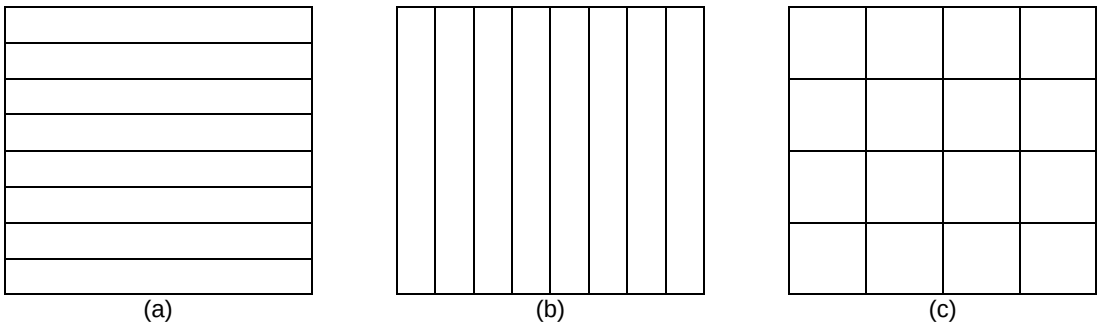


Figura 11. Paralelismo espacial basado en partición de la imagen. (a) Partición por filas. (b) Partición por Columnas. (c) Partición por bloques. Figura adaptada por el autor a partir de (Bailey, 2011).

Uno de los cuellos de botella más comunes en procesamiento de imágenes es el tiempo y ancho de banda requeridos para leer la imagen de memoria y escribir el resultado en memoria nuevamente. El procesamiento de flujo de datos o (*Stream Processing*) permite afrontar dicho problema convirtiendo el paralelismo espacial en paralelismo temporal. La imagen se lee y se escribe a un barrido equivalente a una taza de un pixel por ciclo de reloj. Este tipo de procesamiento es más efectivo si se utiliza como entradas las vecindades de pixeles, de lo contrario los requerimientos de memoria cache pueden ser excesivos. Para la mayoría de operaciones la latencia es baja comparada con la carga y procesamiento de una imagen completa. En este caso, el tiempo de procesamiento de la imagen suele estar gobernado por la velocidad de *frame*. En la Figura 12 se muestra de forma gráfica como funciona este tipo de procesamiento (Bailey, 2011).

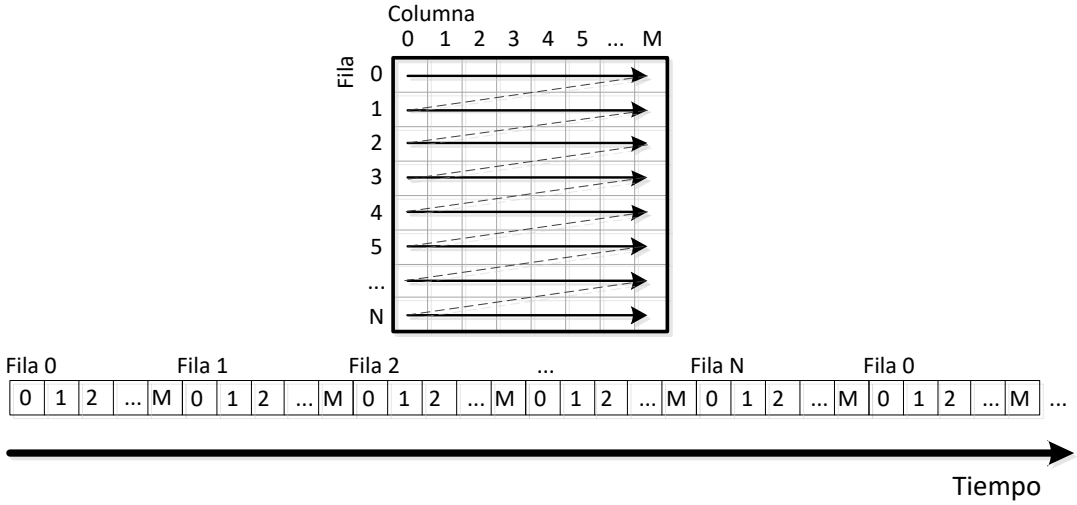


Figura 12. Procesamiento de flujo de datos. Conversión de paralelismo espacial en paralelismo temporal. Figura adaptada por el autor a partir de (Bailey, 2011).

3.3.3 Técnicas de mapeo de algoritmos en FPGA

A diferencia de la implementación en Software, la implementación en FPGA de los algoritmos de procesamiento de imágenes, empleando *stream processing,* conlleva a un incremento en el rendimiento de dichos algoritmos. Sin embargo, se requiere más esfuerzo en el diseño del hardware, lo que involucra el desarrollo o aplicación de técnicas de mapeo de algoritmos a recursos de FPGA con el fin de obtener los resultados deseados. De acuerdo con (Gribbon, Bailey, & Johnston, 2005, 2006) existen tres tipos de restricciones en el proceso de mapeo: restricción de tiempo, restricción de ancho de banda de memoria, y restricción de recursos disponibles. La utilización de los recursos depende de la eficiencia del diseño. Se debe tener cuidado en el nivel de diseño lógico. Las siguientes secciones describen las técnicas estándar utilizadas para abordar las dos primeras restricciones.

3.3.3.1 Restricción de tiempo

La restricción de tiempo es común en aplicaciones de tiempo real, donde los requerimientos de velocidad de los datos imponen una estricta limitación. A velocidades

de video, por ejemplo, todo el procesamiento necesario para cada píxel debe realizarse a la velocidad del reloj de píxel (o más rápido). Para cumplir con esta restricción generalmente se requiere de *pipelining* a bajo nivel (Gribbon et al., 2005).

La segmentación o *pipelining* de bajo nivel divide una operación en pequeñas etapas de manera que la operación completa se ejecuta por segmentos. Como resultado se reduce el retardo de propagación a una sola etapa. Debido a que únicamente una porción de la operación se ejecuta en un ciclo de reloj, la velocidad puede incrementarse. Para ejecutar una operación completa se requiere más de un ciclo de reloj. Si el hardware es duplicado, se pueden procesar varios pixeles a la vez. El número de pixeles que pueden ser procesados simultáneamente depende del total de etapas de *pipeline* (Bailey, 2011).

Para ilustrar este procedimiento, consideremos la siguiente ecuación.

$$y = ax^2 + bx + c$$
$$y = (ax + b)x + c \qquad (6)$$

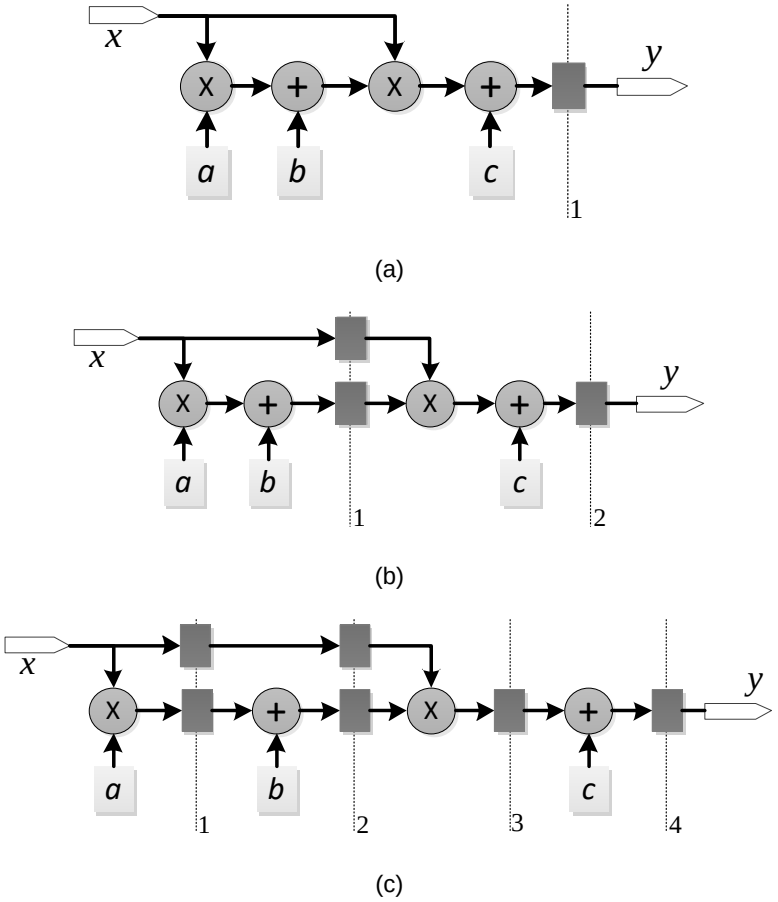


Figura 13. Ejemplo de ejecucion *Pipeline*. (a) Realizar el calculo en un solo ciclo de reloj. (b) Realizar el calculo en dos ciclos de reloj (dos etapas pipeline). (c) Realizar el calculo en cuatro ciclos de reloj (cuatro etapas Pipeline). Figura adaptada por el autor a partir de (Bailey, 2011).

La ecuación anterior puede ser implementada usando uno, dos o cuatro estados de pipeline como se muestra en la Figura 13. La latencia óptima se puede obtener al implementar 2 estados de pipeline ya que se mejora el *throughput* del sistema. Esto se deba a que las operaciones de suma y multiplicación son uniformemente distribuidas entre las dos etapas. En el caso de cuatro etapas de pipeline, las etapas 1 y 3 poseen operaciones de multiplicación, mientras que las etapas 2 y 4 poseen operaciones de suma. Una operación de multiplicación requiere más tiempo que una operación de suma. Es decir, que la distribución de tiempo en las etapas no es uniforme. Esto causa desequilibrio, e incrementa la latencia del sistema. Este efecto puede atenuarse mediante un método conocido como *retiming*. Dicho método, consiste en ejecutar una parte de la multiplicación en la primera etapa y la parte restante en la etapa siguiente, lo cual permite obtener una distribución de tiempo equivalente entre las etapas del *Pipeline*. Por lo tanto la selección del número de etapas de pipeline es sumamente importante en diseños con FPGAs (Bailey, 2011).

3.3.3.2 Restricción de ancho de banda de memoria

La restricción de ancho de banda de memoria se presenta cuando se requiere acceder a memoria con demasiada frecuencia creando un cuello de botella en el procesamiento. Este es el caso del enfoque estándar de software en el cual, se lee cada píxel requerido por una operación desde la memoria, se aplica la operación y se escriben los resultados en memoria nuevamente. Este proceso se repite para cada píxel dentro de la imagen. Un proceso similar se repite para cada operación que se requiera aplicar a la imagen. La lectura y la escritura continuas en memoria en muchas situaciones pueden evitarse utilizando una arquitectura de almacenamiento en caché de los datos que se reutilizan con frecuencia (Bailey, 2011; Gribbon et al., 2005).

Un método en el cual el almacenamiento chache es implementado en operaciones de procesamiento de imágenes es el almacenamiento intermedio de filas (*Row Buffering*). Si se considera una operación espacial con una ventana 3X3, donde cada pixel de salida es una función de 8 pixeles vecinos en la ventana. En una implementación normal se deben leer nueve pixeles en cada posición de la ventana (por cada ciclo de reloj para *Stream* Processing) y además, cada pixel debe ser leído nueve veces mientras se desplaza la ventana a través de la imagen. Por lo general, los pixeles que son adyacentes horizontalmente son requeridos en ciclos de reloj consecutivos, por lo tanto pueden ser almacenados en un buffer y retrasados por registros (Bailey, 2011; Gribbon et al., 2005). En la Figura 13 se presenta gráficamente el funcionamiento de dos *Row Buffer*. Como se puede apreciar, cada vez que ingresa un nuevo pixel en la dirección de escaneo de la ventana, los valores de los pixeles de las filas anteriores se encuentran disponibles para ser procesados, evitando de esta forma la lectura repetitiva en memoria de esos pixeles. La implementación en FPGA de este método de almacenamiento en cache se muestra en la Figura 15. En ésta figura se pueden apreciar los buffers de almacenamiento de filas, los cuales son típicamente memorias de tipo FIFO con una capacidad de almacenamiento de

$M = N - w$ pixeles, donde N es el ancho de la imagen en pixeles y w el ancho de la ventana. La ventana está conformada por $W = w^2$ registros interconectados de forma horizontal como se observa en la figura. Estos registros pueden leerse simultáneamente de manera que se permita el procesamiento de la operación vinculada a dicha ventana.

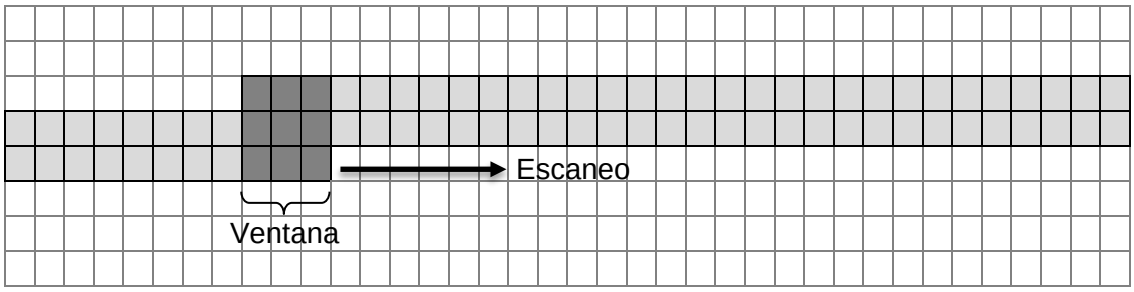


Figura 14. Escaneo de la imagen mediante una ventana 3X3 empleando el metodo de cacheo *Row Buffering*. Figura adaptada por el autor a partir de (Bailey, 2011).

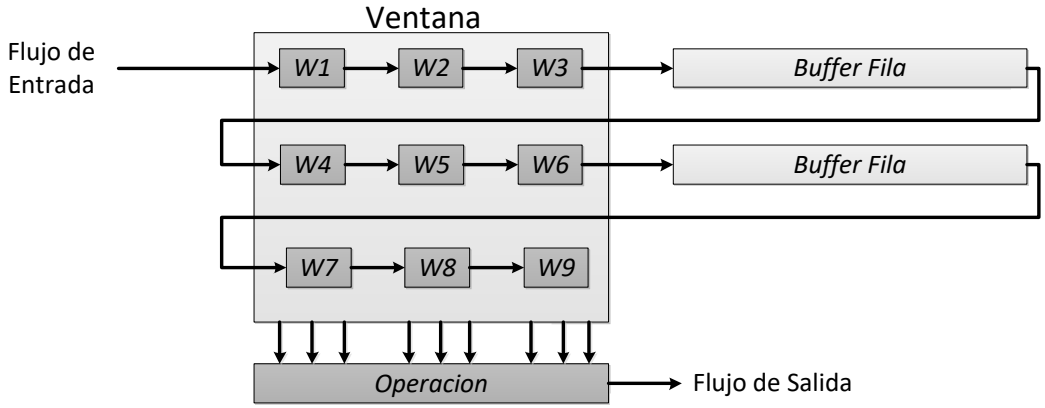


Figura 15. Aquitectura *Row Buffering* para FPGA. Adaptacion a partir de (Bailey, 2011).

Existen otros métodos que permiten evitar la restricción de ancho de banda de memoria, algunos de ellos son: el almacenamiento de doble buffer, *Frame Buffer*, memoria cache, entre otros.

3.4 Modelos Ocultos de Markov

Un Modelo Oculto de Markov (HMM) es un modelo estadístico que describe una secuencia estocástica $O = O_1, O_2, ..., O_T$, como una observación indirecta de una secuencia aleatoria oculta $Q = Q_1, Q_2, ..., Q_T$, (Rabiner, 1989).

En un modelo de Márkov normal, el estado es visible directamente para el observador, por lo que las probabilidades de transición entre estados son los únicos parámetros. En un modelo oculto de Márkov, el estado no es visible directamente, sino que sólo lo son las variables influidas por el estado. Cada estado tiene una distribución de probabilidad sobre los posibles símbolos de salida. Consecuentemente, la secuencia de símbolos generada por un HMM proporciona cierta información acerca de la secuencia de estados.

Un HMM discreto de primer orden se define formalmente mediante los siguientes elementos:

- Un conjunto $S = \{S_1, S_2, ..., S_k\}$ de estados ocultos. Aunque estos estados están ocultos existen algunas aplicaciones prácticas, donde los estados pueden tener un significado físico. Por ejemplo en reconocimiento de escritura o reconocimiento de voz. Un estado en el tiempo t se denota como Q_t (Rabiner, 1989).

- Una matriz de transiciones $A = \{a_{ij}\}$, de dimensiones k x k, donde el elemento $a_{ij} \geq 0$ representa la probabilidad de ir del estado S_i al estado S_j. Además se debe cumplir que $\sum_{j=1}^{k} a_{ij} = 1$. A esta matriz se le conoce con el nombre de matriz estocástica (Rabiner, 1989).

- Un conjunto $V = \{v_1, v_2, ..., v_m\}$ de símbolos de observación. Este es el alfabeto, y corresponde a las salidas físicas del proceso que está siendo modelado (Rabiner, 1989).

- Una matriz de emisión $B = \{b(j \mid S_i)\}$ de dimensiones (k x m), que indica la probabilidad de emitir un símbolo v_j en el estado S_i. Donde $b(j \mid S_i) = P\left[O_t = v_j \mid O_t = S_i\right]$, con $b(j \mid S_i) \geq 0$ y $\sum_{j=1}^{m} b(j \mid S_i) = 1$ (Rabiner, 1989).

- Una distribución de probabilidad inicial $\pi = \{\pi_i\}$, donde: $\pi_i = \pi(S_i) = P\left[q_1 = S_i\right]$ con $\pi_i \geq 0$ y $\sum_{i=1}^{k} \pi_i = 1$.

Un modelo Oculto de Markov (HMM) se denota de forma habitual como una tupla de la siguiente forma: $\lambda = (\mathrm{S}, \mathrm{V}, \mathrm{A}, \mathrm{B}, \pi)$ (Rabiner, 1989).

3.4.1 Problemas básicos de un HMM

Existen tres problemas básicos que deben ser resueltos para que el modelo resulte útil en aplicaciones del mundo real:

Problema 1: Dada la secuencia de observaciones $O = O_1, O_2, ..., O_T$ y un modelo $\lambda = (\mathrm{A}, \mathrm{B}, \pi)$ ¿cómo calcular de manera eficiente la probabilidad $P(\mathrm{O} \mid \lambda)$ de observar la

secuencia dado el modelo? El problema de evaluación tiene un interés particular cuando se desea elegir entre varios modelos posibles, ya que se puede elegir aquel que mejor explique una secuencia de observaciones (Rabiner, 1989).

La solución al problema de evaluación del modelo es un algoritmo conocido como procedimiento de avance y retroceso. Este algoritmo se basa en el uso de dos variables intermedias. La variable de avance α y la variable de retroceso β. La variable de avance se define en la ecuación (7) y representa la probabilidad de observar la secuencia $O = O_1, O_2, ..., O_t$ hasta el tiempo t iniciando en el estado S_i (Rabiner, 1989). En las ecuaciones (8), (9) y (10) se presenta el procedimiento a seguir para calcular dicha variable.

$$\alpha(i) = P(\mathrm{O}_1, \mathrm{O}_2, ..., \mathrm{O}_t, \mathrm{q}_t = \mathrm{S}_i \mid \lambda) \tag{7}$$

1. Inicialización

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \le \mathrm{i} \le \mathrm{k} \tag{8}$$

2. Inducción

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{k} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \qquad 1 \le \mathrm{t} \le \mathrm{T\text{-}1} \quad 1 \le j \le k \tag{9}$$

3. Terminación

$$P(O \mid \lambda) = \sum_{i=1}^{k} \alpha_T(i) \tag{10}$$

La variable de retroceso se define como $\beta_t(\mathrm{i}) = \mathrm{P}(\mathrm{O}_{t+1}, ...\mathrm{O}_T \mid \mathrm{Q}_t = \mathrm{S}_i, \lambda)$ y representa la probabilidad de observar los símbolos $\mathrm{O}_{t+1}, ...\mathrm{O}_T$, iniciando en el estado S_i en el tiempo t (Rabiner, 1989). Esta variable se calcula recursivamente como se muestra en las ecuaciones (11) y (12).

$$\beta_T(i) = 1 \qquad 1 \le \mathrm{i} \le \mathrm{k} \tag{11}$$

$$\beta_t(i) = \sum_{j=1}^{k} a_{ij} b(O_{t+1} \mid S_j) \beta_{t+1}(j) \qquad t = T-1, ..., 1 \qquad 1 \le \mathrm{i} \le \mathrm{k} \tag{12}$$

Problema 2: Dada la secuencia de observaciones $O = O_1, O_2, ..., O_T$ y un modelo $\lambda = (\mathrm{A}, \mathrm{B}, \pi)$ ¿cómo escoger una secuencia de estados correspondiente $\hat{Q} = \hat{Q}_1, \hat{Q}_2, ..., \hat{Q}_T$, de manera que explique de la mejor manera las observaciones? En este caso el objetivo es encontrar la secuencia de estados $\hat{Q} = \hat{Q}_1, \hat{Q}_2, ..., \hat{Q}_T$ de tal forma que

$\hat{Q} = \arg\max_{Q} P(\mathrm{O}, \mathrm{Q} \mid \lambda)$. Este problema se resuelve mediante el algoritmo de Viterbi (Rabiner, 1989). Este procedimiento se inicia definiendo la siguiente variable.

$$\delta_t(i) = \max_{Q_1,...,Q_{t-1}} P(\mathrm{Q}_1, \mathrm{Q}_2, ..., \mathrm{Q}_t = \mathrm{S}_i, \mathrm{O}_1, \mathrm{O}_2, ..., \mathrm{O}_t \mid \lambda) \tag{13}$$

Esta variable representa la mejor puntuación o mayor probabilidad a lo largo de una trayectoria simple, en el tiempo *t*. Para obtener la secuencia de estados del argumento de δ_i debe ser almacenado para cada *t* y cada *i* obteniendo como resultado un vector $\psi_t(i)$. El algoritmo de Viterbi se define mediante los siguientes pasos recursivos:

1. Inicialización

$$\delta_1(i) = \pi_i b(O_1 \mid \mathrm{S}_i) \qquad 1 \le \mathrm{i} \le \mathrm{k}$$
$$\psi_1(i) = \phi \qquad 1 \le \mathrm{i} \le \mathrm{k}$$

2. Recursión

$$\delta_t(i) = \max_{1 \le i \le k} \left[\delta_{t-1}(i) a_{ij} \right] b(\mathrm{O}_t \mid \mathrm{S}_j) \qquad 1 \le \mathrm{j} \le \mathrm{k} \qquad 2 \le \mathrm{t} \le \mathrm{T}$$
$$\psi_t(i) = \arg\max_{1 \le i \le k} \left[\delta_{t-1}(i) a_{ij} \right] \qquad 1 \le \mathrm{j} \le \mathrm{k} \qquad 2 \le \mathrm{t} \le \mathrm{T}$$

3. Terminación

$$\hat{P} = \max_{1 \le i \le k} \left[\delta_T(i) \right]$$
$$\hat{Q}_T = \arg\max_{1 \le i \le k} \left[\delta_T(i) \right]$$

4. Retroceso de la Secuencia de Estados

$$\hat{Q}_t = \psi_{t+1}(\hat{Q}_{t+1}) \qquad \mathrm{t=T\text{-}1,T\text{-}2,...,1}$$

Problema 3: ¿Cómo ajustar los parámetros del modelo $\lambda = (\mathrm{A}, \mathrm{B}, \pi)$ para maximizar la $P(\mathrm{O} \mid \lambda)$? Este es el problema más difícil de resolver y se soluciona usualmente empleando el criterio de máxima probabilidad ML. El algoritmo que permite implementar este criterio es el algoritmo conocido como *Baum-Welch* (Rabiner, 1989). Con el fin de describir el procedimiento de re-estimación de los parámetros del modelo HMM se emplean dos variables.

- $\xi_t(i,j)$: representa la probabilidad de pasar del estado S_i en el tiempo *t* al estado S_j en el tiempo *t+1*, dada una secuencia de observaciones y el modelo. Esta variable es calculada utilizando las variables de avance y retroceso.

$$\xi_t(i,j) = P(\mathrm{Q}_t = \mathrm{S}_i, \mathrm{Q}_{t+1} = \mathrm{S}_j \mid \mathrm{O}, \lambda)$$
$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b(\mathrm{O}_{t+1} \mid \mathrm{S}_j) \beta_{t+1}(\mathrm{j})}{\sum_i \sum_j \alpha_t(i) a_{ij} b(\mathrm{O}_{t+1} \mid \mathrm{S}_j) \beta_{t+1}(\mathrm{j})} \tag{14}$$

- $\gamma_t(i)$: Representa la probabilidad de iniciar en el estado S_i en el tiempo *t*, dada una secuencia de observaciones y un modelo. La variable es calculada como se muestra a continuación.

$$\gamma_t(i) = P(Q_t = S_i \mid O, \lambda)$$
$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_i \alpha_t(i)\beta_t(i)} \tag{15}$$
$$\gamma_t(i) = \sum_{j=1}^{k} \xi_t(i,j)$$

Una vez calculadas las anteriores variables se procede a estimar los parámetros del modelo A, B y π de la siguiente manera:

$$\bar{\pi}_i = \gamma_t(i) \tag{16}$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{17}$$

$$\bar{b}(\mathrm{v}_j \mid \mathrm{S}_i) = \frac{\sum_{t=1}^{T-1} \gamma_t(i) \quad s.t.\ \mathrm{O}_t = v_j}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{18}$$

4 Descripción del Sistema de Reconocimiento de Gestos

En este trabajo se presenta el diseño e implementación en FPGA de un sistema de reconocimiento de gestos basado en visión. El sistema está diseñado para capturar, procesar y reconocer 7 gestos dinámicos de las manos en tiempo real. Los gestos seleccionados corresponden a los comandos básicos de control de un robot móvil tomado como ejemplo para demostrar las posibilidades de aplicación del sistema de reconocimiento de gestos diseñado. El usuario debe realizar los gestos empleando únicamente la mano derecha sin la necesidad de usar ningún accesorio adicional. El gesto es interpretado de acuerdo con la trayectoria obtenida como resultado del movimiento de la mano. En la Figura 16 se presentan los gestos dinámicos empleados en este trabajo y su significado para el control del robot móvil.

Figura 16. Gestos empleados en este trabajo para controlar a distancia un robot. Fuente autor.

4.1 Etapas del Sistema de Reconocimiento de Gestos

El sistema de reconocimiento de gestos de las manos desarrollado en este trabajo está compuesto por cinco etapas principales: captura de los gestos, pre-procesamiento, extracción de características, reconocimiento y comunicación y control del robot (ver Figura 17). Los gestos son capturados en un ambiente con iluminación y fondo uniformes empleando un sistema de visión estereoscópico compuesto por dos cámaras ArduCam que poseen el sensor OV5642 de Omnivision.

En la etapa de Pre-procesamiento se aplica el filtro de mediana a las imágenes capturadas con el fin de eliminar el ruido producido en la adquisición. Una vez realizado el proceso de filtrado, se calcula el mapa de disparidad empleando los algoritmos de correspondencia estéreo: Transformada Census Dispersa y Suma de Distancias de Hamming. Para detectar las manos se emplea la segmentación por color de piel junto con la segmentación del mapa de disparidad. Adicionalmente, se utiliza la operación morfológica de apertura para eliminar el ruido producido en la segmentación.

En la etapa de Extracción de características se obtiene el gesto mediante el cálculo del centroide de la forma de la mano a partir de una secuencia consecutiva de imágenes. El gesto puede contener ruido, razón por lo cual se emplea un filtro de media móvil que permite suavizar el gesto realizado. La trayectoria resultante del gesto es un patrón espacio-temporal compuesto por puntos del centroide de la mano. Uno de los retos en reconocimiento de gestos es determinar aquellos movimientos que tienen un significado (gesto) de otro tipo de movimientos (no gesto); a este proceso se conoce como "G*esture Spotting*" (M. O. S. M. Elmezain, 2010). El proceso de "G*esture Spotting*" se lleva a cabo mediante la detección del movimiento de la mano, teniendo en cuenta posiciones estáticas de inicio y finalización. Una vez realizado este proceso, se extraen las características que describen el gesto ejecutado. En este trabajo se emplea la orientación entre dos puntos consecutivos de la trayectoria de la mano para describir un gesto cualquiera. Posteriormente, estas características son transformadas mediante un proceso de cuantización en el que se calculan 8 posibles valores discretos que son necesarios para el uso de los clasificadores basados en HMM.

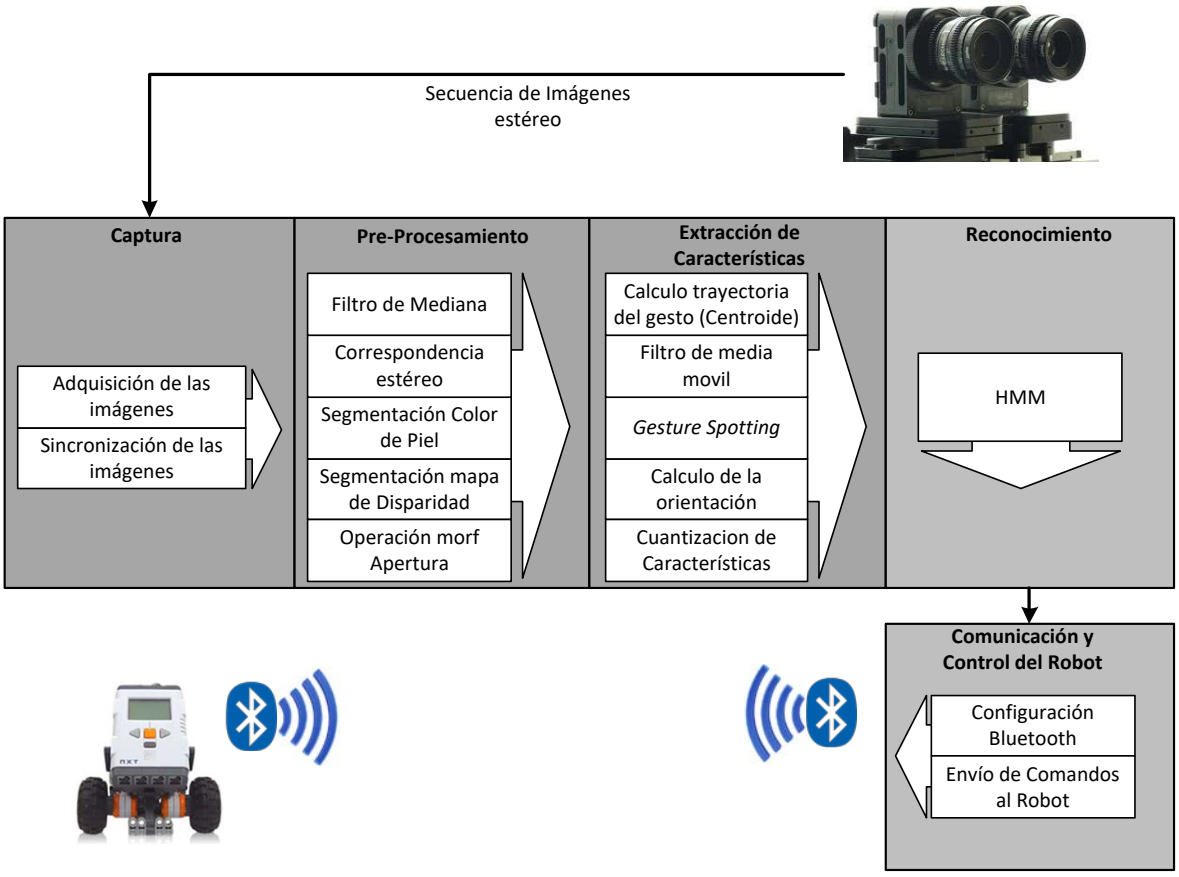


Figura 17. Esquema conceptual del sistema de reconocimiento de gestos de las manos. Fuente autor.

En la etapa de Reconocimiento se emplean HMMs como clasificadores. Los HMM son modelos probabilísticos idóneos para el modelado de patrones espacio-temporales como es el caso de los gestos empleados en este proyecto. En este trabajo se emplearon 7

HMMs con arquitectura LRB, uno por cada gesto. El entrenamiento de los HMM se realizó empleando la metodología de validación cruzada denominada "*K-Fold Cross-Validation*". Este tipo de validación permite seleccionar un modelo con el mejor rendimiento a partir de varios modelos opcionales cada uno con diferentes parámetros.

En la etapa de comunicación y control del robot se empleó un robot móvil LEGO Mindstorms NXT. El sistema de reconocimiento detecta el gesto realizado y envía al robot el comando correspondiente via Bluetooth. El robot recibe la indicación y ejecuta la acción programada de acuerdo con la acción realizada por el usuario.

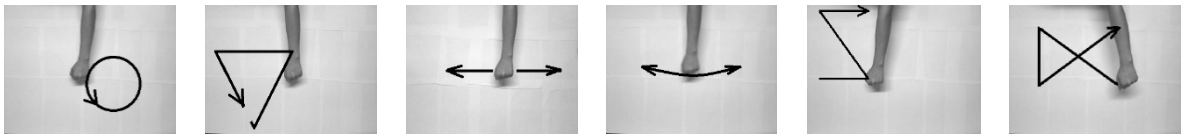
4.2 Proceso de Diseño del sistema de Reconocimiento de Gestos.

El sistema de reconocimiento de gestos de las manos presentado en este trabajo se desarrolló mediante la ejecución de tres pasos fundamentales: *i*) desarrollo de algoritmos, en esta etapa se determina la secuencia de operaciones de procesamiento de imágenes requeridas para transformar la imagen o imágenes de entrada en el resultado deseado. Además, se desarrollan los algoritmos adicionales necesarios para procesar la información extraída de las imágenes. *ii*) determinar cuáles algoritmos o porciones de estos se implementan mediante una arquitectura de hardware personalizada (FPGA) o en su defecto en un procesador convencional (software). Esta selección se realiza teniendo en cuenta la porción del algoritmo que puede ser paralelizada y la capacidad computacional requerida por este. Para la selección de la arquitectura en hardware se debe transformar el algoritmo en operaciones, las cuales tienen un equivalente de procesamiento físico. *iii*) implementación de los algoritmos sobre la arquitectura elegida de acuerdo con los parámetros de selección previamente establecidos.

4.2.1 Desarrollo de algoritmos

En este trabajo se empleó el software de desarrollo MATLAB® para el diseño y selección de los algoritmos requeridos en este caso. Asimismo, se utilizó la base datos SKIG (L. Liu & Shao, 2013), de la que se seleccionaron seis gestos dinámicos para realizar las pruebas preliminares en software (ver Figura 18). De esta primera fase se obtuvieron como resultado los algoritmos de procesamiento de imágenes que permiten extraer el gesto de la escena, y los algoritmos de tratamiento de la trayectoria del gesto que permiten realizar su descripción de forma adecuada. Los algoritmos seleccionados son: filtro de mediana, segmentación por color de piel, segmentación del mapa de disparidad, operación morfológica de apertura, cálculo de la trayectoria del gesto a partir del centroide de la forma de la mano, filtro de media móvil aplicado a la trayectoria del gesto, detección del gesto a partir del movimiento y cálculo de las características basadas en la orientación. El cómputo de la correspondencia estéreo se desarrolló de forma independiente teniendo en cuenta que la base SKIG no dispone de información estereoscópica. Sin embargo, ésta incluye la información de profundidad que fue utilizada para la selección de los algoritmos mencionados. Por lo tanto, el algoritmo de

correspondencia estéreo fue comprobado empleando las imágenes estéreo de la base de datos presentada en (Scharstein, Ugbabe, & Szeliski, 2011). Para el entrenamiento de los HMM se empleó el Toolbox para MATLAB® desarrollado por Kevin Murphy (Murphy, 1998). La evaluación de los HMM a partir de una secuencia de observaciones, se realizó preliminarmente mediante el algoritmo de avance empleando un script de MATLAB® con el fin de comprobar su funcionamiento y facilitar su posterior implementación.



Circle *Triangle* *Right-Left* *Wave* *"Z"* *Cross*

Figura 18. Gestos de la base de datos SKIG empleados para la primera fase del diseño del sistema de reconocimiento. Fuente (L. Liu & Shao, 2013).

La implementación en software de los algoritmos empleados en este trabajo se toma como modelo de referencia para la verificación de resultados obtenidos en hardware. El resultado de dichas comparaciones indica si la implementación en FPGA funciona correctamente.

4.2.2 Selección de la arquitectura

La selección de la arquitectura se lleva a cabo mediante la evaluación de los algoritmos empleados teniendo en cuenta su costo computacional y su posibilidad de ser paralelizados. Los algoritmos implementados en hardware en este trabajo son: filtro de mediana, correspondencia estéreo, segmentación por color de piel, segmentación del mapa de disparidad la operación morfológica de apertura y la extracción de la trayectoria del gesto a partir del centroide de la forma de la mano. Adicionalmente, se implementó en hardware el filtro de media móvil y la detección del gesto a partir del movimiento. La arquitectura elegida en este trabajo para la implementación de los algoritmos en hardware es una arquitectura full pipeline para procesamiento en línea de datos o también llamada "*Stream Processing*" (Bailey, 2011), la cual es una forma de paralelismo temporal y permite procesar las imágenes a la misma velocidad de captura.

Los algoritmos de alto nivel como la extracción de características, los HMM, y el control de la aplicación son implementados en software sobre un procesador convencional ya que la cantidad de información que deben procesar es baja y corresponde únicamente un punto con coordenas x,y,z por cada imagen capturada.

4.2.3 Implementación

La etapa final en el proceso de diseño es la implementación del sistema. En esta, los algoritmos se proyectan sobre los recursos de hardware elegidos según la definición de la arquitectura. Las implementaciones basadas en FPGA requieren el diseño del hardware específico encargado de llevar a cabo las operaciones de procesamiento necesarias (Bailey, 2011). En la Figura 19 se presenta el flujo de diseño para la implementación en

FPGA del sistema de reconocimiento empleado en este trabajo. Para el diseño e implementación del sistema en hardware se utilizó la herramienta de desarrollo EDA de Altera Quartus II v 14.1.

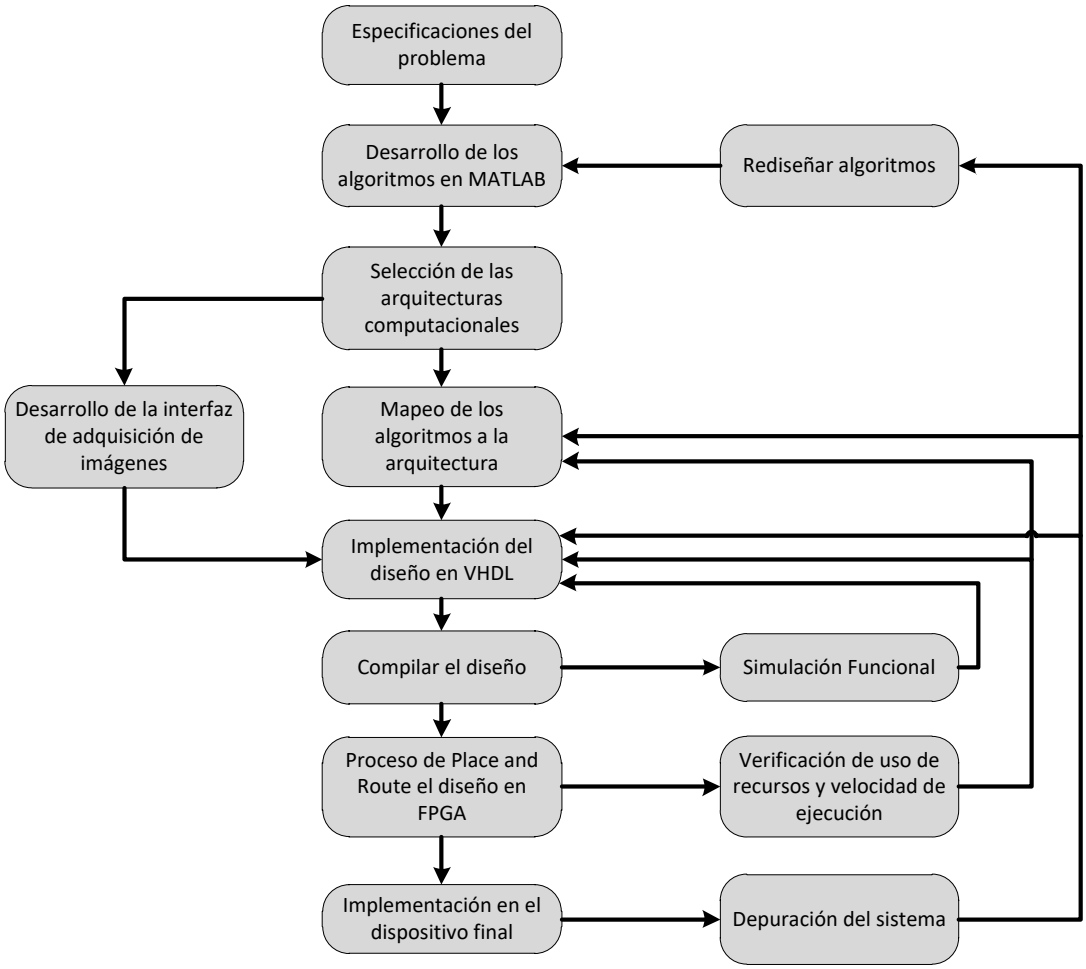


Figura 19. Flujo de diseño empleado para la implementación en FPGA del sistema de reconocimiento. Figura adaptada por el autor a partir de (Bailey, 2011)

El sistema de reconocimiento fue implementado en la tarjeta de desarrollo DE1-SoC (ver anexo A) de Terasic, la cual dispone de una FPGA CYCLONE V SoC 5CSEMA5F31C6 (ver anexo A) de la compañía Altera®, así como del hardware necesario para el desarrollo de aplicaciones de visión por computador. La descripción de cada una de las etapas del sistema de reconocimiento fue realizada empleando el lenguaje de descripción de hardware VHDL.

El diseño e implementación del sistema de reconocimiento se llevó a cabo empleando la metodología de diseño *Top-Down* siguiendo una estructura jerárquica. En la Figura 20 se presenta un esquema general de la arquitectura desarrollada para el sistema de reconocimiento de gestos descrito en este trabajo. Las etapas de captura y pre-procesamiento están implementadas en hardware. La etapa de extracción de

características está implementada en hardware y software. Las etapas de reconocimiento y control del robot están implementadas en software. Las aplicaciones desarrolladas en software se implementaron mediante el sistema de procesador físico (HPS-*Hard Processor System*) disponible dentro del dispositivo SoC. Adicionalmente, se implementaron módulos de hardware para realizar la depuración del sistema y para realimentar al usuario sobre la realización de los gestos. Estos módulos son: un multiplexor de video, un controlador VGA para visualizar las imágenes en una pantalla y un controlador de memoria RAM para sincronizar la velocidad de captura y visualización.

El sistema de procesador HPS, es un recurso de hardware que ofrecen los sistemas SoC y que permite implementar aplicaciones en software que no necesariamente requieren ser implementadas en hardware directamente. Este tipo de procesadores ya están físicamente implementados y hacen parte de los recursos de hardware disponible en la FPGA SoC, por lo cual su inclusión en una aplicación no hace uso de recursos adicionales y permite tener todo el hardware reconfigurable disponible para implementación de operaciones de alto rendimiento. En dispositivos FPGA convencionales se deben emplear recursos lógicos para sintetizar un procesador (tal como el NIOS II de Altera o el MicroBlase de Xilinx, entre otros) con el fin de ejecutar aplicaciones de software. Esto hace que la cantidad de recursos lógicos para el diseño e implementación de hardware personalizado se vea disminuida.

En los siguientes capítulos se presenta de forma detallada el diseño e implementación, tanto en hardware como en software del sistema de reconocimiento de gestos. Cada uno de los módulos de hardware diseñados son descritos desde el diseño de su arquitectura hasta la simulación y comparación de los resultados con las implementaciones realizadas en MATLAB®.

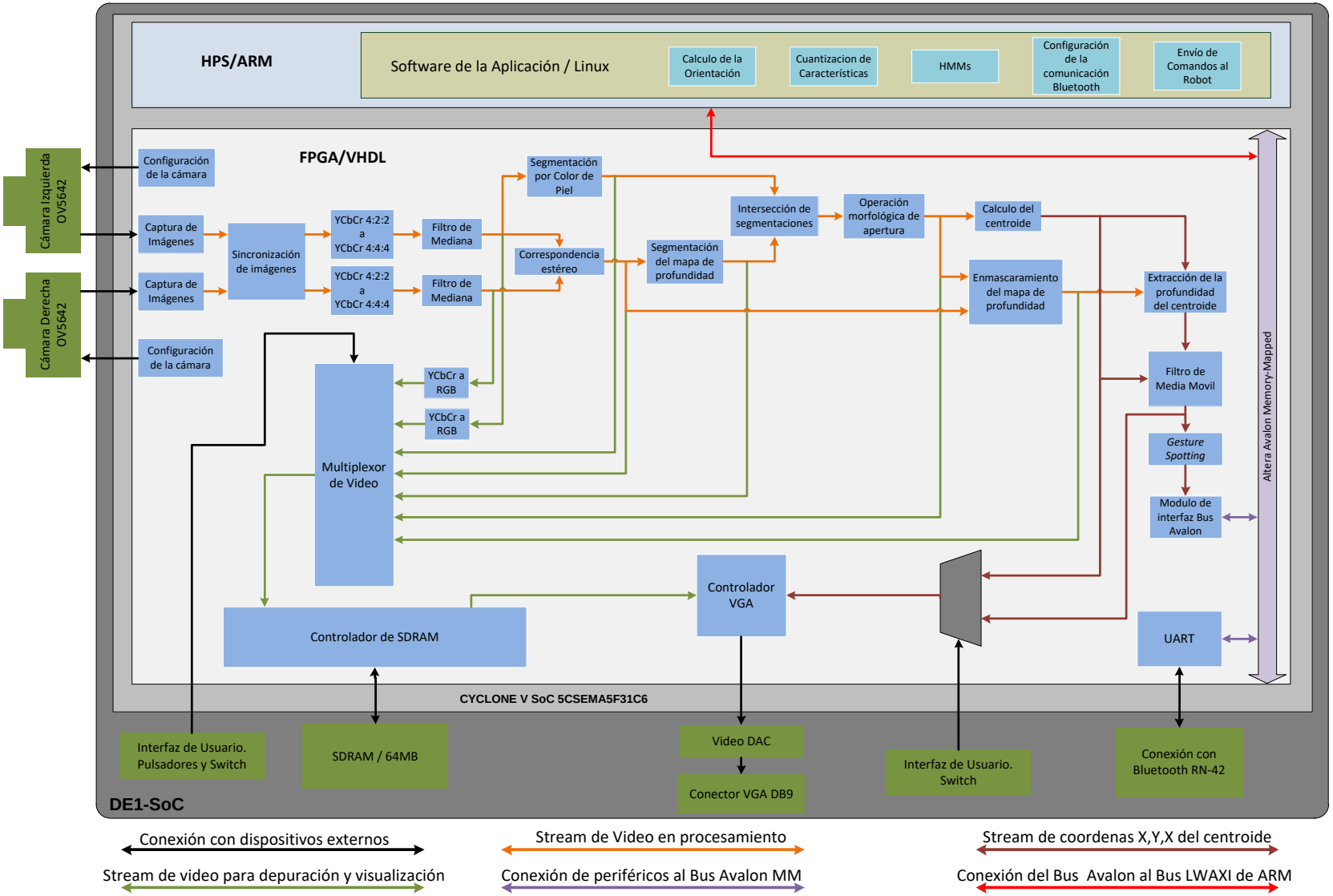


Figura 20. Diagrama general que expone la arquitectura del sistema de reconocimiento desarrollado. Fuente autor.

5 Captura de gestos

La adquisición de los gestos empleados en este trabajo se lleva a cabo mediante un sistema de visión estereoscópico el cual consta de dos cámaras digitales ArduCam separadas por una distancia entre sus ejes focales de 30.48mm. Las cámaras empleadas constan de un sensor CMOS OV5642 y de una lente objetivo de 3MP para CCTV con montura tipo CS. La lente de las cámaras tiene una distancia focal de 4mm, una apertura de 1:1.4 y formato de imagen de 1/2.5". En la Figura 21 se presenta el arreglo de cámaras desarrollado en este trabajo para la captura de imágenes estereoscópicas.

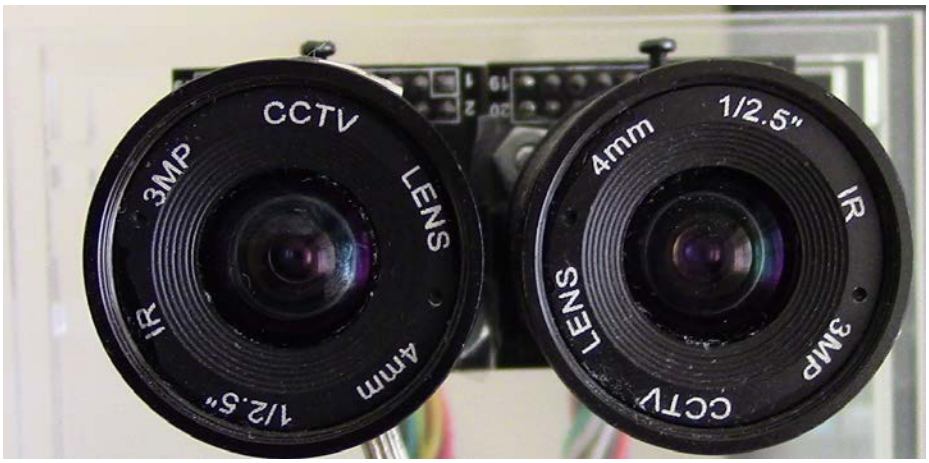


Figura 21. Sistema de cámaras estereoscópico desarrollado para la aplicación. Fuente autor.

El sistema de cámaras se conecta a la tarjeta de desarrollo DE1-SOC a través de sus puertos de expansión GPIO. La cámara derecha del sistema se conecta al puerto GPIO_0 y la cámara izquierda al puerto GPIO_1. En la Figura 22 se presenta la interconexión requerida entre una cámara y el dispositivo FPGA. Como se observa en esta figura, la conexión entre la cámara y la FPGA requiere de siete señales de conexión. Las señales SDA y SCL pertenecen al bus I2C mediante el cual se realiza la inicialización de la cámara. Las señales HREF, VSYNC, DATA[9:0] y PCLK son las señales que contienen la información de las imágenes capturadas. La señal XCLK es la fuente de reloj externa requerida por la cámara para funcionar de forma adecuada. El sensor se polariza a un voltaje de 3.3V (ver Anexo A).

Para realizar la captura de las imágenes y efectuar su respectivo procesamiento dentro de la FPGA es necesario el diseño e implementación de cuatro módulos de hardware. El módulo de configuración de la cámara es el encargado de realizar la inicialización del sensor mediante la selección de la resolución de la imagen y el *frame-rate*, entre otros parámetros. El módulo de captura es el encargado de recibir una imagen pixel por pixel haciendo un barrido de izquierda a derecha y de arriba hacia abajo iniciando desde la parte superior izquierda del sensor. Debido a que la captura de las imágenes derecha e

izquierda es realizada por módulos de hardware independientes, es posible que se produzca un desfase entre estos módulos, debido a que durante el inicio de la captura cualquiera de las cámaras se puede retrasar unos cuantos ciclos de reloj con respecto a la otra. Para corregir esto se emplea un módulo de hardware que sincroniza las dos imágenes, ya que la correspondencia estéreo y demás algoritmos del pre-procesamiento así lo requieren. Por último, se implementó un módulo que convierte el muestreo YCbCr 4:2:2, empleado por las cámaras, al formato 4:4:4 requerido para obtener un pixel en sus tres componentes de color, lo cual es necesario para realizar el pre-procesamiento posterior. A continuación se presenta el diseño de cada uno de los módulos empleados en esta etapa del sistema de reconocimiento de gestos.

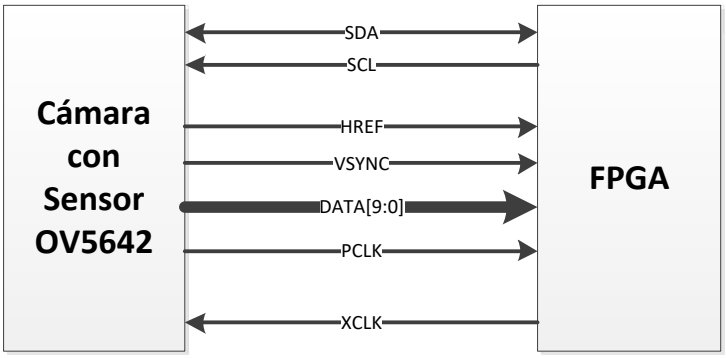


Figura 22. Interfaz de conexión entre el sensor OV5642 y la FPGA CYCLONE V SoC. Fuente autor.

5.1 Módulo de Configuración de la Cámara

Las cámaras digitales empleadas en este trabajo requieren un proceso de inicialización antes de comenzar la captura de imágenes. La inicialización de las cámaras consiste en configurar los registros internos del sensor OV5642 con el fin de establecer los siguientes parámetros de funcionamiento: resolución de captura de las imágenes, *frame-rate*, submuestreo de la imagen (*Binning*), modelo de color, y submuestreo de crominancia. En la Tabla 7 se presenta los parámetros de funcionamiento de la cámara empleados en este trabajo. En el Anexo B se presentan los valores de configuración de cada uno de los registros internos del sensor los cuales permiten establecer los parámetros de funcionamiento de la cámara antes mencionados. Los valores de configuración de la cámara fueron tomados de (Freescale Semiconductor, 2013). (Para más información sobre los registros de configuración del sensor OV5642 remítase al anexo A).

Tabla 7. Parámetros de configuración del sensor OV5642

Resolución de la Cámara	**Frame-rate**	**Submuestreo de la imagen (*binning*)**	**Modelo de Color**	**Submuestreo de Color**
640X480	30 fps	2X2	YCbCr	4:2:2

Para realizar la configuración de la cámara desde el dispositivo FPGA se realizó el diseño de un módulo de hardware que permite enviar cada uno de los valores de configuración

de la cámara empleando el protocolo de comunicación serial síncrona I2C. Para escribir la información en cada uno de los registros de configuración del sensor es necesario realizar la secuencia mostrada en la Figura 23. En primer lugar se envía la palabra de control que contiene la dirección de la cámara, después se envía la parte alta y la parte baja de la dirección del registro que se desea configurar, y por último se envía el dato. La cámara responde con una señal ACK=’0’ cada vez que recibe de manera correcta un byte.

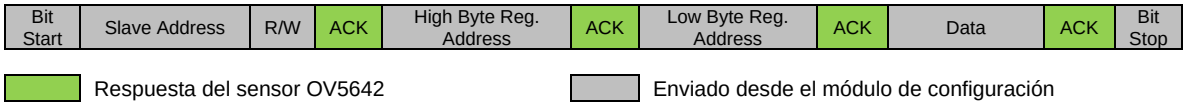


Figura 23. Secuencia de envío de datos mediante bus I2C para configuración de los registros del sensor OV5642.

En este trabajo se utilizó el módulo de configuración de la cámara TRDB_D5M desarrollado por Terasic, el cual fue modificado para su uso con el sensor OV5642. Este módulo se compone de cuatro bloques de hardware principales: divisor de frecuencia, módulo de comunicación I2C, tabla de búsqueda LUT (*Lookup-Table*) y una máquina de estados finitos. En la Figura 24 se presenta el diagrama del módulo de configuración en el que se muestran los bloques funcionales mencionados y su interconexión. En la Tabla 8 se describen los puertos de entrada-salida del módulo de configuración.

Como se observa en la Figura 24 se usó un módulo divisor de frecuencia para obtener una señal de reloj de 400KHz (a partir de una señal de entrada de 50 MHz), la cual es utilizada en la comunicación I2C. El módulo de comunicación I2C se encarga de realizar la transmisión de la información siguiendo la secuencia establecida en la Figura 23. Para esto, el modulo tiene una entrada de datos de 32 bits que contiene los 4 bytes que se requieren para la transmisión, una entrada GO que habilita la comunicación y dos salidas END y ACK que informan el estado de la transmisión. El módulo de configuración almacena en una tabla de búsqueda (LUT) tanto las direcciones de los registros como los datos que deben enviarse al sensor OV5642. La LUT fue configurada para la aplicación requerida teniendo en cuenta que se necesita un total de 988 posiciones de 24 bits cada una (16 bits de dirección del registro y 8 bits para el dato correspondiente).

Para realizar el envío de los datos de configuración disponibles en la LUT se emplea una FSM (*Finite State Machine*) de 3 estados tipo Mealy, la cual funciona de la siguiente forma: en el estado *S0* se lee la posición de la LUT apuntada por un índice y a su vez se concatena con la palabra de control 0X78. Esta palabra de control corresponde a la dirección del sensor OV5642 como esclavo para escritura mediante I2C. En este mismo estado se da inicio a la transmisión mediante GO=’1’y se hace la transición al estado *S1*. En el estado *S1* se verifican las señales END y ACK. END=’1’ indica la finalización de una transmisión y ACK=’0’ indica que la comunicación fue exitosa. Si estas dos condiciones se cumplen entonces se hace transición al estado *S2* de lo contrario la FSM se reinicia. En el estado *S2* se incrementa el índice de la LUT y se hace transición al estado *S0*. Este

procedimiento se repite hasta finalizar la lectura de todas las posiciones de la LUT. En el anexo C se presenta el diagrama de estados que muestra el comportamiento aquí descrito. En el anexo D se presentan los archivos de descripción de hardware en VHDL que modelan el comportamiento del módulo de configuración de la cámara.

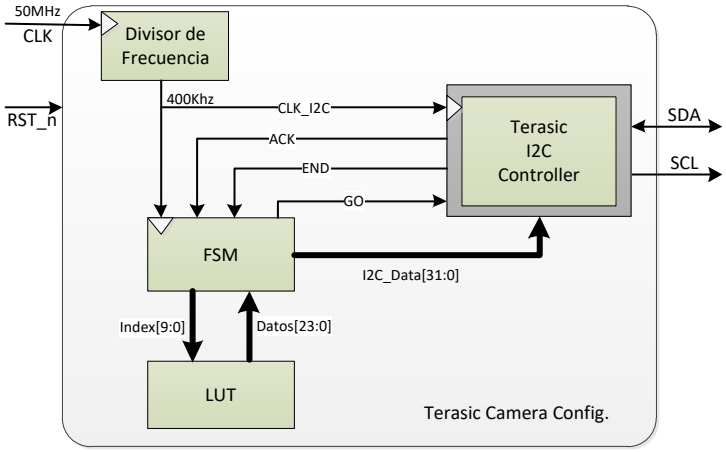


Figura 24. Arquitectura del módulo de inicialización del sensor OV5642. Fuente autor.

Tabla 8. Descripción de los puertos de entrada-salida del módulo de configuración de la cámara.

Puerto de entrada	**Función**	**Nº de Bits**
CLK	Reloj del módulo conectado a 50MHz	1
RST_n	Reset asíncrono activo bajo del módulo de configuración.	1
Puerto de salida	**Función**	**Nº de Bits**
SCL	Reloj para comunicación I2C	1
Puerto de entrada/salida	**Función**	**Nº de Bits**
SDA	Datos para comunicación I2C	1

5.2 Módulo de Captura de Imágenes

Para capturar las imágenes provenientes de la cámara se debe tener en cuenta el comportamiento de las señales de video empleadas por el sensor para transmitir la información hacia la FPGA. La Figura 25 muestra el diagrama de tiempos para el sensor OV5642. Como se observa, el sensor envía un pixel (10 bits) en cada flanco ascendente de la señal de reloj (PCLK). Para determinar si el pixel enviado por el sensor es válido se debe tener en cuenta el comportamiento de las señales VSYNC y HREF. Cuando VSYNC es ‘0’ y HREF es ‘1’ los pixeles enviados son válidos, de lo contrario se considera que la información que no corresponde a la imagen capturada. Cuando la señal VSYNC presenta un flanco descendente significa que inicia la transmisión de una nueva imagen, mientras que un flanco ascendente de esta señal indica que se ha enviado una imagen completa. Cuando la señal HREF es ‘1’ el sensor se encuentra enviando una fila de la imagen pixel a pixel.

El sensor OV5642 envía una imagen haciendo un barrido de los pixeles de arriba hacia abajo y de izquierda a derecha, por lo tanto el primer pixel enviado corresponde a la esquina superior izquierda y el ultimo pixel enviado corresponde a la esquina inferior

derecha de la imagen (ver Figura 12). Esta característica permite implementar un procesamiento paralelo de imágenes aplicando los conceptos de paralelismo temporal.

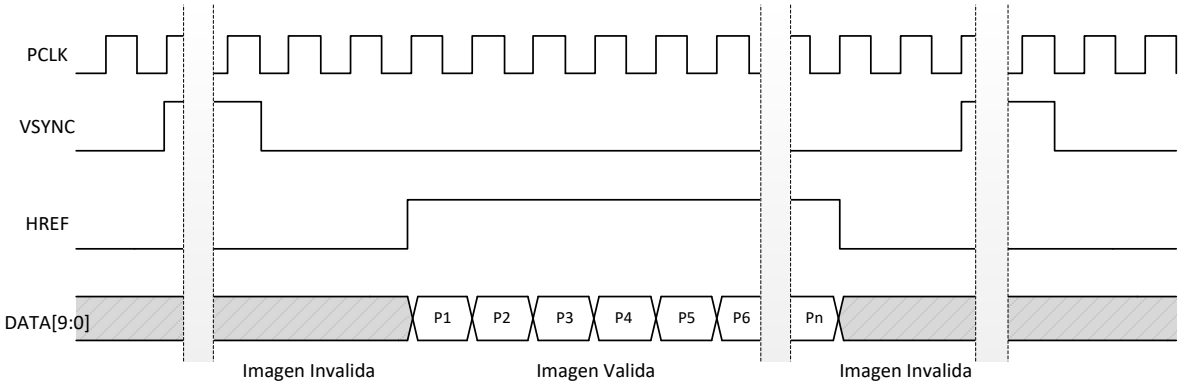


Figura 25. Diagrama de tiempos de las señales provenientes del sensor OV5642 empleadas en la captura de imágenes. Fuente autor.

En este trabajo se realizó la configuración de la cámara con submuestreo de crominancia YCbCr422. Este formato de submuestreo configura el sensor para enviar en un ciclo de reloj la componente de luminancia Y, y en el siguiente ciclo la componente de crominancia de un pixel Cb o Cr, según corresponda. Con el fin de facilitar la conversión al formato YCbCr444 se diseñó el módulo de captura. En este, la secuencia de entrada (8 bits) es transformada mediante la concatenación de la componente Y con la componente siguiente (Cb o Cr), para obtener un pixel con dos componentes (16 bits) en un solo ciclo de reloj. En la Figura 26 se presenta la secuencia de entrada al módulo de captura y la secuencia de salida generada. Como se observa, la componente Y1 ingresa como un pixel independiente de la componente Cb. Sin embargo, el pixel realmente está formado por estas dos componentes, por lo tanto, al agruparlas se obtiene un solo dato que se interpreta como un pixel a la salida Po_1.

Formato de entrada al módulo de captura

Pi_1	Pi_2	Pi_3	Pi_4	Pi_5	Pi_6	Pi_7	Pi_8	...	Pi_{n-3}	Pi_{n-2}	Pi_{n-1}	Pi_n
Y_1	Cb	Y_2	Cr	Y_3	Cb	Y_4	Cr	...	Y_{m-1}	Cb	Y_m	Cr

Formato de salida del módulo de Captura

Po_1	Po_2	Po_3	Po_4	...	Po_{m-1}	Po_m
Y_1	Y_2	Y_3	Y_4	...	Y_{m-1}	Y_m
Cb	Cr	Cb	Cr	...	Cb	Cr

Figura 26. Captura de una fila de la imagen con submuestreo de crominancia 4:2:2. Fuente autor.

En la Figura 27 se presenta la estructura interna del módulo de captura y se describe el proceso de concatenación de los datos capturados. Este se lleva a cabo mediante la interconexión de registros en cascada, enviando la información de los dos últimos registros a través de un puerto. Para determinar si un pixel capturado es válido se desarrollaron dos bloques de hardware. El primero determina el comportamiento de los puertos i_fval (conectado a VSYNC) e i_lval (conectado a HREF) de acuerdo con el diagrama de tiempos de la Figura 25. El segundo es un contador que determina la

posición del pixel capturado dentro de la fila. La señal de validez del pixel capturado se genera usando el bit menos significativo del contador, ya que los pixeles de salida se obtienen a la mitad de la frecuencia de la entrada (ver Figura 26). Adicionalmente, se emplea otro contador para determinar la fila en la que se encuentra el pixel. Los valores de conteo de las filas y columnas son enviados al exterior mediante dos puertos independientes. Los puertos i_clk e i_rst son las entradas de reloj y reset del módulo y están conectadas a todos los bloques internos. En la Tabla 9 se presenta una descripción detallada de los puertos de entrada-salida del módulo diseñado. En el anexo E se presentan los resultados de simulación del módulo de captura obtenidos mediante el software Modelsim-Altera.

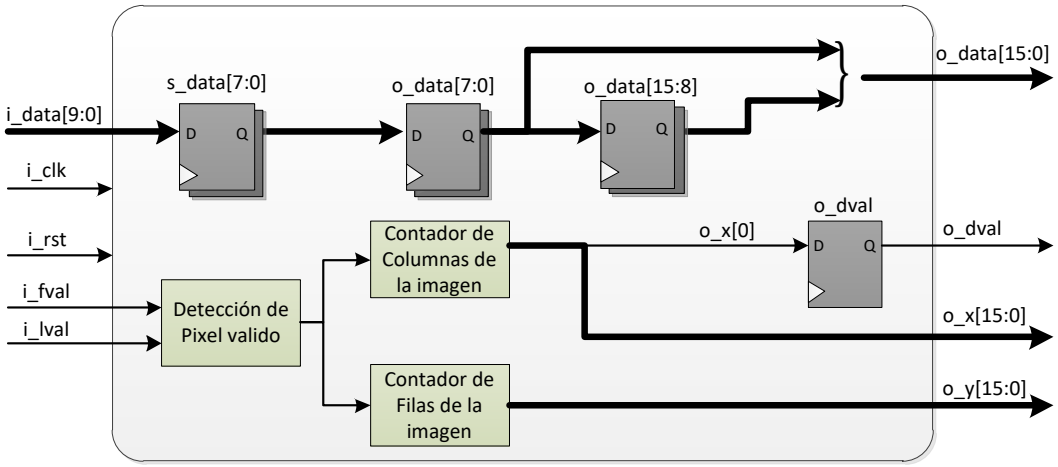


Figura 27. Arquitectura interna del módulo de captura de imágenes. Fuente autor.

Tabla 9. Descripción de los puertos de entrada-salida del módulo de captura de imágenes.

Puerto de entrada	Función	Nº de Bits
i_rst	Reset asíncrono del módulo, su funcionamiento es activo bajo	1
i_clk	Reloj de captura de la imagen proveniente de la cámara. Se conecta a PCLK	1
i_fval	Frame Valido proveniente de la cámara. Se conecta a VSYNC	1
i_lval	Línea Valida proveniente de la cámara. Se conecta a HREF	1
i_data	Datos del sensor provenientes de la cámara. Se conecta a DATA[9:2]	8
i_start	Inicio de captura de imágenes interfaz de usuario	1
I_end	Pausa de captura de imágenes interfaz de usuario	1
Puerto de salida	**Función**	**Nº de Bits**
o_dval	Determinar la validez del pixel capturado para las siguientes etapas	1
o_data	Pixel capturado con destino a las siguientes etapas	16
o_x	Coordenada x dentro de la imagen a la que pertenece el pixel capturado	16
o_y	Coordenada y dentro de la imagen a la que pertenece el pixel capturado	16

5.3 Módulo de Sincronización de Imágenes Estéreo

Las imágenes estéreo obtenidas directamente de las cámaras no están sincronizadas teniendo en cuenta que se emplean módulos de hardware independientes para configuración y captura. Cada cámara posee su propia señal de reloj por lo que no es posible garantizar la captura simultánea de los pixeles de las dos imágenes. Además, en las posteriores etapas de procesamiento, en especial en la correspondencia estéreo, se requiere que en un ciclo de reloj se disponga del pixel correspondiente a la misma

coordenada para la imagen izquierda y derecha. Para esto, se diseñó un módulo de hardware que permite corregir o sincronizar las imágenes capturadas. Este módulo captura de manera independiente las dos imágenes y posteriormente las envía hacia los siguientes módulos de procesamiento empleando una sola señal de reloj. En la Figura 28 se presenta la arquitectura del módulo de sincronización desarrollado en este trabajo y en la Tabla 10 se presenta una descripción de sus puertos de entrada-salida.

EL módulo de sincronización está compuesto por tres bloques de hardware: una FSM y dos memorias FIFO, una para la imagen izquierda y otra para la imagen derecha. Las memorias FIFO almacenan los pixeles provenientes de los módulos de captura de las cámaras izquierda y derecha, respectivamente. Cuando estas memorias han almacenado una cantidad mayor a N pixeles, la FSM inicia el proceso de lectura de N pixeles en ambas memorias. La FSM genera también una salida que indica la validez de los pixeles enviados al exterior del módulo de sincronización. En el proceso de lectura de las memorias FIFO se emplea un reloj independiente de las señales de reloj usadas en la escritura. Esto permite que los siguientes procesos tengan a su disposición las imágenes izquierda y derecha, así como el flujo de datos necesario para aplicar el procesamiento paralelo.

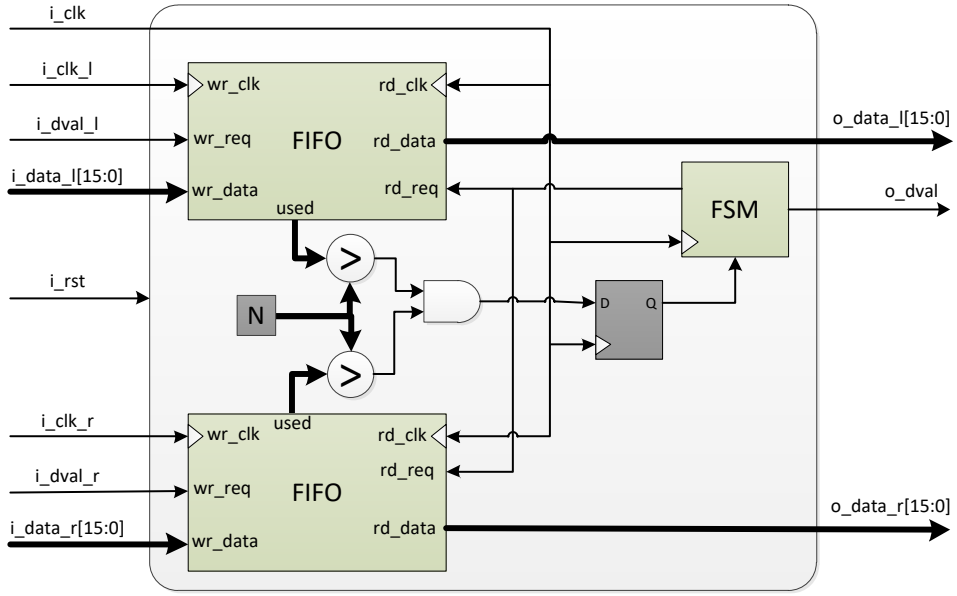


Figura 28. Arquitectura interna del módulo de sincronización de imágenes estéreo. Fuente autor.

Tabla 10. Descripción de los puertos de entradas-salida del módulo de sincronización.

Parámetros	**Función**	
N	Este parámetro permite elegir la cantidad mínima de pixeles que requieren las imágenes para sincronización	
Puerto de entrada	**Función**	**Nº de Bits**
i_clk	Reloj del módulo para sincronización de imágenes estéreo	1
i_rst	Reset del módulo activo a nivel lógico bajo	1
i_clk_l	Reloj proveniente de la cámara izquierda	1
i_dval_l	Entrada que determina la validez de un pixel proveniente de la cámara izquierda	1
i_data_l	Dato de entrada que contiene un pixel proveniente de la cámara izquierda	16

i_clk_r	Reloj proveniente de la cámara derecha	1
i_dval_r	Entrada que determina la validez de un pixel proveniente de la cámara derecha	1
i_data_r	Dato de entrada que contiene un pixel proveniente de la cámara derecha.	16
Puerto de salida	**Función**	**Nº de Bits**
o_dval	Salida que indica la validez de la sincronización para un pixel de las cámaras izquierda y derecha	1
o_data_l	Dato de salida que contiene un pixel proveniente de la cámara izquierda.	16
o_data_r	Dato de salida que contiene un pixel proveniente de la cámara derecha.	16

5.4 Módulo de Conversión de YCbCr422 a YCbCr444

Como se mencionó anteriormente, el sensor empleado en este trabajo fue configurado para capturar imágenes mediante submuestreo de crominancia YCbCr422. Este tipo de submuestreo permite obtener las imágenes de las cámaras a una frecuencia de pixel menor comparado con el uso de un muestreo completo. Esto se debe a que la cantidad de información para cada pixel está limitada a dos componentes de color únicamente: la componente de luminancia Y y una sola componente de crominancia (Cb o Cr). Sin embargo, las etapas de procesamiento posteriores requieren de imágenes con pixeles representados en las tres componentes de color. Por esta razón, es necesario hacer un proceso de conversión que consiste en transformar 1 pixel de dos componentes en 1 pixel de tres componentes o YCbCr444. El módulo de hardware diseñado para realizar esta conversión se presenta en la Figura 29. En la Tabla 11 se presenta una descripción detallada de los puertos de entrada-salida que componen el modulo diseñado.

Para el diseño del módulo de hardware que realiza esta conversión se debe considerar la información enviada desde el módulo de captura, de acuerdo a expuesto en la sección 5.2. Los pixeles capturados correspondientes a las columnas pares de la imagen contienen las componentes Y y Cb, mientras que los pixeles de las columnas impares contienen las componentes Y y Cr respectivamente (ver Figura 26). Teniendo en cuenta este orden de captura para los pixeles en una fila de la imagen, el proceso de conversión de YCbCr422 a YCbCr444 se desarrolla empleando el método de replicación simple de las componentes Cb y Cr presentado en (Gregg Hawkes, 2001). Este método de replicación consiste en tomar dos pixeles consecutivos ($P_1=\{Y_1,Cb_1\}$ y $P_2=\{Y_2,Cr_2\}$) y a partir de dichos pixeles obtener dos nuevos pixeles replicando las componentes Cb y Cr para los que se mantienen sus componentes de luminancia Y, de tal forma que el resultado es de dos pixeles en tres componentes ($P_1=\{Y_1,Cb_1,Cr_2\}$ y $P_2=\{Y_2,Cb_1,Cr_2\}$).

En el presente trabajo se realizó el diseño de un módulo de hardware para realizar la conversión de YCbCr422 a YCbCr444. Como se observa en la Figura 29, el módulo de hardware desarrollado para esta conversión consta de tres elementos fundamentales: una FSM, un multiplexor y registros. La FSM determina la validez de un pixel de entrada y evalúa la columna a la que pertenece dicho pixel dentro de la imagen. En caso de que la columna sea par, el dato de entrada es almacenado en los registros Y1 y Cb1, y si la columna es impar, el dato de entrada es almacenado en los registros Y2, Cr1 y Cr2, y el registro Cb2 carga la información contenida en Cb1. Para enviar el dato correcto al exterior, la FSM controla un multiplexor enviando a la salida Y1, Cb1 y Cr1 si el pixel

pertenece a una columna par, o Y2, Cb2 y Cr si pertenece a una columna impar. El módulo de hardware desarrollado tiene una latencia fija de tres ciclos de reloj.

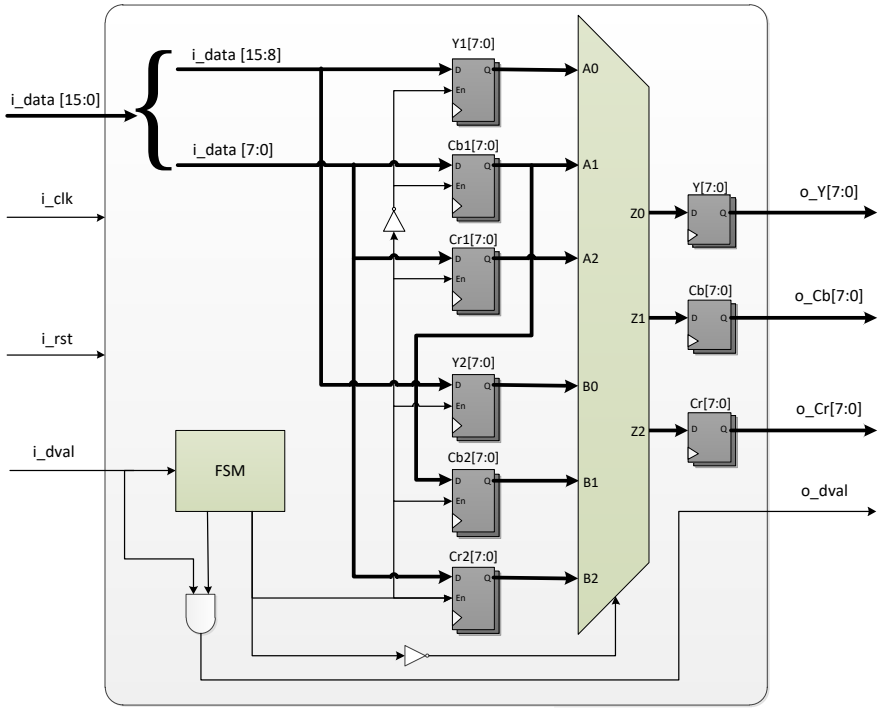


Figura 29. Arquitectura interna del módulo de conversión de YCbCr422 a YCbCr444. Fuente autor.

Tabla 11. Descripción de los puertos de entrada-salida del módulo de conversión se YCbCr 4:2:2 a 4:4:4.

Puerto de entrada	Función	Nº de Bits
i_clk	Entrada de Reloj del modulo	1
i_rst	Entrada de reset asíncrono activo a nivel lógico bajo	1
i_dval	Entrada que determina la validez de un pixel	1
i_data	Dato de entrada que contiene un pixel en formato YCbCr422	16
Puerto de salida	**Función**	**Nº de Bits**
o_y	Dato de salida de la componente de luminancia Y	8
o_cb	Dato de salida de la componente de Crominancia Azul Cb	8
o_cr	Dato de salida de la componente de Crominancia Roja Cr	8
o_dval	Salida que determina la validez de un pixel en sus tres componentes YCbCr	1

En el anexo D se presenta la descripción en VHDL de los modulos de sincronización y de conversión YCbCr422 a YCbCr444. En el anexo E se presentan los resultados de simulación obtenidos mediante el software Modelsim-Altera para cada caso.

6 Pre-procesamiento

En este capítulo se describen los algoritmos involucrados en la etapa de pre-procesamiento del sistema de reconocimiento. El pre-procesamiento consiste en la aplicación de algoritmos en un orden determinado con el fin de extraer la forma de la mano de la escena. En primer lugar, se aplica el filtro de mediana a las imágenes de entrada con el fin de eliminar el ruido que se produce en el sensor al momento de la captura. Una vez filtradas las imágenes se obtiene el mapa de disparidad mediante un algoritmo de correspondencia estéreo. El resultado de esta operación se filtra mediante el filtro de mediana con el fin de suavizar el resultado obtenido. Para la extracción de la forma de la mano de la escena se aplica un proceso de segmentación conformado por la segmentación de color de piel y la segmentación del mapa de disparidad. El proceso de segmentación produce ruido que se filtra mediante la operación morfológica de apertura.

Los algoritmos de la etapa de pre-procesamiento implementados en hardware poseen una arquitectura computacional para *Stream Processing*. Esta arquitectura permite aprovechar el flujo de datos proveniente de la cámara ya que éstos ingresan al dispositivo mediante el formato de escaneo Ráster el cual entrega un pixel en cada flanco ascendente de la señal de reloj. El modulo diseñado posee una señal de control que indica la validez del dato entregado por cada módulo, lo que permite que la imagen sea procesada al mismo tiempo que es capturada. En la Figura 30 se presenta un esquema general de la etapa de pre-procesamiento implementada en FPGA. En esta figura se muestra el flujo de los datos y de la señales de control a través de los módulos que intervienen en esta etapa.

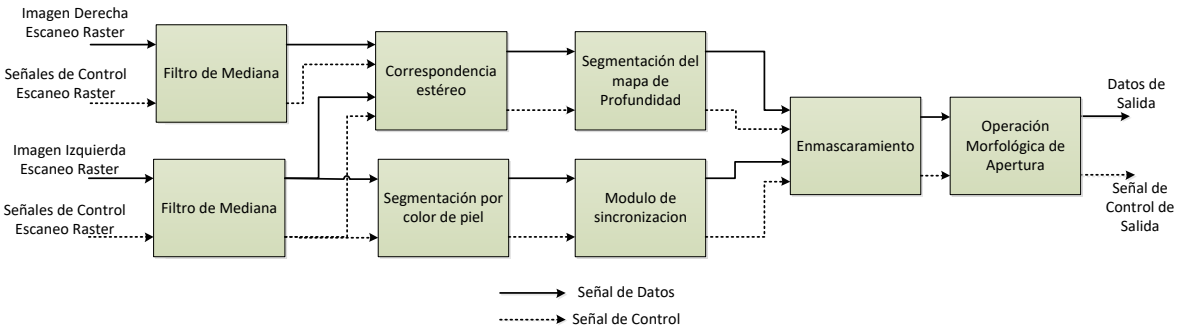


Figura 30. Interconexión de los componentes que conforman la etapa de pre-procesamiento. Fuente autor.

6.1 Filtro de Mediana

Las imágenes obtenidas de las cámaras contienen ruido que se debe principalmente a las características del sensor utilizado y al método de submuestreo de crominancia empleado en la etapa de captura. Para esta razón, se utiliza el filtro de mediana que permite eliminar el ruido sin distorsionar la información contenida en las imágenes. Aunque este filtro

elimina eficientemente el ruido de tipo impulsivo (conocido como sal y pimienta), también se puede emplear en la eliminación del ruido gaussiano, ya que a diferencia de otros filtros como el de promedio, la imagen resultante no presenta demasiada borrosidad. El filtro de mediana consiste en reemplazar el valor de un pixel de la imagen por la mediana de su vecindad W. Esto permite corregir los pixeles cuya intensidad es muy diferente a la de sus vecinos, obteniendo un suavizado o eliminación del ruido en la imagen.

Por definición el filtro de mediana es un filtro espacial no lineal (Bailey, 2011). Este filtro toma una vecindad W para cada pixel dentro de la imagen y ordena de menor a mayor los valores de los pixeles dentro de dicha vecindad. La salida del filtro es el valor de la mediana obtenida en dicho proceso de ordenamiento. Debido a la característica del filtro se requiere un costo computacional relativamente grande si se implementa en software sobre una arquitectura de procesamiento convencional. Sin embargo, dicho filtro puede implementarse de forma paralela, haciéndolo ideal para su aplicación en hardware.

```
1:  procedure MedianaIm(I_i)
2:      Input: Una Imagen de entrada I_i
3:      Output: Imagen Filtrada I_o
4:      I_W ← ancho de la imagen I_i en pixeles
5:      I_H ← alto de la imagen I_i en pixeles
6:      N_w ← Tamaño de la ventana del filtro
7:      V[N_w × N_w] ← Vector Temporal para construccion de la ventana del filtro
8:      for j ← 2 , I_W do          ▷ Recorrido de la ventana sobre las columnas de la imagen
9:          for i ← 2 , I_H do          ▷ Recorrido de la ventana sobre las filas de la imagen
10:             w ← 0
11:             for l ← 1 , N_w do          ▷ Extracción de la Ventana a procesar, en un vector V
12:                 for k ← 1 , N_w do
13:                     w ← w + 1
14:                     V[w] ← I_i[i − 2 + l, j − 2 + k]
15:                 end for
16:             end for
17:         end for
18:         I_o[i, j] ← MEDIANA(V)          ▷ Calculo de la mediana sobre la ventana V
19:     end for
20:     return I_o
21: end procedure
```

Figura 31. Algoritmo Correspondiente al filtro de mediana aplicado a una imagen de entrada I_i

Tomando como referencia el proceso de diseño propuesto en la sección 4.2, para el diseño e implementación de algoritmos de procesamiento de imágenes en FPGAs es preferible realizar inicialmente la implementación de dicho algoritmo en una herramienta de software. Esto permite comprender mejor su funcionamiento y desarrollar la arquitectura de hardware requerida. En este trabajo se realizó el diseño del filtro de mediana en MATLAB® y luego se procedió a desarrollar el respectivo mapeo en la arquitectura de hardware deseada. En la Figura 31 se presenta el algoritmo

correspondiente a la aplicación del filtro de mediana a una imagen. En esta figura se puede observar el recorrido de la ventana sobre la imagen y la aplicación del filtro para cada una de las ventanas desplazadas. En la Figura 32 se presenta el algoritmo correspondiente al cálculo de la mediana siguiendo el método de burbuja para el ordenamiento de datos. En el anexo F se presenta el desarrollo del filtro de mediana empleando el software MATLAB®.

```
1:  procedure MEDIANA(V)
2:      Input: Un vector con el contenido de la ventana a procesar V
3:      Output: Calculo de la mediana M
4:      N_v ← Tamaño del vector V
5:      for j ← 1 to N_v do                ▷ Algoritmo de burbuja para ordenar los elementos de V
6:          for i ← 1 to N_v − j do
7:              if V[i] > V[i+1] then
8:                  aux ← V[i]
9:                  V[i] ← V[i+1]
10:                 V[i+1] ← aux
11:             end if
12:         end for
13:     end for
14:     return V[N_v/2 + 1]                ▷ Retorno del valor correspondiente a la mediana
15: end procedure
```

Figura 32. Algoritmo Correspondiente al cálculo de la mediana aplicada a un vector de entrada *V*

6.1.1 Arquitectura en hardware

El diseño del filtro de mediana en FPGA se realiza para que sea compatible con la arquitectura computacional (*Stream Processing*). El filtro de mediana recibe un nuevo pixel en cada ciclo de reloj, lo procesa y envía un pixel procesado al siguiente módulo para su posterior tratamiento. El módulo de hardware para el filtro de mediana está diseñado empleando las técnicas de mapeo de algoritmos conocidos como segmentación (*pipelining)* y almacenamiento en caché (*Row Buffering*). La descripción detallada de estas técnicas de mapeo se presenta en la sección 3.3.3. Dichas técnicas de mapeo permiten procesar un nuevo pixel en cada ciclo de reloj y conlleva a ejecutar la operación mediante paralelismo temporal. El modulo diseñado está conformado por tres bloques funcionales: generador de ventana móvil 3X3, filtro de mediana y generador de señal de validez para el pixel de salida. El flujo de los datos de entrada y las señales de control de un bloque funcional a otro se muestra en la Figura 33.

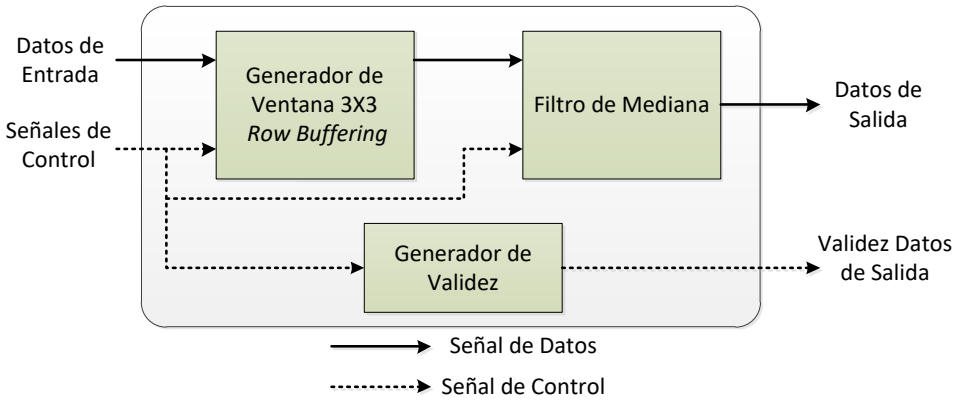


Figura 33. Modulo de hardware del filtro de mediana. Flujo de las señales de datos y control. Fuente autor.

En la Figura 34 se muestra el diagrama esquemático del módulo de hardware correspondiente al filtro de mediana. En este diagrama se muestra la interfaz de conexión del módulo. El módulo posee dos parámetros genéricos N y M empleados en el proceso de síntesis del componente. El parámetro N permite seleccionar la cantidad de bits de los datos de entrada y salida del módulo. El parámetro M establece el ancho de la imagen para la construcción de los *Row Buffer*. En cada flanco ascendente de la señal de reloj (i_clk) se captura un dato de entrada y a su vez se genera un nuevo dato en la salida. La señal de reinicio asíncrono (i_rst_n) es activa a nivel lógico bajo y permite reestablecer en cualquier momento el funcionamiento del módulo. La señal de pixel valido (o_valid) se establece en alto si el pixel de salida es válido, siempre y cuando el pixel de entrada sea válido (i_valid a nivel lógico alto). La señal de entrada i_valid actúa como habilitador del módulo. Cuando i_valid está en bajo los datos dentro del módulo no cambian y los datos de entrada se ignoran.

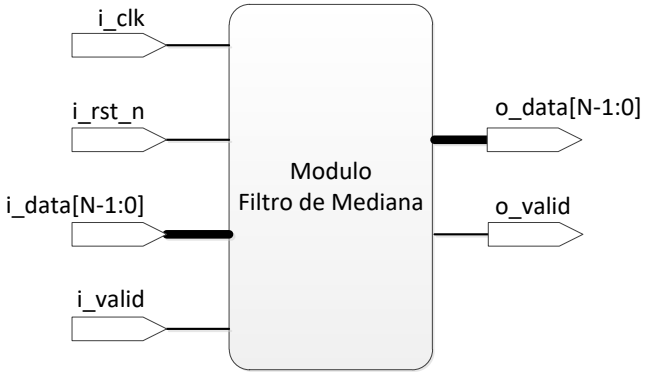


Figura 34. Diagrama esquemático del modulo de filtro de Mediana. Fuente autor.

6.1.2 Generador de ventana móvil

El método de mapeo para almacenamiento caché *Row Buffering* se emplea construir una arquitectura de ventana móvil. Este tipo de arquitectura permite acceder a los pixeles en filas consecutivas de la imagen sin la necesidad de almacenar la imagen completa en

memoria externa (ver sección 3.3.3.2), por lo que se mantiene el flujo de datos continuo requerido en las etapas de procesamiento siguientes. En este trabajo se seleccionó una ventana de 3X3 de tal forma que se requieren dos *Row Buffer* que permiten cachear los valores de los pixeles de dos filas precedentes. También se requieren nueve registros para almacenar los valores de la ventana antes de que sean procesados por el modulo.

La arquitectura de ventana lee los pixeles provenientes del exterior fila por fila en una exploración ráster. Un pixel de entrada ingresa al registro W_1 y se desplaza a la derecha en cada ciclo de reloj (ver Figura 35). Los pixeles se desplazan continuamente de izquierda a derecha sobre los registros y buffers. Cuando los dos *Row Buffer* se llenan y el primer pixel de la imagen llega al registro W_5 se obtiene la primera ventana 3X3 lista para ser procesada.

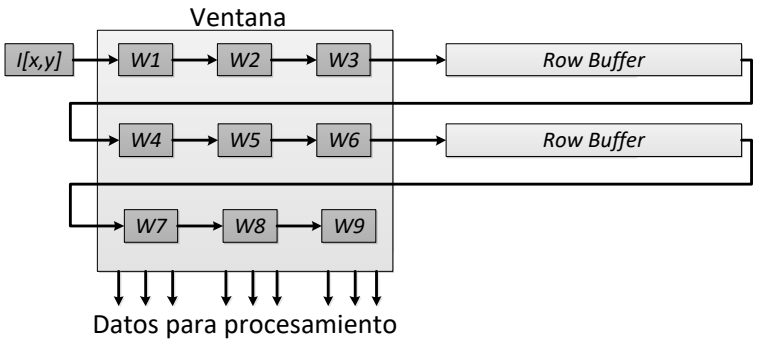


Figura 35. Aquitectura de cache empleando *Row Buffering*. Adaptacion a partir de (Bailey, 2011).

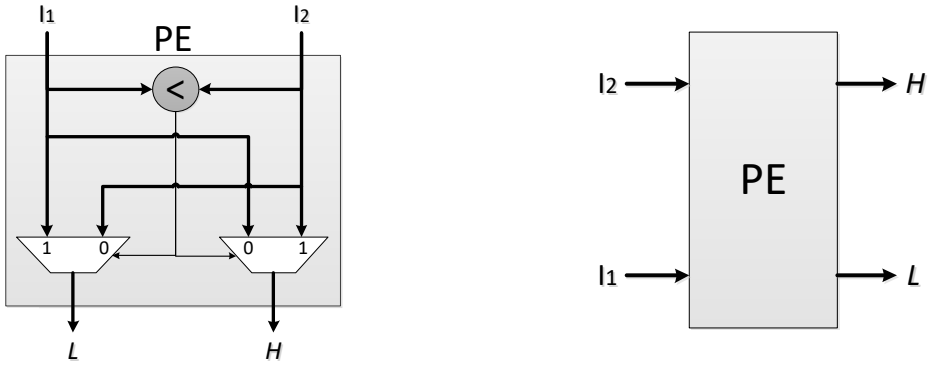
6.1.3 Red de ordenamiento para el cálculo de la mediana

El filtro de mediana toma los nueve datos de entrada provenientes del generador de ventana móvil y obtiene su mediana. Como se describió en el algoritmo de la Figura 32, el filtro de mediana consiste en el ordenamiento de varios elementos, para posteriormente seleccionar el valor de la mediana resultante de dicho proceso. La implementación en hardware de este algoritmo se lleva a cabo mediante redes de ordenamiento. Una red de ordenamiento (*Sorter Network*, SN) está definida como una red de operaciones elementales denotadas como comparar&intercambiar (*compare&Swap*, CS*)* (Vasicek & Sekanina, 2008). La cantidad de operaciones CS en una SN está determinada por la cantidad de elementos a ordenar. Las SN se caracterizan porque su implementación en hardware es completamente paralela y permiten el desarrollo una arquitectura *pipeline*.

De acuerdo con (Habermann, 1972; Lakshmivarahan, Dhall, & Miller, 1984) existen varias SN dentro de las cuales se encuentra la red de ordenamiento de transposición impar-par. Esta red de ordenamiento consiste en realizar comparaciones de forma similar al proceso realizado por el algoritmo de burbuja. El funcionamiento de este método se lleva a cabo mediante la comparación de parejas de elementos con índices impar/par. Si una pareja de elementos se encuentra en el orden incorrecto los elementos son reordenados. El siguiente paso consiste en realizar la comparación de elementos adyacentes con índices

par/impar. El proceso se repite de forma alterna hasta que la lista de elementos se encuentre completamente ordenada.

El elemento de procesamiento CS es el componente fundamental de la SR de transposición impar-par. En la Figura 36 se presenta la arquitectura correspondiente, la cual consta de un comparador de magnitud y dos multiplexores. Los datos de entrada I_1 e I_2 ingresan al comparador el cual evalúa si $I_2 > I_1$. En caso que sea verdadero, la salida del comparador se pone a 1 controlando los multiplexores para que los datos salgan en el mismo orden de ingreso (sin intercambio) por lo cual $H = I_2$ y $L = I_1$. En caso que la comparación sea falsa los datos de entrada son intercambiados obteniendo $H = I_1$ y $L = I_2$



(a) (b)

Figura 36. Elemento de procesamiento de comparacion e intercambio CS. (a) arquitectura interna. (b) diagrama esquemático. Fuente autor.

La implementación en FPGA de la SN se realiza mediante capas de elementos de procesamiento CS interconectados siguiendo el proceso descrito por (Habermann, 1972). En la primera capa de la red se toman los valores provenientes del generador de ventana móvil, y se agrupan por parejas tomando los datos con índices impar-par (de arriba hacia abajo) para cada CS. La segunda capa tiene un comportamiento similar, pero esta vez tomando los datos con índices par-impar. Para llevar a cabo el ordenamiento de 9 datos se requieren 9 capas de procesamiento con 4 elementos de comparación por capa. Para llevar a cabo el procesamiento *pipeline*, basta con poner registros entre cada una de las capas de procesamiento, obteniendo una arquitectura con 9 etapas de pipeline. En la Figura 37 se presenta la construcción completa de la SN.

Los valores que ingresan a la SN son ordenados de mayor a menor, siendo la posición 9 el valor máximo y la posición 1 el valor mínimo. Por lo tanto, para obtener la mediana de los datos de entrada se toma el valor obtenido en la posición 5. Para obtener un valor valido de mediana se debe tener en cuenta la latencia de la SN y la latencia del generador de ventana móvil, por lo que se requiere de un módulo de hardware independiente que permita determinar dicho estado.

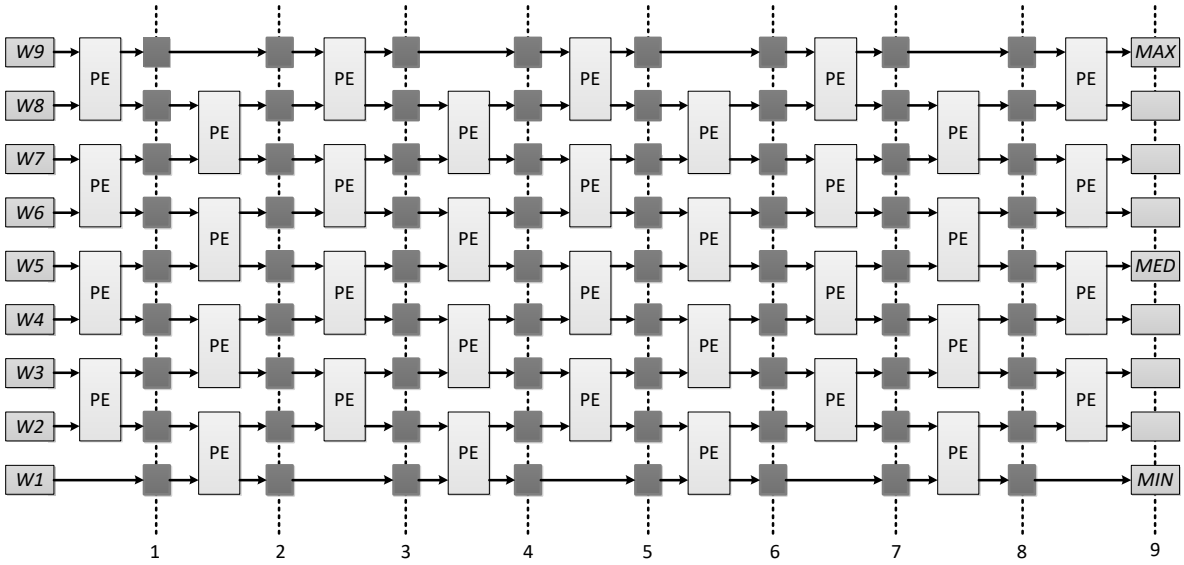


Figura 37. Aquitectura full-pipeline.del filtro de mediana basada en una red de ordenamiento de transposicion impar-par. Fuente autor.

6.1.4 Generador de la señal de validez de los datos de salida

Con el fin de saber si si los datos de salida del filtro son validos se requiere de una señal de validez denominada *o_valid*.

El módulo del filtro de mediana empieza a producir datos validos cuando los primeros pixeles se han propagado a través del generador de ventana y de la SN. De acuerdo con el diagrama de tiempos de la Figura 38 se observa que la generación de datos validos (señal *o_valid* en alto) se debe iniciar después del ingreso de n datos válidos al módulo. A partir de ese momento, los datos validos de salida se generan siempre y cuando la señal de control (*i_valid*) se encuentra en alto. Esto implica que existe una latencia n en el procesamiento la cual está determinada por el ancho de la imagen a procesar, el tamaño de la ventana y la cantidad de etapas *pipeline* de la SN.

La latencia total n del módulo del filtro de mediana se calcula teniendo en cuenta la latencia producida por el generador de ventana móvil y la SN. Para obtener la primera ventana válida a ser procesada se requiere de $M+2$ datos validos de entrada, siendo M el ancho de la imagen. El procesamiento de una ventana mediante la SN requiere de 9 datos o ciclos de reloj válidos. Por lo tanto, la latencia total del módulo es de $n = M+11$ ciclos validos de reloj.

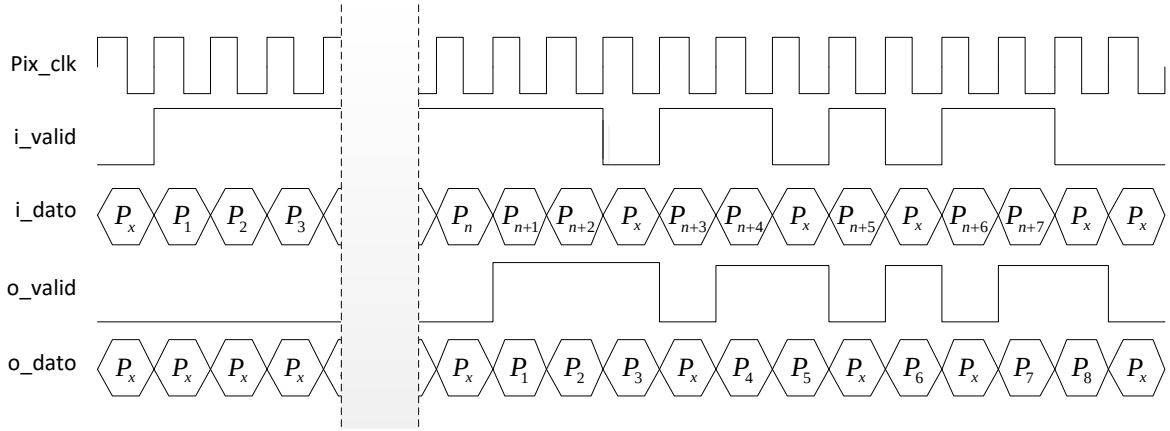


Figura 38. Comportamiento del filtro de mediana en el tiempo. Fuente autor.

Teniendo en cuenta el comportamiento y la latencia total de modulo del filtro de mediana, se desarrolla un bloque funcional que permite generar de forma correcta la señal de salida *o_valid*. En la Figura 39 se presenta el circuito esquemático correspondiente a dicho bloque funcional. Para la espera de los primeros ciclos de reloj validos se emplea un contador ascendente el cual se incrementa hasta el valor $M+10$. Los conteos son controlados por la señal de entrada *i_valid* y por un comparador de magnitud. El comparador de magnitud controla a su vez la señal de salida *o_valid*. Por lo tanto, mientras el contador sea menor a $M+10$ la señal de validez permanece en bajo sin importar el estado de la entrada *i_valid*. Cuando el contador es igual a $M+10$ la señal de validez es controlada por la entrada *i_valid*.

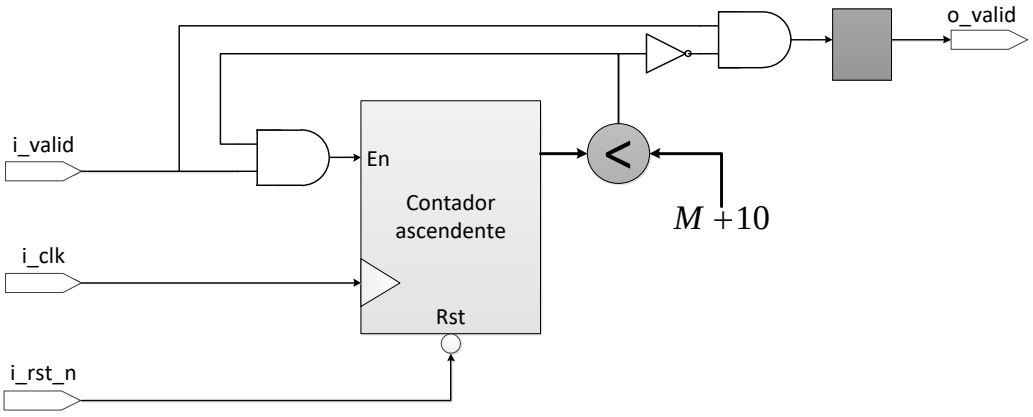


Figura 39. Generador de la señal de validez de los pixeles de salida procesados. Fuente autor.

6.2 Correspondencia Estéreo

El algoritmo de correspondencia estéreo se implementa inicialmente en software mediante el uso de MATLAB®. El mapa de disparidad obtenido en hardware es comparado con el mapa de disparidad obtenido en software; el resultado de dicha comparación indica si la

implementación en FPGA funciona correctamente. A continuación se presenta, de manera detallada, el desarrollo del módulo de correspondencia estéreo, iniciando por su implementación en software y el desarrollo posterior de su arquitectura en hardware.

Según (Fife, 2011; Fife & Archibald, 2013) el método de visión estéreo basado en la transformación no paramétrica Census proporciona una exactitud de correlación superior comparado con otros métodos de visión estéreo basados en el área. Mediante este método, el cálculo del mapa de disparidad se lleva a cabo aplicando, en primer lugar, la transformación de Census a cada una de las imágenes estéreo. En este trabajo se empleó la transformada de Census dispersa al 50% como se explicó en las secciones 3.1.1 y 3.1.2., ya que se obtienen resultados similares a la transformación original, pero disminuyendo a la mitad la cantidad de recursos (especialmente memoria) requeridos para la implementación en FPGA (Humenberger et al., 2010). Una vez realizada la transformación, se evalúa el grado de similitud entre las imágenes mediante el método de correlación denominado Suma de Distancias de Hamming (SHD). Para mejorar el resultado del mapa de disparidad se emplea el algoritmo de comprobación de consistencia Izquierda-Derecha LRCC (Left-Right Consistency Check).

En la Figura 40 se presenta el algoritmo que describe su implementación en software de la transformada de Census dispersa al 50%. La imagen se recorre de arriba hacia abajo y de izquierda a derecha. Para cada ventana centrada en $I_i[i, j]$ se obtiene una cadena de $\frac{W_c^{\,2}-1}{2}$ bits, resultado de la comparación del pixel central y sus vecinos teniendo en cuenta la máscara de la Figura 7.b. El resultado de la transformada Census es una matriz de dimensiones $I_H \times I_W \times \frac{W_c^{\,2}-1}{2}$, siendo I_W e I_H el ancho y alto de la imagen original y W_c el tamaño de la ventana de Census.

Después de aplicar la transformada Census a la imagen izquierda y derecha, se calcula el mapa de disparidad siguiendo el algoritmo mostrado en la Figura 41. Este Algoritmo permite calcular los mapas de disparidad tomando como referencia la imagen izquierda M_{L2R} y la imagen derecha M_{R2L}. El cálculo del mapa de disparidad se lleva a cabo desplazando sobre las imágenes, las ventanas de comparación en la dirección de arriba hacia abajo y de izquierda a derecha. Cada ventana en la imagen de referencia se compara con d ventanas en la imagen objetivo. Para obtener el mapa de disparidad (*left-to-right*) *L2R* la ventana de la imagen izquierda centrada en $[i, j]$ se compara con d ventanas a la izquierda de la imagen derecha. Para el mapa de disparidad (*right-to-left*) *R2L* la ventana de la imagen derecha centrada en $[i, j-d]$se compara con d ventanas a la derecha de la imagen izquierda. La comparación de dichas ventanas se lleva a cabo mediante la función de coste SHD. El resultado de comparar la ventana de referencia con las ventanas objetivo en cada mapa de disparidad, es un vector de longitud d . El valor de

disparidad está determinado por la posición del vector, cuyo valor de comparación entre ventanas es mínimo.

procedure TRANSFORMADA CENSUS DISPERSA(I_i,W_c)
  Input: I_i: Imagen de entrada en escala de grises
    W_c: Tamaño de la ventana de Census

  Output: I_o: Imagen Transformada

  $I_W \leftarrow$ ancho de la imagen I_i en pixeles
  $I_H \leftarrow$ alto de la imagen I_i en pixeles
  $I_o[I_W, I_H, \frac{W_c^2-1}{2}] \leftarrow$ Declaracion de la imagen de salida e inicializacion en 0
  $d \leftarrow \frac{W_c-1}{2}$ parámetro requerido para la transformada dispersa de Census

  for $i \leftarrow d+1$, $I_W - d$ **do** ▷ Recorrido por las Filas de la imagen
    for $j \leftarrow d+1$, $I_H - d$ **do** ▷ Recorrido por las Columnas de la imagen
      for $m \leftarrow 1$, $d+1$ **do** ▷ Recorrido Filas impares de la ventana w
        for $n \leftarrow 1$, d **do** ▷ Recorrido Columnas pares de la ventana w
          if $I_i[i-(d+1)+2\times m-1, j-(d+1)+2\times n] \geq I_i[i,j]$ **then**
            $I_o\left[i, j, int\left[\frac{w\times(2\times(m-1))+2\times n+1)}{2}\right]\right] \leftarrow 1$
          end if
        end for
      end for
      for $m \leftarrow 1$, d **do** ▷ Recorrido Filas pares de la ventana w
        for $n \leftarrow 1$, $d+1$ **do** ▷ Recorrido Columnas impares de la ventana w
          if $I_i[i-(d+1)+2\times m, j-(d+1)+2\times n-1] \geq I_i[i,j]$ **then**
            $I_o\left[i, j, int\left[\frac{w\times(2\times(m-1)+1)+2\times n)}{2}\right]\right] \leftarrow 1$
          end if
        end for
      end for
    end for
  end for
  return I_o
end procedure

Figura 40. Algoritmo para la implementacion Software de la transformada Census Dispersa.

El cálculo de la diferencia entre dos ventanas empleadas para calcular el mapa de disparidad se lleva a cabo mediante la Suma de Distancias de Hamming. El algoritmo que describe este método se presenta en la Figura 42. Para encontrar el grado de diferencia entre dos ventanas de igual tamaño se aplica la operación lógica XOR entre las ventanas y luego se cuentan los 1’s obtenidos en la operación. Esta operación lógica permite encontrar la diferencia entre cadenas de bits, ya que si dos bits de cadenas diferentes son comparados, el resultado es 1 si los bits son diferentes y 0 si son iguales. Esto indica que la cantidad de 1’s permite determinar el grado de diferencia entre las ventanas, 0 indica

que las ventanas no tienen elementos diferentes y un valor $S > 0$ indica que las ventanas se diferencian por S bits entre sí.

procedure MAPA DE DISPARIDAD(I_{left},I_{rigth},W_h,d)
  Input: I_{left}: Transformada census de la Imagen izquierda
    I_{rigth}: Transformada census de la Imagen derecha
    W_h: Tamaño de la ventana de correlacion de SHD
    d: cantidad de pixeles de disparidad

  Output: M_{L2R}: Mapa de disparidad Resultante imagen izquierda como referencia
    M_{R2L}: Mapa de disparidad Resultante imagen derecha como referencia

  $I_W \leftarrow$ ancho de la imagen I_i en pixeles
  $I_H \leftarrow$ alto de la imagen I_i en pixeles
  $W_d \leftarrow \frac{W_h - 1}{2}$ Parámetro para recorrido de la ventana
  $Costo_{L2R}[d] \leftarrow 0$ Vector para almacenar el calculo de similitud para las d disparidades
  $Costo_{R2L}[d] \leftarrow 0$ Vector para almacenar el calculo de similitud para las d disparidades

  for $i \leftarrow W_d + 1$, $I_H - W_d$ **do** ▷ Recorrido por las Filas de la imagen
    for $j \leftarrow W_d + d$, $I_W - W_d$ **do** ▷ Recorrido por las Columnas de la imagen
      for $k \leftarrow 1$, d **do** ▷ Recorrido por las d ventanas candidatas
        $Costo_{L2R}[k] \leftarrow SHD\left\{ I_{left}\left[Ventana[i,j]\right], I_{rigth}\left[Ventana[i,j-k]\right] \right\}$
        $Costo_{R2L}[k] \leftarrow SHD\left\{ I_{left}\left[Ventana[i,j-d+k]\right], I_{rigth}\left[Ventana[i,j-d]\right] \right\}$
      end for
      $M_{L2R} \leftarrow \arg\min_d \{Costo_{L2R}[d]\}$
      $M_{R2L} \leftarrow \arg\min_d \{Costo_{R2L}[d]\}$
    end for
  end for
  return M_{R2L}, M_{L2R}
end procedure

Figura 41. Algoritmo para la implementacion Software del calculo de la disparidad.

Para mejorar el resultado del mapa de disparidad existe un método denominado Comprobación de Consistencia Izquierda-Derecha (LRCC). Este método es ampliamente para la perfección de los resultados de cualquier método de correspondencia estéreo. Este método comprueba la correspondencia entre los mapas de disparidad obtenidos previamente. Si un pixel con coordenadas $[x_l, y_l]$ en la imagen izquierda tiene una disparidad d entonces el pixel en las coordenadas $[x_r - d, y_r]$ debe tener disparidad $-d$. Los pixeles para los que difieren las estimaciones de disparidad se rechazan. Este control es bastante eficaz y es particularmente bueno para eliminar coincidencias incorrectas debido a oclusiones.

```
procedure SHD(Ventana_1,Ventana_2)
    Input: Ventana_1: Ventana de Hamming de tamaño W_h
           Ventana_2: Ventana de Hamming de tamaño W_h

    Output: S_o: Similitud existente entre Ventana_1 y Ventana_2

    S_o ← 0
    for i ← 1 , W_h do                          ▷ Recorrido por las Columnas de las ventanas
        for j ← 1 , W_h do                      ▷ Recorrido por las Filas de las ventanas
            S_o ← S_o + ∑ Xor(Ventana_1[i,j], Ventana_2[i,j])   ▷ Calculo de la similitud
        end for
    end for
    return S_o
end procedure
```

Figura 42. Algoritmo para la implementacion Software de SHD.

```
procedure LRCC(M_L2R,M_R2L,T_h)
    Input: M_L2R: Mapa de disparidad de la imagen izquierda como referencia
           M_R2L: Mapa de disparidad de la imagen derecha como referencia
           T_h: umbral de verificación de consistencia

    Output: M_final: Resultado de la verificación de consistencia izquierda-derecha

    I_W ← ancho del mapa de disparidad en pixeles
    I_H ← alto del mapa de disparidad en pixeles

    for i ← 1 , I_W do                          ▷ Recorrido por las Columnas de la imagen
        for j ← 1 , I_H do                      ▷ Recorrido por las Filas de la imagen
            xl ← j
            xr ← xl − M_L2R[i, xl]
            xlp ← xr + M_R2L[i, xr]
            if |xl − xlp| < T_h then
                M_final[i,j] ← M_L2R[i,j]
            else
                M_final[i,j] ← 0
            end if
        end for
    end for
    return M_final
end procedure
```

Figura 43. Algoritmo para la implementacion Software de LRCC.

En la Figura 43 se presenta el algoritmo que describe la implementación en software del método de comprobación LRCC. El procedimiento se realiza sobre toda la imagen recorriendo los pixeles de izquierda a derecha y de arriba hacia abajo. Cada pixel del mapa de disparidad M_{L2R} se compara con el pixel d posiciones a la izquierda del mapa de disparidad M_{R2L}. Si el valor de la comparación es menor al valor de umbral T_h seleccionado, el pixel en M_{L2R} se considera válido, de lo contrario se pone a 0 indicando

que es un pixel incorrecto. En el anexo F se presenta el desarrollo de la correspondencia estéreo empleando el software Matlab®.

6.2.1 Arquitectura en hardware

Según (Dunn & Corke, 1997; Fife & Archibald, 2013; Miyajima & Maruyama, 2003) muchos de los sistemas de alto rendimiento para visión estéreo en tiempo real basados en el área (como el utilizado en este trabajo), emplean una arquitectura equivalente a la de la Figura 44. Esta permite la ejecución de los algoritmos mediante paralelismo temporal ya que es una arquitectura computacional de *Stream Processing*. Esto implica que, en cada ciclo de reloj, ingresa un pixel y a su vez se genera una respuesta.

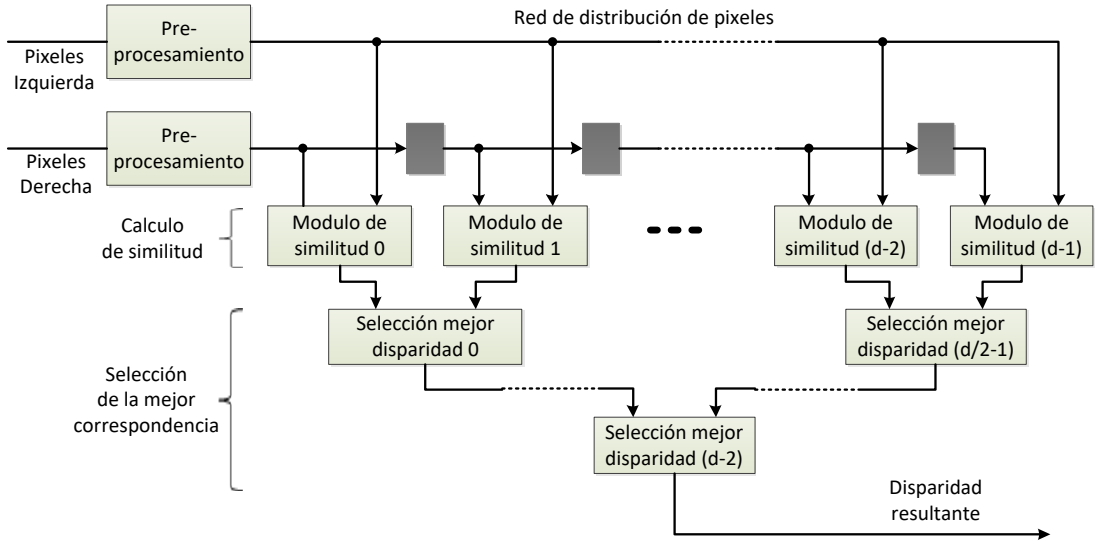


Figura 44. Aquitectura general de correlacion estéreo. Figura adaptada por el autor a partir de (Fife, 2011; Fife & Archibald, 2013).

Como se observa en la Figura 44, las imágenes que ingresan al módulo de correlación estéreo son pre-procesadas antes de calcular el mapa de disparidad. Este pre-procesamiento está basado comúnmente en filtros tales como LoG o Mediana, entre otros. En este trabajo, el método de pre-procesamiento empleado corresponde a la transformada Census dispersa. Una vez aplicado el método de pre-procesamiento a las imágenes estéreo, el flujo de pixeles de una imagen se retrasa con respecto a la otra de tal forma que, en cada ciclo de reloj, un pixel de la imagen original y cada pixel de la imagen desplazada ingresan a los respectivos módulos de similitud. Por lo tanto, cada módulo de similitud calcula el valor de semejanza para un par de pixeles a un nivel de disparidad específico. Es decir, el módulo de similitud 0 calcula la semejanza para todos los pares de pixeles con disparidad 0, el módulo de similitud 1 calcula la semejanza para todos los pixeles con disparidad 1, y así sucesivamente, para un total de d niveles de disparidad. En la mayoría de los casos los módulos de similitud se basan en la sumatoria sobre una ventana de pixeles. En este trabajo, el método de similitud empleado corresponde a la Suma de Distancias de Hamming (SHD). Finalmente, cada módulo de "selección-del-mejor" genera la disparidad que tiene la mejor puntuación de semejanza,

eligiendo entre la puntuación de su nivel de disparidad y la mejor de todos los niveles de disparidad anteriores. La salida del módulo selección-del-mejor $d-1$ corresponde a la salida de los valores de disparidad para cada pixel de entrada, obteniendo como resultado el mapa de disparidad (Dunn & Corke, 1997; Fife, 2011; Fife & Archibald, 2013).

Aunque esta arquitectura permite la ejecución paralela de la correspondencia estéreo, tiene una ruta crítica a través de los módulos de selección de disparidad, que limita demasiado la máxima frecuencia de reloj a la cual puede funcionar adecuadamente. La mencionada ruta crítica se compone de $\log_2(d)$ módulos de selección, lo que implica que para grandes valores de disparidad d se tiene un retardo de propagación que puede ser excesivo (Fife, 2011; Fife & Archibald, 2013).

En (Fife, 2011; Fife & Archibald, 2013) se propone una modificación a esta arquitectura, de manera que la ejecución se realiza mediante *Pipeline*. Esta arquitectura permite calcular d disparidades en cada ciclo de reloj, incrementando así su velocidad. Además, permite calcular los mapas de disparidad con las imágenes izquierda y derecha como referencia, ideales para la implementación del algoritmo LRCC (Miyajima & Maruyama, 2003). Sin embargo, los módulos de similitud deben ser diseñados internamente con *Pipeline* para maximizar el rendimiento del sistema. En la Figura 45 se presenta la arquitectura pipeline de correlación estéreo.

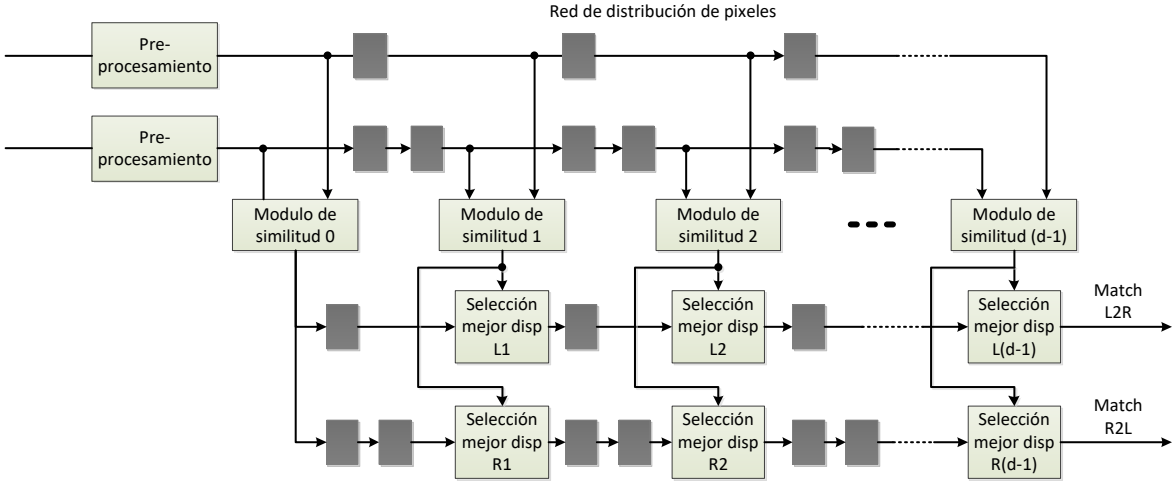


Figura 45. Aquitectura *Pipeline* de correlacion estéreo. Figura apadata por el autor a partir de (Fife, 2011; Fife & Archibald, 2013).

En la Figura 46 se presenta el diagrama esquemático del módulo de correspondencia estéreo y la interfaz de conexión de dicho módulo. Las señales *i_data_L* e *i_data_R* corresponden a los pixeles de las imágenes izquierda y derecha respectivamente. Estos datos son capturados por el modulo en cada flanco ascendente de la señal de reloj *i_clk*. La señal i_tresh_LRCC permite seleccionar el umbral para el cálculo de la comprobación de correspondencia izquierda-derecha (LRCC). La señal *o_data* corresponde al mapa de disparidad, el cual es generado pixel a pixel en cada flanco ascendente de la señal de

reloj *i_clk*. Las señales de control *i_rst_n, i_valid* y *o_valid* tienen un comportamiento similar que en el filtro de mediana. Además, el modulo posee varios parámetros que permiten seleccionar su comportamiento en el proceso de síntesis. El parámetro W_c se usa para seleccionar el tamaño de la ventana de Census, el cual debe ser impar mayor o igual a 3. El parámetro W_h permite seleccionar el tamaño de la ventana de Hamming, que debe ser impar mayor o igual a 3. El parámetro M establece el ancho de la imagen a procesar. El parámetro D selecciona la cantidad de niveles de disparidad que se quieren calcular, mientras el parámetro N establece el número de bits de los pixeles que ingresan al módulo de procesamiento.

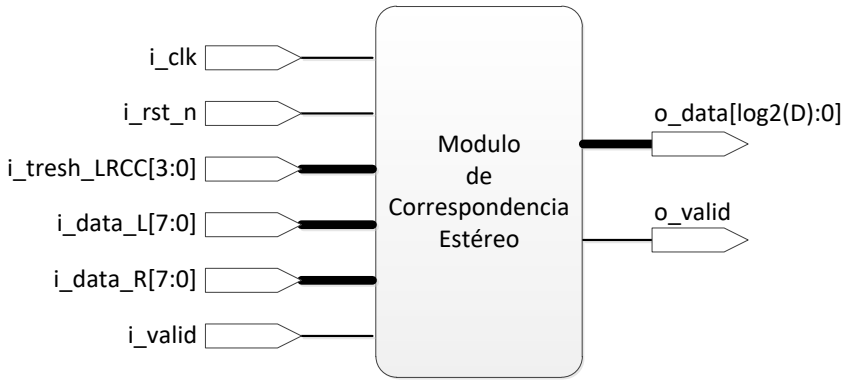


Figura 46. Diagrama esquemático del modulo de correspondencia estéreo. Fuente autor.

6.2.2 Módulo de similitud SHD

El módulo de similitud calcula la Suma de Distancias de Hamming (SHD) sobre una ventana de tamaño $W_h \times W_h$. De acuerdo con (Dunn & Corke, 1997) la implementación del módulo de similitud se debe realizar en dos etapas: en la primera se calcula la Distancia de Hamming comparando dos pixeles (pre-procesados previamente mediante Census), usando la operación lógica XOR y contabilizando el número de bits 1's resultante. En la segunda etapa, se suman los bits sobre una ventana de tamaño $W_h \times W_h$. El resultado corresponde a la medida de similitud de los pixeles de entrada. En la Figura 47 se presenta la estructura empleada en este trabajo para la implementación del módulo de similitud SHD.

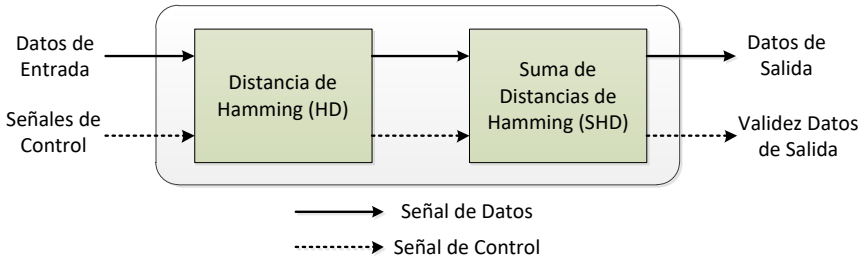


Figura 47. Modulo de similitud SHD. Flujo de las señales de datos y control. Fuente autor.

6.2.2.1 Calculo de la Distancia de Hamming

En la Figura 48 se presenta el hardware diseñado en este trabajo para el cálculo de la distancia de Hamming. Los pixeles que ingresan al módulo de similitud, provenientes de la transformación de Census, son comparados mediante la operación lógica XOR. Como el tamaño de cada pixel pre-procesado es de $n = \frac{W_c^2 - 1}{2}$ bits, se requieren n compuertas lógicas para realizar dicha comparación. Ver Figura 48 (a). El resultado del proceso de comparación se emplea para obtener el valor de la distancia de Hamming como se explicó en la sección 3.1.3. Para cumplir con este procedimiento se diseñó una configuración pipeline de sumadores en árbol como se aprecia en la Figura 48 (b). El diseño *Pipeline* permite aumentar el rendimiento de la arquitectura, en la cual la ruta crítica de la suma total se divide en etapas. Esto permite trabajar a velocidades de reloj mayores, únicamente limitada por la capa de sumadores más lenta. La cantidad de etapas pipeline depende principalmente de la cantidad de bits obtenidos en la transformación de Census. De esta manera, se requieren $\log_2(n)$ etapas de pipeline y $n-1$ sumadores. El valor máximo de la Distancia de Hamming está representado en $\log_2(n)$ bits.

Al igual que el módulo del filtro de mediana diseñado, el hardware para el cálculo de la distancia de Hamming entrega datos válidos después de los primeros $\log_2(n)$ ciclos de reloj. Para informar de este hecho a la etapa de procesamiento siguiente se diseña un circuito generador de validez, el cual tiene la misma arquitectura mostrada en la Figura 39. Para el caso del módulo de cálculo de la Distancia de Hamming el valor de comparación empleado en el generador de validez es $\log_2(n)$.

6.2.2.2 Suma de Distancias de Hamming (SHD)

Después de calcular la Distancia de Hamming de un par de pixeles que ingresan al módulo de similitud, el siguiente paso consiste en sumar las distancias de Hamming dentro de una ventana. Este procedimiento se puede llevar a cabo mediante un generador de ventana móvil, similar al empleado en el filtro de mediana, pero en este caso con la información de las distancias de Hamming. Una vez construida la ventana, el siguiente paso consiste en realizar el proceso de suma mediante una configuración *pipeline* de sumadores en árbol. Aunque dicha implementación es funcionalmente correcta, resulta sumamente ineficiente en el uso de recursos físicos de la FPGA, principalmente en cuanto a bits de memoria. En la ecuación (19) se presenta el cálculo necesario para conocer la cantidad total de bits requeridos por este tipo de implementación. En esta ecuación d corresponde a los niveles de disparidad a ser calculados, y por ende al número de módulos de similitud requeridos, M es el ancho de la imagen, W_h el tamaño de la ventana de Hamming y n el número de bits de la transformada Census. Este tipo de implementación se realizó de forma preliminar, la cual requería más de 200% de los bits de memoria disponibles en el dispositivo, por lo que resultaba totalmente inviable para su utilización en este trabajo.

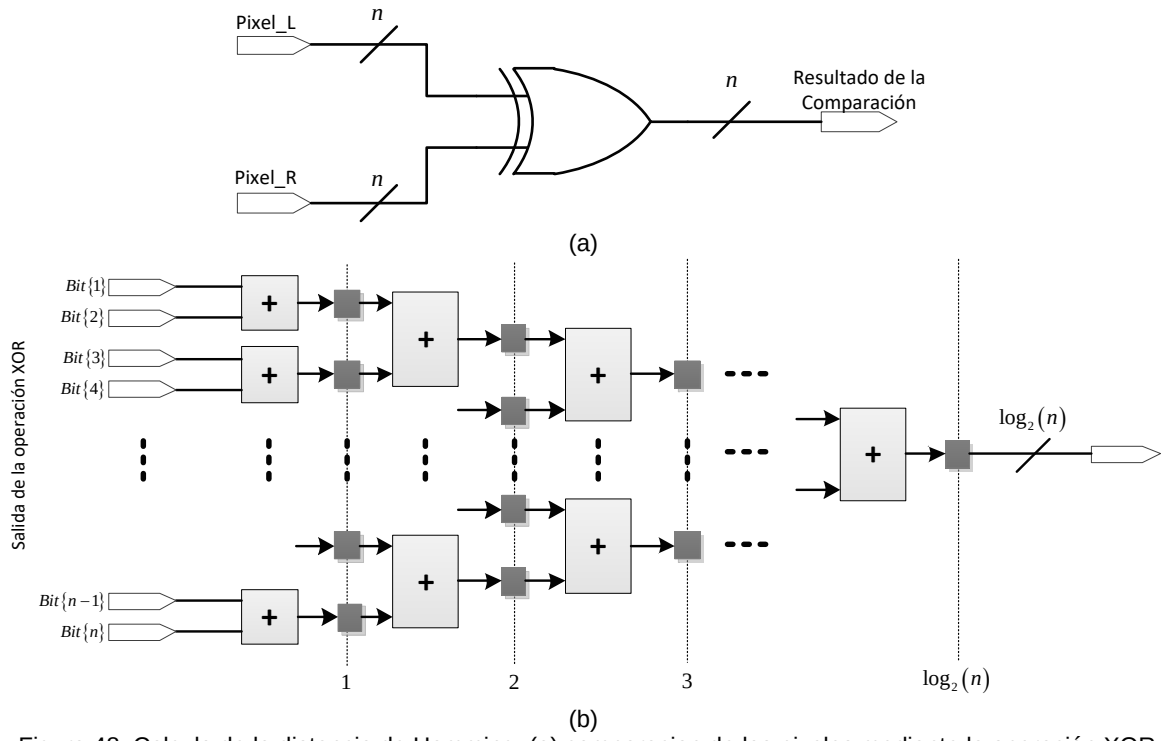


Figura 48. Calculo de la distancia de Hamming. (a) comparacion de los pixeles mediante la operación XOR. (b) Configuracion de sumadores en arbol para obtener la cantidad de 1's. Fuente autor.

$$(W_h - 1) \times (M - W_h) \times \log_2(n) \times d \tag{19}$$

Los trabajos desarrollados por (Dunn & Corke, 1997; Faugeras et al., 1993; Miyajima & Maruyama, 2003) presentan un método alternativo, al generador de ventana móvil, para el cálculo de la Suma de Distancias de Hamming. Este método emplea una técnica de optimización de suma de ventanas que consiste en usar pequeños buffers para calcular las sumas de las filas superior e inferior de la ventana de correlación de Hamming. Las sumas de filas se emplean posteriormente para calcular la suma total de la ventana. Este método reduce en gran medida el uso de memoria requerida por el módulo de similitud.

Aunque este método es muy bueno en el uso de bits de memoria dentro de la FPGA, en (Fife, 2011; Fife & Archibald, 2013) se presenta una modificación que permite disminuir la memoria requerida hasta en un 30%. Esta modificación consiste en calcular primero la suma de las columnas, dentro de la ventana, y luego emplear esas sumas para calcular la suma total de la ventana. En la Figura 49 se presenta este método de forma gráfica con el fin de ilustrar de forma más clara su funcionamiento. Este método requiere, para la suma de las columnas, únicamente de un búfer de memoria de longitud M, donde cada posición del búfer requiere un máximo de $\log_2(n \times W_h)$ bits. Para el cálculo de la suma total de la ventana se requiere de un búfer de W_h posiciones, equivalentes al ancho de la ventana de Hamming. En la ecuación (20) se presenta el cálculo para la máxima cantidad de bits de memoria requeridos por este tipo de implementación.

$$(W_h + M) \times \log_2(n \times W_h) \times d \quad (20)$$

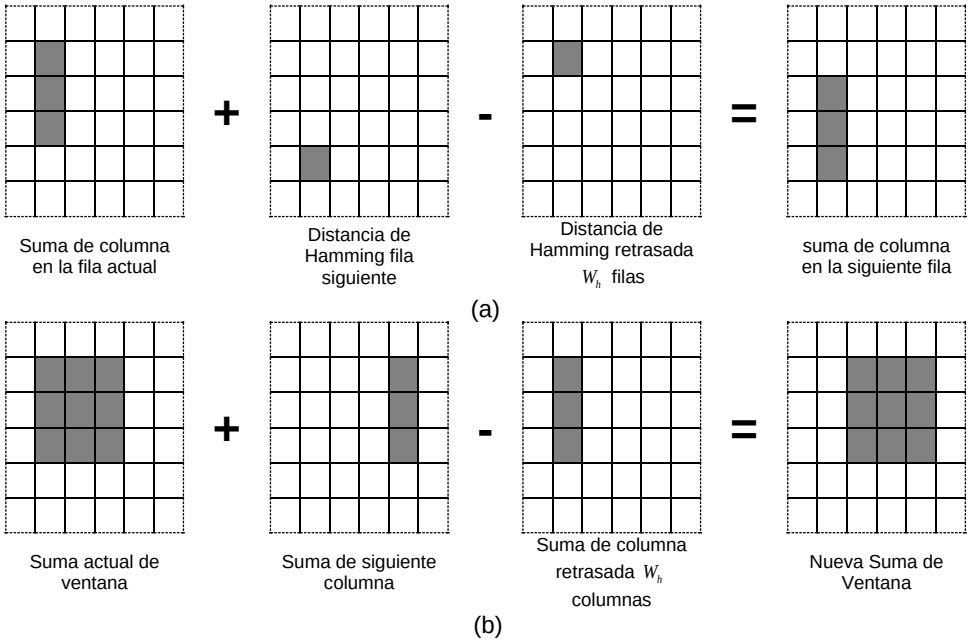


Figura 49. Calculo de la distancia de Hamming. (a) procedimiento para sumar las columnas de la ventana. (b) procedimiento para calcular la suma total de la ventana a partir de la suma de columnas. Figura adaptada por el autor a partir de (Fife, 2011)

En la Figura 50 se presenta una comparación del uso de bits de memoria requerido por la implementación basada en el generador de ventana móvil y la implementación basada en la optimización de suma de ventanas. La figura presenta los bits de memoria requeridos para diferentes tamaños de ventana de Hamming W_h. La comparación se realizó para $d = 64$ niveles de disparidad, tamaño de la ventana de Census $W_c = 7$ y ancho de la imagen $M = 640$. Como se observa, la curva de color azul, que representa la implementación del generador de ventana móvil, se incrementa rápidamente a medida que la ventana de Hamming crece. Esto conlleva a requerir hasta 3 Mbit de memoria para una ventana de Hamming de 17X17. En cambio, la curva de color rojo, que representa la implementación del método de optimización de suma de ventana, se incrementa levemente con un requerimiento de memoria que no supera los 400 Kbits. Por lo tanto, se puede concluir que la optimización de suma de ventana es extremadamente eficiente en el uso de bits de memoria haciéndola idónea para su uso en este trabajo.

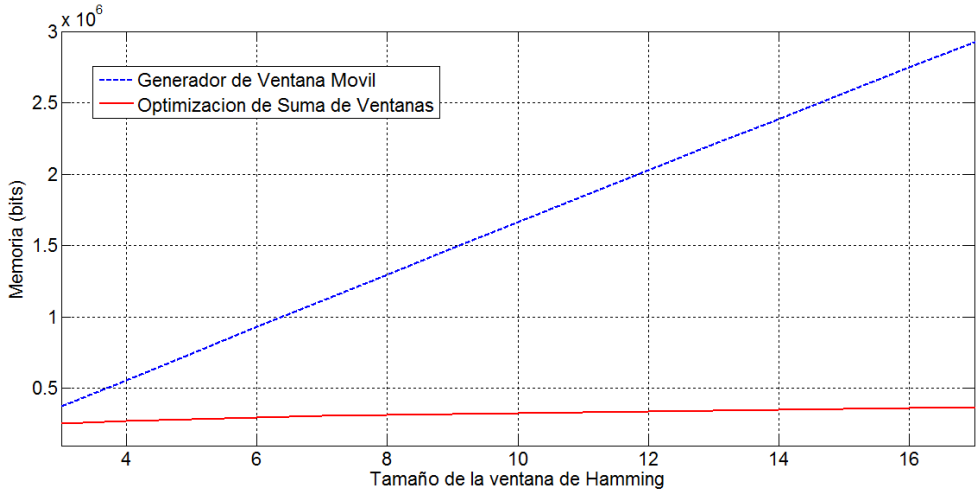


Figura 50. Requerimientos de memoria del modulo de similitud empleando generador de ventana movil (azul) y Optimización de Suma de Ventanas (rojo). Fuente…

Con el fin de hacer más eficiente el cálculo, la implementación en hardware de la SHD mediante el método de optimización de suma de ventana, requiere obtener el valor de la distancia de Hamming para los pixeles superior e inferior de la ventana de correlación de Hamming. Esto implica que la transformada Census entrega dos pixeles sobre la misma columna, pero separados verticalmente por W_h filas. Por consiguiente, al módulo de similitud no ingresan dos, sino cuatro pixeles, los cuales son comparados mediante dos módulos de cálculo de la Distancia de Hamming, uno para el pixel inferior $HD(x,y)$ y otro para el pixel superior $HD(x,y-W_h)$.

En la Figura 49 (a) se lleva a cabo la operación de resta entre las Distancias de Hamming de los pixeles inferior $HD(x,y)$ y superior $HD(x,y-W_h)$ de la ventana de Hamming para la x columna de la imagen. El resultado de esta operación es sumado con el valor anterior acumulado en esa misma columna. Para obtener la suma total sobre la ventana de Hamming (Figura 49 (b)) se ejecuta la operación de resta entre el valor acumulado en la columna x y el valor acumulado en la columna $x-W_h$. El resultado de esta operación es acumulado para obtener el valor de la Suma de distancias de Hamming total sobre la ventana. En la Figura 51 se presenta la arquitectura respectiva desarrollada en este trabajo. Esta arquitectura cuenta con cuatro etapas de pipeline que permiten aumentar el desempeño del módulo. El bloque denominado *Row Buffer* es una memoria FIFO de M posiciones en la que se almacenan las sumas acumuladas en cada columna de la imagen. El bloque denominado *Column buffer* es una memoria FIFO de W_h posiciones en la que se almacenan las sumas acumuladas de las ultimas W_h columnas necesarias para realizar el cálculo de la suma total sobre la ventana.

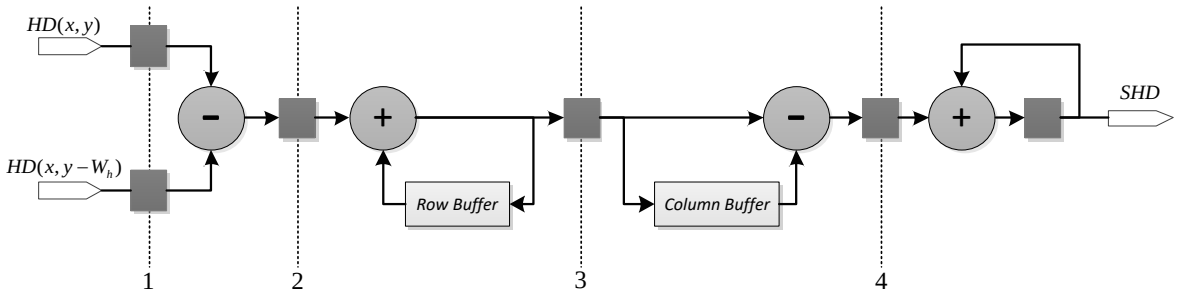


Figura 51. Calculo de la suma de distancias de Hamming mediante la optimizacion de suma de ventana. Adaptacion a partir de (Fife & Archibald, 2013)

Para obtener un valor valido de SHD se requiere construir un bloque que genera la señal de validez de los datos entregados por este módulo. La arquitectura del generador de validez es similar a la expuesta en la Figura 39, sin embargo, el valor de comparación empleado para el módulo de SHD es de $\frac{(W_h+1)\times(\mathrm{M}+1)}{2} - M + 3$.

6.2.3 Transformada Census Dispersa

La arquitectura desarrollada para el módulo de similitud requiere que se le ingresen los pixeles inferior y superior de la ventana de correlación de Hamming ($Pixel(x,y)$ y $Pixel(x, y-W_h)$ de ambas imágenes), con el fin de calcular la SHD de forma correcta. Según (Fife & Archibald, 2013) la forma más eficiente para obtener dichos pixeles consiste en calcularlos en el módulo de pre-procesamiento, en este caso en el módulo de la transformada Census. Teniendo en cuenta lo anterior, el módulo de la transformada Census fue desarrollado para generar dos datos sobre la misma columna de la imagen. El modulo está conformado por tres componentes principales: un generador de ventanas para obtener las ventanas requeridas en la formación de los dos datos de salida, la trasformación de Census que se aplica a cada una de las ventanas, y un generador de validez que indica si los datos generados son correctos. El flujo de los datos de entrada y las señales de control de un componente a otro se muestran en la Figura 52.

6.2.3.1 Generador de ventanas

Para obtener un solo dato de salida de la transformada Census se requiere tener como entrada una ventana de $W_c \times W_c$ pixeles. Para la construcción de esta ventana en hardware se necesita almacenar en cache $W_c - 1$ filas de la imagen. Esto se logra mediante el diseño de un generador de ventana móvil como el mostrado en la Figura 35. Sin embargo, el módulo de transformada Census debe generar dos datos los cuales deben pertenecer a la misma columna de la imagen pero con una diferencia entre sí de W_h filas. Para lograr esto, se debe diseñar un generador de dos ventanas con dicha separación, lo que implica almacenar en cache $W_c + W_h - 1$ filas de la imagen. En la Figura 53 se presenta la arquitectura del generador de ventanas mediante el método de caché

row buffering. En esta arquitectura, V_1 corresponde a la ventana para calcular la transformada de Census del pixel inferior y V_2 corresponde a la ventana para calcular la transformada Census del pixel superior, requeridas por el módulo de similitud.

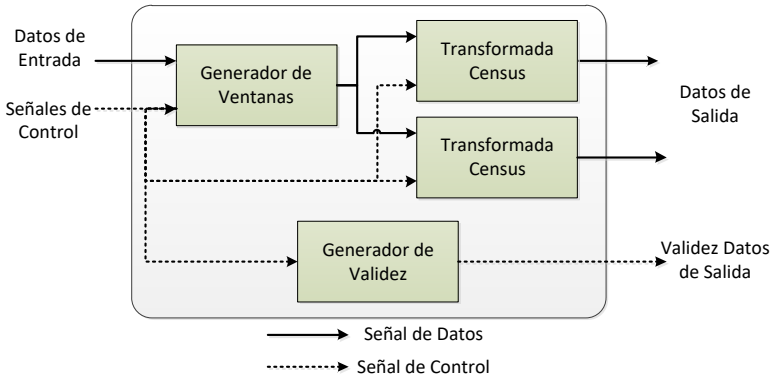


Figura 52.Modulo de la Transformada Census. Flujo de la señales de datos y de control. Fuente autor.

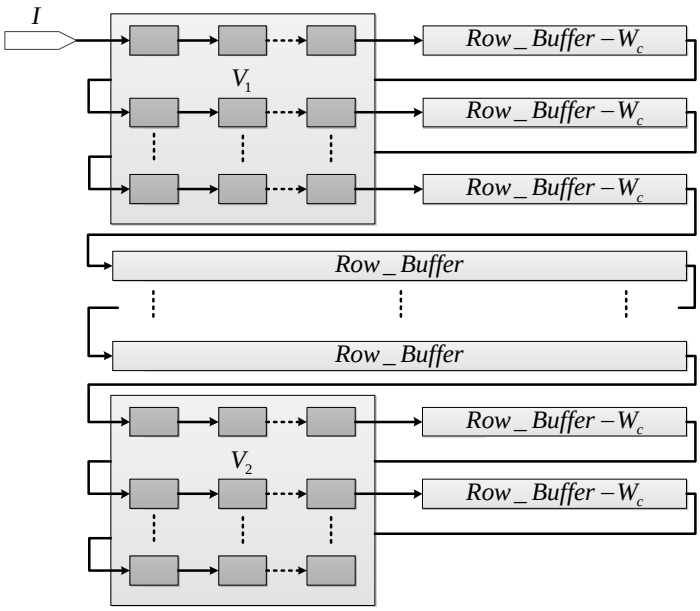


Figura 53.Modulo de la Transformada Census. Flujo de la señales de datos y de control. Fuente autor.

6.2.3.2 Transformación de Census Dispersa

La trasformación de Census dispersa consiste en comparar el pixel central de una ventana con los demás pixeles dentro de la ventana siguiendo un patrón disperso. En este trabajo se realizó la implementación de la transformada Census empleando un patrón de dispersión del 50% como se muestra en la Figura 7 (b). Los cuadros de color gris representan los pixeles dentro de la ventana que son comparados con el pixel central.

Una ventana de Census V_k dispersa al 50% posee tiene un total de $n=\frac{W_c^{\,2}-1}{2}$ pixeles a ser comparados con el pixel central C_v de dicha ventana. En la Figura 54 se presenta la arquitectura empleada para el cálculo de la transformada Census. En esta arquitectura se emplean n comparadores de N bits cada uno. Los comparadores evalúan si el pixel central C_v es mayor que un pixel P_i (perteneciente al patrón de dispersión utilizado). Si la comparación es verdadera el resultado es 1, de lo contrario el resultado es 0. La concatenación de los bits obtenidos en el proceso de comparación da como resultado una cadena de n bits correspondiente a la transformación de Census dispersa T_c.

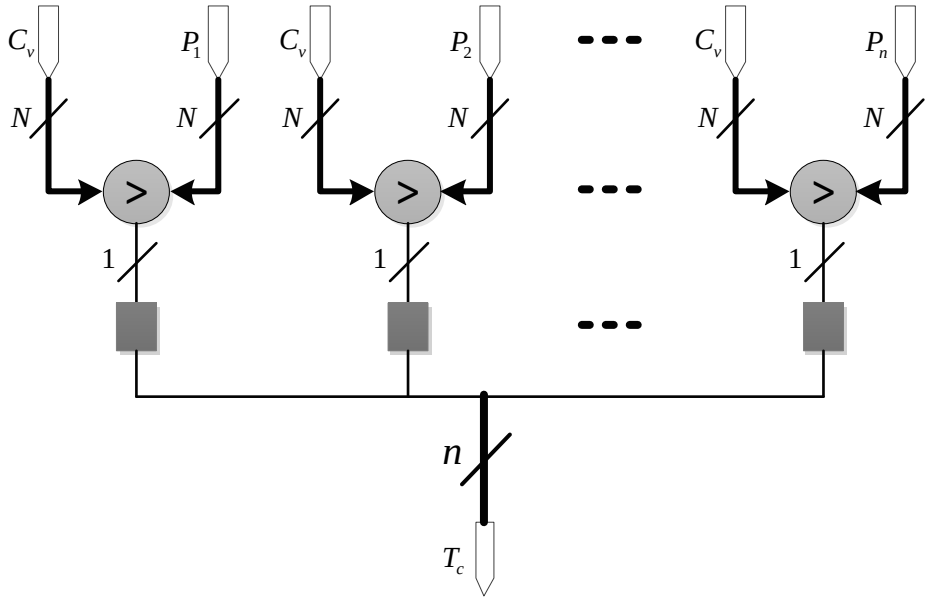


Figura 54.Calculo de la transformada de census dispersa. Fuente autor.

Para obtener un valor valido de la transformada de Census dispersa se requiere un componente que produzca la señal de validez con una arquitectura similar a la presentada en la Figura 39. Sin embargo, el valor de comparación empleado para el módulo de la transformada Census es de $\frac{(W_c+1)\times(\mathrm{M}+1)}{2}-M$.

6.2.4 Calculo de la disparidad

En la arquitectura mostrada en la Figura 45, la selección del valor de disparidad se realiza mediante un arreglo de módulos interconectados que evalúan los valores de similitud obtenidos para los d niveles de disparidad contemplados. Estos módulos buscan el mejor valor de similitud mediante la comparación de las similitudes calculadas. El nivel de disparidad cuya similitud es mejor que las demás corresponde al valor de disparidad del pixel analizado. El criterio de selección del mejor valor de similitud empleado en este trabajo es el mínimo, por lo cual, el arreglo de módulos desarrollados busca el nivel de disparidad con el menor valor de similitud.

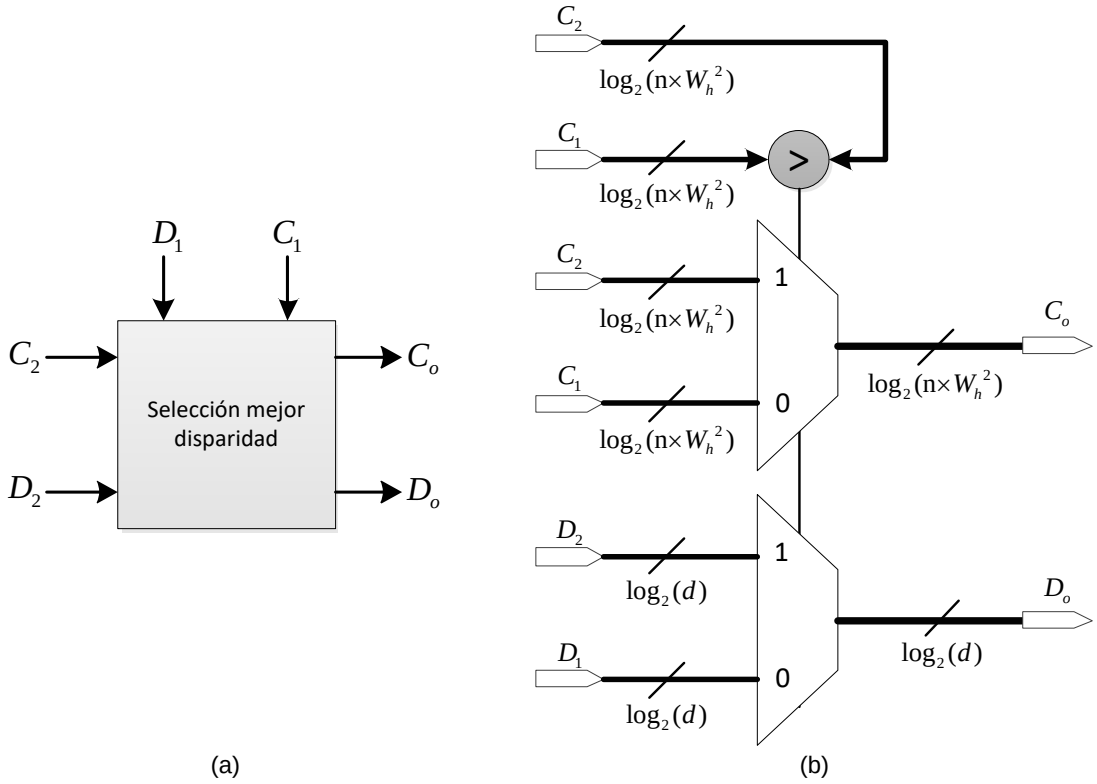


Figura 55.Modulo de selección del mejor valor de disparidad. (a) diagrama esquematico del módulo. (b) arquitectura interna del módulo. Fuente autor.

En la Figura 55 (a) se presenta el diagrama esquemático del módulo de selección del mejor valor de disparidad. Las entradas C_1 y D_1 del módulo corresponden al valor de similitud C_1 del nivel de disparidad D_1, y las entradas C_2 y D_2 del módulo corresponden al valor de similitud C_2 del nivel de disparidad D_2. El módulo realiza la comparación de las similitudes C_1 y C_2, y genera las salidas C_o y D_o. En la Figura 55 (b) se presenta la arquitectura del módulo de selección desarrollado en este trabajo. El módulo está compuesto por un comparador de magnitud y dos multiplexores. El comparador evalúa si el valor de similitud C_1 es mayor que el valor de similitud C_2. Si la comparación es verdadera la salida C_o toma el valor C_2 y la salida D_o toma el valor de D_2. Si el resultado de la comparación es falso C_o toma el valor C_1 y la salida D_o toma el valor de D_1. La cantidad de bits requeridos por el modulo está determinada por los niveles de disparidad d, el tamaño de la ventana de Census W_c y el tamaño de la venta de Hamming W_h.

6.2.5 Comprobación de consistencia Izquierda-derecha (LRCC)

La comprobación de consistencia izquierda-derecha (LRCC) es un método ampliamente utilizado para el refinamiento del mapa de disparidad. Este método requiere el cálculo

previo de los mapas de disparidad tomando como referencia las imágenes izquierda (L2R) y derecha (R2L) del par estéreo. La arquitectura de hardware mostrada en la Figura 45, permite hacer el cálculo de los dos mapas de disparidad requeridos por este método. Sin embargo, existe un retraso de d pixeles del mapa de disparidad R2L con respecto al mapa de disparidad L2R. Este desfase se corrige, antes de calcular la LRCC, implementando d registros en cascada con el mapa de disparidad L2R.

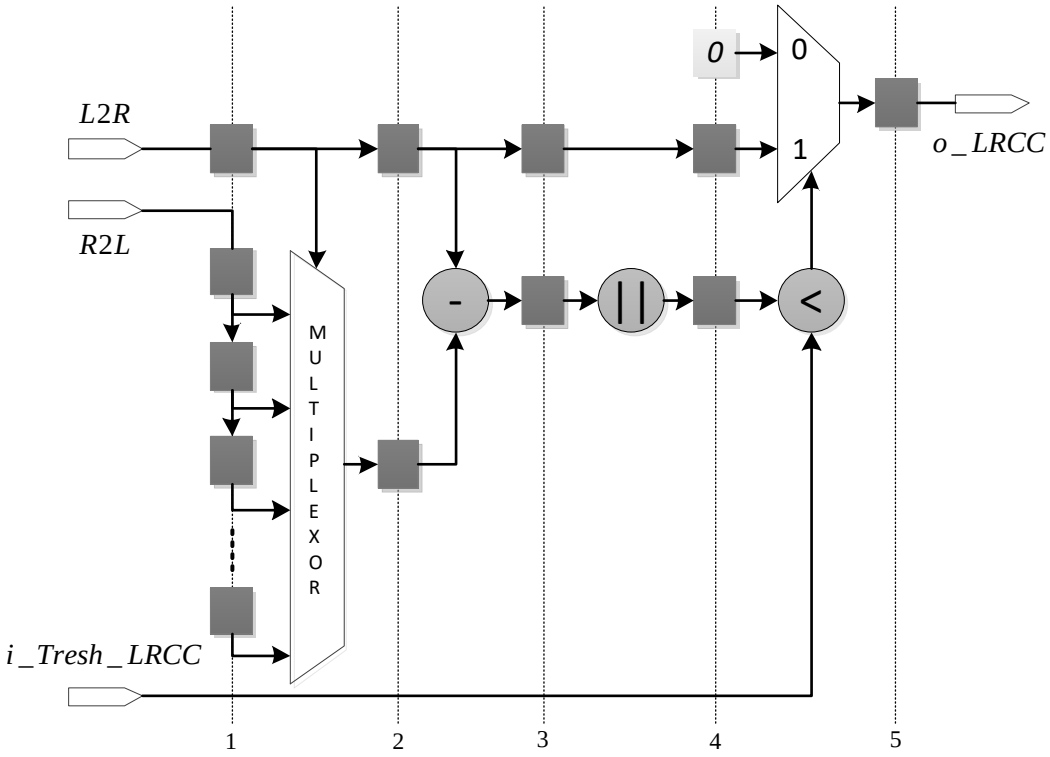


Figura 56.Calculo de la comprobacion de consistencia izquierda-derecha LRCC. Fuente autor.

El diseño en FPGA del módulo LRCC para el refinamiento del mapa de disparidad se realiza teniendo en cuenta la definición que se presenta en la ecuación (21). De acuerdo con esta ecuación, el valor absoluto de la diferencia entre la disparidad para un pixel en el mapa L2R y la disparidad de un pixel del mapa R2L en función de la disparidad del pixel en el mapa L2R, debe ser menor a un valor de umbral T_h. Idealmente, T_h debe ser igual a 1, indicando que la disparidad de un pixel del mapa L2R y su equivalente en el mapa R2L son iguales. Si la desigualdad se cumple significa que los pixeles comparados en ambos mapas de disparidad son consistentes. De lo contrario se considera que el valor de disparidad para el pixel comparado no corresponde a un valor de disparidad permitido.

$$\left| d_{L2R}(x) - d_{R2L}(d_{L2R}(x)) \right| < T_h \tag{21}$$

En la Figura 56 se presenta el hardware desarrollado para el cálculo de la LRCC. Los pixeles del mapa R2L ingresan a un arreglo de registros conectados en cascada. Estos

registros junto con un multiplexor permiten seleccionar el valor de disparidad del mapa R2L en función de la disparidad de un pixel del mapa L2R. Dado que el valor máximo de disparidad calculado está determinado por d niveles de disparidad, en total se requieren d registros para éste propósito. El módulo de hardware calcula el valor absoluto de la diferencia entre el valor de disparidad a la salida del multiplexor y el valor de disparidad del mapa L2R y compara el resultado de la operación con el valor de umbral seleccionado. En caso de obtener un resultado favorable, el valor de disparidad del mapa L2R se envía al exterior, de lo contrario en su lugar se envía un 0 indicando que dicho valor no es consistente en ambos mapas de disparidad.

El módulo de hardware fue diseñado para ejecutar el refinamiento del mapa de disparidad mediante 5 etapas de *pipeline* con el fin de mantener un diseño robusto para todo el módulo de correspondencia estéreo. Debido al *Pipeline*, el mapa de disparidad tiene un retraso inicial de cinco pixeles válidos, por lo que es necesario implementar un componente para la generación de la señal de validez de salida. Dicho generador de validez tiene la arquitectura mostrada en la Figura 39 y emplea un valor de comparación de 4 para el caso particular del módulo LRCC.

6.3 Segmentación

La segmentación es un método de procesamiento de imágenes empleado en la extracción regiones de interés. El proceso de segmentación etiqueta los pixeles de la región que se desea extraer del fondo de la imagen. En este trabajo se emplean dos métodos de segmentación para extraer la región de la forma de la mano: segmentación por color de piel y segmentación del mapa de disparidad.

La segmentación por color de piel permite asignar un valor de 1 a los pixeles que poseen un color similar al de la piel humana, y 0 para los demás pixeles presentes en la escena (ver anexo G). La segmentación del mapa de disparidad permite extraer lar regiones u objetos que se encuentran en un rango de niveles de disparidad. Los pixeles correspondientes a las regiones extraídas del mapa de disparidad se etiquetan con un 1 y los pixeles ignorados se etiquetan con 0. La selección del rango de niveles de disparidad se realiza en este caso teniendo en cuenta la zona de profundidad donde se ejecuta el movimiento de la mano durante la realización de un gesto. La combinación de estos dos métodos de segmentación permite extraer de manera más efectiva la región de la forma de la mano, que será analizada posteriormente para caracterizar el gesto realizado.El proceso de segmentación fue desarrollado inicialmente en software mediante el uso de MATLAB®.

En la Figura 57 se presenta el algoritmo que describe la implementación en software de la segmentación por color de piel para imágenes en el modelo de color YCbCr. El proceso de segmentación se lleva a cabo a nivel de pixel mediante un recorrido de la imagen de arriba hacia abajo y de izquierda a derecha. Para cada pixel se evalúan las componentes

de crominancia de acuerdo con los valores de umbral seleccionados. Ver ecuación (22). Si los valores de crominancia se encuentran dentro de los rangos establecidos el pixel es etiquetado con un valor de 1, de lo contrario se etiqueta con un valor de 0.

```
1: procedure SKIN SEGMENTATION(IM_YCbCr)
2:     Input: IM_YCbCr: Imagen en el modelo de color YCbCr
3:
4:     Output: IM_segm: Segmentacion por color de piel
5:
6:     I_W ← ancho de la imagen en pixeles
7:     I_H ← alto de la imagen en pixeles
8:
9:     for i ← 1 , I_W do                    ▷ Recorrido por las Columnas de la imagen
10:        for j ← 1 , I_H do                 ▷ Recorrido por las Filas de la imagen
11:            if 77 < C_b[i, j] < 127 && 133 < C_r[i, j] < 173 then
12:                IM_segm[i, j] ← 1
13:            else
14:                IM_segm[i, j] ← 0
15:            end if
16:        end for
17:    end for
18:    return IM_segm
19: end procedure
```

Figura 57.Algoritmo para la implementacion en software de la segmentacion por color de piel. Fuente …

$$P_{skin} = \begin{cases} 1 & \text{if } 77 \le C_b \le 127 \quad \& \; 133 \le C_r \le 173 \\ 0 & Other\ Case \end{cases} \tag{22}$$

La Figura 58 presenta el algoritmo que describe la implementación en software de la segmentación del mapa de disparidad. El proceso de segmentación se realiza a nivel de pixel recorriendo la imagen de arriba hacia abajo y de izquierda a derecha. Para cada pixel del mapa de disparidad se evalúa la disparidad de acuerdo con los umbrales seleccionados $T_L < d[i,j] < T_H$. Estos umbrales se seleccionan experimentalmente teniendo en cuenta el espacio requerido por la persona al realizar un gesto. Si el valor de disparidad $d[i,j]$ se encuentra dentro de los limites el pixel se etiqueta con un 1, de lo contrario se etiqueta con 0.

```
1: procedure DISPARITY SEGMENTATION(MD)
2:     Input: MD: mapa de disparidad con d niveles de disparidad
3:
4:     Output: MD_segm: Segmentación del mapa de disparidad resultante
5:
6:     I_W ← ancho de la imagen en pixeles
7:     I_H ← alto de la imagen en pixeles
8:     T_H ← umbral superior de comparación
9:     I_L ← umbral inferior de comparación
10:
11:    for i ← 1 , I_W do                    ▷ Recorrido por las Columnas de la imagen
12:        for j ← 1 , I_H do                ▷ Recorrido por las Filas de la imagen
13:            if T_L < d[i, j] < T_L then
14:                MD_segm[i, j] ← 1
15:            else
16:                MD_segm[i, j] ← 0
17:            end if
18:        end for
19:    end for
20:    return MD_segm
21: end procedure
```

Figura 58.Algoritmo para la implementacion en software de la segmentacion del mapa de disparidad.

Finalmente, se intersectan las dos segmentaciones resultantes obteniendo la extracción de la región que corresponde a la forma de la mano del gesto realizado. En el anexo F se presenta el desarrollo del proceso de segmentación empleando el software Matlab®.

6.3.1 Arquitectura en hardware

El diseño del proceso de segmentación en hardware se lleva a cabo mediante una arquitectura computaciónal *Stream Porcessing*. El módulo de hardware de la segmentación está compuesto por tres componentes principales: la segmentación de color de piel, la segmentación del mapa de disparidad y la intersección de las segmentaciones. El flujo de los datos de entrada y las señales de control de un bloque funcional a otro se muestra en la Figura 59.

En la Figura 60 se presenta el diagrama esquemático del módulo de segmentación diseñado. En este diagrama se muestra la interfaz de conexión del módulo. Las señales *i_Cb* y *i_Cr* corresponden a las componentes de crominancia de la imagen de entrada en el modelo de color YCbCr. La señal *i_L2R* corresponde a la entrada de los pixeles del mapa de disparidad. Estos datos son capturados por el modulo en cada flanco ascendente de la señal de reloj *i_clk*. La señal *o_data* corresponde al resultado de la segmentación y es generada pixel a pixel en cada flanco ascendente de la señal de reloj *i_clk*. Las señales de control *i_rst_n, i_valid* y *o_valid* tienen un comportamiento similar a los modulos de filtro de mediana y correspondencia estéreo. Además, el modulo posee varios parámetros que permiten seleccionar su comportamiento en el proceso de síntesis. El parámetro D selecciona la cantidad de niveles de disparidad disponibles en el mapa

de disparidad. El parámetro N permite seleccionar el número de bits en que están representadas las componentes de crominancia.

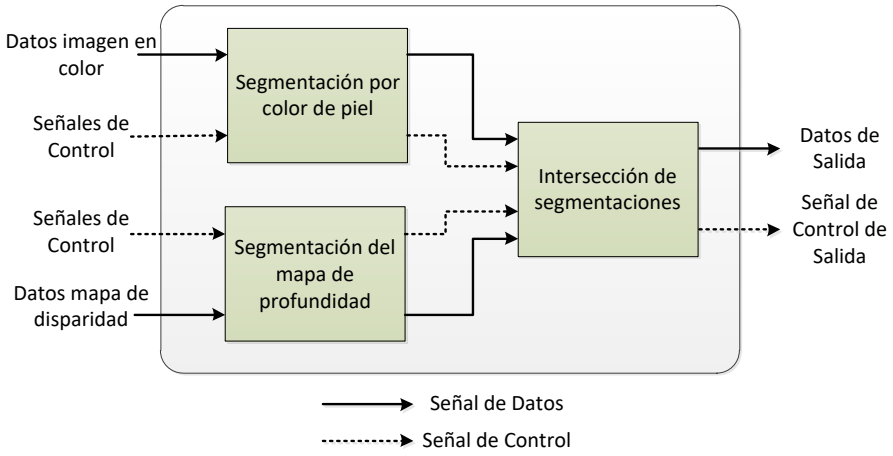


Figura 59.Modulo de Segmentacion. Flujo de las señales de datos y control. Fuente autor.

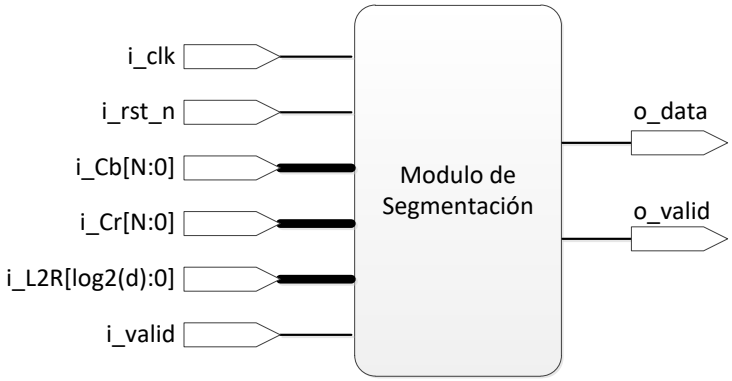


Figura 60.diagrama esquematico del modulo de segmentacion en hardware. Fuente autor.

A continuación se presenta una descripción detallada de los componentes internos que conforman la arquitectura del módulo de segmentación.

6.3.2 Segmentación del color de piel

El diseño en Hardware de la segmentación por color de piel se realizó mediante la arquitectura mostrada en la Figura 61. En esta se observa la comparación de cada componente de color con los valores de umbral establecidos en la ecuación (22). Para este módulo se requieren 4 comparadores de N bits cada uno. Cada comparador da como resultado un 1 si la comparación es verdadera y un 0 si el resultado de la comparación es falsa. Si los valores de crominancia de un pixel de entrada están dentro de los valores de comparación establecidos, el módulo genera un 1 en la salida, de lo contrario genera un 0. Cuando la salida es igual a 1 indica que el pixel pertenece a una región de color de piel.

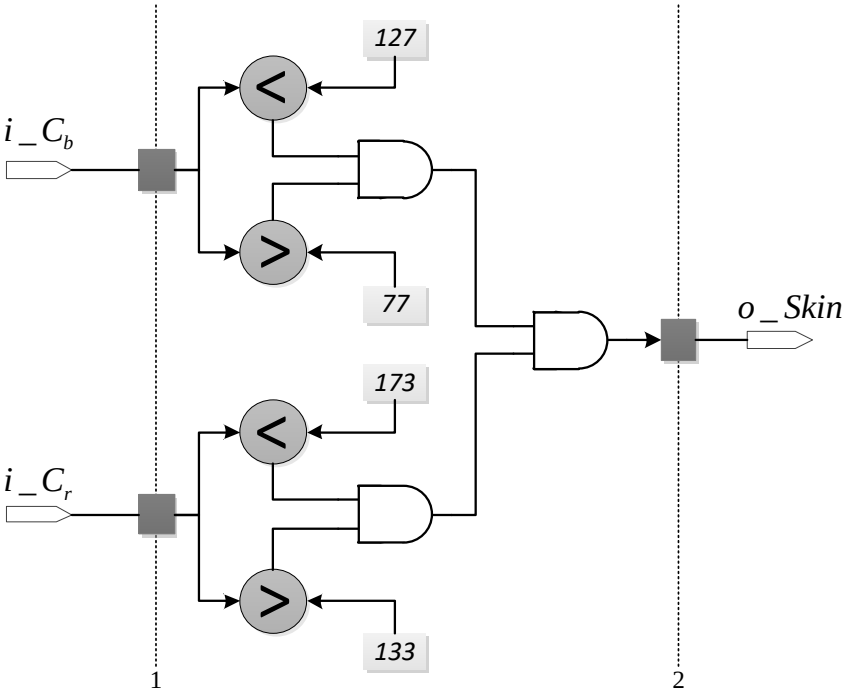


Figura 61.Calculo de la segmentacion por color de piel. Fuente autor.

Dado que el modulo diseñado para el cálculo de la segmentación por color de piel posee registros a la entrada y a la salida, se necesitan dos ciclos de reloj válidos para obtener un dato de segmentación valido a la salida. Por tal razón se requiere un componente adicional que indique si el pixel generado es valido o no. Este componente se diseña con base en la arquitectura mostrada en la Figura 39.

6.3.3 Segmentación del mapa de disparidad

El diseño en Hardware de la segmentación del mapa de disparidad se realizó mediante la arquitectura mostrada en la Figura 62. En esta figura se observa la comparación del valor de disparidad de entrada teniendo en cuenta los valores de umbral T_L y T_H seleccionados. Este módulo posee una arquitectura similar a la empleada en la segmentación por color de piel. En este caso se requieren únicamente 2 comparadores de N bits cada uno. Cada comparador da como resultado un 1 si la comparación es verdadera, y un 0 si el resultado de la comparación es falsa. Si el valor de disparidad de un pixel de entrada está dentro de los valores de comparación establecidos, el módulo genera un 1 en la salida, de lo contrario genera un 0. Cuando la salida es igual a 1 indica que el pixel pertenece a una zona espacial definida para la realización del gesto. La selección de los valores de umbral se realizó experimentalmente, obteniendo los mejores resultados para $d = 48$ niveles de disparidad y $T_L = 33$ y $T_H = d - 2$.

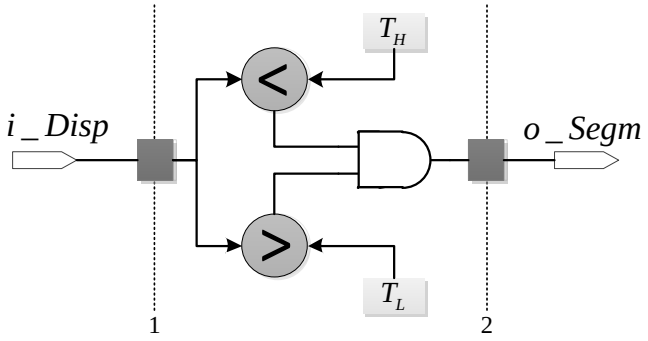


Figura 62.Calculo de la segmentacion del mapa de disparidad. Fuente autor.

Dado que el modulo diseñado para el cálculo de la segmentación del mapa de disparidad posee registros a la entrada y a la salida se necesitan dos ciclos de reloj válidos para obtener un dato de segmentación valido a la salida. Por lo tanto, se requiere un componente adicional que indique si el pixel generado es valido o no. Este componente es similar al requerido por el módulo de segmentación por color de piel.

Finalmente, las segmentaciones se intersectan mediante una operación lógica AND para obtener el resultado final de la segmentación teniendo en cuenta que se trata de imágenes binarias.

El proceso de intersección requiere que la segmentación por color de piel y la segmentación del mapa de disparidad estén totalmente sincronizadas. En este caso, la segmentación del mapa de disparidad está desfasada con respecto a la segmentación por color de piel debido a la latencia existente en el módulo de cálculo de la correspondencia estéreo. Para sincronizar las dos segmentaciones basta con retrasar la segmentación por color de piel la cantidad de pixeles de latencia del mapa de disparidad como se observa en la Figura 30. La imagen segmentada es una imagen binaria, la cual puede contener pixeles no deseados que pueden ser eliminados mediante un filtrado morfológico.

6.4 Operación Morfológica de Apertura

La operación morfológica de apertura es un filtro ampliamente utilizado en imágenes binarias, especialmente imágenes segmentadas. Esta operación morfológica permite eliminar los pixeles no deseados obtenidos como resultado de un proceso de segmentación. En este trabajo se emplea este tipo de filtrado con el fin de eliminar pixeles aislados que no pertenecen a la forma de la mano extraída en el proceso de segmentación explicado en la sección anterior.

La operación morfológica de apertura consiste en aplicar de forma consecutiva una operación morfológica de erosión seguida de una operación morfológica de dilatación empleando el mismo elemento estructurante (ver anexo G). La operación morfológica de apertura se desarrolló inicialmente en software mediante el uso de MATLAB®. Esta implementación permite comprender la estructura y funcionamiento de las operaciones

morfológicas Erosión y Dilatación, para el desarrollo posterior de su arquitectura en FPGA. Además, la implementación en software se considera como modelo de referencia para la verificación de los resultados obtenidos en hardware.

En la Figura 63 se presenta el algoritmo que describe la implementación en software de la operación morfológica de erosión sobre una imagen binaria. El proceso de erosión se lleva a cabo recorriendo los pixeles de la imagen de arriba hacia abajo y de izquierda a derecha. Los elementos dentro de una ventana V centrada en un pixel con coordenadas $[i, j]$ son intersectados con un elemento estructurante EE. Si el resultado de la intersección es nuevamente el elemento estructurante, el pixel se pone a 1 permaneciendo en la imagen como parte del objeto. Si por el contrario es diferente, entonces el pixel es puesto a 0 erosionando la región del objeto al que pertenece dicho pixel.

```
1:  procedure EROSION(I_bin, W)
2:      Input: I_bin: imagen binaria
3:
4:      Output: O_bin: imagen erosionada
5:
6:      I_W ← ancho de la imagen en pixeles
7:      I_H ← alto de la imagen en pixeles
8:      EE[W, W] ← Elemento estructurante de tamaño W × W
9:
10:     for i ← W²/2 , I_W − W²/2 do          ▷ Recorrido por las Columnas de la imagen
11:         for j ← W²/2 , I_H − W²/2 do      ▷ Recorrido por las Filas de la imagen
12:             if Ventana[i, j] ∧ EE = EE then
13:                 O_bin[i, j] ← 1
14:             else
15:                 O_bin[i, j] ← 0
16:             end if
17:         end for
18:     end for
19:     return O_bin
20: end procedure
```

Figura 63.Algoritmo para la implementacion en software de la operación morfológica de Erosión.

En la Figura 64 se presenta el algoritmo que describe la implementación en software de la operación morfológica de dilatación sobre una imagen binaria. El proceso de dilatación se lleva a cabo recorriendo los pixeles de la imagen de arriba hacia abajo y de izquierda a derecha. Los elementos dentro de una ventana V centrada en un pixel con coordenadas $[i, j]$ son comparados con un elemento estructurante EE. Si existe un elemento de la ventana, que pertenezca a la región del objeto y que coincida con al menos una parte del patrón del elemento estructurante $\neg \text{Ventana}[i, j] \wedge \text{EE} \neq \text{EE}$, el pixel central de la ventana es puesto a 1 dilatando la región del objeto con dicho pixel. Si por el contrario no hay coincidencia, entonces la región del objeto no es dilatada con dicho pixel.

```
1: procedure DILATACION(I_bin,W)
2:     Input: I_bin: imagen binaria
3:
4:     Output: O_bin: imagen dilatada
5:
6:     I_W ← ancho de la imagen en pixeles
7:     I_H ← alto de la imagen en pixeles
8:     EE[W,W] ← Elemento estructurante de tamaño W × W
9:
10:    for i ← W²/2 , I_W − W²/2 do          ▷ Recorrido por las Columnas de la imagen
11:        for j ← W²/2 , I_H − W²/2 do       ▷ Recorrido por las Filas de la imagen
12:            if ¬Ventana[i,j] ∧ EE ≠ EE then
13:                O_bin[i,j] ← 1
14:            else
15:                O_bin[i,j] ← 0
16:            end if
17:        end for
18:    end for
19:    return O_bin
20: end procedure
```

Figura 64.Algoritmo para la implementacion en software de la operación morfológica de Dilatación.

Para realizar la operación morfológica de apertura basta con erosionar la imagen primero y aplicar luego la operación de dilatación. En el anexo F se presenta el desarrollo de las operaciones morfológicas erosión y dilatación empleando el software MATLAB®.

6.4.1 Arquitectura en hardware

El diseño del filtro morfológico de apertura en hardware se lleva a cabo mediante una arquitectura computacional *Stream Porcessing*. El módulo de hardware del filtro morfológico de apertura está compuesto por dos componentes principales. El filtro morfológico de erosión y el filtro morfológico de dilatación. El flujo de los datos de entrada y las señales de control de un bloque funcional a otro se muestra en la Figura 65.

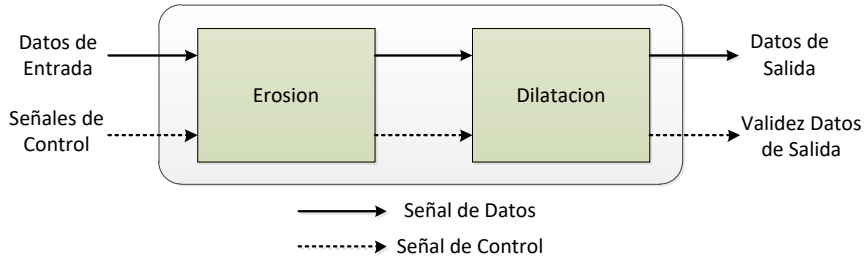


Figura 65. Modulo de la operación morfológica de apertura. Flujo de las señales de datos y control. Fuente autor.

En la Figura 66 se presenta el diagrama esquemático del módulo del filtro morfológico de apertura desarrollado. En este diagrama se presenta la interfaz de conexión del módulo.

La señal *i_dato* corresponde a la imagen binaria obtenida en el módulo de segmentación. Los datos son capturados por el modulo en cada flanco ascendente de la señal de reloj *i_clk*. La señal *o_dato* corresponde al resultado del filtro morfológico de apertura, el cual es generado pixel a pixel en cada flanco ascendente de la señal de reloj *i_clk*. Las señales de control *i_rst_n, i_valid* y *o_valid* tienen un comportamiento similar a los módulos de filtro de mediana, correspondencia estéreo y segmentación. Adicionalmente, el modulo posee varios parámetros que permiten seleccionar su comportamiento en el proceso de síntesis. El parámetro W se usa para seleccionar el tamaño de la ventana requerida por el filtro. El parámetro M establece el ancho de la imagen a procesar.

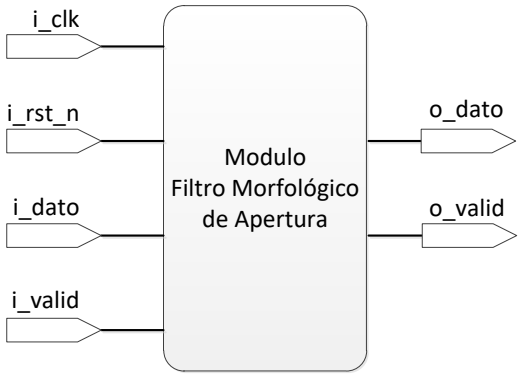


Figura 66.Diagrama esquemático del módulo de la operación morfológica de apertura en hardware. Fuente autor.

6.4.2 Erosión y dilatación

El diseño en hardware de los filtros morfológicos de erosión y dilatación se realiza mediante tres componentes: generador de ventana móvil, filtro morfológico y generador de validez. En la Figura 67 se presenta un esquema general del diseño de los filtros morfológicos empleados en este trabajo. Según (Bailey, 2011, p. 264) la erosión y la dilatación son operaciones morfológicas duales, ya que la dilatación de la imagen es equivalente a la erosión del fondo y viceversa. Teniendo en cuenta esta característica, es posible diseñar un módulo de hardware que permita implementar cualquiera de las dos operaciones morfológicas.

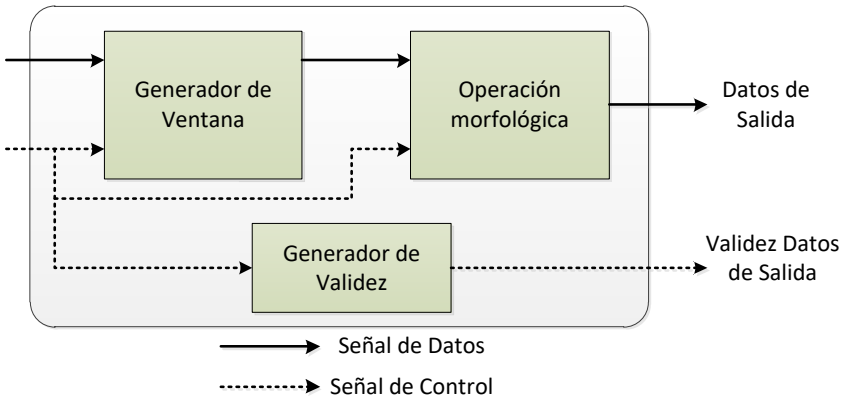


Figura 67.Diagrama general de la implementacion en hardware de un filtro morfológico. Fuente autor.

Dado que los filtros morfológicos son filtros locales, se requiere de una ventana de pixeles para realizar el procesamiento respectivo. Para tal fin se diseña un generador de ventana móvil que almacena en caché las ultimas $W-1$ ventanas, sin la necesidad de disponer de la imagen completa. De esta manera se puede acceder a todos los pixeles de la ventana en paralelo. El generador de ventana se diseña con base en la arquitectura mostrada en la Figura 35. Con el fin de diseñar un módulo con la capacidad de aplicar a una imagen el filtro de erosión o dilatación, la arquitectura de ventana es modificada agregando una compuerta XOR ente los pixeles que ingresan al generador y una línea de selección de operación. En la Figura 68 se presenta la estructura del generador de ventana diseñado para el filtrado morfológico.

En diseño del filtro morfológico se realiza teniendo en cuenta dos elementos importantes. Una ventana V y un elemento estructurante EE. El elemento estructurante contiene el patrón empleado por el filtro para realizar la operación requerida. Tanto el elemento estructurante como la ventana deben tener el mismo tamaño, el cual está determinado por el parámetro de síntesis W. La cantidad de elementos de la ventana es $k=W^2$, siendo la misma cantidad de elementos del elemento estructurante.

En la Figura 69 se presenta el diseño realizado para el filtro morfológico. Cada elemento de V es enmascarado por un elemento de EE mediante la función lógica $f_i=\overline{EE_i}+V_i$ para $1\le i\le k$. Esto permite tener en cuenta en el proceso de filtrado únicamente los pixeles de la ventana para los cuales $EE_i=1$. Después de enmascarar los pixeles se aplica la operación lógica AND ente los elementos f_i obtenidos en el proceso de enmascaramiento lo que resulta en una operación morfológica de erosión.

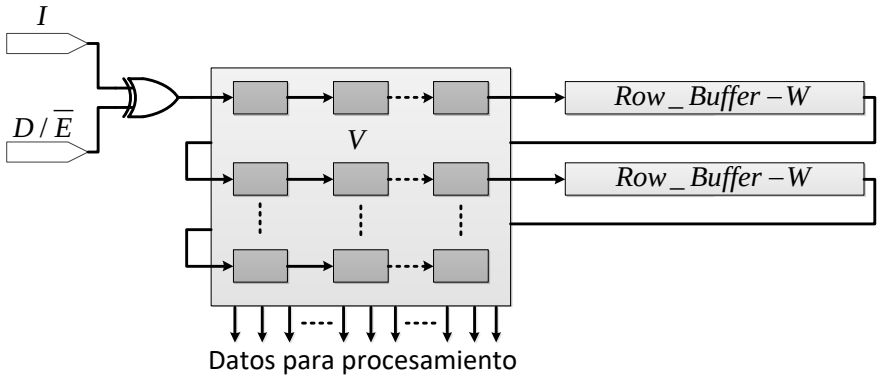


Figura 68.Arquitectura del generador de ventana para el filtrado morfológico. La señal $D/\overline{E}$ permite seleccional la operación morfológica a emplear. Figura adaptada por el autor a partir de (Bailey, 2011).

La operación morfológica de dilatación se obtiene complementando los pixeles de entrada y el resultado de las operaciones AND. Dicho complemento hace que los pixeles pertenecientes al fondo de la imagen tomen un valor de 1 dentro del generador de ventana, por lo cual al erosionarlos lo que se obtiene realmente es una dilatación del objeto. La operación lógica AND se realiza mediante una estructura de árbol con pipeline. La cantidad de estados de pipeline es $\log_2(k)$+2, ya que la estructura de árbol aumenta o disminuye dependiendo del tamaño de la ventana utilizada. La selección del modo de operación del módulo de procesamiento se realiza mediante la entrada $D/\overline{E}$. Si dicha entrada es 0, el modulo funciona como filtro morfológico de erosión, si la entrada es 1 el modulo funciona como filtro de dilatación.

Dado que el modulo diseñado para el cálculo de la segmentación del mapa de disparidad posee un generador de ventana móvil y una estructura de procesamiento *pipeline*, se necesitan $\frac{(W+1)\times(M+1)}{2} - M + \log_2(k) + 2$ ciclos de reloj válidos para obtener un dato valido a la salida. Por tal razón, se requiere un componente adicional que indique si el pixel es valido o no. Este se diseña con base en la arquitectura mostrada en la Figura 39. Este generador de validez emplea un valor de comparación de $\frac{(W+1)\times(M+1)}{2} - M + \log_2(k) + 1$ para su implementación en el módulo de filtrado morfológico.

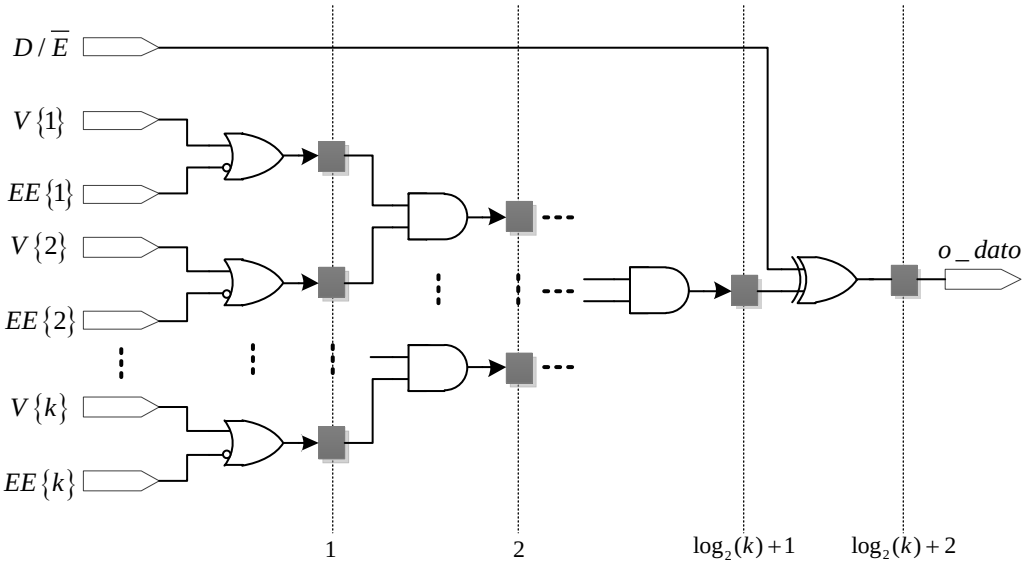


Figura 69.Arquitectura para las operaciones morfológicas básica Erosion y dilatacion. La señal $D/\overline{E}$ permite seleccional la operación morfológica a emplear. Fuente autor.

Los modulos de segmentación, correspondencia estéreo, filtro de mediana y filtro morfológico de apertura fueron descritos de forma paramétrica en lenguaje VHDL. En el anexo D se realiza la descripción de cada uno de estos modulos y en el anexo E se presentan los resultados de simulación obtenidos mediante el software Modelsim-Altera.

7 Extracción de características

La extracción de características es la tercera etapa de un sistema de reconocimiento de gestos basado en visión. En esta etapa se toman las imágenes pre-procesadas con el fin de extraer la información necesaria que describe características únicas para un determinado gesto. Dichas características se utilizan para distinguir un gesto de otro en una etapa posterior de reconocimiento.

En este capítulo se presenta el proceso utilizado en este trabajo para la extracción de características de un gesto a partir de las imágenes obtenidas en la etapa de pre-procesamiento. Este proceso se desarrolla mediante la ejecución de cuatro fases consecutivas: cálculo del centroide de la mano para la extracción de la trayectoria del gesto, filtro de media móvil para suavizado de la trayectoria del gesto, detección de gesto válido mediante condiciones de inicio-fin y extracción de características basadas en la orientación de la trayectoria del gesto. A continuación se presentan los algoritmos empleados en cada una de las fases del proceso, así como su implementación en FPGA.

7.1 Cálculo del Centroide de la Forma de la Mano

Un gesto dinámico cualquiera se construye a partir de la trayectoria del movimiento de la mano, también conocida como la trayectoria del gesto. En la mayoría de situaciones, es posible diferenciar un gesto de otro simplemente observando la trayectoria característica de cada uno. En un sistema de reconocimiento basado en visión, la trayectoria del gesto se construye mediante la unión de los valores de centroide de la forma de la mano obtenidos a partir de una secuencia de imágenes (ver Figura 70). En este trabajo el centroide de la forma de la mano se obtiene a partir de la imagen producida en la etapa de pre-procesamiento. Dicha imagen es una imagen binaria que contiene como único objeto la región de la forma de la mano capturada en un instante del tiempo.

Figura 70.Trayectoria de centroide de la mano para el gesto “Rotar a la Derecha 90º”. Fuente autor.

Existen diversos métodos que permiten realizar el cálculo del centroide de una región contenida en una imagen digital. Un método ampliamente utilizado para este fin es el uso de los momentos geométricos bidimensionales de primer orden. En la ecuación (23) se presenta la definición de los momentos geométricos bidimensionales de orden $p+q$ para una imagen digital de M filas por N columnas (Binh et al., 2005). La función $f(x,y)$

corresponde a la región u objeto de análisis dentro de la imagen. Dicha función en este trabajo corresponde a la región de la forma de la mano. El momento M_{00} describe el área de $f(x,y)$ en pixeles, y la relación con los momentos M_{10} y M_{01} permite calcular el centroide (x_c, y_c) de la región $f(x,y)$ como se muestra en la ecuación (24) .

$$M_{pq} = \sum_{x=1}^{N} \sum_{y=1}^{M} x^p y^q f(x,y) \tag{23}$$

$$x_c = \frac{M_{10}}{M_{00}} \qquad\qquad y_c = \frac{M_{01}}{M_{00}} \tag{24}$$

En la Figura 71 se presenta el algoritmo que describe el procedimiento para calcular el centroide de la forma de la mano mediante el uso de los momentos geométricos bidimensionales de primer orden. El centroide es calculado a partir de la imagen binaria obtenida en la etapa de pre-procesamiento. La imagen se recorre pixel a pixel de arriba hacia abajo y de izquierda a derecha. Para cada pixel de la imagen se calculan los momentos M_{00}, M_{10} y M_{01}.

```
1:  procedure CENTROIDE(I_bin)
2:      Input: I_bin: Imagen en binaria resultado del proceso de segmentación
3:                    y filtrado en la etapa de pre-procesamiento
4:
5:      Output: x_c: Coordenada x del centroide
6:              y_c: Coordenada y del centroide
7:
8:      I_W ← ancho de la imagen en pixeles
9:      I_H ← alto de la imagen en pixeles
10:     M_00 ← 0
11:     M_10 ← 0
12:     M_01 ← 0
13:
14:     for i ← 1 , I_W do                    ▷ Recorrido por las Columnas de la imagen
15:         for j ← 1 , I_H do                ▷ Recorrido por las Filas de la imagen
16:             M_00 ← M_00 + I_bin[i, j]
17:             M_10 ← M_10 + i × I_bin[i, j]
18:             M_01 ← M_01 + j × I_bin[i, j]
19:         end for
20:     end for
21:     x_c ← M_10 / M_00
22:     y_c ← M_01 / M_00
23:     return x_c, y_c
24: end procedure
```

Figura 71.Algoritmo para el calculo del centroide de la forma de la mano en una imagen binaria.

Se puede apreciar que la entrada del algoritmo es una imagen de tamaño $I_W \times I_H$. Sin embargo, el resultado son únicamente dos datos x_c y y_c que corresponden a las coordenadas del centroide calculado. Esto implica que la información de una imagen completa es codificada únicamente en dos datos que corresponden a la posición de la mano en un instante del tiempo t durante la realización de un gesto. Por consiguiente, la trayectoria del gesto, observada en imágenes consecutivas, se representa por un conjunto de puntos como se muestra en la ecuación (25), dónde (x_t, y_t) se refiere al punto del centroide de la mano en el tiempo *t* y *T* es la longitud de la trayectoria del gesto de la mano.

$$Trayectoria\_Gesto = \{(x_1, y_1), (x_2, y_2), ..., (x_t, y_t), ..., (x_T, y_T)\} \quad (25)$$

La implementación en hardware para el cálculo del centroide se desarrolló tomando como referencia el algoritmo presentado en la Figura 71. El diseño realizado para el módulo de hardware está compuesto por un generador de coordenadas que permite conocer la posición de un pixel dentro de la imagen, tres sumadores-acumuladores empleados para calcular los momentos geométricos bidimensionales de primer orden, y dos divisores para el cálculo de las coordenadas *x* e *y* del centroide. En la Figura 72 se presenta la arquitectura desarrollada.

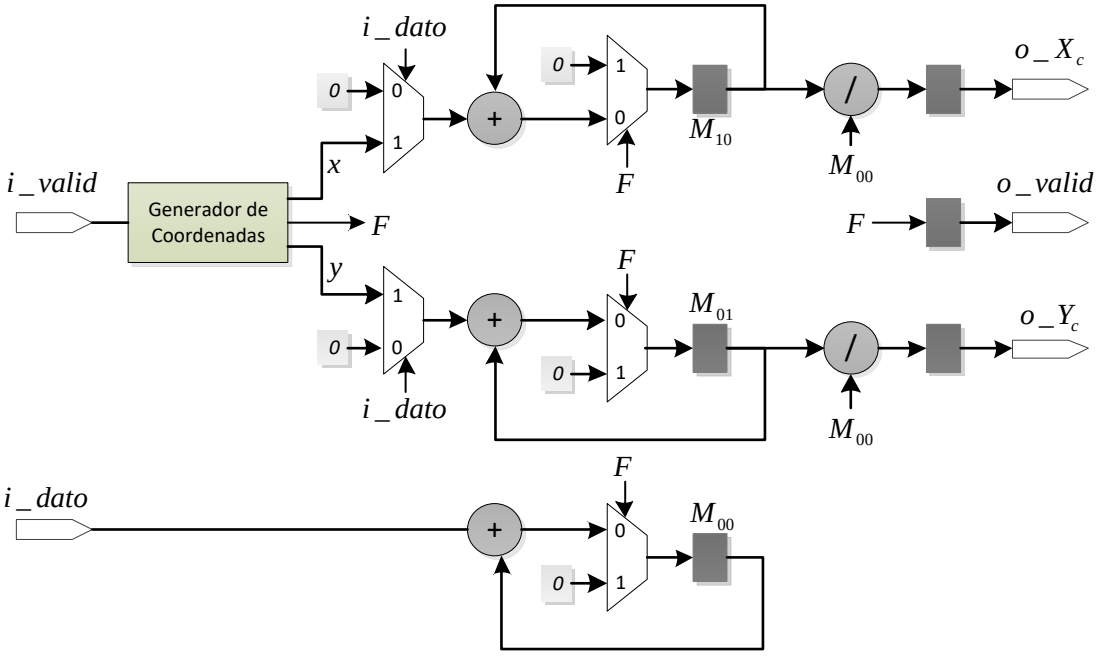


Figura 72. Arquitectura diseñada para el calculo del centroide. Fuente autor.

En esta arquitectura el generador de coordenadas produce tres señales de salida: las coordenadas *x* e *y*, y una salida F que es puesta a 1 durante un ciclo de reloj, indicando que se ha procesado una imagen completa. Cada vez que se ha procesado una imagen

completa las coordenadas *x*, e *y* se ponen a 0 nuevamente con el fin de esperar el ingreso de una nueva imagen al módulo para ser procesada. Los sumadores-acumuladores para el cálculo de los momentos geométricos, están controlados por el valor binario del pixel de entrada y por la señal F del generador de coordenadas. El valor del pixel habilita (pixel de la región del objeto) o deshabilita (pixel del fondo) los acumuladores dependiendo de su valor. La señal F reinicia los acumuladores con el fin de preparar el módulo para procesar una nueva imagen. Así mismo, la señal F permite el paso del valor calculado del centroide e informa al exterior de dicho evento.

En la Figura 73 se presenta el diagrama esquemático del módulo para el cálculo del centroide de la mano. En dicho diagrama se presenta la interfaz de conexión del módulo con el exterior. La señal *i_dato* corresponde al valor de los pixeles provenientes de la operación morfológica de apertura y la señal *i_valid* indica la validez de dichos pixeles. En cada flanco ascendente de la señal de reloj *i_clk* un pixel válido de la imagen binaria es capturado por el módulo. Las señales *o_Xc* y *o_Yc* corresponden al valor calculado del centroide. Estas señales son generadas por el módulo cuando una imagen completa ha sido analizada. La señal *o_valid* indica cuando se ha generado un nuevo valor de centroide válido. La señal *i_rst_n* tiene un comportamiento similar que en los módulos de hardware ya explicado en el capituro anterior. Adicionalmente, el módulo posee dos parámetros que permiten seleccionar su comportamiento en el proceso de síntesis. Los parámetros M y N permiten establecer el ancho y el alto de la imagen a ser procesada.

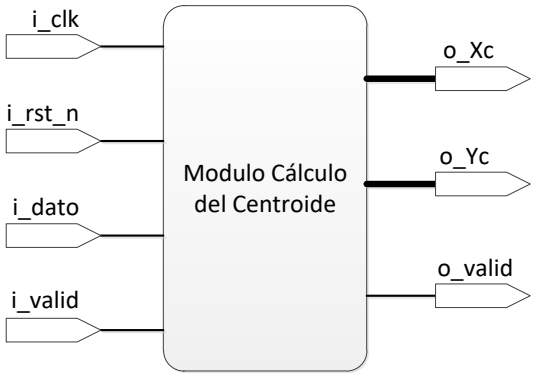


Figura 73.Diagrama esquemático del módulo para calcular el centroide de la mano en hardware. Fuente autor.

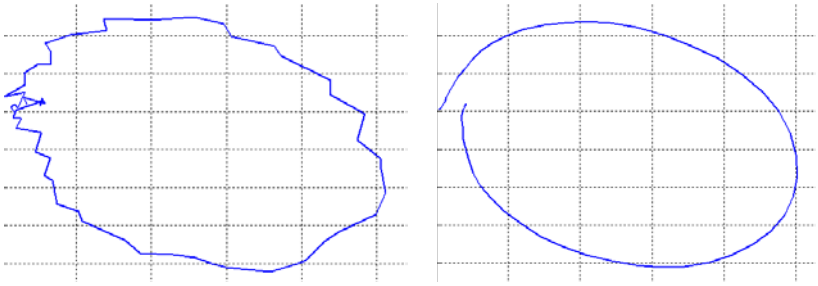
7.2 Suavizado del Gesto: Filtro de Media Móvil

La trayectoria del gesto de la mano se determina conectando los puntos del centroide como se describe en la sección anterior. Las imágenes de entrada suelen ser inestables debido al cambio en las condiciones de iluminación entre una imagen y otra, fondos desordenados y temblores durante el movimiento, lo cual causará cambios frecuentes y agudos en el centroide. Con el fin de superar eficazmente estos cambios inesperados, los puntos de trayectoria se suavizan utilizando el filtro de media móvil, el cual es un caso especial del filtro FIR regular. Este filtro entrega como salida el valor promedio de las

últimas N muestras de la entrada. En la ecuación (26) se presenta la ecuación que describe este tipo de filtro.

$$y[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n-k] \tag{26}$$

Para el suavizado del gesto se emplean dos filtros de media móvil, uno para la coordenada x del centroide y el otro para la coordenada y. En la Figura 74 se presenta un ejemplo de aplicación para el suavizado del gesto “Rotar a la Derecha 90°”. La Figura 74 (a) muestra la trayectoria del gesto en la que se evidencia presencia de ruido. En la Figura 74 (b) se muestra el resultado del suavizado de la trayectoria del gesto mediante la aplicación de los dos filtros de media móvil de longitud 4.



(a) (b)

Figura 74.Suavizado del gesto “Rotar a la Derecha 90º” mediante el filtro de media móvil de longitud 4. Fuente autor

La implementación en hardware del filtro de media móvil se lleva a cabo mediante la forma recurrente mostrada en la ecuación (27). Esta forma recurrente permite realizar la implementación de dicho filtro con menos hardware aritmético comparado con la forma normal de la ecuación (26). En la Figura 75 se muestra la arquitectura desarrollada para el cálculo en hardware del filtro de media móvil, la cual fue diseñada para realizar operaciones en aritméticas en punto fijo, con 10 bits para la parte entera y 8 bits para la parte fraccionaria.

$$y[n] = y[n-1] + \left(\frac{x[n]}{N} - \frac{x[n-N]}{N}\right) \tag{27}$$

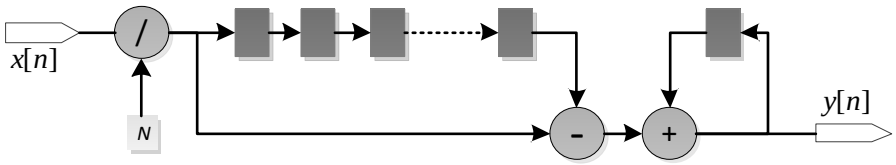


Figura 75.Arquitectura en hardware para el calculo del filtro de media movil de longitud N. Fuente autor.

La Figura 76 muestra el diagrama esquemático del módulo de hardware diseñado para el suavizado de gestos en el que se presenta la interfaz de conexión del módulo con el exterior. Las señales *i_Xc* e *i_Yc* corresponden al centroide de la forma de la mano

calculado previamente. Los datos del centroide son capturados en los flancos ascendentes de la señal de reloj *i_clk*, siempre y cuando la señal *i_valid* esté en alto. Las señales *o_Xc* y *o_Yc* corresponden a la salida del centroide resultado del suavizado del gesto. La señal *o_valid* indica que se ha procesado un nuevo valor de centroide. La señal *i_rst_n* tiene el mismo comportamiento que en los módulos explicados en secciones anteriores. El módulo de hardware tiene adicionalmente un parámetro N, el cual permite seleccionar la longitud del filtro en el proceso de síntesis. En este trabajo se encontró experimentalmente que el valor adecuado para el suavizado del filtro es N=4.

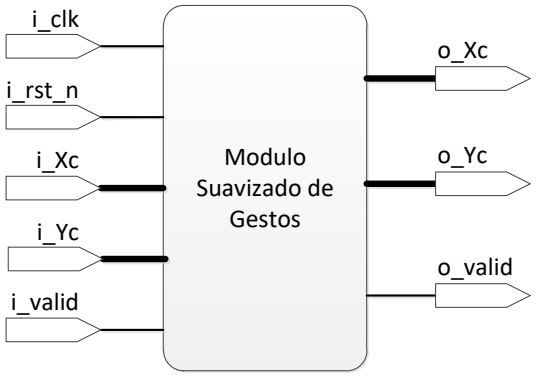


Figura 76.Diagrama esquemático del módulo de suavizado de la trayectoria de gestos en hardware. Fuente autor.

7.3 Detección del Gesto, "*Gesture Spotting*"

Según (M. O. S. M. Elmezain, 2010; M Elmezain, Al-Hamadi, Appenrodt, et al., 2008; M Elmezain et al., 2009; Mahmoud Elmezain & Al-Hamadi, 2012) un gesto se puede dividir en dos tipos. Gesto comunicativo y gesto no comunicativo. El gesto comunicativo (gesto-principal) está definido como una parte de la trayectoria de la mano que transporta información explicita, mientras que un gesto no comunicativo representa movimientos involuntarios los cuales son usados para conectar gestos principales. Los gestos no comunicativos se realizan al inicio (pre-gesto) y al final (pos-gesto) de un gesto principal.

En un sistema de reconocimiento de gestos, uno de los problemas más difíciles de resolver es la detección del gesto principal ("*Gesture Spotting*"). Normalmente el proceso de detección se realiza mediante la búsqueda de los puntos de inicio y de finalización del gesto a partir de una secuencia continua de movimientos de la mano. Sin embargo, este proceso resulta demasiado complejo debido a la evidente variabilidad de la forma, realización y duración de un gesto, así como de la información sin significado asociada al mismo (pre-gesto y pos-gesto). Por esta razón, los gestos empleados en este trabajo se realizan teniendo en cuenta una posición relativa de inicio y finalización en común, simplificando la realización del pre-gesto y pos-gesto respectivamente. Esto permite que el gesto sea detectado únicamente cuando la mano se pone en movimiento.

La realización de un gesto se lleva a cabo en tres fases, teniendo en cuenta la composición natural de un gesto: en la primera fase (pre-gesto), la mano debe estar detenida en la posición de inicio seleccionada, la segunda fase (gesto-principal) consiste en la ejecución del gesto moviendo la mano sobre la trayectoria deseada hasta llegar nuevamente a la posición de inicio, en la tercera fase (pos-gesto) la mano se detiene indicando la finalización del gesto ejecutado. A continuación la mano permanece detenida hasta el inicio de un nuevo gesto. En la Figura 77 se presenta un ejemplo que ilustra las fases de ejecución correspondiente al gesto "Rotar a la Derecha 90º". En dicha figura se muestra el comportamiento del centroide durante la realización del gesto en 80 imágenes consecutivas. La grafica de color azul corresponde a la coordenada x del centroide y la gráfica de color rojo corresponde a la coordenada y del mismo.

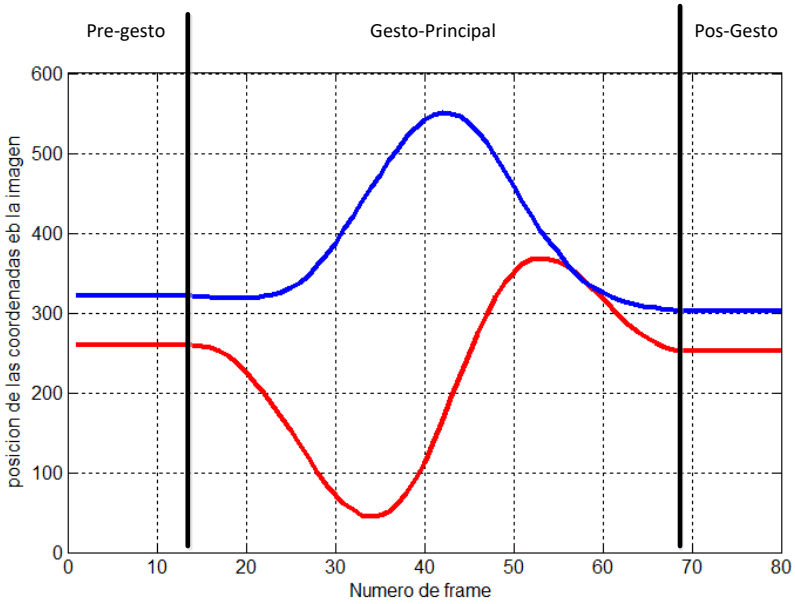


Figura 77. Fases de la ejecucion del gesto "Rotar derecha 90º". Fuente autor.

De acuerdo con el comportamiento del gesto mostrado en la Figura 77. El pre-gesto y pos-gesto corresponden a la ausencia de movimiento. Por esta razón, el valor del centroide en estas fases de la ejecución del gesto no debe ser tenido en cuenta como parte del mismo. Esto permite que el gesto-principal sea detectado mediante la observación del movimiento del centroide en imágenes consecutivas. Para determinar el movimiento del centroide en el tiempo t, se calcula la diferencia de dicho centroide con respecto a K valores de centroide anteriores. Si existe una diferencia superior a un valor de umbral T_m el centroide (x_t, y_t) es considerado válido y por consiguiente forma parte de un gesto-principal.

La Figura 78 presenta el algoritmo desarrollado para la detección del gesto-principal a partir del movimiento de las coordenadas del centroide en imágenes consecutivas. Para determinar el movimiento del centroide se calcula la diferencia de cada una de sus coordenadas con K valores previos. Si alguna de las coordenadas presenta una

diferencia mayor a un valor de umbral el centroide es considerado válido y por consiguiente el gesto en ese instante de tiempo es detectado. Los valores de K y T_m fueron calculados experimentalmente en este trabajo obteniendo $K = 8$ y $T_m = 30$.

procedure DETECCION DEL GESTO($C_x[t]$,$C_y[t]$)
    Input: $C_x[t]$: Coordenada x del centroide en un instante de tiempo t
        $C_y[t]$: Coordenada y del centroide en un instante de tiempo t

    Output: G_{valido}: salida que indica la validez del gesto en el tiempo t

    $F_x[i] \leftarrow |C_x[t] - C_x[t-i]|$ para $3 \leq i \leq K$
    $F_y[i] \leftarrow |C_y[t] - C_y[t-i]|$ para $3 \leq i \leq K$

    if $F_x[i] > T_m$ o $F_y[i] > T_m$ **then**
        $G_{valido}[t] \leftarrow 1$
    else
        $G_{valido}[t] \leftarrow 0$
    end if
    return G_{valido}
end procedure

Figura 78. Algortimo que describe el proceso para la deteccion de un gesto.

La implementación en hardware de la detección de gestos se realiza a partir del algoritmo presentado en la Figura 78. En la Figura 79 se presenta el diseño en hardware realizado para la detección de un gesto-principal. Este módulo realiza el proceso de detección del gesto mientras es ejecutado por el usuario. El módulo captura cada valor de centroide proveniente del módulo de suavizado del gesto y calcula la diferencia con respecto a los valores anteriores. Si una de las diferencias calculadas es mayor que el valor de umbral seleccionado, la salida *o_Gvalido* se pone en alto indicando que los valores de centroide capturados corresponden a un gesto-principal.

En la Figura 80 se presenta el diagrama esquemático correspondiente al módulo de detección de gestos en el que se indica la interfaz de conexión del módulo con el exterior. Las señales *i_Xc* y *i_Yc* corresponden a los valores de centroide provenientes del módulo de suavizado de gesto. Los valores de centroide son capturados en cada flanco ascendente de la señal de reloc *i_clk*, siempre y cuando la señal *i_valid* este en un nivel lógico alto. Las señales *o_Xc* y *o_Yc* corresponden a los valores de centroide de salida detectados como parte de un gesto-principal. La señal *o_gesto* indica la detección de un gesto-principal, la cual permanece en 1 durante la detección del gesto y pasa a 0 cuando no hay detección de un gesto. La señal *o_valid* indica la validez de los datos de salida.

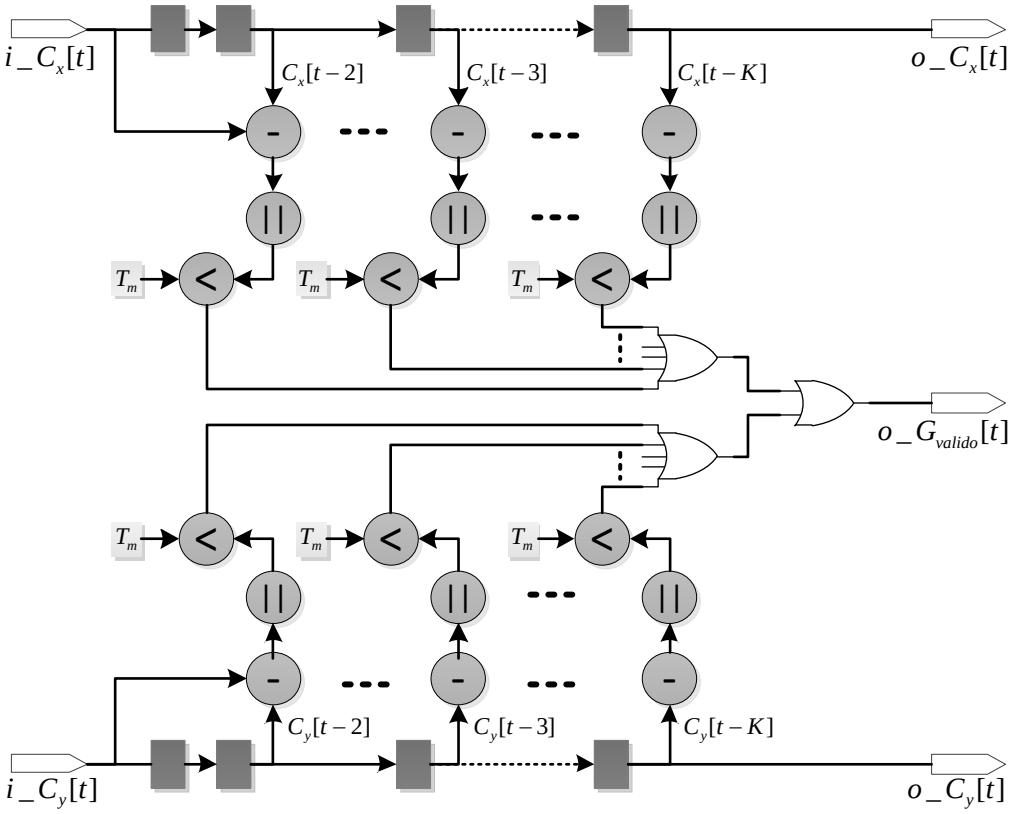


Figura 79. Implementacion en hardware de la deteccion de gesto válido. Fuente autor.

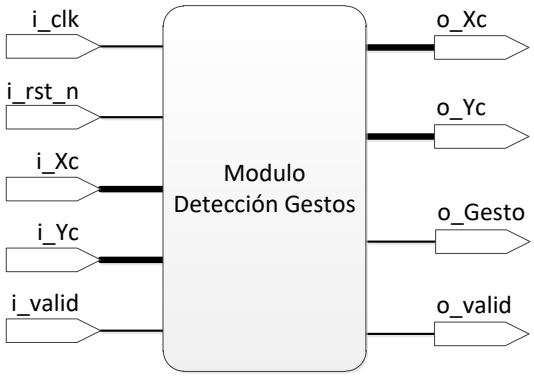


Figura 80. Diagrama esquemático de modulo de deteccion de gestos implementado en hardware. Fuente autor.

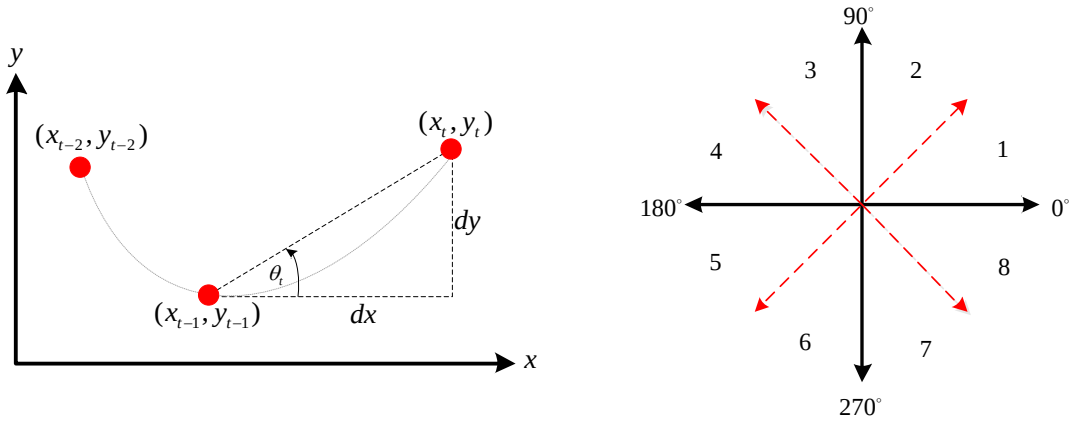
7.4 Extracción de Características

La selección de las mejores características para el reconocimiento de la trayectoria de los gestos de la mano juega un papel importante en el funcionamiento del sistema. Dichas características son empleadas en la etapa de reconocimiento para distinguir adecuadamente un gesto de otro. En la literatura existen actualmente tres características básicas: localización, orientación y velocidad. Según (M Elmezain, Al-Hamadi, Appenrodt, et al., 2008) la característica de orientación es la mejor en términos de resultados de

exactitud. Por esta razón, la orientación se selecciona como la característica principal del sistema de reconocimiento en este trabajo.

La trayectoria de un gesto es un patrón espacio-temporal que está conformado por un conjunto de puntos del centroide de la mano (x_{mano}, y_{mano}). La característica de orientación empleada para describir el gesto permite calcular la dirección de la mano cuando recorre el espacio cartesiano durante la realización del gesto. Por lo tanto, la orientación está determinada por dos puntos consecutivos de la trayectoria del gesto como se muestra en la ecuación (28), dónde θ_t representa la orientación del centroide en el instante de tiempo t, y T representa la longitud total del gesto.

$$\theta_t = \tan^{-1}\left(\frac{y_t - y_{t-1}}{x_t - x_{t-1}}\right); \quad t = 2,3,...,T \tag{28}$$



(a) (b)

Figura 81. Calculo de la orientacion. (a) orientacion entre dos puntos consecutivos del centroide. (b) codificacion discreta de 1 a 8 dividiendo la orientacion en 45º. Fuente autor.

Una vez extraída la orientación (ver Figura 81 (a)) se debe realizar un proceso de normalización o cuantización para obtener los símbolos discretos que se utilizan como entrada a los HMMs. Este proceso de cuantización se realiza dividiendo la orientación por 45º, lo que permite obtener el alfabeto de símbolos finito requerido por el sistema de reconocimiento basado en HMMs.

Cualquier valor de orientación es representado mediante un símbolo específico, por ejemplo si la orientación está dada en un rango de 0 a 45 º el símbolo corresponde al 1. Si la orientación está en el rango 135º a 180º el símbolo asignado corresponde al 4 y así sucesivamente para los demás valores de orientación (ver Figura 81 (b)). Este proceso de cuantización permite transformar el gesto realizado en una secuencia de observaciones discretas. En la Figura 82 se presenta el algoritmo que describe la extracción de la orientación θ_t y su valor de cuantización O_t para dos valores de centroide consecutivos (x_t, y_t) y (x_{t-1}, y_{t-1}). El valor de cuantización O_t ingresa directamente al sistema de

reconocimiento basado en HMMs, ya que dicho valor corresponde a una observación discreta del gesto en el instante de tiempo t.

procedure CARACTERISTICAS(Cx_t,Cx_{t-1},Cy_t,Cy_{t-1})
    Input: Cx_t: Coordenada x del centroide en un instante de tiempo t
        Cx_{t-1}: Coordenada x del centroide en un instante de tiempo $t-1$
        Cy_t: Coordenada y del centroide en un instante de tiempo t
        Cy_{t-1}: Coordenada y del centroide en un instante de tiempo $t-1$

    Output: O_t: símbolo correspondiente a la cuantizacion de la orientación calculada

    $dx \leftarrow (Cx_t - Cx_{t-1})$
    $dy \leftarrow (Cy_t - Cy_{t-1})$

    if $dx \geq 0$ y $dy > 0$ **then**
        $\theta_t \leftarrow \arctan\left(\frac{dy}{dx}\right)\left(\frac{180}{\pi}\right)$
    else if $dx < 0$ y $dy \geq 0$ **then**
        $dx \leftarrow |dx|$
        $\theta_t \leftarrow 180^\circ - \arctan\left(\frac{dy}{dx}\right)\left(\frac{180}{\pi}\right)$
    else if $dx < 0$ y $dy < 0$ **then**
        $dx \leftarrow |dx|$
        $dy \leftarrow |dy|$
        $\theta_t \leftarrow 180^\circ + \arctan\left(\frac{dy}{dx}\right)\left(\frac{180}{\pi}\right)$
    else if $dx > 0$ y $dy < 0$ **then**
        $dy \leftarrow |dy|$
        $\theta_t \leftarrow 360^\circ - \arctan\left(\frac{dy}{dx}\right)\left(\frac{180}{\pi}\right)$
    end if
    $O_t \leftarrow truncar\left(\frac{\theta_t}{45^\circ}\right)$
    return O_t
end procedure

Figura 82. Algoritmo que describe la extraccion de la orientacion y su valor de cuantizacion para dos valores de centroide consecutivos. Fuente …

7.4.1 Interfaz de conexión FPGA-HPS y extracción de características

En este trabajo, el proceso de extracción de la orientación como característica que describe la trayectoria del centroide, se lleva a cabo en el sistema de procesador físico (HPS) del dispositivo Cyclone V SOC empleado en este proyecto. La decisión de emplear este recurso del dispositivo radica principalmente en tres razones importantes: el cálculo de la orientación y su cuantización no requieren procesamiento intensivo, el HPS no hace uso adicional de recursos lógicos de la FPGA, y el HPS tiene la posibilidad de realizar fácilmente operaciones aritméticas con representación en punto flotante.

La razón por la que el cálculo de la orientación no requiere procesamiento intensivo se debe principalmente a que la operación involucra únicamente dos puntos (valores de centroide) consecutivos, a diferencia de una imagen la cual contiene demasiada información que debe ser procesada aplicando varias transformaciones simultáneamente

durante el tiempo de captura. Por tal motivo, la implementación del cálculo de la orientación se puede desarrollar en un procesador convencional.

El HPS del dispositivo Cyclone V SoC contiene un sistema basado en microprocesador que incluye una unidad de microprocesador (MPU) con dos núcleos ARM Cortex-A9, un controlador de memoria Flash, un controlador de memoria SDRAM, memoria on-Chip, interfaces de conexión a periféricos, etc. Este sistema es un recurso físicamente implementado dentro del chip Cyclone V SOC totalmente independiente de la zona FPGA propiamente dicha, lo que significa que su uso no requiere hardware adicional de la FPGA para su funcionamiento como es el caso de los SoftCores como el Nios II. Por lo tanto, el uso del HPS permite que se tengan disponibles todos los recursos lógicos reconfigurables de la FPGA para el desarrollo del hardware requerido en el procesamiento de imágenes, tales como la extracción del centroide, suavizado del gesto y detección del gesto-principal.

Debido a que la extracción de la característica de orientación y el reconocimiento de gestos basado en HMMs requieren de operaciones aritméticas con representación numérica de punto flotante (como la operación $\tan^{-1}$), y dado que la intensidad de procesamiento para dichos cálculos no es alta comparada con el procesamiento de las imágenes, estas operaciones se pueden implementar perfectamente en el sistema HPS.

Para realizar la extracción de la orientación como característica principal en el reconocimiento de gestos, es necesario establecer comunicación entre el HPS y el hardware de procesamiento desarrollado en la FPGA con el fin de que el gesto detectado en hardware pueda ser manipulado por el procesador. Sin embargo, el hardware desarrollado en la FPGA no puede comunicarse directamente con el HPS. Para esto, el dispositivo cuenta con interfaces especiales para este fin.

De acuerdo con la documentación oficial del fabricante del dispositivo (ver anexo A) el sistema HPS y la FPGA se comunican internamente mediante tres interfaces específicas: *HPS-to-FPGA Bridge*, *FPGA-to-HPS Bridge* y Lightweight HPS-to-FPGA Bridge. El *Lightweight HPS-to-FPGA Bridge* permite al procesador acceder fácilmente a los periféricos desarrollados en la FPGA a través del bus *Avalon-Memory-Maped (Avalon-MM)*. En este tipo de interfaz el procesador se comporta como maestro mediante el bus *Avalon-MM* y los periféricos en el lado de la FPGA como esclavos del bus *Avalon-MM*. Las demás interfaces poseen un funcionamiento similar con características más avanzadas las cuales no son abordadas en este trabajo.

Para que el procesador pueda acceder a la información del gesto detectado se realiza el diseño de un módulo de hardware adicional en la FPGA. Dicho módulo está diseñado para almacenar los datos correspondientes a un gesto detectado y los mantiene hasta el momento en el que el procesador pueda acceder a ellos. Este módulo permite sincronizar los datos obtenidos en hardware con el procesamiento desarrollado en el sistema HPS. El

módulo de almacenamiento desarrollado recibe el *stream* de datos provenientes del módulo de detección del gesto y los almacena en una memoria FIFO. Para extraer los datos de la memoria FIFO se diseña una interfaz esclavo compatible con el bus Avalon-MM, la cual se conecta con el sistema HPS a través del *Lightweight HPS-to-FPGA Bridge*. En el anexo A se presenta la documentación del bus Avalon-MM empleada para el diseño de la interfaz esclavo teniendo en cuanta las especificaciones de compatibilidad con dicho bus. En la Figura 83 se presenta un diagrama de bloques que ilustra la interconexión entre el sistema de procesamiento en FPGA y el sistema HPS a través del *Lightweight HPS-to-FPGA Bridge* mediante conexión del bus Avalon-MM.

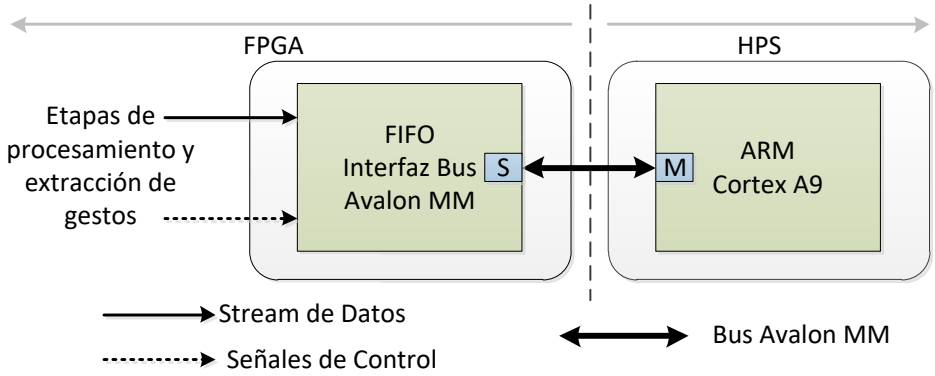


Figura 83. Interconexion entre el procesamiento en desarrollado en FPGA y el HPS a traves del *Lightweight HPS-to-FPGA Bridge*. Fuente autor.

Tabla 12. Mapa de registros del módulo de memoria FIFO.

Registros Mapeados en Memoria								
Offset en bytes	**Nombre del Registro**	**R/W**	**Descripción de Bits**					
			31…5	4	3	2	1	0
0	Dato_X	R	Valor de la coordenada x del centroide					
4	Dato_Y	R	Valor de la coordenada y del centroide					
8	Reservado	-	-	-	-	-	-	-
12	Control	RW	-	FIFO - FULL	FIFO-CLR	Gesto	Dato FIFO	GO
16	Longitud_X	R	Cantidad de datos disponibles para x					
20	Longitud_Y	R	Cantidad de datos disponibles para y					

El procesador del sistema HPS interactúa y se comunica con el hardware desarrollado en FPGA mediante el bus Avalon-MM a través de seis registros de 32 bits mapeados en memoria. En la Tabla 12 se presenta el mapa de registros que dispone el módulo de almacenamiento FIFO diseñado. Los registros Dato_X y Dato_Y son de solo lectura y permiten al procesador leer el valor del centroide para un gesto detectado. Los registros Longitud_X y Longitud_Y son registros de solo lectura y permiten determinar la cantidad de datos disponibles en el módulo a la espera de ser leídos por el procesador. Cada vez que se hace una lectura en los registros Dato_X ó Dato_Y el valor de los registros Longitud_X o Longitud_Y se decrementan. El registro de control es un registro de lectura-escritura el cual permite controlar el comportamiento del módulo FIFO.

Tabla 13. Descripción de los bits que conforman el registro de control del módulo FIFO.

Bits del Registro de Control			
Numero de bit	**Nombre del Bit**	**R/W**	**Descripción**
0	GO	R/W	Habilita la escrituras de memoria FIFO desde el módulo de procesamiento en FPGA 1=habilitar 0=deshabilitar
1	Dato FIFO	R	Indica si existen datos disponibles para ser leídos. 1=hay datos disponibles. 0=no hay datos disponibles
2	Gesto	R	Indica si el gesto capturado es válido 1= No se ha detectado gesto válido 0= se ha detectado un gesto válido
3	FIFO-CLR	R/W	Borra el contenido de la memoria FIFO 1= Borrar memoria FIFO. Cuando finaliza el borrado se pone a 0 automáticamente
4	FIFO-FULL	R	Indica que la memoria FIFO se ha llenado. 0= la memoria tiene espacio disponible 1= la memoria se ha llenado por completo

El registro de control dispone de bits para el control del módulo y bits que indican su estado. El bit 0 (GO) es un bit de lectura y escritura que permite habilitar o deshabilitar el ingreso de datos provenientes del sistema de procesamiento y detección de gestos. El bit 1 (Dato_FIFO) es un bit de solo lectura e indica la disponibilidad de datos del centroide para ser leídos por el procesador. El bit 2 (Gesto) es un bit de solo lectura e informa si el valor del centroide corresponde a un gesto detectado. El bit 3 (FIFO-CLR) es un bit de lectura-escritura y permite borrar el contenido de la memoria FIFO. Para borrar la memoria FIFO este bit debe ponerse en 1 lógico, éste pasa automáticamente pasa a 0 cuando ha finalizado el borrado de los datos de la memoria. El bit 4 es un bit de solo lectura e indica que la memoria FIFO ha alcanzado su máxima capacidad. En la Tabla 13 se describe la función que desempeña cada bit del registro de control.

La implementación de la extracción de características y del sistema de reconocimiento se lleva a cabo mediante el desarrollo de una aplicación de software para un sistema operativo Linux que se ejecuta en el HPS del dispositivo. Dicha aplicación accede a los datos del gesto detectado a través de los registros mapeados en memoria, extrae las características y evalúa los HMM para clasificar el gesto realizado. La extracción de las características se realiza mediante una función de software a partir del algoritmo presentado en la Figura 82. La implementación en software del sistema de reconocimiento se presenta en el siguiente capítulo.

Los módulos de suavizado de gestos y de detección de gestos fueron desarrollados siguiendo una descripción paramétrica en lenguaje VHDL. En el anexo D se presenta la descripción de los módulos. En el anexo E se muesran los resultados de simulación obtenidos mediante el software Modelsim-Altera para caso.

8 Reconocimiento

En este capítulo se presenta el desarrollo de la última etapa que conforma el sistema de reconocimiento propuesto en este trabajo. Esta etapa toma las características extraídas en la etapa anterior y realiza un proceso de clasificación para determinar el gesto al que pertenecen, con el fin de enviar un comando para controlar las acciones de un Robot LEGO NXT. En este trabajo se emplean los Modelos Ocultos de Markov HMM para realizar el proceso de clasificación de los gestos de las manos utilizados.

Los HMM pueden ser empleados de diferentes formas para hacer tareas de reconocimiento de gestos. Una forma bastante común (utilizada en este trabajo), consiste en modelar cada uno de los gestos mediante un HMM. El reconocimiento de gestos empleando esta estructura se realiza mediante dos pasos principales. i) construir un HMM λ para cada gesto y estimar los parámetros (A,B,π) mediante un proceso de entrenamiento realizado con el algoritmo de Waum-Welch, utilizando como datos de entrenamiento un conjunto de vectores de observación de cada gesto. ii) reconocer un gesto desconocido a partir de la secuencia de observaciones obtenida en la etapa de extracción de características. En este paso se calcula la probabilidad de cada modelo con respecto a la secuencia de observaciones del gesto. El modelo cuya probabilidad es mayor que la probabilidad de los demás modelos corresponderá al gesto de entrada realizado. En la Figura 84 se presenta un diagrama de bloques del sistema de reconocimiento basado en HMM utilizado en este trabajo.

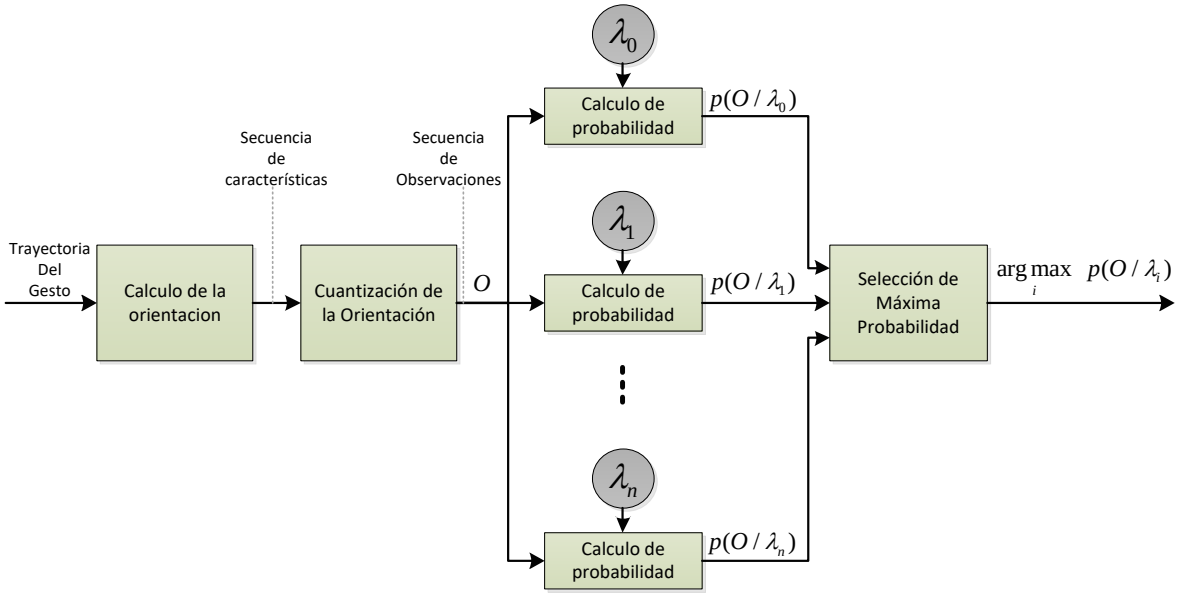


Figura 84. Diagrama de bloques de la etapa de reconocimiento de gestos empleando HMM. Adaptacion a partir de (M. O. S. M. Elmezain, 2010; Rabiner, 1989).

8.1 Selección del Modelo de los HMMs

Para construir un HMM se debe hacer la selección del modelo de acuerdo a tres parámetros importantes: selección de la topología del modelo (ergodica, o izquierda-derecha), selección de los símbolos de observación a ser empleados por el modelo (discretos o continuos) y selección del tamaño del modelo (número de estados ocultos). Desafortunadamente no existe una regla teórica que permita seleccionar estos parámetros de una manera sencilla, principalmente porque el modelo depende de los datos a ser modelados (Rabiner, 1989).

Un HMM puede construirse a partir de tres topologías diferentes: totalmente conectado (modelo ergódico) en el que cualquier estado puede ser alcanzado desde cualquier otro estado, izquierda-derecha en el que cada estado puede ir a los siguientes estados o regresar a sí mismo, y LRB (left-right banded) en el cual cada estado únicamente puede ir al siguiente estado o regresar a sí mismo. En la Figura 85 se presenta un ejemplo de la topología LRB de 6 estados. Para aplicaciones de reconocimiento de patrones se ha demostrado que la topología izquierda-derecha (LR) obtiene mejores resultados que la topología ergódica (FC) (N. Liu, Lovell, Kootsookos, & Davis, 2004; Siriboon, Jirayusakul, & Kruatrachue, 2002). En el caso particular de reconocimiento de gestos de la mano, la topología LRB presenta mejores resultados sobre la topología LR convencional (M Elmezain, Al-Hamadi, Appenrodt, et al., 2008; M Elmezain, Al-Hamadi, & Michaelis, 2008; M Elmezain et al., 2009; Mahmoud Elmezain, Al-Hamadi, Appenrodt, & Michaelis, 2009; N. Liu et al., 2004). Para el presente trabajo se selecciona la topología LRB para la construcción de los HMM del sistema de reconocimiento.

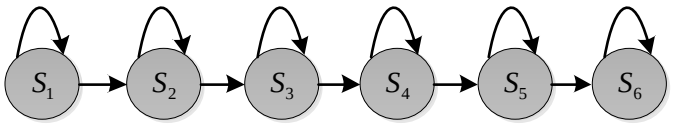


Figura 85. Topología LRB (Left-Right Banded) de 6 estados para un HMM. Figura adaptada por el autor a partir de (M. O. S. M. Elmezain, 2010; Rabiner, 1989)

Una vez seleccionada la topología del HMM se seleccionaron a continuación los símbolos de observación a ser empleados por el modelo los cuales pueden ser discretos o continuos. Las observaciones en el reconocimiento de gestos de la mano usualmente son consideradas como símbolos discretos (M Elmezain, Al-Hamadi, Appenrodt, et al., 2008; M Elmezain, Al-Hamadi, & Michaelis, 2008; M Elmezain et al., 2009; Mahmoud Elmezain et al., 2009; N. Liu et al., 2004; Siriboon et al., 2002). En este trabajo, la secuencia de observaciones se obtiene mediante la cuantización de la secuencia de orientaciones calculadas en la etapa de extracción de características. Por consiguiente, los símbolos de observación empleados por el modelo son los símbolos discretos 1-8.

Decidir la cantidad de estados que requiere el modelo para describir adecuadamente una secuencia de observaciones en particular es una tarea compleja. Cuando el número de

muestras de datos de entrenamiento es insuficiente, el uso de un número excesivo de estados causa un problema de sobre-entrenamiento. Además, el poder de discriminación de los HMMs disminuye cuando se utiliza un número insuficiente de estados, debido a que más de una parte segmentada del patrón gráfico se modela en un solo estado. Sin embargo, existen varios métodos y criterios desarrollados con el propósito de seleccionar adecuadamente dicho número de estados tales como, el criterio de información de Akaike (AIC), el criterio de Información de Bayes (BIC), validacion cruzada, etc (Le, Chatelain, & Berenguer, 2014). En este trabajo se emplea el método de validación cruzada *k-fold-Cross-Validation* para seleccionar el número de estados adecuado de los HMMs empleados en el sistema de reconocimiento propuesto.

8.2 Inicialización de los Parámetros del HMM Con Topología LRB

Antes de iniciar el algoritmo iterativo de Baum-Welch, se deben asignar los valores iniciales para todos los parámetros en los HMM. Una buena inicialización de los parámetros (A,B,π) de los HMMs permite alcanzar mejores resultados en el sistema de reconocimiento. Según (M Elmezain, Al-Hamadi, & Michaelis, 2008; N. Liu et al., 2004) la inicialización de los parámetros de los HMM se debe realizar como se muestra a continuación, teniendo en cuenta que la matriz de transiciones A es el primer parámetro a ser inicializado conforme se define en la ecuación (29).

$$A=\begin{pmatrix} a_{11} & 1-a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & 1-a_{22} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix} \tag{29}$$

Los elementos de la diagonal a_{ii} de la matriz de transición pueden ser seleccionados para indicar de forma aproximada la duración promedio d en cada estado. En la ecuación (30) se presenta el cálculo de la duración promedio por estado, siendo T la longitud de la trayectoria del gesto realizado y N el número de estados. En este caso se tomó un valor promedio de $T=60$ ya que los gestos empleados tienen longitudes variables entre 50 a 80 datos. En la ecuación (31) se presenta el cálculo de las transiciones a_{ii}.

$$d=\frac{T}{N} \tag{30}$$

$$a_{ii}=1-\frac{1}{d} \tag{31}$$

La matriz de observación B está determinada por la ecuación (32). Dado que los estados HMM son discretos, todos los elementos de la matriz B se pueden inicializar con el

mismo valor para todos los estados. B es una matriz de $N \times M$ donde b_{im} es la probabilidad de emitir un símbolo v_m en el estado i.

$$B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1M} \\ b_{21} & b_{22} & \cdots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & \cdots & b_{Nm} \end{pmatrix} ; \; b_{im} = \frac{1}{M} \tag{32}$$

Dado que las transiciones de estados solo pueden ejecutarse sobre el mismo estado o al siguiente en cada observación. El vector de probabilidad inicial π debe inicializarse como se muestra en la ecuación (33). Esto asegura el correcto inicio del modelo en el primer estado.

$$\pi = (1 \quad 0 \quad \cdots \quad 0)^T \tag{33}$$

8.3 Entrenamiento de los HMM

El entrenamiento de los parámetros inicializados (A, B, π) de un HMM se lleva a cabo mediante el algoritmo de Baum-Welch (ver sección 3.4.1). Este algoritmo toma como entradas los parámetros del modelo inicializados y un conjunto de vectores discretos para estimar una nueva matriz de transiciones A y una nueva matriz de observaciones B. El modelo entrenado debe ser lo suficientemente flexible para representar una secuencia de prueba no utilizada durante el entrenamiento. El entrenamiento se repite hasta que las matrices de transición y observación converjan o se llegue al máximo numero de iteraciones (M Elmezain, Al-Hamadi, & Michaelis, 2008).

Para realizar el proceso de entrenamiento de cada uno de los HMM del sistema de reconocimiento de gestos, se emplea el toobox de HMM para MATLAB desarrollado por Kevin Murphy (Murphy, 1998). El entrenamiento se lleva a cabo mediante la función *dhmm_em* de dicho toolbox. Esta función realiza el entrenamiento de un HMM con salidas discretas mediante el algoritmo de Baum-Welch. Los parámetros de entrenamiento utilizados son: máximo número de iteraciones 5000 y tolerancia de convergencia 0.0001. El proceso de entrenamiento se realiza de forma *off-line* en MATLAB. Para realizar el entrenamiento se tomaron 60 muestras por cada gesto, realizados por 3 personas diferentes, obteniendo un total de 1260 muestras.

8.4 Validación Cruzada *K-fold* y Selección del Número de Estados de los HMMs

La validación cruzada es un método estadístico de evaluación y comparación de algoritmos de aprendizaje a partir de la división de los datos en dos segmentos: uno

utilizado para entrenar un modelo y el otro utilizado para validar el modelo entrenado. En la validación cruzada típica, los conjuntos de entrenamiento y validación deben cruzarse en rondas sucesivas de manera que cada conjunto de datos tenga una oportunidad de ser validado.

La forma básica de validación cruzada es la validación cruzada k-fold (Refaeilzadeh, Tang, & Liu, 2009). En este tipo de validación, los datos primero se dividen en k segmentos o grupos de igual tamaño. Posteriormente se realizan k iteraciones de entrenamiento y validación de tal manera que dentro de cada iteración un grupo diferente de datos se emplea para la validación, mientras que los restantes k-1 grupos se utilizan para el entrenamiento. La Figura 86 muestra un ejemplo tomando un valor de k = 3. La sección más oscura de los datos se utiliza para el entrenamiento mientras que las secciones más claras se utilizan para la validación. En aplicaciones de aprendizaje automático, la validación cruzada de 10-fold (k = 10) es la más común (Refaeilzadeh et al., 2009).

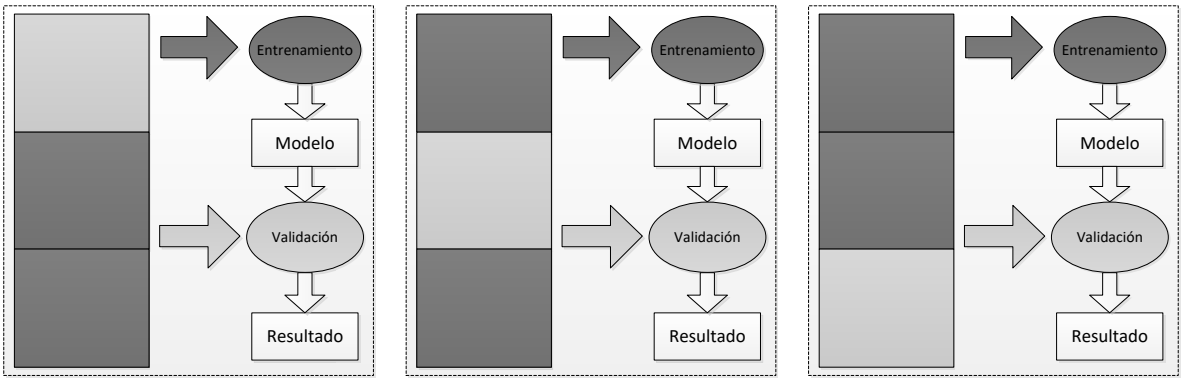


Figura 86. Procedimiento de validacion cruzada 3-fold. Figura adaptada por el autor a partir de (Refaeilzadeh et al., 2009)

La validación cruzada es fundamental en la selección de los parámetros adecuados que maximizan el rendimiento de un modelo. En este trabajo, el método de validación cruzada k-fold se emplea para seleccionar de forma apropiada el tamaño (número de estados ocultos) de los HMMs empleados en el sistema de reconocimiento de gestos propuesto. La validación cruzada k-fold para la selección del tamaño adecuado de los HMMs se lleva a cabo de la siguiente forma:

i. Construir Q HMMs con un número n de estados ocultos cada uno. Así mismo, inicializar los parámetros (A, B, π) como se explicó en la sección 8.2. La cantidad Q de HMMs a construir depende de la cantidad de gestos a reconocer. En el caso particular de este trabajo se construyen 7 HMMs.

ii. Dividir en K grupos iguales los datos disponibles de cada gesto. Se debe disponer de la misma cantidad de datos para cada uno de los gestos empleados en el

entrenamiento. En este trabajo se emplea una base de datos con 180 muestras para cada gesto.

iii. Realizar el entrenamiento de cada HMM por separado, empleando los datos del gesto que le corresponda al modelo. Para realizar dicho entrenamiento se deben emplear K-1 grupos obtenidos en la etapa anterior, dejando 1 para validación. El proceso de entrenamiento se lleva a cabo como se explicó en la sección 8.3.

iv. Validar el clasificador basado en HMMs mediante los datos que no fueron utilizados en el entrenamiento. Este procedimiento se lleva a cabo mediante la evaluación de los datos de validación de cada gesto, empleando todos los HMM entrenados. Esta evaluación se realiza mediante el cálculo de la probabilidad $P(O/\lambda_i)$ en cada HMM, mediante el algoritmo de *forward* (ver sección 3.4.1). Una vez calculadas las probabilidades de los HMM, cada uno de los datos de validación es clasificado como un determinado gesto cuyo modelo obtiene el máximo valor de probabilidad. Por último, se obtiene la exactitud del clasificador ACC_k a partir de la matriz de confusión con los datos de validación.

v. Repetir los pasos *iii* y *iv* K veces y calcular la exactitud de validación cruzada empleando la ecuación (34).

$$CV(\lambda) = \frac{1}{K}\sum_{k=1}^{K} ACC_k(\lambda) \tag{34}$$

vi. Repetir los pasos del i al v para HMMs de diferentes tamaños y obtener la exactitud de la validación cruzada para cada modelo.

Después de realizar el proceso de validación para el clasificador basado en HMMs con diferentes tamaños, finalmente se selecciona la cantidad de estados que produce el mayor valor de exactitud del clasificador. En este trabajo se realiza el proceso de validación cruzada empleando de 2 hasta 20 estados mediante la validación cruzada *k-fold* con un valor de k=10. El entrenamiento y validación se realiza de manera *off-line* mediante el software MATLAB y el mejor resultado obtenido en dicho proceso es implementado en el sistema de reconocimiento de gestos desarrollado en el sistema SoC empleado.

8.5 Implementación de la Etapa de Reconocimiento en el HPS

De acuerdo con lo explicado en la sección 3.4.1 el problema 1 de un HMM es evaluar la probabilidad de una secuencia de observaciones dado un modelo λ. Dicho problema se puede resolver de forma eficiente mediante el algoritmo *Forward*. EL algoritmo de *Forward* se ejecuta en tres etapas principales: inicialización, recursión y finalización. La

inicialización calcula el valor de la variable α del algoritmo *Forward* a partir de los parámetros (A,B,π) del HMM entrenado y del primer símbolo observado. En la Figura 87 se presenta una descripción del procedimiento de inicialización del algoritmo *Forward*.

```
1: procedure HMM FW INIT(A,B,π,O_1)
2:     Input: A: Matriz de Transición de Estados del HMM
3:            B: Matriz de Observaciones del HMM
4:            π: Vector de Probabilidad inicial del HMM
5:            O_1: Primer simbolo de la secuencia de observaciones
6:
7:     Output: α: variable del algoritmo de forward inicializada
8:
9:     for i ← 1, N do
10:        α_1[i] ← π[i] × B[i][O_1]
11:    end for
12:    return α
13: end procedure
```

Figura 87. Inicializacion del algoritmo de forward.

```
1: procedure HMM FW ITER(A,B,π,O_t,α_{t-1})
2:     Input: A: Matriz de Transición de Estados del HMM
3:            B: Matriz de Observaciones del HMM
4:            π: Vector de Probabilidad inicial del HMM
5:            O_t: símbolo de la secuencia de observaciones en el instante de tiempo t
6:            α_{t-1}: variable del algoritmo de forward calculada en t − 1
7:
8:     Output: α_t: valor de la variable de forward obtenida a partir de la observacion O_t
9:
10:    for i ← 1, N do
11:        α_t[i] ← 0
12:        for j ← 1, N do
13:            α_t[i] ← α_t[i] + α_{t-1}[i] × A[j][i]
14:        end for
15:        α_t[i] ← α_t[i] × B[i][O_t]
16:    end for
17:    return α_t
18: end procedure
```

Figura 88. Evaluacion del algoritmo de forward para un simbolo observado en un instante de tiempo t.

La etapa de recursión actualiza el valor de la variable α en el instante de tiempo t tomando el símbolo observado en dicho instante de tiempo y los parámetros (A,B,π) del HMM entrenado. En la Figura 88 se presenta el procedimiento que describe la etapa de recursión del algoritmo *Forward* en un instante de tiempo t.

La etapa de finalización calcula la probabilidad de la secuencia observada, dado el modelo λ con parámetros (A,B,π). En esta etapa, la probabilidad de observar toda la

secuencia está dada por la suma de todos los estados finales posibles presentes en la variable α calculada para el último símbolo observado. En la Figura 89 se presenta el procedimiento que describe la finalización del algoritmo *Forward* para el cálculo de la probabilidad de un HMM, dada una secuencia de observaciones O.

```
1: procedure HMM FW END(α_{t-1})
2:     Input: α_{t-1}: variable del algoritmo de forward calculada en t − 1
3:
4:     Output: P: el valor de la probabilidad calculada P(O/λ)
5:
6:     P ← 0
7:     for i ← 1, N do
8:         P ← P + α_{t-1}[i]
9:     end for
10:    return P
11: end procedure
```

Figura 89. Finalizacion del algoritmo de Forward para calcular la probabilidad total del HMM.

La etapa de reconocimiento es implementada en el HPS mediante el desarrollo de una aplicación de software para ser ejecutada sobre un sistema operativo Linux. Para el desarrollo de la aplicación de software se tiene en cuenta la guía planteada en la documentación del fabricante, la cual puede ser consultar en el anexo A. Dicha aplicación está diseñada para interactuar con el sistema de procesamiento de imágenes desarrollado en FPGA (ver sección 7.4.1), reconocer el gesto realizado mediante el clasificador basado en HMMs y enviar un comando al robot LEGO NXT para ejecutar la acción según corresponda.

La aplicación está diseñada mediante una FSM que permite implementar el clasificador basado en HMMs de la Figura 84. Dicha FSM extrae el valor de un símbolo de observación en cualquier instante del tiempo t y calcula la probabilidad de dicho símbolo mediante el algoritmo *Forward* aplicado a cada uno de los HMMs previamente entrenados. Además, la FSM clasifica la trayectoria del gesto observada de acuerdo con el modelo cuya probabilidad es mayor a la probabilidad de los demás modelos empleados. El resultado de esta clasificación se emplea para enviar un comando al robot con la acción correspondiente al gesto realizado.

En la Figura 90 se presenta el diagrama de estados de la FSM empleada para realizar la etapa de reconocimiento. En la Tabla 14 se presenta la descripción de las entradas utilizadas para determinar las transiciones de la FSM. Dichas entradas se obtienen a partir del registro de control mapeado en memoria del periférico de almacenamiento FIFO explicado en la sección 7.4.1. La FSM inicia en el estado S0. Si hay un gesto valido disponible en ese momento, la maquina permanece en S0 y da la orden de borrar el contenido del componente FIFO. En caso de no existir gesto disponible la FSM pasa al estado S1 y habilita el componente FIFO para recibir datos.

En el estado S1 se evalúa si existe gesto disponible, de ser así, la maquina realiza una transición al estado S2, de lo contrario se devuelve a S0 iniciando el proceso nuevamente. Cuando la FSM se encuentra en el estado S2 existen tres posibilidades: i) hay un gesto válido pero no hay datos disponibles para ser procesados por lo tanto permanece en el estado S2, ii) hay un gesto válido y también hay datos disponibles, en este caso se extrae un dato de centroide de la memoria FIFO y se hace transición al estado S3, iii) si no hay gesto válido se hace una transición al estado S5 indicado la finalización del gesto.

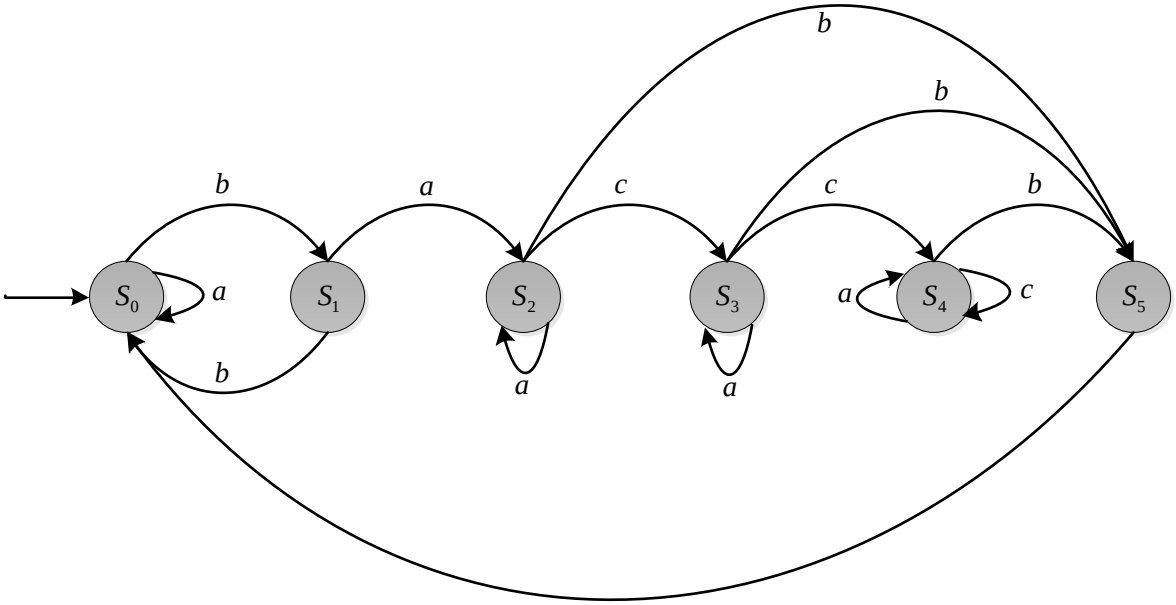


Figura 90. Diagrama de estados de la etapa de reconocimiento desarrollada en software. Fuente autor.

Tabla 14. Descripción de las entradas de la FSM del sistema de reconocimiento.

Entradas de la FSM	A	B	C
Descripción	Gesto valido	Gesto no valido	Gesto válido y datos disponibles en FIFO

Cuando la FSM se encuentra en el estado S3 se realiza cualquiera de las siguientes acciones: i) si existe un gesto válido pero no hay datos disponibles para procesar, la FSM permanece en el estado S3, ii) si existe gesto válido y hay datos disponibles, la FSM hace transición al estado S4, extrae un valor de centroide del periférico de almacenamiento FIFO, realiza la extracción del primer símbolo de observación (ver Figura 82) e inicializa el algoritmo *Forward,* iii) si no hay gesto válido, la FSM hace transición al estado S5.

Cuando la FSM se encuentra en el estado S4 se realiza cualquiera de las siguientes acciones: i) si existe gesto válido pero no hay datos disponibles, la FSM permanece en el estado S4, ii) si existe gesto válido y datos disponibles, la FSM permanece en el estado S4, extrae un valor de centroide del periférico de almacenamiento FIFO, obtiene el

símbolo de observación correspondiente y realiza la etapa de recursión del algoritmo *Forward,* iii) si no existe gesto valido, la FSM hace transición al estado S5.

En el estado S5 se realiza la etapa de finalización del algoritmo *Forward*, se clasifica el gesto realizado según el HMM con el mayor valor de probabilidad obtenida y se envía al robot el comando según corresponda. En este estado la FSM finaliza el proceso de reconocimiento de un gesto, por lo cual, hace una transición al estado S0 para iniciar el proceso con un gesto nuevo.

8.5.1 Control del robot Lego Mindstorms NXT

Para controlar las acciones que realiza el robot, el sistema de reconocimiento envía los comandos a través de una comunicación Bluetooth. Para establecer este tipo de comunicación con el robot se emplea el dispositivo Bluetooth RN-42. La comunicación entre la aplicación desarrollada y el dispositivo Bluetooth se lleva a cabo mediante un IP *core* RS232 del Programa Universitario de Altera®, el cual permite configurar el dispositivo y realizar el emparejamiento con el Robot. Este IP *core* se comunica con el HPS mediante el bus Avalon-MM a través del *Lightweight HPS-to-FPGA Bridge* de forma similar al módulo explicado en la sección 7.4.1. En el anexo A se presentan las especificaciones del módulo RS232 empleado. En la Figura 91 se presenta un diagrama de bloques donde se muestran los componentes que intervienen en la comunicación del robot con el sistema de reconocimiento.

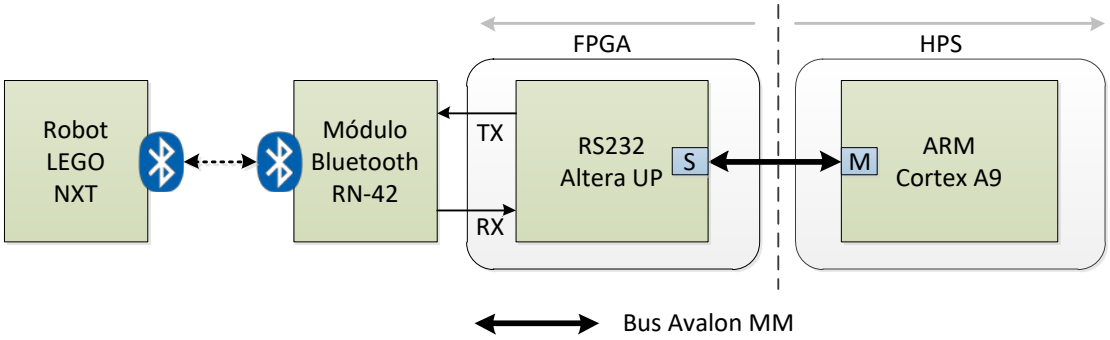


Figura 91. Comunicación entre el sistema de reconocimeinto y el Robot LEGO NXT. Fuente autor.

El dispositivo Bluetooth RN-42 es configurado para operar a 115.200 baudios, 8 bits de datos, 1 bit de parada y 0 bits de paridad. Además, la aplicación de software desarrollada para el sistema de reconocimiento realiza el emparejamiento con el robot LEGO NXT mediante el dispositivo RN-42 como maestro. Estas configuraciones se realizan siguiendo el manual de referencia del dispositivo, el cual puede consultarse en el anexo A.

Para enviar las acciones que se desean ejecutar en el robot de acuerdo con el gesto realizado, es necesario emplear el protocolo de intercambio de mensajes establecido por el fabricante del robot (ver anexo A). Este protocolo de intercambio de mensajes requiere el envío de una cadena de bytes desde el dispositivo maestro. En la Figura 92 se presenta la descripción de dicho protocolo. Los bytes 0 y 1 corresponden a la longitud

total en bytes del mensaje a enviar. Siendo el byte 0 la parte LSB y el byte 1 la parte MSB. El byte 2 permite configurar al receptor para enviar el estado de la transmisión después de recibir el mensaje (0x80 retorna estado, 0x00 no retorna estado). El byte 3 establece que se realizara una escritura. El byte 4 selecciona el buzón donde será enviado el mensaje (0-9). En byte 5 establece la longitud en bytes del mensaje a enviar (N+1). A partir del byte 6 se dispones de N posiciones para enviar un mensaje. El byte N+6 informa la finalización del mensaje, por lo tanto, siempre debe ser 0x00. De acuerdo con lo anterior, en la Tabla 15 se presentan los comandos enviados al robot desde la aplicación, teniendo en cuenta el protocolo establecido por el fabricante del robot.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	N bytes	Byte N+6
Longitud LSB	Longitud MSB	Retorno estado	0x09	Mailbox	N+1	Mensaje	0x00

Figura 92. Protocolo de intercambio de mensajes via bluetooth de LEGO NXT. Fuente autor.

Tabla 15. Lista de comandos empleados para el control del robot de acuerdo con el gesto realizado.

Gesto identificado	Comando a enviar
Adelante	{0x0D,0x00,0x80,0x09,0x00,0x09,'A','d','e','l','a','n','t','e',0x00}
Atrás	{0x0A,0x00,0x80,0x09,0x00,0x06,'A','t','r','a','s',0x00}
Derecha	{0x0C,0x00,0x80,0x09,0x00,0x08,'D','e','r','e','c','h','a',0x00}
Izquierda	{0x0E,0x00,0x80,0x09,0x00,0x0A,'I','z','q','u','i','e','r','d','a',0x00}
Rotar a la Derecha 90º	{0x0C,0x00,0x80,0x09,0x00,0x08,'R','o','t','a','r','_','D',0x00}
Rotar a la Izquierda 90º	{0x0C,0x00,0x80,0x09,0x00,0x08,'R','o','t','a','r','_','I',0x00}
Stop	{0x09,0x00,0x80,0x09,0x00,0x05,'S','t','o','p',0x00}

Los comandos mostrados en la tabla anterior son enviados al robot únicamente si el sistema de reconocimiento clasifica el gesto realizado como uno de los 7 gestos previstos. El robot recibe el comando y extrae únicamente el mensaje que se encuentra contenido dentro del mismo. De acuerdo con la información disponible en dicho mensaje, el robot inicia la ejecución de la acción correspondiente.

En el anexo H se presentan los códigos fuente de la aplicación de software diseñada para llevar a cabo la etapa de reconocimiento e interacción con el robot LEGO NXT. Así mismo, se presenta el programa desarrollado en el robot para detectar los comandos y ejecutar las acciones correspondientes a cada uno.

9 Resultados

En este capítulo se presentan los resultados obtenidos en cada una de las cuatro etapas que componen el sistema de reconocimiento de gestos propuesto en este trabajo. Los algoritmos de procesamiento de imágenes son evaluados individualmente tomando como referencia su implementación en software. Además, se presentan los resultados obtenidos por el funcionamiento total del sistema implementado en la tarjeta de desarrollo DE1-SoC.

Cada uno de los módulos de hardware para el procesamiento de imágenes desarrollados es comprobado funcionalmente mediante la descripción de un *test-bench* ejecutado sobre MODELSIM®. Adicionalmente, se utiliza la librería DSPBUILDER de Altera® para realizar la comprobación del hardware diseñado, empleando imágenes en el entorno Simulink de MATLAB®. Los resultados de simulación empleando Simulink son comparados con los resultados obtenidos mediante la ejecución de los algoritmos implementados en MATLAB. Para las pruebas de los algoritmos en software se emplea MATLAB R2013b sobre un computador con sistema operativo Windows 7 y un procesador Intel Core i7-4790 a una frecuencia de 3.6GHz.

9.1 Captura de Gestos

La captura de los gestos se realiza mediante un sistema de visión estéreo conformado por dos cámaras ArduCam las cuales poseen un sensor OV5642. Para adquirir las imágenes se emplean cuatro módulos de hardware los cuales permiten configurar, capturar y sincronizar las imágenes estéreo con resolución de 640x480 pixeles y una velocidad de captura de 30fps (ver capítulo 5). En la Figura 93 se presenta un par de imágenes estéreo obtenido mediante el sistema de captura diseñado. En esta figura se observa que las imágenes capturadas presentan una distorsión de barril, la cual puede corregirse empleando un proceso posterior de rectificación.

(a) (b)

Figura 93. Imágenes estéreo obtenidas en la etapa de captura. (a) imagen izquierda. (b) imagen derecha. Fuente autor.

La correspondencia estéreo requiere que las imágenes sean rectificadas previamente. Sin embargo, el desarrollo de dicho proceso de rectificación en este trabajo puede resultar

demasiado complicado debido a dos razones principales. En primer lugar la extracción de las imágenes de prueba desde la FPGA hacia el PC, para obtener los paramentos de calibración y rectificación en MATLAB, es demasiado dispendiosa. Esto se debe a que se requiere el diseño de un hardware adicional que permita enviar los pixeles capturados hacia el computador a través de una interfaz en común. Por otra parte, el diseño e implementación de la rectificación de imágenes estéreo en hardware resulta demasiado compleja debido al tipo de operaciones aritméticas involucradas, así como a las estrictas restricciones asociadas al procesamiento mediante *stream-processing*. Por lo tanto, en este trabajo se optó por realizar el cálculo del mapa de disparidad sin llevar a cabo el proceso de rectificación en la etapa de captura.

Para compensar la falta del proceso de rectificación, el sistema de cámaras estéreo se construye cuidadosamente buscando mantener (manualmente) una alineación horizontal aceptable que permita calcular el mapa de disparidad lo suficientemente bien para los fines de este trabajo. Cabe mencionar que los resultados obtenidos, a pesar de no implementar el proceso de rectificación, son bastante buenos y aunque las imágenes presentan la distorsión de barril esto no efacta significativamente el funcionamiento del sistema.

9.2 Etapa de Pre-procesamiento

La etapa de pre-procesamiento está compuesta por cuatro fases de procesamiento de imágenes. Filtro de mediana, correspondencia estéreo, segmentación y filtrado morfológico. Estas fases se aplican consecutivamente a las imágenes capturadas con el fin de extraer la región de la mano presente en la escena, para posteriormente extraer la trayectoria del centroide y llevar a cabo el reconocimiento del gesto realizado. Esta etapa se explica más detalladamente en el capítulo 6.

A b C d

Figura 94. Resultados del filtro de medianta aplicado a la imagen *lenna*. a) imagen original. b) imagen con ruido sal y pimienta al 5%. c) imagen filtrada en FPGA. d) imagen filtrada en MATLAB. Fuente autor.

El filtro de mediana es aplicado a las imágenes capturadas, con el fin de reducir el ruido producido por la cámara en el proceso de adquisición. Según (Ahmed, Elatif, & Alser, 2015) el máximo rendimiento del filtro de mediana se obtiene para una ventana de tamaño 3X3 bajo diferentes densidades de ruido de tipo sal y pimienta. Por consiguiente, en este trabajo se implementó en hardware un filtro de mediana con un tamaño de ventana 3X3.

Como se aprecia en las imágenes de la Figura 94, el resultado del filtro de mediana en la implementación en hardware Figura 94 (c) es igual a la implementación en software Figura 94 (d). Sin embargo, para evaluar de forma cuantitativa el grado de similitud de ambos resultados con respecto la imagen original Figura 94 (a) se emplea la PSNR (Relación Señal a Ruido de Pico).

La PSNR es una métrica ampliamente utilizada en la comparación de imágenes y está definida como se muestra en la ecuación (35). En esta, MSE representa el error cuadrático medio entre la imagen original o de referencia R y la imagen a comprar Q y n es la cantidad de bits empleados para representar un pixel.

$$MSE = \frac{1}{N \times M} \sum_{i=1}^{M} \sum_{j=1}^{N} (R(i,j) - \mathrm{Q(i,j)}^2$$
$$PSNR = 10 \log_{10} \left(\frac{2^n - 1}{MSE} \right) \tag{35}$$

Otra característica importante a comprar entre las dos implementaciones del filtro de mediana, es el tiempo de ejecución requerido por dicho filtro. Para obtener el tiempo de ejecución de los algoritmos en MATLAB se emplea la función *timeit*. El tiempo de ejecución en hardware está determinado por la latencia del filtro como se muestra en la ecuación (36), donde T_{PCLK} es el periodo de pixel, M es el ancho de la imagen y N el alto de la misma. Para la implementación en hardware se emplea una frecuencia de pixel de $100MHz$.

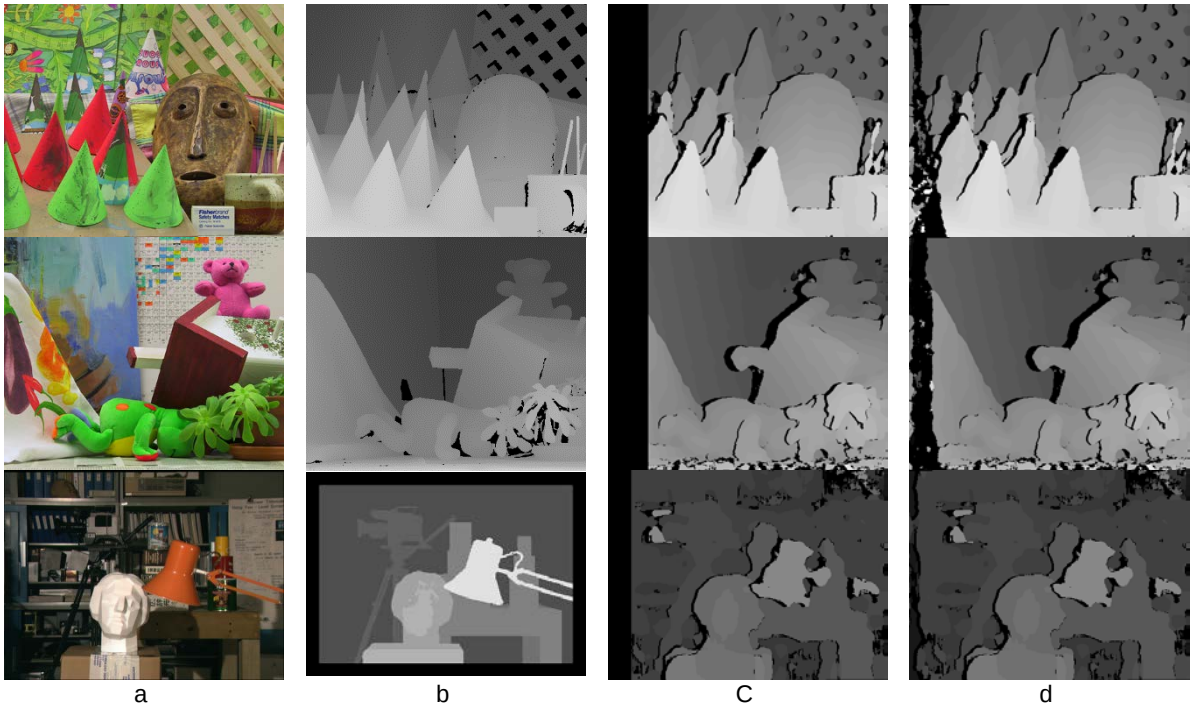
$$t_{FPGA} = T_{PCLK} \times (M + 11 + M \times N) \tag{36}$$

En la Tabla 16 se presentan los resultados de comparación empleando la PSNR y el tiempo de ejecución requerido por ambas implementaciones. Como se observa en la tabla, la PSRN es de 36.1848 dB para las dos imágenes resultantes del proceso de filtrado, lo que indica que la implementación en hardware obtiene exactamente los mismos resultados que la ejecución en MATLAB. Sin embargo, el tiempo de ejecución en hardware es de solo 2.63 *ms* mientras que en software requiere 4.2 *ms*. Esto implica que la implementación en hardware es aproximadamente 1.6 veces más rápida que su contraparte en software.

Tabla 16. Resultados del filtro de mediana implementado en Software y Hardware.

Implementación	PSNR	TE
Hardware	36.1848 dB	2.63 *ms*
MATLAB	36.1848 dB	4.2 *ms*

Después de filtrar las imágenes capturadas, la segunda fase del pre-procesamiento del gesto consiste en calcular el mapa de disparidad de las imágenes estéreo capturadas. El método para calcular el mapa de disparidad en este trabajo, está conformado por la transformada Census dispersa al 50% y la SHD. De acuerdo con (Fife, 2011; Fife & Archibald, 2013), los mejores resultados del mapa de disparidad se obtienen cuando se selecciona un tamaño de ventana de Census de $W_c = 7$ y un tamaño de ventana de SHD de $W_h = 13$. Para evaluar el funcionamiento del módulo desarrollado en hardware se emplean las imágenes estéreo *cones, Teddy* y *Tsukuba* obtenidas de la base de datos de Middlebury (Scharstein & Szeliski, 2002, 2003).



a b C d

Figura 95. Resultados de ejecución del algoritmo de correspondencia estéreo implementado en Software y Hardware. a) imagen izquierda del par estéreo empleado: *cones, teddy* y *tsukuba*. b) *ground-truth* c) mapa de disparidad en MATLAB. d) mapa de disparidad en FPGA. Fuente autor.

En la Figura 95 se presentan las imágenes estéreo empleadas para evaluar el comportamiento del algoritmo de correspondencia estéreo implementado en hardware. Para cada una de las imágenes estéreo empleadas se calcula el mapa de disparidad tanto en software como en hardware. Las imágenes de la columna (a) corresponden a la imagen izquierda del par estéreo. Las imágenes de la columna (b) corresponden al *ground-truth* del mapa de disparidad de cada una de las imágenes empleadas y las columnas (c) y (d) corresponden al mapa de disparidad obtenido en software y hardware respectivamente. En esta figura se puede observar que los mapas de disparidad calculados en software y hardware son similares lo que indica que la arquitectura de hardware implementada cumple perfectamente con las especificaciones del algoritmo de correspondencia estéreo seleccionado.

En la Tabla 17 se presenta el porcentaje de pixeles coincidentes de los mapas de disparidad calculados en este trabajo con respecto a los mapas de disparidad *ground-truth*. En dicha tabla se puede observar que la implementación software tiene un porcentaje de efectividad bajo comparado con los resultados obtenidos en hardware. Esto se debe a que los pixeles de la franja izquierda del mapa de disparidad obtenido en software son tomados como oclusión por el algoritmo LRCC (ver Figura 95 (c)). Sin embrago, al comprar los dos mapas de disparidad, sin tener en cuenta dichos pixeles de oclusión, se obtiene como resultado que no existe diferencia alguna, por consiguiente, se verificó que la implementación hardware perfectamente equivalente a la implementación en software. Por otro lado, el mapa de disparidad obtenido en este trabajo posee un porcentaje de efectividad similar a los resultados obtenidos por otros trabajos en la literatura. Por ejemplo, (Fife, 2011) obtiene como resultados: *cones* 87.76%, *teddy* 87.03% y *Tsukuba* 90.92%. Esto indica que el rendimiento del algoritmo de correspondencia estéreo implementado en hardware presenta un resultado bastante bueno comparado con métodos similares reportados en la literatura.

Tabla 17. Porcentaje de pixeles correctamente coincidentes del algoritmo de correspondencia estéreo implementado en Software y Hardware.

Imágenes de prueba	Matlab	Hardware
Cones	80.8071	88.0041
Teddy	80.6516	87.1227
Tsukuba	87.7729	90.7899

Tabla 18. Tiempo de ejecución de la correspondencia estéreo en software y en hardware.

Imágenes de prueba	Matlab	Hardware
Cones	375.6181 s	1.73 ms
Teddy	378.5892 s	1.73 ms
Tsukuba	127.6772 s	1.14 ms

En la Tabla 18 se presenta el tiempo de ejecución requerido por el algoritmo de correspondencia estéreo para calcular el mapa de disparidad. De acuerdo con los resultados obtenidos, se observa que la implementación en hardware es sumamente rápida ya que se encuentra en el orden de los *ms* mientras que la implementación en software supera los 100 segundos. Los resultados de tiempo de ejecución en software fueron calculados mediante la función *timeit* de MATLAB. El tiempo de ejecución en hardware está determinado por la latencia del módulo de correspondencia estéreo como se observa en la ecuación (37), siendo M el ancho de la imagen y d los niveles de disparidad. Para calcular el tiempo de ejecución en hardware se emplearon los paramentos expuestos en la Tabla 19.

$$t_{FPGA} = T_{PCLK} \times \left[\frac{(M+1)(W_c + W_h + 2)}{2} - 2M + \log_2\left(\frac{W_c^{\,2}}{2}\right) + 2d + 11 + M \times N \right] \qquad (37)$$

Tabla 19. Parámetros utilizados para calcular el tiempo de ejecución en hardware.

Imágenes de prueba	Tamaño	M	W_c	W_h	d	T_{pclk}
Cones	450x375	450	7	13	64	10 *ns*
Teddy	450x375	450	7	13	64	10 *ns*
Tsukuba	384x288	384	7	13	32	10 *ns*

Se llevaron a cabo varias pruebas experimentales con diferentes tamaños de ventana de Census y ventana de SHD. Esto fue necesario debido a que, como se mencionaba anteriormente, en este trabajo no se realizó la implementación del proceso de rectificación en la etapa de captura. De acuerdo con las pruebas realizadas, se encontró que con valores de $W_c = 7$, $W_h = 29$ y $d = 48$ se obtiene un mapa de disparidad lo suficientemente bueno para el desarrollo de este trabajo. En la Figura 96 se presenta el mapa de disparidad resultante empleado las imágenes capturadas por el sistema de reconocimiento con los parámetros antes mencionados.

A b c

Figura 96. Mapa de disparidad calculado para una imagen durante la realizacion del gesto. a) captura de la imagen izquierda. b) captura de la imagen derecha. c) mapa de disparidad calculado por el sistema desarrollado. Fuente autor.

Una vez calculado el mapa de disparidad, el siguiente paso en la etapa de pre-procesamiento del gesto es la segmentación. Para extraer la región de la mano de la escena se emplean dos tipos de segmentación. La segmentación de color de piel y la segmentación del mapa de disparidad. La primera permite extraer las regiones de la imagen con un color similar al de la piel, mientras que la segunda permite extraer solo los objetos que se encuentran a cierta distancia de la cámara.

En la Figura 97 se presentan los resultados obtenidos de la segmentación de color de piel. La imagen (a) corresponde a la imagen izquierda del par estéreo capturado por el sistema. Las imágenes (b) y (c) son los resultados de la segmentación obtenidos en software y hardware respectivamente. Como se observa, el resultado de la segmentación en hardware es exactamente igual al resultado en software. Por lo tanto, se puede afirmar que el hardware desarrollado cumple adecuadamente con las especificaciones de diseño planteadas para este algoritmo.

A B c

Figura 97. Segmentacion por color de piel. (a) Imagen izquierda del par estéreo capturado. (b) segmentacion por color de piel en MATLAB. (c) Segmentacion por color de piel en FPGA. Fuente autor.

En la Figura 98 se presenta el resultado obtenido de la segmentación del mapa de disparidad. La figura Figura 98 (a) corresponde a la imagen izquierda del par estéreo capturado por el sistema, mientras que las imágenes en las Figuras 98 (b) y (c) corresponden al resultado de la segmentación del mapa de disparidad obtenido en software y hardware respectivamente. Al igual que la segmentación por color de piel, la segmentación del mapa de disparidad en hardware y software son iguales. Por consiguiente, el hardware desarrollado corresponde al algoritmo de segmentación especificado.

A B c

Figura 98. Segmentacion del mapa de disparidad. (a) mapa de disparidad calculado a partir de las imágenes estéreo capturadas. (b) segmentacion del mapa de disparidad en MATLAB. (c) segmentacion del mapa de disparidad en FPGA. Fuente autor.

Como se observó en las imágenes anteriores la segmentación en hardware y software presentan los mismos resultados. Sin embargo, el tiempo de ejecución en hardware es mucho menor al requerido para la ejecución en software. En la Tabla 20 se presentan los resultados de tiempo de ejecución. En mencionada tabla, se observa que en software la segmentación del mapa de disparidad tiene un tiempo de ejecución menor con respecto a la segmentación por color de piel. Por otro lado, la implementación en hardware de las segmentaciones tardan exactamente el mismo tiempo en ejecutarse. Para calcular el tiempo de ejecución en software se emplea la función *timeit* de MATLAB. El tiempo de ejecución en hardware se calcula empleando la ecuación (38), donde M y N son el ancho y alto de la imagen y T_{PCLK} El periodo de pixel del sistema de procesamiento.

$$t_{FPGA} = T_{PCLK}\left[M \times N + 2\right] \tag{38}$$

Tabla 20. Tiempo de ejecución de la segmentación del mapa de disparidad en software y en hardware.

Segmentación	Matlab	Hardware
Color de Piel	41.7 *ms*	3.07 *ms*
Mapa de disparidad	4.8 *ms*	3.07 *ms*

El resultado final del proceso de segmentación se obtiene intersectando las segmentaciones calculadas por color de piel y del mapa de disparidad. Esto permite eliminar aquellos objetos o regiones de la imagen que pueden afectar el correcto funcionamiento del sistema. Sin embargo, el resultado de segmentación suele tener ruido el cual debe filtrarse para obtener la región de la mano de forma más limpia. Para llevar a cabo este proceso de filtrado se aplica la operación morfológica de apertura.

A B C

Figura 99. Operación morfológica de apertura. (a) imagen segmentada. (b) operación morfológica de apertura en MATLAB. (c) operación morfológica de apertura en FPGA. Fuente autor.

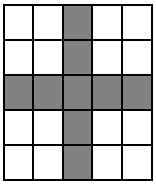


Figura 100. Elementro estructurante empleado en la operación morfológica de apertura. Fuente autor.

En la Figura 99 a) se presenta el resultado del proceso de filtrado morfológico aplicado a la imagen segmentada. Como se puede observar, el resultado obtenido del proceso de segmentación es muy bueno. Sin embargo, existe una cantidad considerable de ruido que debe ser eliminado. Para esto, se llevaron a cabo varias pruebas experimentales con diferentes tamaños de ventanas de procesamiento, obteniendo el resultado más favorable para una ventana de tamaño 11x11 empleando el elemento estructurante de la Figura 100. Las imágenes mostradas en la Figura 99 (b) y la Figura 99 (c) muestran el resultado obtenido del filtrado morfológico de apertura con el tamaño de ventana seleccionado para la implementación en software y hardware, respectivamente. En dichas figuras se puede apreciar que el ruido es eliminado en su totalidad aunque la forma de la mano tiende a deformarse incrementando su area. Esta deformación no resulta critica ya que en este

trabajo no se emplean características basadas en la apariencia por lo tanto el resultado de la operación morfológica es aceptable.

Adicionalmente, en las Figura 99 b) y c) se puede observar que el resultado de la operación morfológica de apertura obtenida en hardware es igual al resultado de la operación en software. Sin embargo, el tiempo de ejecución en hardware es mucho menor que la ejecución en software como se muestra en la Tabla 21. Para calcular el tiempo de ejecución en software se emplea la función *timeit* de MATLAB. El tiempo de ejecución en hardware se calcula empleando la ecuación (39), donde M y N son el ancho y alto de la imagen, W_o el tamaño de la ventana del filtro y T_{PCLK} el periodo de pixel del sistema de procesamiento.

$$t_{FPGA} = T_{PCLK}\left[M \times N + 2 \times \left(\frac{(W_o + 1)(M + 1)}{2} - M + \log_2\left(W_o^{\,2}\right) + 2 \right) \right] \tag{39}$$

Tabla 21. Tiempo de ejecución de la operación morfológica de apertura en software y en hardware.

Matlab	Hardware
1.2633 s	42.050 ms

9.3 Extracción de Características

La etapa de extracción de características está compuesta por cuatro fases: calculo del centroide, suavizado de la trayectoria del gesto, detección del gesto y extracción y cuantización de la orientación; cada una de estas fases se explica en el capítulo 7. En las Figuras 116 a 122 se presenta el resultado de la extracción de características obtenido por el sistema de reconocimiento durante la realización de cada gesto.

De acuerdo a lo explicado en la sección 7.4 en este trabajo se emplea la orientación de dos valores de centroide consecutivos como única característica que describe cada gesto. Por lo tanto, en cada imagen procesada se obtiene un símbolo discreto (1-8) dependiendo de la trayectoria descrita por la mano en ese instante de tiempo. La cantidad de símbolos extraídos para un gesto depende de su duración, la cual se ve representada en la cantidad de *frames* procesados.

En la Figura 101 se presentan las características obtenidas para el gesto “Adelante”. En dicha figura se observa que el gesto tiene una duración aproximada de 45 *frames* y los símbolos predominantes son (7, 6, 2 y 3).

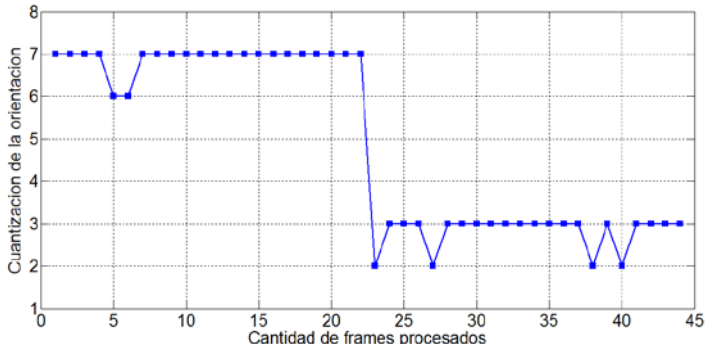


Figura 101. Secuencia de simbolos discretos que representan el gesto “Adelante”. Fuente autor.

En Figura 102 se presentan las características obtenidas para el gesto “Atrás”. En esta figura se observa que el gesto tiene una duración aproximada de 40 *frames* y los símbolos predominantes son (2, 1, 3, 8, 7 y 6).

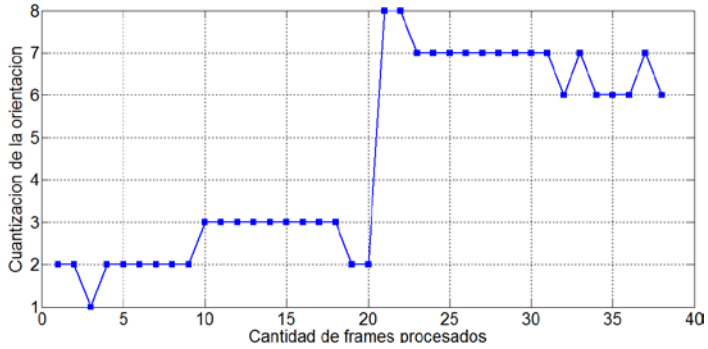


Figura 102. Secuencia de simbolos discretos que representan el gesto “Atras”. Fuente autor.

En Figura 103 se presentan las características obtenidas para el gesto “Derecha”. En dicha figura se observa que el gesto tiene una duración aproximada de 45 *frames* y los símbolos predominantes son (5,6 y1).

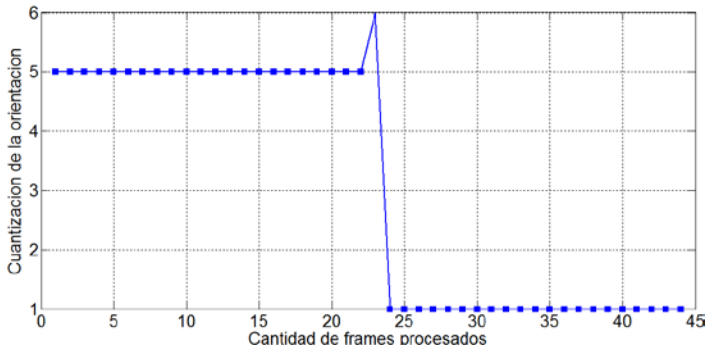


Figura 103. Secuencia de simbolos discretos que representan el gesto “Derecha”. Fuente autor.

En Figura 104 se presentan las características obtenidas para el gesto “Izquierda”. En esta figura se observa que el gesto tiene una duración aproximada de 50 *frames* y los símbolos predominantes son (1, 7, 6, 5 y 4).

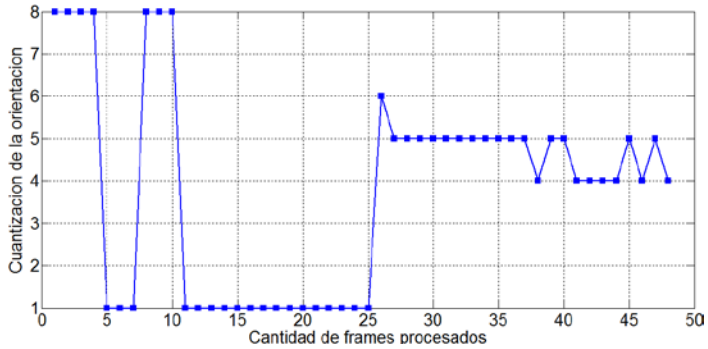


Figura 104. Secuencia de simbolos discretos que representan el gesto “Izquierda”. Fuente autor.

En Figura 105 se presentan las características obtenidas para el gesto “Rotar Derecha 90º”. En esta figura se observa que el gesto tiene una duración aproximada de 60 *frames*. Este gesto emplea todos los símbolos (1-8).

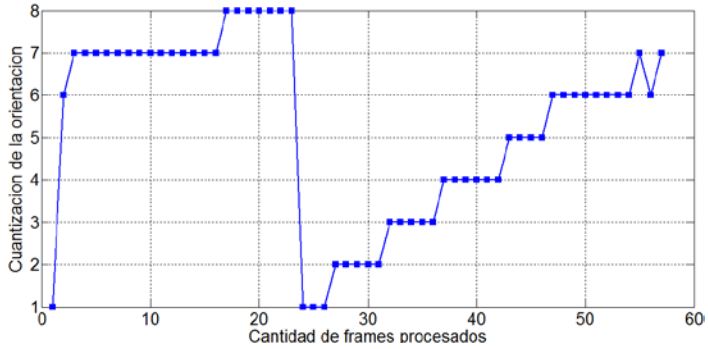


Figura 105. Secuencia de simbolos discretos que representan el gesto “Rotar Derecha 90º”. Fuente autor.

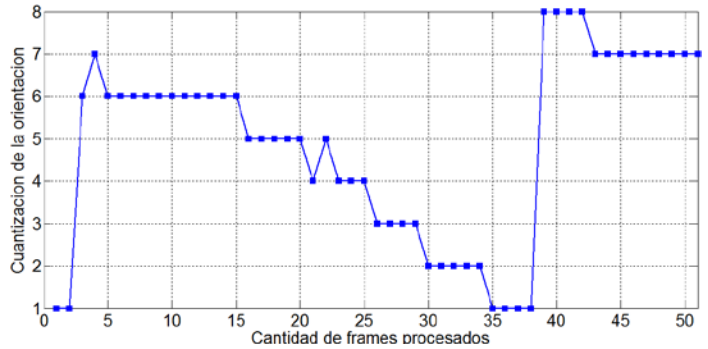


Figura 106. Secuencia de simbolos discretos que representan el gesto “Rotar Izquierda 90º”. Fuente autor.

En Figura 106 se presentan las características obtenidas para el gesto “Rotar Izquierda 90º”. En dicha figura se observa que el gesto tiene una duración aproximada de 51 *frames*. Este gesto, al igual que el gesto anterior emplea todos los símbolos discretos para su descripción, sin embargo el orden en que aparecen los símbolos a media que se realiza el gesto es diferente.

En Figura 107 se presentan las características obtenidas para el gesto “Stop”. En esta figura se observa que el gesto tiene una duración aproximada de 80 *frames*. Este gesto es el que requiere una mayor cantidad de *frames* para ser representado y requiere de todos los símbolos a excepción del 3 para su representación.

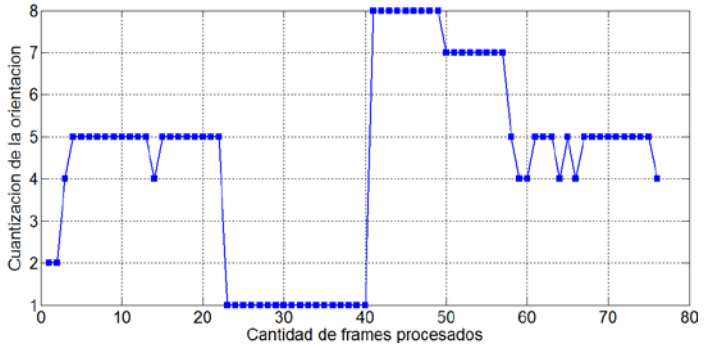


Figura 107. Secuencia de simbolos discretos que representan el gesto “Stop”. Fuente autor.

De acuerdo con las características obtenidas para cada uno de los gestos se puede observar que existe una diferencia significativa en la realización de cada uno de ellos, lo que permite al clasificador basado en HMMs hacer el proceso de reconocimiento con mayor facilidad.

9.4 Reconocimiento de Gestos

Como se explicó en el capítulo 8, la validación cruzada k-fold permite seleccionar adecuadamente el número de estados ocultos de los HMMs empleados en la etapa de reconocimiento. Para llevar a cabo la validación cruzada se construyó una base de datos compuesta por 1260 muestras correspondientes a los siete gestos empleados en este trabajo. Las muestras de la base de datos se obtuvieron de tres personas diferentes, tomando 60 muestras por persona para cada uno de los gestos.

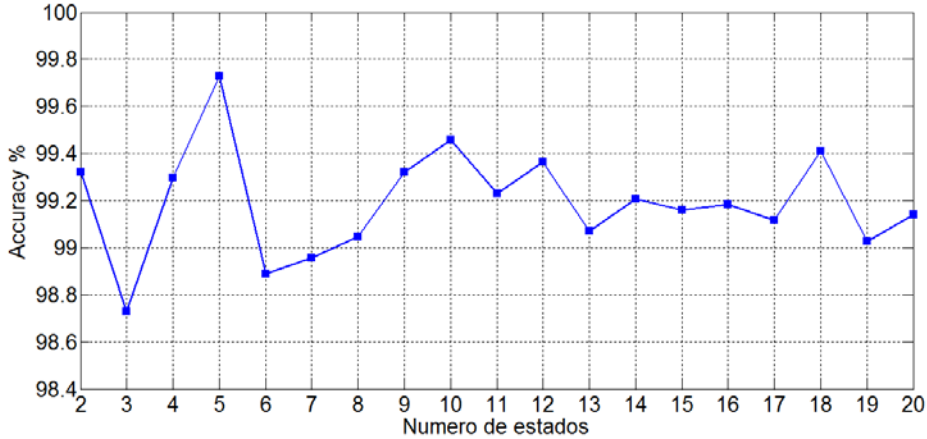


Figura 108. Resultados de la validacion cruzada 10-fold para determinar la exactitud del clasificador basado en HMMs con diferente numero de estados ocultos. Fuente autor.

En la Figura 108 se presentan los resultados de la validación cruzada 10-fold realizada al clasificador basado en HMMs. La validación se realizó para diferentes tamaños del clasificador empleando desde 2 hasta 20 estados ocultos. Como se observa en esta figura, el clasificador compuesto por 5 estados presenta el máximo rendimiento, obteniendo un 99,7% de exactitud. Por otra parte, el clasificador tiene el rendimiento más bajo con 98,7% cuando se emplean 3 estados ocultos. Teniendo en cuenta los resultados obtenidos mediante el método de validación cruzada seleccionado, la implementación del sistema de reconocimiento basado en HMMs se lleva a cabo empleando 5 estados ocultos.

9.5 Implementación del Sistema de Reconocimiento de Gestos en FPGA-SoC

El sistema de reconocimiento de gestos fue implementado en la tarjeta de desarrollo DE1-SOC, la cual posee un dispositivo FPGA-SoC Cyclone V. En la Figura 109 se presenta la implementación del sistema de reconocimiento de gestos. Como se observa, las cámaras y el modulo Bluetooth RN42 están conectados a través de los puertos GPIO de la tarjeta de desarrollo DE1-SoC. El puerto VGA se emplea para depurar el funcionamiento del sistema mediante la visualización de las imágenes procesadas en un monitor. Además, los pulsadores, switches y LED se emplean para interactuar con el sistema de reconocimiento como se muestra en la Tabla 22.

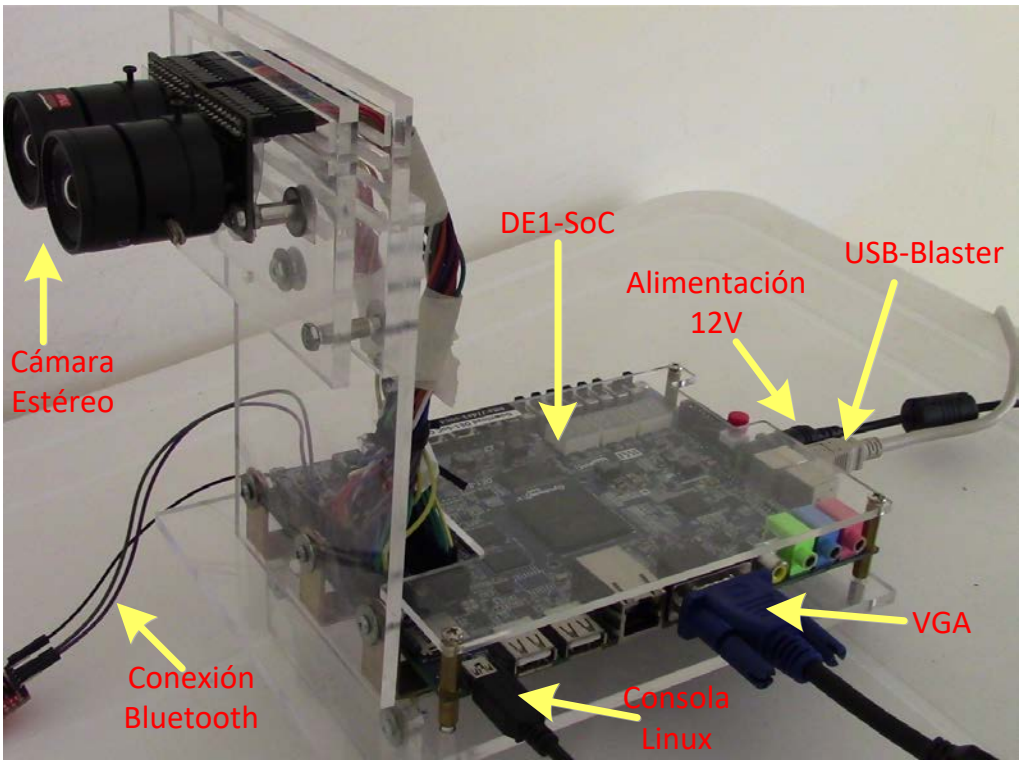


Figura 109. Implementacion del sistema de reconocimiento de gestos. Fuente autor.

Tabla 22. Descripción de la funcionalidad de los botones, switches y leds de la tarjeta De1-SoC en el sistema de reconocimiento de gestos.

Nombre	Descripción
KEY[0]	Reset del sistema de procesamiento de imágenes desarrollado
KEY[1]	Configuración de la cámaras para detener el ajuste automático de exposición a la luz
KEY[2]	Pausa la captura de imágenes
KEY[3]	Inicia la captura de imágenes
SW[0]	Habilita la captura de imágenes
SW[9:7]	Selección de la imagen a visualizar por puerto VGA. "000" Imagen derecha del sistema estéreo "001" Imagen izquierda del sistema estéreo "010" Mapa de disparidad "011" Segmentación por color de piel "100" Segmentación del mapa de disparidad "101" Segmentación después de aplicar la operación morfológica de apertura
LEDR[4]	Detección de gesto principal.

El sistema de reconocimiento de gestos diseñado se implementa en la tarjeta de desarrollo DE1-SoC empleando el software de síntesis de Altera Quartus II v 14.1. En la Tabla 23 se presentan los resultados del proceso de síntesis. Como se observa, el módulo de correspondencia estéreo consume cerca del 37% de los recursos lógicos y el 16 % de los bits de memoria disponibles en el dispositivo. La etapa de captura consume el 35% de los bits de memoria. Esto se debe al módulo de sincronización, el cual debe poseer la suficiente memoria para sincronizar las imágenes estéreo durante la captura. Los demás módulos de hardware no superan el 10% de uso de los recursos de la parte FPGA. En total, el sistema desarrollado hace uso de cerca del 60% de los recursos disponibles, mas un 10% adicional utilizado por el hardware de depuración, el cual no hace parte del sistema de reconocimiento.

Tabla 23. Recursos del dispositivo FPGA-SoC utilizados por el sistema de reconocimiento de gestos.

	ALMs		Registros		Bits de Memoria		HPS	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
Recursos disponibles	32070	-	64140	-	4065280	-	1	-
Etapa de Captura	1010	3.1	1371	2.2	1048576	25.8	0	0
Filtro de mediana	2298	7.1	3876	6.1	61056	1.5	0	0
Correspondencia estéreo	11977	37.2	30897	48.2	661556	16.3	0	0
Segmentación	49	0.2	27	0.04	0	0	0	0
Operación morfológica de apertura	1350	4.1	3259	5.4	101888	25	0	0
Calculo de centroide	1013	3.1	921	1.4	0	0	0	0
Filtro de media móvil	65	0.2	194	0.3	0	0	0	0
Detección del gesto	301	1	562	0.8	0	0	0	0
Extracción de características y reconocimiento de gestos	911	2.8	1409	2.1	14336	0.3	1	100%
Hardware de depuración	2967	9.2	3303	5.1	454581	11.2	0	0
Total de recursos consumidos	21941	68	45819	71.4	2341913	57.6	1	100%

El sistema de reconocimiento de gestos se desarrolló mediante una descripción paramétrica en lenguaje VHDL. Esta descripción permite seleccionar cualquier tamaño de

imágenes a procesar así como otras características de funcionamiento del sistema. Es importante tener en cuenta que la descripción paramétrica se ve reflejada en el proceso de síntesis, por lo cual existen limitantes en la cantidad utilizada de recursos del dispositivo según la configuración requerida. El sistema realiza el procesamiento de las imágenes al mismo tiempo que son capturadas por la cámara.

El tiempo total de procesamiento de una imagen completa está determinado por la ecuación (40) empleando los siguientes parámetros: imágenes VGA (640x480), ventana de Census $W_c = 7$, ventana de Hamming $W_h = 29$, niveles de disparidad $d = 48$, ventana de la operación morfológica $W_o = 11$ y frecuencia de pixel de $100MHz$. De acuerdo con las especificaciones de funcionamiento del sistema se obtiene un tiempo de procesamiento total de 3.252940 ms, lo que implica que el sistema está en la capacidad de procesar a una velocidad de aproximadamente 307 fps.

$$
\begin{aligned}
L_{me} &= M + 11 \\
L_{sm} &= \frac{(M+1)(W_c + W_h + 2)}{2} - 2M + \log_2\left(\frac{W_c^{\,2}}{2}\right) + 2d + 11 \\
L_{op} &= 2 \times \left[\frac{(M+1)(W_o + 1)}{2} - M + \log_2\left(W_o^{\,2}\right) + 2\right] \\
L_{se} &= 2 \\
t_{total} &= T_{PCLK} \times \left[M \times N + L_{me} + L_{sm} + L_{op} + L_{se}\right]
\end{aligned}
\tag{40}
$$

Aunque la velocidad de procesamiento que sistema desarrollado puede llegar a alcanzar es bastante alta, en este trabajo se empleó una velocidad de procesamiento de 30fps, debido principalmente a las limitaciones en la velocidad de captura de las cámaras empleadas este trabajo.

El sistema de reconocimiento de gestos completo implementado en el dispositivo FPGA-SoC, fue puesto a prueba controlando las acciones de un robot LEGO NXT. La prueba fue llevada a cabo mediante la ejecución de cada gesto 100 veces para evaluar el comportamiento del sistema con gestos diferentes a los empleados en el proceso de entrenamiento y validación utilizado. El resultado de dicha prueba se presenta en la matriz de confusión de la Tabla 24.

De acuerdo con la matriz de confusión obtenida en el proceso de evaluación, se puede determinar que el sistema de reconocimiento de gestos diseñado e implementado en el dispositivo FPGA-SoC tiene una tasa total de reconocimiento del 99,7%. El sistema presenta excelentes resultados con tasas de reconocimiento superiores al 99%. Adicionalmente, se observa que el sistema de reconocimiento tiende a confundir el gesto “Derecha” con el gesto “Stop” y el gesto “Rotar Derecha 90º” con el gesto “Stop”. Sin

embargo, esta confusión no afecta demasiado el rendimiento total del sistema de reconocimiento.

Tabla 24. Matriz de confusión del sistema de reconocimiento implementado.

	Adelante	**Atrás**	**Derecha**	**Izquierda**	**Rotar Derecha 90º**	**Rotar Izquierda 90º**	**Stop**
Adelante	100	0	0	0	0	0	0
Atrás	0	100	0	0	0	0	0
Derecha	0	0	99	0	0	0	1
Izquierda	0	0	0	100	0	0	0
Rotar Derecha 90º	0	0	0	0	99	0	1
Rotar Izquierda 90º	0	0	0	0	0	100	0
Stop	0	0	0	0	0	0	100

10 Conclusiones y Trabajos Futuros

En este trabajo se presenta el desarrollo de un sistema de reconocimiento de gestos de las manos, implementado en un dispositivo FPGA-SoC y empleando visión estereoscópica, con aplicación en la interacción Hombre-Robot. El sistema de reconocimiento desarrollado está compuesto por cuatro etapas principales: captura, pre-procesamiento, extracción de características y reconocimiento del gesto realizado. El sistema es implementado en un solo chip de la siguiente manera. Las primeras tres etapas del sistema son implementadas en la zona FPGA del dispositivo, mientras que la etapa de reconocimiento se implementa en la zona HPS mediante una aplicación desarrollada para Linux. El sistema está en la capacidad de reconocer siete gestos de las manos y controlar las acciones de un robot LEGO NXT de acuerdo con el gesto identificado. Los gestos empleados en este trabajo son realizados empleando la mano derecha únicamente. El sistema de reconocimiento identifica los gestos con una tasa de reconocimiento superior al 99 %.

La metodología de diseño en hardware de los algoritmos de procesamiento de imágenes del sistema de reconocimiento de gestos basada en el desarrollo de dichos algoritmos en software, es sumamente útil ya que permite al diseñador conocer en profundidad su comportamiento y facilitar de esta forma su posterior mapeo en hardware. Además, la implementación en software de los algoritmos empleados en este trabajo se toma como modelo de referencia para la verificación de resultados. Esto permite evaluar de forma efectiva si los resultados obtenidos por el hardware desarrollado corresponden a los resultados esperados.

La captura y procesamiento de las imágenes estéreo, realizadas por las tres primeras etapas del sistema de reconocimiento, presentan un alto poder de procesamiento logrando extraer la región de la mano, para cada una de las imágenes capturadas, en un tiempo inferior a 3.5 ms. Esto permite realizar un procesamiento de imágenes en tiempo real con una velocidad de procesamiento superior a los 300fps. La implementación en hardware de los algoritmos de procesamiento de imágenes utilizados en este trabajo tiene un rendimiento muy superior a la ejecución en software de dichos algoritmos.

El algoritmo de correspondencia estéreo empleado en este trabajo presenta excelentes resultados para el cálculo del mapa de disparidad teniendo en cuenta que las imágenes estéreo no son rectificadas en la etapa de captura. A pesar de la elevada complejidad computacional requerida por el algoritmo de correspondencia estéreo, es importante resaltar que el cálculo del mapa de disparidad se realiza en tiempo real, durante la captura de las imágenes provenientes de las cámaras utilizadas.

El proceso de validación cruzada *k-fold* es bastante útil para la evaluación del rendimiento de diferentes clasificadores o de diferentes parámetros de un mismo clasificador. En este trabajo se emplearon los HMMs como clasificadores del sistema de reconocimiento de gestos. La validación cruzada *k_fold* permite seleccionar de manera adecuada la cantidad de estados ocultos requeridos por dicho clasificador. Este proceso de entrenamiento y validación se realizó de forma *off-line* empleando el software MATLAB. El entrenamiento de los HMMs se lleva a cabo mediante el algoritmo de Waum-Welch, disponible en el toolbox de HMMs para MATLAB desarrollado por Kevin Murphy. De acuerdo con los resultados obtenidos se establece que el clasificador obtiene su máximo rendimiento cuando se utilizan 5 estados ocultos.

El HPS empleado para implementar la etapa de reconocimiento del sistema desarrollado tiene ventajas en comparación con el uso de procesadores embebidos o *Soft-Cores*. En primer lugar, el sistema HPS es un recurso físico del dispositivo FPGA-SoC, lo cual implica que su utilización no requiere de los recursos lógicos de la FPGA. Esto permite al usuario diseñar hardware de alto rendimiento en la zona reconfigurable del dispositivo, sin destinar elementos lógicos para la síntesis del procesador requerido para la ejecución de software. Por otra parte, el HPS posee dos procesadores que permiten realizar procesamiento multi-hilo de alto rendimiento. Además, el HPS soporta la ejecución de un sistema operativo Linux, esto permite al usuario desarrollar aplicaciones personalizadas con mayor rapidez a diferencia del desarrollo de aplicaciones *bare-metal* requeridas en la mayoría de procesadores *soft-Core*. La única desventaja encontrada en el desarrollo de este trabajo está relacionada con la documentación del sistema HPS. Dicha documentación no explica claramente el funcionamiento de los puentes (*bridges*) entre la zona HPS y la zona FPGA. Esto dificulta un poco la conexión del procesador con módulos personalizados diseñados en la FPGA.

El uso de dispositivos FPGA en aplicaciones de reconocimiento de gestos es una excelente alternativa frente a los sistemas y tecnologías convencionales que realizan tareas similares. El alto rendimiento que puede obtenerse de las FPGAs y su bajo consumo de potencia las convierte en una tecnología ideal para el desarrollo y mejoramiento continuo de sistemas de reconocimiento de gestos en un amplio espectro de aplicaciones de interacción Hombre-Computador.

10.1 Trabajos Futuros

El sistema de reconocimiento de gestos presentado en este trabajo está orientado a la interacción Hombre-Robot empleando los gestos de las manos como principal modalidad de interacción. Aunque el sistema desarrollado tiene un funcionamiento bastante bueno, su interacción con el robot se encuentra un poco limitada. Esta limitación de debe principalmente a la restricción empleada en la realización de los gestos, ya que la correcta identificación de las tres fases que componen un gesto es una tarea bastante compleja de realizar en tiempo real y más aún si esto implica el desarrollo de una arquitectura en

hardware que se encargue de dicha tarea. Por esta razón, se plantea el desarrollo o evaluación de algoritmos que permitan realizar el reconocimiento de gestos teniendo en cuenta una interacción con mayor naturalidad.

El sistema de reconocimiento de gestos desarrollado tiene grandes ventajas sobre los sistemas convencionales, ya que se permite realizar procesamiento con un rendimiento elevado y un bajo consumo de energía gracias al paralelismo propio de las FPGAs. Esto brinda la oportunidad de desarrollar sistemas, no solo de interacción Hombre-Robot, sino en otros campos de aplicación tales como en aplicaciones para personas invidentes, desarrollo de sistemas de interacción para personas con discapacidad auditiva, sistemas de entretenimiento, domótica, entre otras.

En este trabajo se implementó en hardware un algoritmo de correspondencia estéreo empleando la transformada Census y la Suma de Distancias de Hamming. Dicho algoritmo permite calcular el mapa de disparidad en imágenes estéreo. Sin embargo, no se realizó la rectificación de las imágenes capturadas como se sugiere en la literatura. Aunque los resultados obtenidos sin el proceso de rectificación son aceptables, se sugiere como trabajo futuro el desarrollo de una arquitectura que permita realizar dicho proceso de rectificación con el fin de obtener un mapa de disparidad más preciso para su aplicación de forma eficiente en otros escenarios diferentes al reconocimiento de gestos.

Finalmente se propone como trabajo futuro, mejorar el proceso de filtrado en hardware de la imagen segmentada, ya que el filtrado morfológico utilizado tiende a modificar demasiado la forma de la mano y no permite obtener información significativa sobre la postura realizada. Esto permitiría no solo emplear la trayectoria del gesto sino la configuración de la mano con el fin de realizar un reconocimiento de gestos más complejos. Adicionalmente, se propone emplear otro tipo de características diferentes a la orientación y evaluar la posibilidad de implementar dichos algoritmos en hardware de forma adecuada.

Bibliografía

Aggarwal, J. K., & Ryoo, M. S. (2011). Human Activity Analysis: A Review. *ACM Comput. Surv.*, *43*(3), 16:1--16:43. http://doi.org/10.1145/1922649.1922653

Ahmed, E. S. A., Elatif, R. E. A., & Alser, Z. T. (2015). Median filter performance based on different window sizes for salt and pepper noise removal in gray and RGB images. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, *8*(10), 343–352.

Al-Rousan, M., Assaleh, K., & Tala'a, A. (2009). Video-based signer-independent Arabic sign language recognition using hidden Markov models. *Applied Soft Computing Journal*, *9*(3), 990–999. http://doi.org/10.1016/j.asoc.2009.01.002

Althoff, F., Lindl, R., & Walchshäusl, L. (2005). Robust multimodal hand- and head gesture recognition for controlling automotive infotainment systems. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-28844476135&partnerID=40&md5=b97f7e1a69ab0ec57362ebc1117d0e2f

Ambrosch, K., Kubinger, W., Humenberger, M., & Steininger, A. (2009). Flexible hardware-based stereo matching. *EURASIP Journal on Embedded Systems*, *2008*(1), 1–12.

Ambrosch, K., Zinner, C., & Kubinger, W. (2009). Algorithmic Considerations for Real-Time Stereo Vision Applications. In *MVA* (pp. 231–234).

Bailey, D. G. (2011). *Design for embedded image processing on FPGAs*. John Wiley & Sons.

Bansal, M., Saxena, S., Desale, D., & Jadhav, D. (2011). Dynamic gesture recognition using hidden markov model in static background. *IJCSI*.

Banz, C., Hesselbarth, S., Flatt, H., Blume, H., & Pirsch, P. (2010). Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation. *Embedded Computer Systems (SAMOS), 2010 International Conference on*. http://doi.org/10.1109/ICSAMOS.2010.5642077

Bellmore, C., Ptucha, R., & Savakis, A. (2011). Interactive display using depth and RGB sensors for face and gesture control. *Image Processing Workshop (WNYIPW), 2011 IEEE Western New York*. http://doi.org/10.1109/WNYIPW.2011.6122883

Bendaoudi, H., & Khouas, A. (2013). Stereo vision IP design for FPGA implementation of obstacle detection system. *Systems, Signal Processing and Their Applications (WoSSPA), 2013 8th International Workshop on*. http://doi.org/10.1109/WoSSPA.2013.6602352

Benko, H., & Wilson, A. (2009). *DepthTouch: Using Depth-Sensing Camera to Enable Freehand Interactions On and Above the Interactive Surface*. Microsoft. Retrieved from https://www.microsoft.com/en-us/research/publication/depthtouch-using-depth-sensing-camera-to-enable-freehand-interactions-on-and-above-the-interactive-surface/

Bergh, M. Van den, Carton, D., Nijs, R. De, Mitsou, N., Landsiedel, C., Kuehnlenz, K., … Buss, M. (2011). Real-time 3D hand gesture interaction with a robot for understanding directions from humans. *2011 RO-MAN*.

http://doi.org/10.1109/ROMAN.2011.6005195

Bergh, M. Van den, & Gool, L. Van. (2011). Combining RGB and ToF cameras for real-time 3D hand gesture interaction. *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*. http://doi.org/10.1109/WACV.2011.5711485

Berkeley Design Technology, I. (2005). Inside DSP on Tools: FPGA Tools Bridge Gap Between Algorithm and Implementation | www.bdti.com. Retrieved August 26, 2016, from http://www.bdti.com/InsideDSP/2005/06/15/Idsp3

Binh, N. D., Shuichi, E., & Ejima, T. (2005). Real-Time Hand Tracking and Gesture Recognition System. In *Proceedings of International Conference on Graphics, Vision and Image Processing (GVIP-05* (pp. 362–368).

Biswas, K. K., & Basu, S. K. (2011). Gesture recognition using Microsoft Kinect®. *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*. http://doi.org/10.1109/ICARA.2011.6144864

Bonato, V., Sanches, A. K., Fernandes, M. M., Cardoso, J. M. P., do Valle Simões, E., & Marques, E. (2004). A Real Time Gesture Recognition System for Mobile Robots. In *ICINCO (2)* (pp. 207–214).

Breuer, P., Eckes, C., & Müller, S. (2007). Hand gesture recognition with a novel IR time-of-flight range camera-A pilot study. *3rd International Conference, MIRAGE 2007: Computer Vision/Computer Graphics Collaboration Techniques*. Universität Siegen, Germany.

Brown, M. Z., Burschka, D., & Hager, G. D. (2003). Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. http://doi.org/10.1109/TPAMI.2003.1217603

Chen, C. P., Chen, Y. T., Lee, P. H., Tsai, Y. P., & Lei, S. (2011). Real-time hand tracking on depth images. *Visual Communications and Image Processing (VCIP), 2011 IEEE*. http://doi.org/10.1109/VCIP.2011.6115983

Chen, F.-S., Fu, C.-M., & Huang, C.-L. (2003). Hand gesture recognition using a real-time tracking method and hidden Markov models. *Image and Vision Computing*, *21*(8), 745–758. http://doi.org/10.1016/S0262-8856(03)00070-2

Chen, L., Wei, H., & Ferryman, J. (2013). A survey of human motion analysis using depth imagery. *Pattern Recognition Letters*, *34*(15), 1995–2006. http://doi.org/10.1016/j.patrec.2013.02.006

Chen, Z.-H., Kim, J.-T., Liang, J., Zhang, J., & Yuan, Y.-B. (2014). Real-time hand gesture recognition using finger segmentation. *Scientific World Journal*, *2014*. http://doi.org/10.1155/2014/267872

Cho, P. C., Li, C. T., & Chen, W. H. (2012). Implementation of low-cost vision-based gesture recognition systems based on FPGA approach (pp. 329–332). Taichung. http://doi.org/10.1109/is3c.2012.90

Colodro, C., Toledo, J., Martínez, J. J., Garrigós, J., & Ferrández, J. M. (2012). Evaluación de algoritmos de correspondencia estereoscópica y su implementación en FPGA. *Jornadas de Computación Reconfigurable Y Aplicaciones (JCRA)*, 88–93.

Coogan, T., Awad, G., Han, J., & Sutherland, A. (2006). Real time hand gesture recognition including hand segmentation and tracking. *2nd International Symposium*

on Visual Computing, ISVC 2006. Dublin City University, Dublin 9, Ireland. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-33845423550&partnerID=40&md5=ff083608b603eade6aa95ad1e3d783b6

Ding, J., Du, X., Wang, X., & Liu, J. (2010). Improved real-time correlation-based FPGA stereo vision system. *Mechatronics and Automation (ICMA), 2010 International Conference on*. http://doi.org/10.1109/ICMA.2010.5588008

Dunn, P., & Corke, P. (1997). Real-time stereopsis using FPGAs. In W. Luk, P. Y. K. Cheung, & M. Glesner (Eds.), *Field-Programmable Logic and Applications: 7th International Workshop, FPL '97 London, UK, September 1--3, 1997 Proceedings* (pp. 400–409). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/3-540-63465-7_245

Elmezain, M., & Al-Hamadi, A. (2012). LDCRFs-based hand gesture recognition. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 2670–2675). Seoul. http://doi.org/10.1109/ICSMC.2012.6378150

Elmezain, M., Al-Hamadi, A., Appenrodt, J., & Michaelis, B. (2008). A Hidden Markov Model-based continuous gesture recognition system for hand motion trajectory. *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. http://doi.org/10.1109/ICPR.2008.4761080

Elmezain, M., Al-Hamadi, A., Appenrodt, J., & Michaelis, B. (2009). A hidden markov model-based isolated and meaningful hand gesture recognition. *International Journal of Electrical, Computer, and Systems Engineering*, *3*(3), 156–163.

Elmezain, M., Al-Hamadi, A., & Michaelis, B. (2008). Real-time capable system for hand gesture recognition using hidden Markov models in stereo color image sequences. In *16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2008, WSCG'2008* (pp. 65–72). Institute for Electronics, Signal Processing and Communications (IESK), Otto-von-Guericke-University, Magdeburg, Germany. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-63249103687&partnerID=40&md5=300265e495117cb1f03a38fcfb67970d

Elmezain, M., Al-Hamadi, A., Pathan, S. S., & Michaelis, B. (2009). Spatio-temporal feature extraction-based hand gesture recognition for isolated american sign language and arabic numbers (pp. 264–269). Salzburg. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-70450255139&partnerID=40&md5=6d4b8b3cf6216720c935c6ec923790b7

Elmezain, M. O. S. M. (2010). *Hand Gesture Spotting and Recognition Using HMMs and CRFs in Color Image Sequences*. Otto-von-Guericke-Universitat Magdeburg.

Fahn, C.-S., & Chu, K.-Y. (2011). Hidden-Markov-model-based hand gesture recognition techniques used for a human-robot interaction system. *14th International Conference on Human-Computer Interaction, HCI International 2011*. Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei 10607, Taiwan. http://doi.org/10.1007/978-3-642-21605-3_28

Faugeras, O., Hotz, B., Mathieu, H., Viéville, T., Zhang, Z., Fua, P., … Proy, C. (1993). *Real time correlation-based stereo: algorithm, implementations and applications*. INRIA. Retrieved from https://infoscience.epfl.ch/record/176855/files/RR-2013.pdf

Fife, W. S. (2011). *Improved Stereo Vision Methods for FPGA-Based Computing*

Platforms. Brigham Young University. Retrieved from http://scholarsarchive.byu.edu/etd/2745

Fife, W. S., & Archibald, J. K. (2013). Improved Census Transforms for Resource-Optimized Stereo Vision. *IEEE Transactions on Circuits and Systems for Video Technology*. http://doi.org/10.1109/TCSVT.2012.2203197

Freescale Semiconductor. (2013). OV5642.c. Boston, MA, USA. Retrieved from https://github.com/boundarydevices/linux-imx6/blob/boundary-imx_3.0.35_4.1.0/drivers/media/platform/mxc/capture/ov5642.c

Fujimura, K., & Liu, X. (2006). Sign recognition using depth image streams. *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*. http://doi.org/10.1109/FGR.2006.101

Ghotkar, A. S., & Kharate, G. K. (2012). Hand segmentation techniques to hand gesture recognition for natural human computer interaction. *International Journal of Human Computer Interaction (IJHCI)*, *3*(1), 15.

Gonzalez, R. C., & Woods, R. E. (2002). *Digital image processing*. *Prentice Hall* (2d ed.). Prentice Hall.

Grammalidis, N., Goussis, G., Troufakos, G., & Strintzis, M. G. (2001). 3-D human body tracking from depth images using analysis by synthesis. *Image Processing, 2001. Proceedings. 2001 International Conference on*. http://doi.org/10.1109/ICIP.2001.958455

Gregg Hawkes. (2001). Application Note: MicroBlaze and Multimedia Development Board Digital Component Video Conversion 4:2:2 to 4:4:4. Xilinx.

Greisen, P., Heinzle, S., Gross, M., & Burg, A. P. (2011). An FPGA-based processing pipeline for high-definition stereo video. *EURASIP Journal on Image and Video Processing*, *2011*(1), 1–13. http://doi.org/10.1186/1687-5281-2011-18

Gribbon, K. T., Bailey, D. G., & Johnston, C. T. (2005). Design Patterns for Image Processing Algorithm Development on FPGAs. *TENCON 2005 - 2005 IEEE Region 10 Conference*. http://doi.org/10.1109/TENCON.2005.301109

Gribbon, K. T., Bailey, D. G., & Johnston, C. T. (2006). Using design patterns to overcome image processing constraints on FPGAs. *Third IEEE International Workshop on Electronic Design, Test and Applications (DELTA'06)*. http://doi.org/10.1109/DELTA.2006.93

Gudis, E., Wal, G. van der, Kuthirummal, S., & Chai, S. (2012). Multi-Resolution Real-Time Dense Stereo Vision Processing in FPGA. *Field-Programmable Custom Computing Machines (FCCM), 2012 IEEE 20th Annual International Symposium on*. http://doi.org/10.1109/FCCM.2012.15

Guerrero-Balaguera, J. D., & Perez-Holguin, W. J. (2015). FPGA-based translation system from colombian sign language to text. *DYNA*, *82*, 172–181. Retrieved from http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0012-73532015000100022&nrm=iso

Guomundsson, S. A., Larsen, R., Aanaes, H., Pardas, M., & Casas, J. R. (2008). TOF imaging in Smart room environments towards improved people tracking. *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*. http://doi.org/10.1109/CVPRW.2008.4563154

Habermann, A. N. (1972). *Parallel neighbor-sort (or the glory of the induction principle)*. *CMU Computer Science Report (available as Technical report AD-759 248, National Technical Information Service, US Department of Commerce, 5285 Port Royal Rd Sprigfield VA 22151)*. Pittsburgh, Pa. 15213.

Hartley, R. ~I., & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision* (Second). Cambridge University Press, ISBN: 0521540518.

Humenberger, M., Zinner, C., Weber, M., Kubinger, W., & Vincze, M. (2010). A fast stereo matching algorithm suitable for embedded real-time systems. In *Computer Vision and Image Understanding* (Vol. 114, pp. 1180–1202). http://doi.org/10.1016/j.cviu.2010.03.012

Ibarra-Manzano, M. A., Almanza-Ojeda, D. L., Devy, M., Boizard, J. L., & Fourniols, J. Y. (2009). Stereo Vision Algorithm Implementation in FPGA Using Census Transform for Effective Resource Optimization. *Digital System Design, Architectures, Methods and Tools, 2009. DSD '09. 12th Euromicro Conference on*. http://doi.org/10.1109/DSD.2009.159

Isakova, N., Başak, S., & Sönmez, A. C. (2012). FPGA design and implementation of a real-time stereo vision system. *Innovations in Intelligent Systems and Applications (INISTA), 2012 International Symposium on*. http://doi.org/10.1109/INISTA.2012.6247007

Jansen, B., Temmermans, F., & Deklerck, R. (2007). 3D human pose recognition for home monitoring of elderly. *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. http://doi.org/10.1109/IEMBS.2007.4353222

Jena, S. K., Majhi, B., Gupta, A., Sehrawat, V. K., & Khosla, M. (2012). FPGA based Real Time Human Hand Gesture Recognition System. *Procedia Technology*, *6*, 98–107. http://doi.org/http://dx.doi.org/10.1016/j.protcy.2012.10.013

Ji, X., & Liu, H. (2010). Advances in View-Invariant Human Motion Analysis: A Review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*. http://doi.org/10.1109/TSMCC.2009.2027608

Keskin, C., Erkan, A., & Akarun, L. (2003). Real time hand tracking and 3d gesture recognition for interactive interfaces using hmm. *ICANN/ICONIPP*, *2003*, 26–29.

Kolb, A., Barth, E., & Koch, R. (2008). ToF-sensors: New dimensions for realism and interactivity. *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*. http://doi.org/10.1109/CVPRW.2008.4563159

Kollorz, E., Penne, J., Hornegger, J., & Barke, A. (2008). Gesture Recognition with a Time of Flight Camera. *Int. J. Intell. Syst. Technol. Appl.*, *5*(3/4), 334–343. http://doi.org/10.1504/IJISTA.2008.021296

Kurakin, A., Zhang, Z., & Liu, Z. (2012). A real time system for dynamic hand gesture recognition with a depth sensor (pp. 1975–1979). Bucharest. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84869786886&partnerID=40&md5=1064a9303d60d64f0e6abed1bd950251

Lakshmivarahan, S., Dhall, S. K., & Miller, L. L. (1984). Parallel Sorting Algorithms. In M. C. Y. B. T.-A. in Computers (Ed.), (Vol. Volume 23, pp. 295–354). Elsevier.

http://doi.org/http://dx.doi.org/10.1016/S0065-2458(08)60467-2

Le, T. T., Chatelain, F., & Berenguer, C. (2014). Hidden Markov Models for diagnostics and prognostics of systems under multiple deterioration modes. In *European Safety and Reliability Conference (ESREL 2014)* (pp. 1197–1204). Wroclaw, Poland: Taylor & Francis - CRC Press/Balkema. Retrieved from https://hal.archives-ouvertes.fr/hal-01027509

Lee, H. K., & Kim, J. H. (1999). An HMM-Based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(10), 961–973. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-0033210184&partnerID=40&md5=a50b9b21daa293081c3f86e3130ac384

Lee, S. H., SNC, C. P. O., & Siew, W. H. (2011). Real time FPGA implementation of hand gesture recognizer system. In *Proceedings of the 15th WSEAS international conference on Computers* (pp. 217–222). World Scientific and Engineering Academy and Society (WSEAS).

Lin, C.-S., & Hwang, C.-L. (1987). New forms of shape invariants from elliptic fourier descriptors. *Pattern Recognition*, *20*(5), 535–545. http://doi.org/http://dx.doi.org/10.1016/0031-3203(87)90080-X

Lin, C.-S., & Jungthirapanich, C. (1990). Invariants of three-dimensional contours. *Pattern Recognition*, *23*(8), 833–842. http://doi.org/http://dx.doi.org/10.1016/0031-3203(90)90130-D

Liu, L., & Shao, L. (2013). Learning Discriminative Representations from RGB-D Video Data. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (pp. 1493–1500). AAAI Press. Retrieved from http://dl.acm.org/citation.cfm?id=2540128.2540343

Liu, N., Lovell, B. C., Kootsookos, P. J., & Davis, R. I. A. (2004). Model structure selection & training algorithms for an HMM gesture recognition system. In F. Kimura & H. Fujisawa (Eds.), (pp. 100–105). Tokyo. http://doi.org/10.1002/0471648272.ch7

Liu, X. L. X., & Fujimura, K. (2004). Hand gesture recognition using depth data. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.* (pp. 529–534). Seoul. http://doi.org/10.1109/AFGR.2004.1301587

Malima, A., Özgür, E., & Çetin, M. (2006). A fast algorithm for vision-based hand gesture recognition for robot control. In *2006 IEEE 14th Signal Processing and Communications Applications* (Vol. 2006). Faculty of Engineering and Natural Sciences, Sabanci University, Tuzla, Istanbul, Turkey. http://doi.org/10.1109/SIU.2006.1659822

Manresa, C., Varona, J., Mas, R., & Perales, F. J. (2005). Hand Tracking and Gesture Recognition for Human-Computer Interaction. *ELCVIA Electronic Letters on Computer Vision and Image Analysis; Vol 5, No 3: Special Issue on Articulated Motion & Deformable Objects - August 2005*. Retrieved from http://elcvia.cvc.uab.es/article/view/109/106

Mattoccia, S. (2013). Stereo Vision Algorithms for FPGAs. *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. http://doi.org/10.1109/CVPRW.2013.96

Mitra, S., & Acharya, T. (2007). Gesture recognition: A survey. *IEEE Transactions on*

Systems, Man and Cybernetics Part C: Applications and Reviews, *37*(3), 311–324. http://doi.org/10.1109/tsmcc.2007.893280

Miyajima, Y., & Maruyama, T. (2003). A Real-Time Stereo Vision System with FPGA. In P. Y. K. Cheung & G. A. Constantinides (Eds.), *Field Programmable Logic and Application: 13th International Conference, FPL 2003, Lisbon, Portugal, September 1-3, 2003 Proceedings* (pp. 448–457). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-540-45234-8_44

Mo, Z., & Neumann, U. (2006). Real-time Hand Pose Recognition Using Low-Resolution Depth Images. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. http://doi.org/10.1109/CVPR.2006.237

Moeslund, T. B., Hilton, A., & Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, *104*(2–3), 90–126. http://doi.org/http://dx.doi.org/10.1016/j.cviu.2006.08.002

Moni, M. A., & Shawkat Ali, A. B. M. (2009). HMM based hand gesture recognition: A review on techniques and approaches (pp. 433–437). Beijing. http://doi.org/10.1109/iccsit.2009.5234536

Muñoz-Salinas, R., Medina-Carnicer, R., Madrid-Cuevas, F. J., & Carmona-Poyato, A. (2008). Depth silhouettes for gesture recognition. *Pattern Recognition Letters*, *29*(3), 319–329. http://doi.org/10.1016/j.patrec.2007.10.011

Murphy, K. (1998). Hidden Markov Model (HMM) Toolbox for Matlab. Retrieved September 3, 2016, from https://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html

Nguyen-Duc-Thanh, N., Lee, S., & Kim, D. (2012). Two-stage Hidden Markov Model in gesture recognition for human robot interaction. *International Journal of Advanced Robotic Systems*, *9*. http://doi.org/10.5772/50204

Nickel, K., Scemann, E., & Stiefelhagen, R. (2004). 3D-tracking of head and hands for pointing gesture recognition in a human-robot interaction scenario. *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*. http://doi.org/10.1109/AFGR.2004.1301593

Nickel, K., & Stiefelhagen, R. (2007). Visual recognition of pointing gestures for human-robot interaction. *Image and Vision Computing*, *25*(12), 1875–1884. http://doi.org/10.1016/j.imavis.2005.12.020

Ohmura, I., Mitamura, T., Takauji, H., & Kaneko, S. (2010). A real-time stereo vision sensor based on FPGA realization of orientation code matching. *Optomechatronic Technologies (ISOT), 2010 International Symposium on*. http://doi.org/10.1109/ISOT.2010.5687334

Oikonomidis, I., Kyriazis, N., & Argyros, A. A. (2011). Efficient model-based 3D tracking of hand articulations using Kinect. In *BMVC* (Vol. 1, p. 3).

Park, C. B., & Lee, S. W. (2011). Real-time 3D pointing gesture recognition for mobile robots with cascade HMM and particle filter. *Image and Vision Computing*, *29*(1), 51–63. http://doi.org/10.1016/j.imavis.2010.08.006

Park, S., & Jeong, H. (2007). Real-time Stereo Vision FPGA Chip with Low Error Rate. *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*. http://doi.org/10.1109/MUE.2007.180

Park, S., Yu, S., Kim, J., Kim, S., & Lee, S. (2012). 3D hand tracking using Kalman filter in depth space. *Eurasip Journal on Advances in Signal Processing*, *2012*(1). http://doi.org/10.1186/1687-6180-2012-36

Pointgrey. (2016). Bumblebee2 FireWire stereo vision camera systems. Retrieved August 18, 2016, from https://www.ptgrey.com/bumblebee2-firewire-stereo-vision-camera-systems

Poppe, R. (2010). A survey on vision-based human action recognition. *Image and Vision Computing*, *28*(6), 976–990. http://doi.org/http://dx.doi.org/10.1016/j.imavis.2009.11.014

Premaratne, P. (2014). *Human Computer Interaction Using Hand Gestures*. *Cognitive Science and Technology*. Adelaide, South Australia, Australia: Springer. http://doi.org/10.1007/978-981-4585-69-9

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*. http://doi.org/10.1109/5.18626

Rahman, M. H., & Afrin, J. (2013). Hand Gesture Recognition using Multiclass Support Vector Machine. *International Journal of Computer Applications*, *74*(1), 39–43.

Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-validation. In *Encyclopedia of database systems* (pp. 532–538). Springer.

Ren, Z., Meng, J., & Yuan, J. (2011). Depth camera based hand gesture recognition and its applications in Human-Computer-Interaction. In *8th International Conference on Information, Communications and Signal Processing, ICICS 2011*. School of EEE, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore. http://doi.org/10.1109/ICICS.2011.6173545

Reyes, M., Domínguez, G., & Escalera, S. (2011). Featureweighting in dynamic timewarping for gesture recognition in depth data (pp. 1182–1188). Barcelona. http://doi.org/10.1109/iccvw.2011.6130384

Scharstein, D., & Szeliski, R. (2002). A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, *47*(1), 7–42. http://doi.org/10.1023/A:1014573219977

Scharstein, D., & Szeliski, R. (2003). High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (Vol. 1, pp. I–I). IEEE.

Scharstein, D., Ugbabe, P., & Szeliski, R. (2011). 2001 Stereo datasets with ground truth. Retrieved September 3, 2016, from http://vision.middlebury.edu/stereo/data/scenes2001/

Schwarz, L. A., Mkhitaryan, A., Mateus, D., & Navab, N. (2011). Estimating human 3D pose from Time-of-Flight images based on geodesic distances and optical flow. *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*. http://doi.org/10.1109/FG.2011.5771333

Schwarz, L. A., Mkhitaryan, A., Mateus, D., & Navab, N. (2012). Human skeleton tracking from depth data using geodesic distances and optical flow. *Image and Vision Computing*, *30*(3), 217–226. http://doi.org/http://dx.doi.org/10.1016/j.imavis.2011.12.001

Sempena, S., Maulidevi, N. U., & Aryan, P. R. (2011). Human action recognition using Dynamic Time Warping. *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*. http://doi.org/10.1109/ICEEI.2011.6021605

Shan, Y., Wang, Z., Wang, W., Hao, Y., Wang, Y., Tsoi, K., … Yang, H. (2012). FPGA based memory efficient high resolution stereo vision system for video tolling. *Field-Programmable Technology (FPT), 2012 International Conference on*. http://doi.org/10.1109/FPT.2012.6412106

Shi, Y., Taib, R., & Lichman, S. (2006). GestureCam: A smart camera for gesture recognition and gesture-controlled web navigation. Singapore. http://doi.org/10.1109/icarcv.2006.345267

Shi, Y., & Tsui, T. (2007). An FPGA-based smart camera for gesture recognition in HCI applications. Tokyo.

Siriboon, K., Jirayusakul, A., & Kruatrachue, B. (2002). HMM topology selection for on-line Thai handwriting recognition. *First International Symposium on Cyber Worlds, 2002. Proceedings.* http://doi.org/10.1109/CW.2002.1180872

Suarez, J., & Murphy, R. R. (2012). Hand gesture recognition with depth images: A review. In *2012 21st IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2012* (pp. 411–417). Center for Robot-Assisted Search and Rescue, Texas AandM University, College Station, TX 77840, United States. http://doi.org/10.1109/ROMAN.2012.6343787

Suryanarayan, P., Subramanian, A., & Mandalapu, D. (2010). Dynamic Hand Pose Recognition Using Depth Data. *Pattern Recognition (ICPR), 2010 20th International Conference on*. http://doi.org/10.1109/ICPR.2010.760

Tang, M. (2011). Recognizing hand gestures with microsoft's kinect. Retrieved from http://www.masters.dgtu.donetsk.ua/2012/fknt/sobolev/library/article5.pdf

Vasicek, Z., & Sekanina, L. (2008). Novel Hardware Implementation of Adaptive Median Filters. *2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*. http://doi.org/10.1109/DDECS.2008.4538766

Wang, X., Xia, M., Cai, H., Gao, Y., & Cattani, C. (2012). Hidden-Markov-Models-based dynamic hand gesture recognition. *Mathematical Problems in Engineering*, *2012*. http://doi.org/10.1155/2012/986134

Weinland, D., Ronfard, R., & Boyer, E. (2011). A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, *115*(2), 224–241. http://doi.org/http://dx.doi.org/10.1016/j.cviu.2010.10.002

Wilson, A. D., & Bobick, A. F. (1999). Parametric hidden Markov models for gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(9), 884–900. http://doi.org/10.1109/34.790429

Woodfill, J. I., Gordon, G., Jurasek, D., Brown, T., & Buck, R. (2006). The Tyzx DeepSea G2 Vision System, ATaskable, Embedded Stereo Camera. *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*. http://doi.org/10.1109/CVPRW.2006.202

Xu, W., & Lee, E. J. (2012). Continuous gesture recognition system using improved HMM algorithm based on 2D and 3D space. *International Journal of Multimedia and*

Ubiquitous Engineering, *7*(2), 335–340. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84865619564&partnerID=40&md5=45533bc3145cfb311a8fc8ebc3d55fbb

Yang, C., Jang, Y., Beh, J., Han, D., & Ko, H. (2012). Gesture recognition using depth-based hand tracking for contactless controller application. *2012 IEEE International Conference on Consumer Electronics (ICCE)*. http://doi.org/10.1109/ICCE.2012.6161876

Yin, X., & Xie, M. (2003). Estimation of the fundamental matrix from uncalibrated stereo hand images for 3D hand gesture recognition. *Pattern Recognition*, *36*(3), 567–584. http://doi.org/http://dx.doi.org/10.1016/S0031-3203(02)00072-9

Zabih, R., & Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In J.-O. Eklundh (Ed.), *Computer Vision --- ECCV '94: Third European Conference on Computer Vision Stockholm, Sweden, May 2--6 1994 Proceedings, Volume II* (pp. 151–158). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/BFb0028345

Zhang, C., & Tian, Y. (2013). Edge enhanced depth motion map for dynamic hand gesture recognition. In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW 2013* (pp. 500–505). Department of Electrical Engineering, City College of New York, United States. http://doi.org/10.1109/CVPRW.2013.80