UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

Profesor Patrocinante:
Dr. Miguel Figueroa

Informe de Tesis para optar al grado de:
Magíster en Ciencias de la Ingeniería con mención en Ingeniería Eléctrica

Detección y Reconocimiento de Rostros en Imágenes Infrarrojas sobre Hardware Digital Dedicado

Concepción, Agosto de 2016

Javier Esteban Soto Salcedo

Universidad de Concepción
Facultad de Ingeniería
Departamento de Ingeniería Eléctrica

Profesor Patrocinante:
Dr. Miguel Figueroa

Detección y Reconocimiento de Rostros en Imágenes Infrarrojas sobre Hardware Digital Dedicado

Javier Esteban Soto Salcedo

Informe de Tesis
para optar al grado de

"Magíster en Ciencias de la Ingeniería con mención en Ingeniería Eléctrica"

Agosto 2016

Resumen

El uso de tecnologías biométricas se presenta comúnmente en la actualidad, siendo un método de identificación confiable, seguro y de fácil uso. El reconocimiento de rostros se presenta como una característica biométrica que no requiere de intervención por parte de las personas a identificar, convirtiéndolo en una herramienta perfecta para utilizar en lugares de acceso restringido o para identificar personas en ambientes abiertos.

En este trabajo se diseñó e implementó un sistema embebido de detección y reconocimiento de rostros en imágenes infrarrojas, el cual que funciona conectado directamente a una cámara infrarroja. Para esto se realizó un estudio bibliográfico de algoritmos de detección y reconocimiento de rostros utilizados ampliamente en la literatura, poniendo énfasis en aquellos que pueden ser implementados eficientemente en hardware. Todos los algoritmos estudiados fueron propuestos para su uso en espectro visible, por lo que fue necesario realizar pruebas en software para comprobar su efectividad en imágenes en espectro infrarrojo de onda larga (Long Wave Infrared, LWIR). Realizando pruebas con tres bases de datos para reconocimiento y una para detección de rostros.

Finalmente, se implementó una versión modificada del algoritmo de Histograma de Gradientes Orientados (Histogram of Oriented Gradients, HOG), con clasificador Gentle AdaBoost para detección. Utilizando una combinación de los algoritmos Patrones Locales Binarios (Local Binary Patterns, LBP) y Análisis de Discriminante Lineal (Linear Discriminant Analysis, LDA), con clasificación por método del vecino más cercano para reconocimiento.

El sistema fue implementado en un sistema en chip (System on chip, SoC) Zynq-7000 de Xilinx, el detector se implementó en el procesador del SoC y el clasificador en lógica programable. El sistema fue probado con las bases de datos LWIR UdeC y UCHThermalFace, para detección y reconocimiento respectivamente. La primera de éstas cuenta con 612 imágenes infrarrojas de 102 sujetos, la segunda cuenta con 1484 imágenes de 53 sujetos. El sistema de detección fue entrenado y probado con 1248 imágenes de rostros y de otros objetos, extraídas de la base de datos LWIR UdeC, con lo que se obtuvo una tasa de acierto de 99.8 %. El sistema de reconocimiento fue entrenado con un 60 % de la base de datos UCHThermalFace y probado con el 40 % restante, con lo que se obtuvo una tasa de acierto de 99.21 %. El sistema completo consume 1.88 W de potencia y puede detectar rostros a un cuadro por segundo, pudiendo clasificar 936 rostros por segundo al funcionar a 100 MHz. En una implementación en hardware del detector de rostros, se alcanzaría una tasa de 305 cuadros por segundo para imágenes de 640×512 píxeles.

"Los mejores libros son los que nos dicen lo que ya sabemos."

- Winston.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Abreviaciones

IR	Infrared.
LWIR	Long Wave Infrared.
FPN	Fixed-Pattern Noise.
PCA	Principal Component Analysis.
LDA	Linear Discriminant Analysis.
LBP	Local Binary Pattern.
MB-LBP	Multi Block Local Binary Pattern.
HOG	Histogram of Oriented Gradients.
AdaBoost	Adaptive Boosting.
DPM	Deformable Parts Models.
SVM	Support Vector Machine.
LUT	Lookup Table.
FPGA	Field Programmable Gate Array.
DSP	Digital Signal Processor.
SIMD	Single Instruction Multiple Data.
VGA	Video Graphics Array.

Capítulo 1. Introducción

1.1. Introducción general

Las tecnologías de identificación de personas por características biométricas son muy atractivas por la facilidad de uso que presentan para el usuario final. Por lo mismo han sido incorporadas como métodos de identificación en bancos, sistemas de salud, accesos a lugares restringidos e incluso en cajeros automáticos. El reconocimiento de rostros es particularmente atractivo ya que no requiere de intervención de las personas a identificar para su funcionamiento, permitiendo realizar la identificación sólo basado en imágenes obtenidas desde una cámara a través del proceso de detección de rostros.

Los temas de detección y reconocimiento facial han sido ampliamente estudiados en la literatura, lo que ha permitido el desarrollo de sistemas confiables, además de sencillos en el caso de detección facial. Esto ha facilitado la masificación del uso de esta tecnología en diversos dispositivos y servicios como cámaras fotográficas o de vigilancia, *smartphones*, consolas de videojuegos o aplicaciones de tratamiento y organización de imágenes en la nube. Sin embargo, todos estos sistemas presentan un problema natural del procesamiento de imágenes en espectro visible, como lo es la dependencia a la iluminación de la escena. Este tema generalmente es abordado realizando un preprocesamiento de las imágenes, utilizando ajuste de contraste o corrección gamma. El uso de imágenes infrarrojas de onda larga elimina este problema, ya que la cámara captura la radiación térmica emitida por los cuerpos, permitiendo su uso en ambiente con baja iluminación, e incluso en ausencia de ella. Aun así, las imágenes infrarrojas son afectadas por otros problemas, como la dependencia a la temperatura o el ruido de patrón fijo (FPN) producto de la no-uniformidad de los sensores infrarrojos.

En el procesamiento de imágenes en tiempo real se requiere de un buen rendimiento por parte de las implementaciones, para poder cumplir con la necesidad de velocidad de procesamiento requerida. La detección y el reconocimiento facial son tareas computacionalmente costosas, para las que existen soluciones confiables como OpenCV en software o las implementaciones de detección facial en cámaras digitales entre otras. A pesar de esto, resulta difícil encontrar sistemas de detección y reconocimiento capaces de procesar cuadros a la frecuencia máxima que entregan las cámaras. Comúnmente, un cuadro de video puede tener más de diez rostros presentes, y es necesario detectarlos y clasificarlos a todos en tiempo real. La implementación en lógica programable garantiza un tiempo fijo de procesamiento en detección y una muy alta

velocidad en el reconocimiento. Además de un bajo consumo de potencia, todo esto por las arquitecturas con alto paralelismo soportadas por los FPGA.

Por lo anterior, resulta de interés realizar el desarrollo de un sistema embebido de detección y reconocimiento facial en espectro infrarrojo funcionando en tiempo real directamente conectado a una cámara.

1.2. Estado del arte

1.2.1. Detección de rostros

El problema de detección de rostros ha sido ampliamente estudiado en la literatura. La detección de objetos corresponde básicamente a un problema de clasificación binaria, donde se generan ventanas móviles dentro de una imagen y se clasifican como pertenecientes a una u otra clase.

Uno de los primeros enfoques para realizar detección de rostros son las características Haar propuestas por Viola y Jones [6]. En este algoritmo se utilizan cinco operadores predefinidos sobre una ventana, éstos son escalados y desplazados para abarcar todas las posibles combinaciones dentro de la ventana, a cada uno de estos nuevos operadores se les conoce como características. Las características Haar son utilizadas para capturar información de texturas presentes en la imagen, basándose en la diferencia de intensidad de píxeles por zonas. En el trabajo realizado por Viola y Jones se utiliza un clasificador en cascada, en el que se divide el conjunto de entrenamiento en un número igual a la cantidad de etapas de la cascada, para cada una de ellas se entrena un clasificador AdaBoost, en los que se reduce la cantidad de rostros clasificados como no rostros, modificando el umbral de pertenencia a una u otra clase. Así, la cascada descarta las imágenes que no corresponden a rostros en las primeras etapas y la reducción de falsos positivos se produce de forma natural con el paso a las otras etapas de la cascada. El clasificador en cascada propuesto mejora significativamente el tiempo requerido en software para procesar cuadros, sobre todo en imágenes con pocos rostros. Por esto, éste es uno de los algoritmos más utilizados en la actualidad, estando presente en OpenCV. El algoritmo propuesto por Viola y Jones obtiene una tasa de acierto máxima de 94.1 % en la base de datos utilizada.

Un algoritmo similar a las características Haar fue propuesto por Zhang et al. [7]. Este algoritmo, llamado características MB-LBP (Multi Block Local Binary Patterns), realiza un proceso

muy similar al algoritmo de Viola y Jones, la única diferencia corresponde a los operadores que se utilizan para construir las características. Mientras en Haar se utilizan cinco operadores predefinidos, en MB-LBP se utiliza un operador que utiliza un vecindario de 3×3 regiones y tiene la capacidad de capturar 255 tipos de texturas diferentes. En este trabajo se argumenta que el operador utilizado puede extraer una mayor cantidad de información desde la imagen. Además de generar un vector de características de mucho menor tamaño, de cerca de un veinteavo de tamaño que el generado en Haar, lo que reduce considerablemente el tiempo necesario durante el entrenamiento del algoritmo. En este trabajo se reporta una tasa de acierto del 93.5 % para la misma base de datos utilizada por Viola y Jones.

Los algoritmos de características Haar y MB-LBP tienen un entrenamiento muy lento, principalmente dado por la cantidad de características que se extraen de una ventana. Otros algoritmos realizan la extracción de características representando la ventana con información de los píxeles presentes, sin utilizar variaciones de macro texturas u operadores predefinidos, por lo que generan descriptores de mucha menor dimensión que Haar o MB-LBP. Entre estos algoritmos se encuentran los histogramas LBP (Local Binary Patterns) que son utilizados como descriptores de las micro texturas presentes en una imagen. Con éstos se realiza la clasificación en base a la similitud de un histograma a la referencia de una u otra clase. Los trabajos que utilizan este algoritmo lo hacen en combinaciones de distintas formas de calcular el operador LBP [8], o en combinación segmentación de piel en base a color [9]. En ambos trabajos se presentan buenos resultados sobre las bases de datos utilizadas, con un 97.8 % y 93.4 % de acierto respectivamente. Sin embargo, los resultados sobre esas bases de datos no son comparados con otros algoritmos como Haar, por lo que no se puede conocer su efectividad relativa a éstos.

Un método alternativo de detección de rostros se basa en encontrar puntos característicos de un rostro en una imagen, como bordes del mentón, nariz, ojos, etc., y en base a éstos realizar la detección facial. Estos algoritmos, conocidos como modelos de partes deformables (Deformable Parts Models, DPM), utilizan algún descriptor como HOG (Histogram of Oriented Gradients) [10, 11] para describir las áreas en torno a los puntos de interés. La principal ventaja de este método es la capacidad de detectar un rostro en condiciones de rotación, inclinación u oclusión, permitiendo detectar los bordes del rostro y simplificando el proceso de encasillamiento de éste. Si bien el uso de este método presenta muchas ventajas, añade complicaciones en el diseño del clasificador, pues se pasa de un problema de clasificación binaria (rostro o no rostro) a un problema multiclase, aumentando la cantidad de operaciones que se deben realizar y de la memoria necesaria para su implementación. El uso de este método presenta buenos resultados, utilizando DPM con HOG [11] en una base de datos estándar se obtiene un acierto de 86.4 %,

mientras para la misma base de datos se obtiene una tasa de acierto de 59.7 % utilizando el algoritmo de Viola y Jones.

En la actualidad la detección de rostros se enfrenta con métodos que utilizan redes neuronales de aprendizaje profundo (*deep learning*). Generalmente las imágenes son utilizadas sin realizar una extracción de características, ya que la misma red neuronal lo realiza. Así, la red es alimentada con los píxeles de diferentes zonas de las imágenes que son divididas en secciones cada vez más pequeñas al pasar de una capa a otra. Las neuronas de la red asignan pesos a cada región según la probabilidad de corresponder o no a un rostro y se realiza la clasificación en base a estas probabilidades. En uno de los trabajos más actuales [12] que utiliza redes neuronales de aprendizaje profundo, se obtienen buenos resultados con rostros rotados y en diferentes poses. Además, como comparativo se entregan resultados para una base de datos estándar, sobre la que se obtiene una tasa de acierto promedio de 84 %, mientras que el algoritmo Haar presenta un acierto promedio de 59.7 %.

La detección de rostros en imágenes en espectro infrarrojo no ha sido un tema muy desarrollado, en comparación a la gran cantidad de trabajos respecto al mismo tema en espectro visible. En general, los sistemas de detección de rostros en imágenes infrarrojas realizan la localización segmentando la imagen según la temperatura. Esto se basa en que la temperatura corporal tiene un rango relativamente constante y cercana a 33,5°C. Al tratarse de imágenes en infrarrojo térmico resulta sencillo encontrar las regiones con temperaturas superiores. En este principio se basan los trabajos de detección existentes [13, 14, 15, 16]. En esos se encuentran métodos simples de detección [15] en el que se segmenta la imagen por temperatura, y se detectan los limites para el encasillamiento encontrando los segmentos horizontales y verticales con mayor cantidad de píxeles con mayor temperatura. Si bien esto funciona, presentando una tasa de acierto de 83.84 % y 52.02 % en las bases de datos utilizadas, limita la búsqueda a sólo una persona por imagen. Una solución más compleja se utiliza en un sistema de activación inteligente de *airbag*. En la que se realiza el mismo procedimiento de segmentación, pero sobre ésta se realiza una búsqueda de elipses afines con un rostro, generando buenos resultados que alcanzan un 90.1 % de acierto.

Otro método [17] utilizado en detección de rostros en espectro infrarrojo, realiza una segmentación de piel en imágenes utilizando un descriptor de texturas. Para esto, con la información del vecindario de cada píxel, se construye un histograma utilizando LBP, que corresponde al vector de características del píxel en cuestión. El histograma generado es clasificado utilizando SVM (Support Vector Machine) obteniendo buenos resultados en las pruebas realizadas, con un 100 % en una base de datos y un 94.5 % en secuencias de video. Si bien este método presenta

buenos resultados y fue implementado eficientemente en hardware, tiene un problema similar al de la segmentación de piel por temperatura. Ya que en ambos casos se detectan píxeles correspondientes a piel, por lo que en ciertas condiciones ambos métodos pueden presentar un alto número de falsos positivos.

La detección de rostros es un problema muy estudiado y presenta un gran número de implementaciones en hardware. La mayoría de las implementaciones utilizan el algoritmo Haar de Viola y Jones, ya sea en implementaciones en FPGA [18] o en arquitecturas mixtas de software-hardware [19, 20]. En estos trabajos se logra un incremento significativo en la velocidad de detección, entre 10 [20] y 72 [19] veces comparado con las implementaciones realizadas en software en cada trabajo. También es posible encontrar implementaciones del algoritmo de características MB-LBP, realizadas en SoC (Systems on Chip) [21] y en procesadores paralelos masivos [22]. Este último utilizando una arquitectura de una instrucción en múltiples datos (Single Instruction Multiple Data, SIMD), con la que se pueden procesar 5 cuadros por segundo en resolución VGA, con un consumo de 250 mW.

Todos los sistemas mencionados requieren un alto uso de recursos, principalmente de memoria, debido a la búsqueda en varias resoluciones, con la que se deben construir imágenes piramidales. Este problema no ocurre al utilizar un detector de piel [17], ya que se realiza la detección en línea, sin necesidad de almacenar la imagen para cambiar la escala. El sistema propuesto en este trabajo es capaz de procesar 313 cuadros por segundo de resolución VGA, consumiendo 266 mW.

1.2.2. Reconocimiento de rostros

El reconocimiento de rostro es un tema muy estudiado en la actualidad. Los algoritmos utilizados para esto se dividen en dos tipos, los métodos holísticos y los basados en características. Los métodos holísticos consideran el rostro completo para el análisis y la clasificación, sin dividirlo ni buscar características relevantes dentro de éste. En los métodos basados en características, se realiza el reconocimiento extrayendo información de regiones relevantes del rostro o generando una descripción con información de la localidad de ciertas regiones. Estos últimos son los más utilizados en la actualidad, pues han presentado mejores resultados en los trabajos realizados.

Los primeros enfoques en el problema de reconocimiento de rostros se realizaron usando métodos holísticos. Entre éstos se encuentran las distintas aplicaciones de métodos basados

en subespacios como el Análisis de Componentes Principales (Principal Component Analysis, PCA) y el Análisis de Discriminante Lineal (Linear Discriminant Analysis, LDA). La aplicación de estos métodos en el problema de reconocimiento de rostro se conocen como Eigenfaces [23] y Fisherfaces [24], respectivamente. Estos algoritmos realizan una proyección de las imágenes faciales a un subespacio lineal de menor dimensión. En PCA este subespacio está diseñado para maximizar la representación de los datos originales, utilizando una matriz de proyección que maximiza la varianza de los vectores proyectados. En LDA el subespacio maximiza la varianza entre los elementos de diferentes clases y la minimiza entre los elementos de una misma clases. Ambos métodos han sido muy utilizados en aplicaciones de reconocimiento de rostros y son parte de las bibliotecas de visión por computador como OpenCV. Si bien el funcionamiento y entrenamiento de ambos algoritmos es muy similar, LDA presenta mejores tasas de acierto, ya que está diseñado para obtener un mejor resultado en problemas de clasificación. En comparativas entre ambos algoritmos es posible ver como LDA requiere un número menor de proyecciones para alcanzar las mismas tasas de acierto que PCA. En particular, en el trabajo donde se introdujo Fisherfaces [24] se logra una tasa de acierto de 92.3 % con 15 proyecciones LDA, mientras que usando PCA se alcanza un 75.6 % de acierto con 30 vectores de proyección para la base de datos utilizada.

Otro enfoque al problema de reconocimiento de rostros son los métodos basados en características. Como se explicó anteriormente, en estos algoritmos se realiza un análisis de distintas regiones de interés dentro de una imagen. El algoritmo basado en características más utilizado corresponde a los histogramas LBP. Este algoritmo es utilizado para representar las microtexturas presentes en una imagen, para su uso en reconocimiento de rostros se divide la imagen en diferentes regiones, generando un histograma para cada una de ellas, con lo que, además de la información de las texturas, se obtiene información espacial de éstas. El gran uso de LBP corresponde principalmente a la sencillez del operador y a los buenos resultados que presenta. Por lo anterior, LBP es un buen candidatos al momento de realizar implementaciones en sistemas de bajo consumo energético [25]. En un estudio comparativo [26] de variaciones de LBP con métodos basados en subespacios, se obtiene una tasa de acierto máxima de 97.91 % para el algoritmo Ahonen [27] (basado en LBP), mientras que en la misma base de datos se obtiene un 57.72 % usando PCA y un 67.84 % con LDA.

Una variación de LBP, llamada Multi-Scale Block LBP [28], remplaza el vecindario de 3×3, por vecindarios de tamaño variable, similar al algoritmo MB-LBP de detección facial. Así, para cada uno de los vecindarios utilizados se generan histogramas, de los que luego se extrae el conjunto de muestras que mejor discriminan entre una y otra clase. Para este algoritmo

son necesarias muchas operaciones durante el entrenamiento, ya que es necesario calcular un histograma para cada posible tamaño de vecindario. Este algoritmo alcanza un acierto máximo de 99.78 %, mientras que para la misma base de datos LBP obtiene un 93.62 %.

En la actualidad los sistemas de reconocimiento facial utilizan redes neuronales de aprendizaje profundo. Uno de los ejemplos más conocidos corresponde al sistema de reconocimiento utilizado por Facebook [29], el cual realiza un alineamiento 3D de las imágenes utilizando histogramas LBP y detectando punto específicos del rostro. Los píxeles de la nueva imagen frontal alimentan a una red neuronal de nueve capas encargada de extraer las características del rostro e identificar a los sujetos. El sistema propuesto alcanza una tasa de acierto de 97.35 % en la base de datos utilizada. En el trabajo se expone que la red neuronal utiliza 120 millones de parámetros, pero aun así tarda sólo 0.23 segundos en procesar una imagen en un computador de escritorio. A pesar de la rapidez indicada, el número de parámetros requeridos hace que este tipo de algoritmos no sean sencillos de implementar en sistemas embebidos, donde la memoria disponible es limitada.

El reconocimiento de rostros en espectro infrarrojo ha sido tratado de manera muy similar al espectro visible, utilizando métodos holísticos como PCA [13, 30]. En estos trabajos se logran altas tasas de acierto, con un máximo de 94 % y 100 % respectivamente. Sin embargo, se presentan dificultades en el reconocimiento en imágenes tomadas con diferente tiempo [30], pues se argumenta que existen diferencias en la temperatura de todo el rostro, especialmente en la nariz. Esto produce una baja de la tasa de acierto de un máximo de 100 % a un máximo de 84 %.

El uso de histogramas LBP en espectro infrarrojo de onda larga, para la representación facial ha sido muy extendido [25, 31, 32]. En un trabajo comparativo [32] se obtiene una tasa de acierto de 97.3 % para LBP y 97.5 % para el LDA, en la base de datos utilizada. Si bien LDA presenta un mejor desempeño, LBP presenta ventajas en el entrenamiento del sistema, ya que, a diferencia de LDA, puede ser aplicado cuando hay sujetos con sólo una imagen disponible. En trabajos posteriores [31, 25] se ha combinado LBP con LDA, aplicando LDA al histograma de texturas LBP. Esto ha permitido un aumento en las tasas de acierto, pero haciendo obligatorio el uso de más de una imagen por sujeto.

En el último tiempo se han realizado estudios que combinan información del espectro visible e infrarrojo. En uno de éstos [33] sólo se utilizan imágenes en espectro visible durante el entrenamiento, usando redes neuronales de aprendizaje profundo para asociar imágenes térmicas a esta base de datos. Utilizando este enfoque se logran tasas de acierto de 83.73 % en la base de

datos utilizada.

Los métodos basados en subespacios lineales requieren de un gran número de operaciones aritméticas para realizar la proyección y de memoria para almacenar los parámetros requeridos, esto limita su uso en aplicaciones en sistemas embebidos. Por esto, las implementaciones de este tipo de algoritmos deben ser variaciones que reduzcan los requerimientos principalmente en memoria. En este sentido, PCA bidireccional [34] fue utilizado siguiendo estas limitantes sobre un FPGA, logrando reducir en 40 veces el uso de memoria en comparación a una implementación normal de LDA. El sistema implementado logra una tasa de acierto de 93.3 %, con un consumo de 157.24 mW, siendo capaz de clasificar más de 7000 rostros por segundo.

Otro algoritmo implementado en hardware es el descriptor de texturas LBP, que en conjunto con LDA fue implementado sobre un FPGA [25]. En éste se logró una tasa de acierto de 98.6 % sobre la base de datos utilizada, sólo un 0.15 % menor a la implementación en software, con el *core* de reconocimiento facial consumiendo sólo 309 mW y pudiendo clasificar más de 8000 rostros por segundo.

1.2.3. Discusión

Los algoritmos de detección facial revisados pueden dividirse en dos grupos, por el entrenamiento similar que presentan. El primer grupo corresponde a los algoritmos Haar y MB-LBP, los cuales son muy similares y presentan un número reducido de operaciones necesarias durante el funcionamiento del sistema, lo que los convierte en buenos candidatos para realizar una implementación en hardware. Sin embargo, presentan un problema común al realizar una búsqueda de rostros de diferente tamaño. En ambos algoritmos el redimensionar una característica es complicado, por lo que se opta por usar imágenes piramidales, aumentando los requerimientos de memoria necesarios para su funcionamiento. El segundo grupo de algoritmos de detección corresponde a los histogramas LBP y HOG, en ambos la construcción del descriptor es similar, pero capturan información diferente. Para una implementación en hardware LBP parece la opción lógica, ya que el operador es sencillo y fácil de aplicar e implementar. Sin embargo, es posible realizar simplificaciones en el cálculo de HOG aprovechando que el cálculo de los ángulos, que es el más costoso de implementar por la función arcotangente, no requiere de mucha precisión. Considerando lo anterior, no existe una diferencia sustancial, en cuanto a rendimiento, entre implementar HOG o LBP.

Como se explicó anteriormente, en los algoritmos de reconocimiento hay dos tipos de mé-

todos, los basados en subespacios y los basados en características. Los algoritmos basados en subespacios utilizan una gran cantidad de parámetros, ya que requieren un parámetro para cada píxel de la imagen. Por esto, en software generalmente se reduce el tamaño de las imágenes al utilizar este tipo de algoritmos, sin embargo esto sería muy costo en hardware, por lo que no es una solución práctica. Los algoritmos basados en características como LBP, presentan una mejor opción para implementar en hardware. En particular LBP es un operador sencillo que genera un descriptor de texturas en base sólo a comparaciones de un vecindario de píxeles, además en la bibliografía se puede corroborar la efectividad de este algoritmo en reconocimiento facial.

Como se pudo ver anteriormente, no hay muchos trabajos en espectro infrarrojo de detección facial, esto puede estar dado porque para su uso en entornos controlados es suficiente con segmentar las imágenes por temperatura. Además, tanto para la detección como para el reconocimiento, sólo existe una implementación en hardware para imágenes en espectro visible. Por lo anterior se presenta una oportunidad para el desarrollo de este trabajo, pues no existen sistemas embebidos de verificación facial en tiempo real en espectro infrarrojo. Por lo que una implementación de esto puede resultar de mucha utilidad para aplicaciones de seguridad, control de accesos o estudios futuros del tema. Así, el principal desafío de este trabajo corresponde a la búsqueda y/o adecuación de un algoritmo que pueda ser eficientemente implementado en hardware programable, los cuales presentan limitaciones de memoria y recursos, pero permiten un diseño dedicado altamente paralelizable.

1.3. Motivación

La implementación de algoritmos de procesamiento de imágenes y visión computacional en hardware presenta un reto importante, ya que el esfuerzo requerido para el diseño de un sistema embebido es mucho mayor comparado a un desarrollo en software. Esto está dado principalmente por los grados de libertad existentes en plataformas basadas en FPGA, que permiten un manejo a muy bajo nivel. Además existen limitaciones técnicas como la cantidad de unidades funcionales presentes, entre las cuales se encuentran las memorias locales de acceso rápido (Block RAMs), que limitan la cantidad de datos a los que se puede acceder de forma rápida recurrentemente. Por esto, no es posible implementar de igual manera soluciones usadas en software, las que, en los casos de algunos algoritmos de clasificación, requieren del uso de muchas matrices de gran tamaño. Debido a esto se debe ser cuidadoso en la selección de algoritmos, además, de ser necesario, se deben realizar simplificaciones en la totalidad o en parte de los algoritmos seleccionados, con el fin de realizar una implementación eficiente y con un desempeño similar al

de un computador de propósito general.

A pesar de lo anterior, una implementación en hardware basado en FPGA tiene grandes beneficios, como la alta tasa de cuadros por segundo que se puede obtener o la muy baja potencia que consumen los circuitos de alto desempeño diseñados.

En síntesis, la dificultad de este trabajo en particular recae en realizar la selección de algoritmos de detección y reconocimiento de rostros, y en diseñar una arquitectura que implemente una variación de éstos de forma eficiente, considerando las limitaciones existentes en el hardware.

1.4. Hipótesis

En este trabajo se postula la posibilidad de realizar una implementación eficiente, en hardware de lógica programable, de algoritmos basados en micro texturas y en orientación de píxeles, para detección y reconocimiento de rostros en imágenes infrarrojas. Esta implementación, además de eficiente, debe ser de bajo consumo energético y debe poder ser ejecutada en tiempo real, ya que el sistema completo funcionará conectado directamente a una cámara infrarroja.

Adicionalmente, se plantea la posibilidad de aumentar la tasa de acierto en la etapa de reconocimiento facial. Para esto se propone diseñar una etapa de procesamiento intermedia entre la detección y reconocimiento de rostros, capaz de identificar la inclinación del rostro. Para esto se plantean dos alternativas a evaluar, la primera corresponde a la resolución de un problema multiclase con inclinaciones predefinidas, y la segunda corresponde a la detección de regiones características del rostro como los ojos. Con ambas alternativas es posible obtener información de la inclinación presente en cada rostro y enviarla al módulo de reconocimiento. El incrementar la tasa de acierto es importante ya que presenta la posibilidad de disminuir la precisión de las representaciones faciales creadas, lo que conlleva una disminución de los recursos de memoria requeridos en esta etapa. Así, el incremento en el acierto dado por la etapa que se propone, no supondrá una mejora sustancial en el resultado, pero será necesario dada las limitaciones técnicas existentes en un FPGA.

1.5. Objetivos

1.5.1. Objetivo general

El objetivo de este trabajo es realizar el diseño de una arquitectura de un sistema de detección y reconocimiento en tiempo real, utilizando imágenes de espectro infrarrojo. Éste será implementado en una tarjeta de desarrollo con lógica programable, utilizando una arquitectura mixta de hardware-software.

1.5.2. Objetivos específicos

1. Creación de una base de datos con imágenes infrarrojas de personas bajo condiciones controladas. Capturando distintas rotaciones horizontales del rostro, diferentes gestos faciales e imágenes grupales.
2. Validación de los algoritmos presentados en detección y reconocimiento de rostros, con la nueva base de datos.
3. Desarrollo y simulación de etapa de estimación de inclinación y encasillamiento de rostros.
4. Desarrollo de interfaz de visualización por HDMI en la tarjeta de desarrollo. En la que se visualizará la salida de la cámara y posteriormente la salida del detector facial. Además mostrará información adicional, como número de personas en la escena y la identidad de éstas.
5. Desarrollo e implementación sobre un FPGA de una arquitectura de detección facial, incluyendo una etapa de encasillamiento.
6. Desarrollo e implementación sobre un FPGA de una arquitectura de reconocimiento facial. Ésta será capaz de identificar los rostros incluidos en la base de datos e informar cuando se detecte un rostro no almacenado.
7. Finalización del sistema, combinando la interfaz con los sistemas de detección y reconocimiento desarrollados.

1.6. Temario

El temario de este trabajo se describe a continuación:

- El Capítulo 2 presenta una revisión teórica de los algoritmos utilizados para el desarrollo de este trabajo.
- El Capítulo 3 describe la metodología utilizada para evaluar el desempeño de los algoritmos y los resultados obtenidos en la pruebas en software realizadas.
- El Capítulo 4 presenta el desarrollo de la implementación del detector y del clasificador de rostros en un sistema embebido, incluyendo la arquitectura y las consideraciones de diseño.
- El Capítulo 5 muestra los resultados obtenidos en cada parte del sistema.
- El Capítulo 6 recopila las conclusiones alcanzadas durante el desarrollo de este trabajo y propone el trabajo a futuro.

Capítulo 2. Fundamentos teóricos

2.1. Introducción

Los procesos de detección y reconocimiento de rostros, son aplicaciones particulares de sistemas de clasificación, para dos clases en el caso de la detección (rostro o no) y para múltiples clases en el caso del reconocimiento. Todo problema de clasificación consta comúnmente de dos etapas: la extracción de características y la clasificación. La primera de estas etapas se realiza para generar un vector de valores que sea capaz de representar un tipo de dato en particular, en este caso imágenes. Esta representación se realiza para disminuir la cantidad de valores originales del tipo de dato a tratar y para obtener un conjunto nuevo que sea de mayor utilidad en el proceso de clasificación. La segunda etapa corresponde a la clasificación del vector de características generado anteriormente, en esta etapa se asocia el descriptor a una de las clases de interés.

En este capítulo se describe el algoritmo utilizado para realizar la implementación del problema de detección y reconocimiento de rostros, el cual corresponde a simplificaciones y modificaciones de otros algoritmos. Además, se describen todos los otros algoritmos estudiados y utilizados para realizar las comparaciones del Capítulo 3.

2.2. Detección de rostros

El problema de detección de rostros corresponde a un problema de clasificación binaria, donde sólo existen dos casos posibles, el que una imagen corresponda a un rostro o a algún otro elemento. En general, los algoritmos de detección facial se utilizan para buscar rostros dentro de una imagen, para esto se toma una región de interés, se clasifica y luego se desplaza horizontalmente para clasificar cada una de las regiones posibles dentro de la imagen. Este proceso de búsqueda se conoce como ventana deslizante y es utilizada por la mayoría de los algoritmos de detección de objetos. El procedimiento anterior sólo puede ser utilizado para encontrar objetos en una sola escala, por lo que para realizar una búsqueda en múltiples escalas es necesario redimensionar la imagen en tantos tamaños como requiera la aplicación, lo que es conocido como imágenes piramidales.

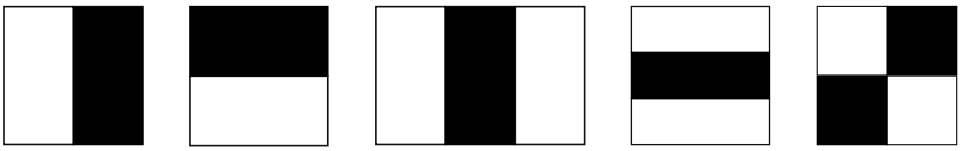


Figura 2.1: Características Haar.

2.2.1. Extracción de características

Características Haar

El detector de objeto Haar fue propuesto por Viola y Jones [6]. Éste consiste en utilizar diferencias de intensidades de píxeles por bloques para generar un descriptor útil para discriminar las diferencias entre un objeto determinado y otros elementos. El detector original utiliza las características de la Figura 2.1, las cuales son desplazadas y redimensionadas para abarcar todas las posibles posiciones y tamaños dentro de una región de tamaño determinado. El valor de cada una de estas características está dado por la suma de las intensidades de los píxeles del área de color negro, menos la suma de las intensidades en el área de color blanco, dado por:

$$\text{Valor} = \sum \text{Píxeles en negro} - \sum \text{Píxeles en blanco}. \tag{2.1}$$

El resultado de este algoritmo genera un vector de características de muy alta dimensionalidad, el que depende del tamaño de la ventana móvil dentro de la imagen. Por ejemplo utilizando regiones de 24×24 píxeles se obtienen más de 160 mil características [6]. La aplicación de las características Haar se ilustra en la Figura 2.2, donde se muestran las cinco características de la Figura 2.1 con un desplazamiento dentro de la región de análisis y con un tamaño diferente al original.

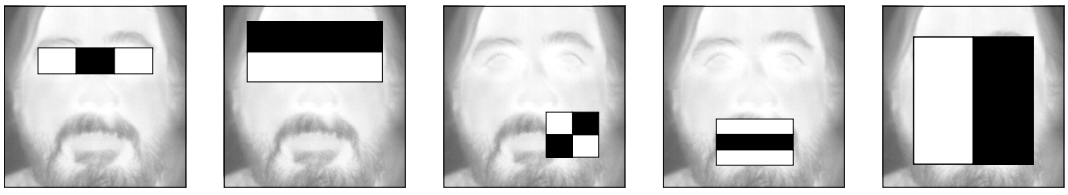


Figura 2.2: Ejemplo de uso de características Haar.

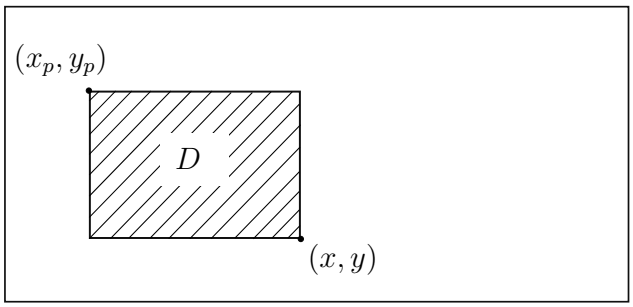


Figura 2.3: Ejemplo de cálculo de suma de intensidades de píxeles usando imágenes integrales.

El cálculo de cada característica Haar tarda un tiempo variable, el que depende de la cantidad de píxeles que deben ser sumados. Con el fin de obtener un tiempo de cálculo fijo por característica y evitar repetir cálculos de sumas de intensidades de píxeles, se introdujo la definición de imágenes integrales [6]. Una imagen integral es creada realizando sumas consecutivas de todos los valores de intensidad de los píxeles previos, horizontal y verticalmente, a uno en particular. Así, cada píxel de una imagen integral posee el valor de la suma de las intensidades de la imagen original desde el origen de la imagen $(0, 0)$ hasta el píxel en cuestión. Esto se puede representar como:

$$I(x, y) = \sum_{y'=0}^{y} \sum_{x'=0}^{x} i(x', y'), \tag{2.2}$$

donde $I(x, y)$ corresponde al valor del píxel (x, y) en la imagen integral e $i(x, y)$ a la intensidad de la imagen original en el píxel (x, y).

Utilizando imágenes integrales, es posible calcular la suma de las intensidades dentro de un área de una imagen en un tiempo constante, ya que, independientemente de su tamaño, sólo se requiere leer el valor de cuatro píxeles de la imagen integral. Por ejemplo para calcular la suma de las intensidades de los píxeles en el área D de la Figura 2.3, sólo es necesario leer lo valores en los píxeles (x, y), (xp, y), (x, yp) y (xp, yp), luego la suma D estará dada por:

$$D = I(x, y) - I(x_p, y) - I(x, y_p) + I(x_p, y_p). \tag{2.3}$$

En el Algoritmo 1 se describe el proceso de extracción de características utilizando el algoritmo Haar. Indicando todas las etapas previas al entrenamiento de un clasificador.

Algoritmo 1 Detección de rostros utilizando características Haar

1: Obtener un conjunto de imágenes de rostros y otro con imágenes sin rostros con un tamaño predeterminado.
2: Generar todas las posibles características para el tamaño de la ventana móvil.
3: Calcular imagen integral de cada imagen.
4: Obtener los valores de cada característica y almacenarlos en el descriptor.
5: Utilizar los descriptores de las imágenes de ambas clases para entrenar un detector facial con un algoritmo de clasificación.

Patrones locales binarios, LBP

El operador LBP es un descriptor de texturas para imágenes en escala de grises. Éste fue introducido por Ojala et al. [35]. Originalmente cada valor de salida está definido por la umbralización de un vecindario de 3×3 píxeles respecto al píxel central. Para cada píxel se genera una etiqueta que es construida concatenando el resultado binario de las comparaciones del píxel con su vecindario. Esta concatenación se realiza desde la esquina noroeste del vecindario, cuya comparación con el píxel central entrega un resultado *booleano* correspondiente al bit más significativo de la etiqueta LBP. Este proceso se repite en forma circular para todo el vecindario, tal como se ilustra en la Figura 2.4, obteniendo un número de 8 bits. El valor resultante del operador LBP se puede definir como:

$$\mathrm{LBP}(x_c, y_c) = \sum_{p=0}^{P-1} 2^p \; s(i_p - i_c), \tag{2.4}$$

donde P corresponde al número de píxeles en el vecindario, x_c y y_c a las coordenadas del píxel central, e i_p corresponde a la intensidad del p-ésimo píxel en torno al píxel central i_c. La función s se define como:

$$s(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{en otro caso} \end{cases}, \tag{2.5}$$

lo que corresponde a la umbralización entre las intensidades de los píxeles del vecindario respecto a la intensidad del píxel central. El orden de la vecindad se muestra en la Figura 2.5.

El operador LBP se aplica sobre toda la imagen, generando una nueva imagen con dos filas y dos columnas de píxeles menos, ya que no es posible aplicar el operador en los extremos de la imagen. Finalmente el histograma de la imagen con LBP aplicado puede ser utilizado como

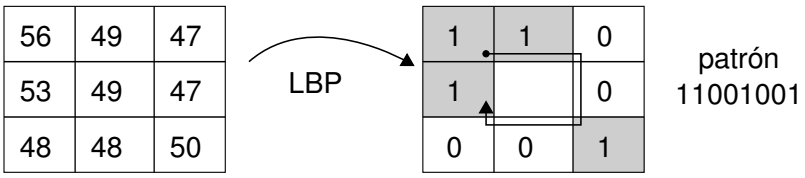


Figura 2.4: Operador LBP con etiqueta 11001001.

descriptor de ésta. LBP ha demostrado ser robusto frente a cambios de iluminación en espectro visible y ante la no-uniformidad en la respuesta de los sensores en imágenes infrarrojas [32].

LBP fue extendido a vecindarios de mayor tamaño [36], definiéndolo como $\text{LBP}_{(P,R)}$ con P puntos equidistantes en un radio R en torno al píxel de interés, donde el operador tiene 2^P etiquetas distintas. En caso de que una coordenada no corresponda a un píxel exacto, su intensidad es calculada por interpolación. Así, el operador original de LBP es definido como $\text{LBP}_{(8,1)}$ y presenta 256 etiquetas distintas.

Otra extensión importante de LBP es la de patrones uniformes. Un patrón LBP es uniforme cuando tiene a lo más dos transiciones de $0 \rightarrow 1$ o de $1 \rightarrow 0$. Por ejemplo los patrones 00111000, 11110001, 00000011 son uniformes y los patrones 10101010 o 10010000 no son uniformes. Los patrones uniformes contienen la mayor parte de las texturas presentes en una imagen y en $\text{LBP}_{(8,1)}$ representan el 90 % según Ojala et al. [36]. Al usar esta extensión, todos los patrones uniformes tienen etiquetas distintas entre sí, y los patrones no uniformes se etiquetan de la misma forma. En $\text{LBP}_{(8,1)}$, de los 256 patrones posibles, 58 son uniformes, reduciendo la cantidad de etiquetas a 59 y el tamaño del histograma descriptor a un 23 % del original. Todos los patrones uniformes de $\text{LBP}_{(8,1)}$ se muestran en la Figura 2.6.

Además, existe una versión de LBP invariante a rotaciones. Éste es generado de igual manera que LBP uniforme, pero para cada patrón se busca el menor valor posible rotando el número binario circularmente. Por ejemplo, considerando el patrón 11000001, si es rotado en una posición hacia la derecha, se obtiene el patrón 11100000. Si se repite este proceso hasta obtener el menor

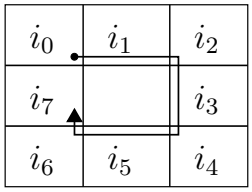


Figura 2.5: Orden de comparaciones para generar patrón LBP.

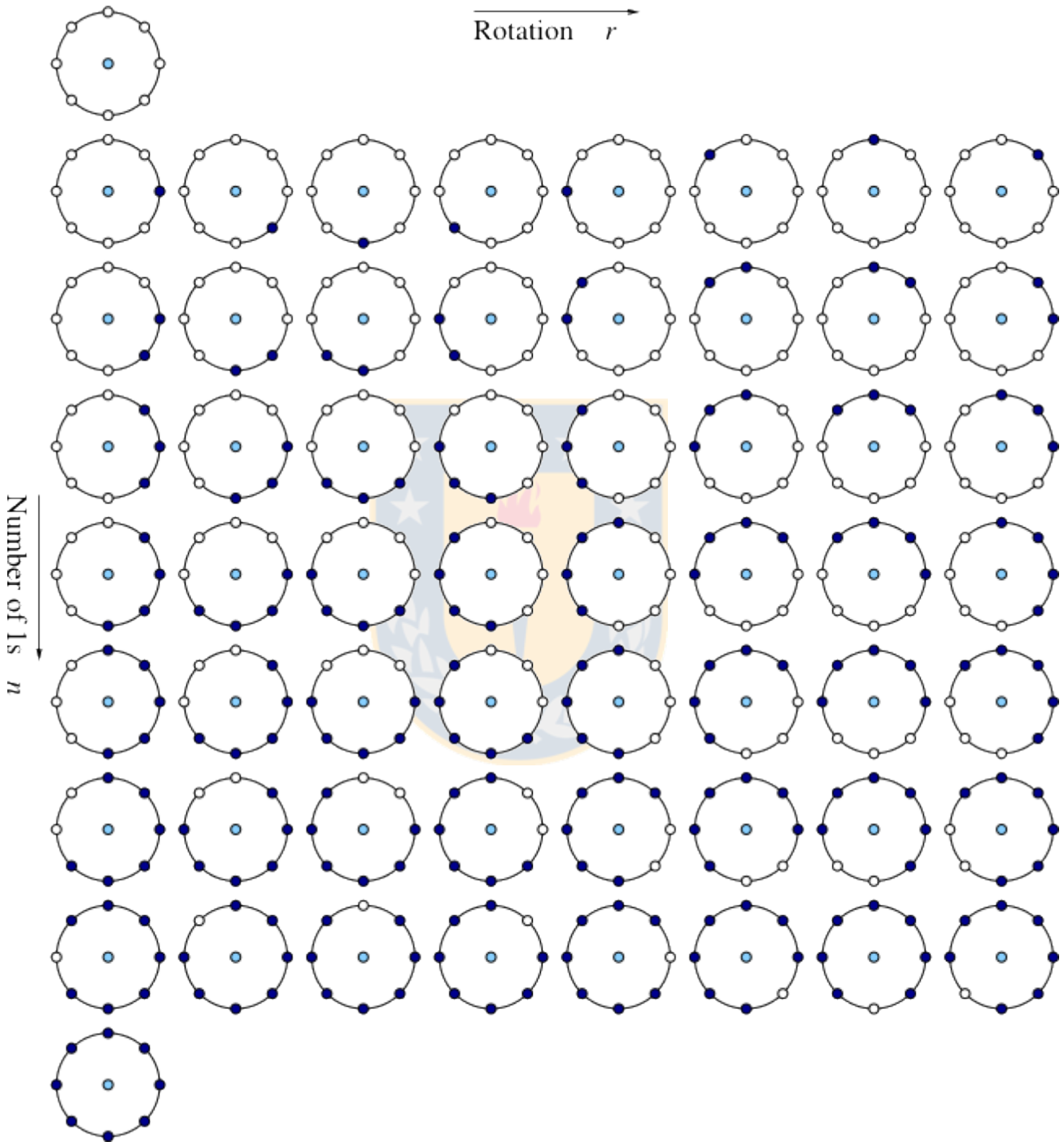


Figura 2.6: Patrones uniformes $LBP_{(8,1)}$ [1].

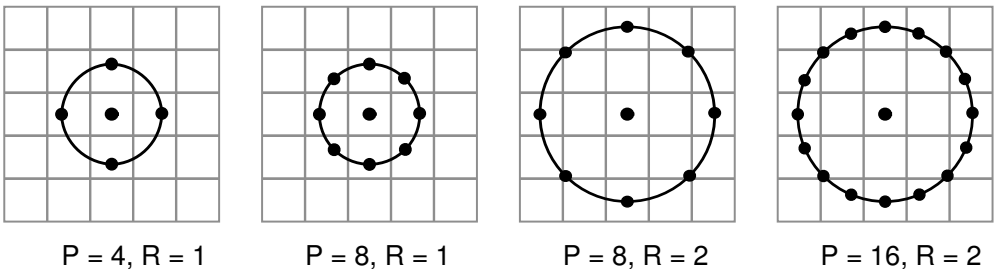


Figura 2.7: LBP en vecindarios de tamaño variable.

valor posible, el patrón final será 00000111. Así, para $LBP_{(8,1)}$ sólo existen 9 etiquetas invariantes a rotación, las que corresponden a la primera columna de la Figura 2.6.

Uno de los usos de LBP para detección de rostros fue propuesto por A. Hadid [8]. En este algoritmo se realiza la extracción de características combinando $LBP_{(8,1)}$ y $LBP_{(4,1)}$, en la Figura 2.8 se ejemplifica el resultado de la aplicación de estos operadores sobre una imagen infrarroja. La extracción de características se realiza generando un descriptor global de la imagen, que corresponde al histograma de $LBP_{(8,1)}$ uniforme. A éste se le añade una serie de descriptores local, para esto se divide la ventana en nueve regiones, como las que se indican en la Figura 2.9. Para cada de las regiones se calcula un histograma con $LBP_{(4,1)}$. Finalmente el descriptor de la ventana se genera concatenando los diez histogramas generados anteriormente. Así, el descriptor además de poseer información de las micro texturas presentes en la ventana, posee información espacial de éstas. El descriptor final tiene 203 muestras $(59 + 9 \times 16)$ y es independiente de la resolución de la ventana de interés considerada. Esto es particularmente útil, ya que al utilizar histogramas normalizados el problema de la búsqueda de rostros de diferentes escalas se reduce

(a) Imagen original (b) $LBP_{(8,1)}$ uniforme (c) $LBP_{(4,1)}$

Figura 2.8: Ejemplo del resultado del operador LBP con intensidades escaladas de 0 a 255.

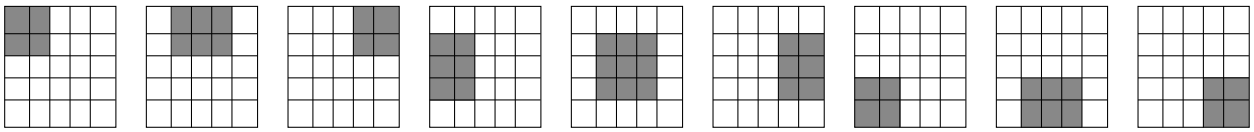


Figura 2.9: Regiones consideradas dentro de una imagen para creación de descriptor LBP.

a modificar el tamaño de la ventana de interés, con lo que se evita el redimensionamiento de la imagen para generar una imagen piramidal con distintas resoluciones.

En el Algoritmo 2 se indican los pasos a seguir para usar LBP en la extracción de características en el problema de detección de rostros.

Características MB-LBP

El operador Multi Block LBP fue propuesto por Zhang et al. [7] como una alternativa a LBP. Esta modificación captura información de macro texturas presentes en una imagen. Para esto, en lugar de utilizar la intensidad de píxeles en un vecindario, se utiliza el promedio de las intensidades dentro de una región de la imagen. Esto se ejemplifica en la Figura 2.10, donde cada rectángulo del vecindario de 3×3 regiones está compuesta por seis píxeles y el patrón MB-LBP es generado de igual forma que un patrón LBP.

Las características MB-LBP se utilizan en detección de rostros de forma muy similar a las características Haar. Sin embargo, las características MB-LBP pueden capturar más información y generan un vector de características de mucho menor dimensión [7]. Por ejemplo en una ventana de 24×24 píxeles existen 8.464 características Esto corresponde a cerca de una veinteava parte de las presentes al utilizar características Haar.

Algoritmo 2 Detección de rostros utilizando histograma LBP

1: Obtener un conjunto de imágenes de rostros y otro con imágenes sin rostros con un tamaño predeterminado.
2: Generar un histograma de $\text{LBP}_{(8,1)}$ uniforme para cada imagen.
3: Dividir cada imagen en 9 regiones solapadas (Fig. 2.9) generando un histograma de $\text{LBP}_{(4,1)}$ para cada una de ellas.
4: Concatenar todos los histogramas por imagen para crear el descriptor final.
5: Utilizar los descriptores de las imágenes de ambas clases para entrenar un detector facial con un algoritmo de clasificación.

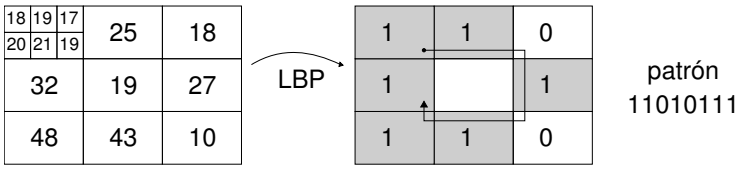


Figura 2.10: Representación de MB-LBP con 6 píxeles por región del vecindario de 3×3 de LBP.

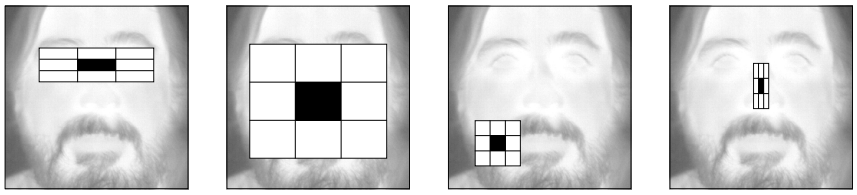


Figura 2.11: Ejemplo de uso de características MB-LBP.

La extracción de características utilizando MB-LBP, se realiza generando todas las posibles combinaciones de posiciones y tamaños dentro de la ventana de interés del vecindario de 3×3 regiones del operador. Para esto, el vecindario original con regiones de un píxel, es redimensionado y desplazado, adoptando todas las posibles combinaciones, también llamadas características, lo que se ejemplifica en la Figura 2.11. Para cada una de estas combinaciones se calcula la intensidad promedio por región utilizando imágenes integrales, tardando un tiempo constante en cada característica. Posteriormente se calcula el patrón LBP correspondiente y se crea el vector de características, que luego es utilizado como el descriptor de la ventana de interés. Cada elemento dentro del descriptor posee un valor entre 0 y 255, éstos son no métricos ya que sólo son representaciones de patrones presentes en la imagen. Por ejemplo, los valores 243 y 244 no tienen, necesariamente, ninguna relación relevante para el problema de clasificación, aun cuando sus representaciones son contiguas. Por esto no es posible utilizar clasificadores que utilicen umbrales o hiperplanos de decisión. La descripción del uso del MB-LBP se muestra en el Algoritmo 3.

Histogramas de gradientes orientados, HOG

El histograma de gradientes orientados fue propuesto por Dalal y Triggs [37] y originalmente fue utilizado para detección de peatones. Este algoritmo de detección de objetos consiste en generar una descripción basada en el gradiente de los píxeles. Esto se realiza creando un histograma que representa los ángulos de los píxeles ponderados por sus magnitudes.

Algoritmo 3 Detección de rostros utilizando características MB-LBP

1: Obtener un conjunto de imágenes de rostros y otro con imágenes sin rostros con un tamaño predeterminado.
2: Generar todas las posibles características para el tamaño de la ventana móvil.
3: Calcular imagen integral de cada imagen.
4: Obtener los patrones MB-LBP de cada característica y almacenarlos en el descriptor.
5: Utilizar los descriptores de las imágenes de ambas clases para entrenar un detector facial con un algoritmo de clasificación.

Antes de describir el algoritmo es necesario comprender la definición de las imágenes gradientes, las que son utilizadas para obtener los ángulos y magnitudes de cada píxel. Para cada imagen en escala de grises existen dos imágenes gradientes, las que representan el desplazamiento de los píxeles en ambos ejes espaciales con respecto a su vecindario. Cada una de ellas es creada realizando la resta entre el píxel siguiente y el anterior al píxel de interés, lo que se realiza para los dos ejes, lo que se ejemplifica en la Figura 2.12. Las imágenes generadas tienen dos píxeles menos de dimensión por eje, ya que no es posible calcular el gradiente en los bordes de la imagen. Un ejemplo de las imágenes gradientes resultantes se ve en la Figura 2.13. Posteriormente se calcula el ángulo y la magnitud de cada píxel, según:

$$\text{Magnitud} = \sqrt{i_x(x,y)^2 + i_y(x,y)^2}, \tag{2.6}$$

$$\text{Ángulo} = atan\left(\frac{i_y(x,y)}{i_x(x,y)}\right), \tag{2.7}$$

donde $i_x(x,y)$ e $i_y(x,y)$ corresponden a los valores del píxel (x,y) en las imágenes gradientes con desplazamiento en el eje X e Y respectivamente.

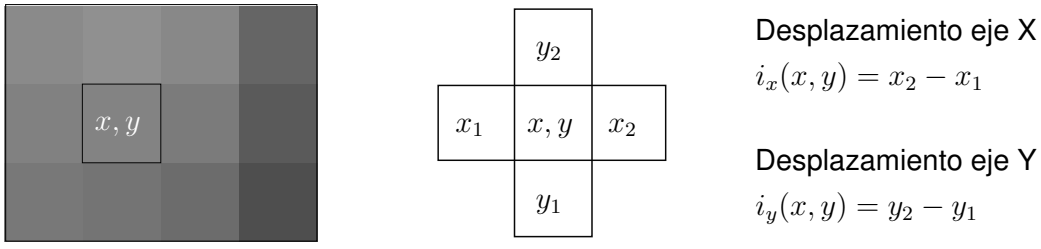
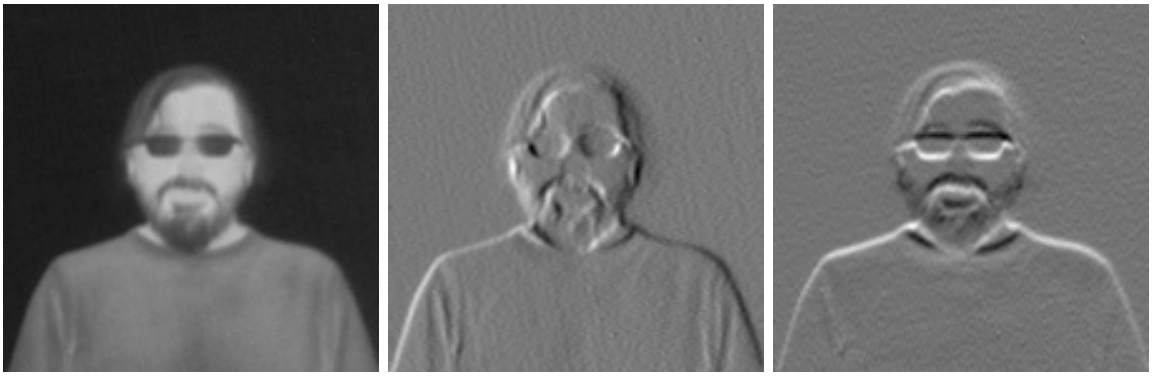


Figura 2.12: Generación de cada píxel de las imágenes gradientes, con sección de una imagen ampliada a la derecha.



(a) Imagen original (b) Desplazamiento en eje X (c) Desplazamiento en eje Y

Figura 2.13: Ejemplo imágenes gradientes.

La construcción del descriptor HOG se realiza calculando las imágenes gradientes de la ventana de interés, y las matrices de magnitud y ángulos a partir de ellas. Posteriormente la ventana es dividida en celdas de un tamaño definido, por ejemplo de 8×8 píxeles. Estas celdas son agrupadas en bloques de 2×2 celdas, las que son solapadas en una celda. Esto quiere decir que cada una de ellas es contenida por cuatro bloques. Excluyendo las de las esquinas que sólo son contenidas por un bloque y las de los bordes que son contenidas por dos bloque. Por ejemplo, para una ventana con 5 celdas verticales y 5 horizontales se tiene un total de 9 bloques.

Para cada una de las celdas dentro de la ventana de interés se calcula el histograma de ángulos sin signo, por lo que sólo se consideran ángulos entre 0 y 180 grados. A los ángulos entre 180 y 360 grados se le restan 180 grados para mapearlos dentro de los límites considerados. Cada una de las muestras del histograma considera un intervalo de 20 grados, por lo que para cada celda se genera un histograma de nueve muestras. La construcción de cada histograma es ponderada por el valor de magnitud de cada píxel. Esto significa que en lugar de incrementar en uno el valor de una muestra en cada ocurrencia de un ángulo, éste es incrementado en el valor de la

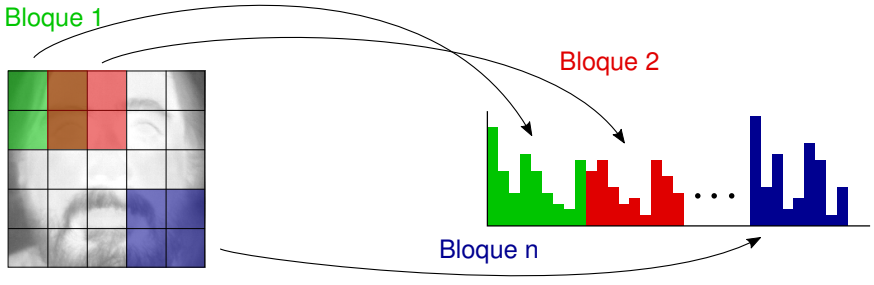


Figura 2.14: Ejemplo de creación de descriptor general HOG.

(a) Imagen original (b) Celdas de 4×4 píxeles (c) Celdas de 8×8 píxeles

Figura 2.15: Ilustración de gradientes generados con el algoritmo HOG, con dirección e intensidad según el ángulo y la magnitud calculada.

Algoritmo 4 Detección de rostros utilizando histograma HOG

1: Obtener un conjunto de imágenes de rostros y otro con imágenes sin rostros con un tamaño predeterminado.
2: Calcular las imágenes gradientes de cada imagen en el conjunto de entrenamiento.
3: Calcular las matrices de magnitud y ángulo para cada imagen.
4: Dividir cada imagen en celdas de 8×8 píxeles y calcular un histograma de los ángulos para cada una de ellas, considerando sólo ángulos de 0 a 180 grados y ponderando cada muestra por la magnitud del gradiente del respectivo píxel.
5: Agrupar las celdas en bloques de 2×2 celdas, solapando cada bloque en una celda (Fig. 2.14).
6: Concatenar los histogramas de cada celda por bloque y normalizarlos, generando los histograma de cada bloques con un tamaño de 4×9 muestras.
7: Concatenar los histogramas de todos los bloques para generar el descriptor final de la imagen.
8: Utilizar los descriptores de las imágenes de ambas clases para entrenar un detector facial con un algoritmo de clasificación.

magnitud del píxel en cuestión. Las muestras que se ubican en las fronteras de cada intervalo, son interpoladas, diviendo su magnitud entre ambas muestras aledañas. Posteriormente los histogramas de las celdas de cada bloque son concatenados y normalizados por bloque. Con esto se extrae información de cada celda y de cada bloque, generando una descripción invariante a cambios de iluminación globales y con una tolerancia a la rotación de 20°. Finalmente el descriptor de la ventana de interés se construye concatenando todos los histogramas de cada bloque. Así, para una ventana con 5 celdas verticales y 5 horizontales se obtiene un descriptor de 576 elementos. El proceso completo se ejemplifica en la Figura 2.14.

Una forma gráfica de visualizar la descripción generada por el algoritmo HOG, es creando una imagen con el ángulo y magnitud resultante de cada celda. Esto se ilustra en la Figura 2.15,

donde cada celda muestra la dirección resultante, con una intensidad de color dependiente de la magnitud de la celda. Esta imagen es sólo ilustrativa y no se utiliza en ninguna parte del procesamiento.

En el Algoritmo 4 se muestran los pasos a seguir para utilizar el algoritmo HOG como descriptor en el problema de detección facial.

HOG simplificado

El algoritmo HOG, al igual que el resto de los descritos anteriormente, realizan un el análisis usando ventanas deslizantes requiere que para cada región se calcule nuevamente el descriptor completo, pues al utilizar celdas, el desplazamiento en un píxel produce que todos los gradientes e histogramas que representan a la región cambien. Otro de los problemas, el que se comparte con el resto de los algoritmos descritos anteriormente, es la necesidad de redimensionar la imagen y volver a realizar el cálculo de los descriptores para cada tamaño de análisis. Ambos problemas se traducen en un muy alto número de operaciones, pues ninguno de los cálculos realizados para una región o un cierto tamaño de análisis pueden ser utilizados en otro, lo que se traduce en un consumo de tiempo importante.

Las simplificaciones propuestas al algoritmo apuntan a: reducir el número de operaciones necesarias para realizar el cálculo de los histogramas por celda; realizar desplazamientos dentro de la imagen sin la necesidad de volver a calcular los histogramas; y a reutilizar los histogramas generados para un tamaño de análisis en otros. Para realizar esto se modificó levemente la generación de los histogramas por celda, remplazando la interpolación, en casos de ángulos múltiplos de 20°, por desigualdades no estrictas, seleccionando la muestra correspondiente del histograma como una función por tramos, según:

$$bin = \begin{cases} 0 & \text{si } 0 \leq angle \leq 20 \\ 1 & \text{si } 20 < angle \leq 40 \\ \vdots & \vdots \qquad \vdots \\ 7 & \text{si } 140 < angle \leq 160 \\ 8 & \text{si } 160 < angle \leq 180 \end{cases}, \tag{2.8}$$

donde *bin* corresponde a la muestra asociada al ángulo *angle* dentro del histograma de una celda.

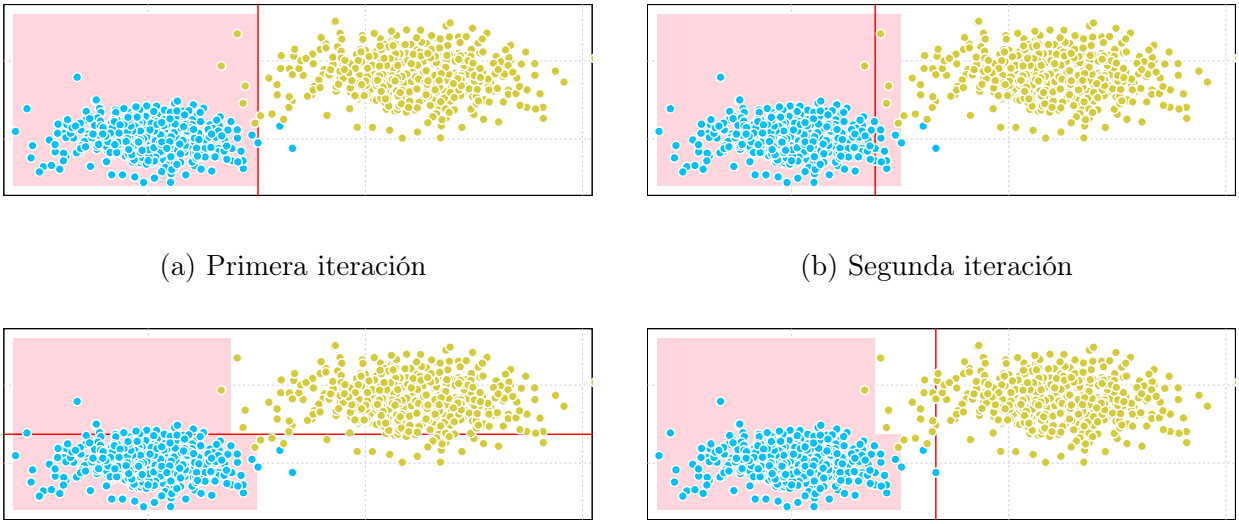
Además de esta modificación, se escogió realizar una búsqueda basada en celdas [38], en lugar de una basada en ventanas, con la que el desplazamiento sobre la imagen se realiza de celda en celda. Esto permite que los histogramas de celdas y de bloques sean generados sólo en una ocasión para un tamaño de imagen. Sin embargo, al realizar esto, se pierde granularidad en la búsqueda, pudiendo reducir la calidad de los resultados de detección, sobre todo en regiones de gran tamaños de análisis, donde la separación entre una y otra celda corresponden a mucha información. Para resolver esto se genera una matriz de histogramas de celdas de menor tamaño que las que realmente se utilizarán. Por ejemplo, si el análisis se realiza considerando regiones de 40×40 píxeles, con celdas de 8×8 píxeles, se utilizarán 4×4 bloques de celdas para representar toda la región. Al realizar una búsqueda en base a celdas, en este caso, cada desplazamiento de la región de análisis será de 8 píxeles, sin embargo, con el enfoque planteado, las celdas serán generadas inicialmente en 4×4 píxeles, permitiendo un desplazamiento en la imagen de ésta cantidad de píxeles. Reduciendo la generación del descriptor, para obtener el tamaño de 8×8 píxeles, a la suma de las 4 celdas que comparten su área en la imagen.

La última consideración realizada para optimizar el funcionamiento del algoritmo HOG, corresponde a que, al utilizar una búsqueda basada en celdas, no es necesario redimensionar las imágenes para realizar el análisis en regiones de mayor tamaño. Ya que, es posible utilizar el hecho de que sumar celdas aledañas equivale a generar celdas que representen mayor área de la imagen. En este caso ocurre el mismo problema anterior, porque las celdas de mayor tamaño generadas tendrán una separación muy grande entre ellas. La solución a esto es la misma expresada en el caso anterior, ya que, manteniendo la celda de tamaño original (4×4 píxeles en el ejemplo), toda las celdas de mayor tamaño tendrán siempre una separación de 4 píxeles. Además, con esto se permite que el incremento de área de cada celda sea también de 4 en 4 píxeles. Así, al utilizar bloques de 4×4 celdas, el tamaño de la región se incrementará en 20 píxeles en cada etapa, permitiendo que la generación de las nuevas celdas se realice de forma iterativa, considerando las celdas del último tamaño analizado y sumando las celdas aledañas de tamaño original a éstas para incrementar su área de representación.

2.2.2. Clasificadores

Adaboost

Los algoritmos de boosting se basan en la creación de un clasificador fuerte mediante la combinación de varios clasificadores débiles. Esto se realiza utilizando una parte de las caracte-



(a) Primera iteración

(b) Segunda iteración

(c) Tercera iteración

(d) Cuarta iteración

Figura 2.16: Ejemplo de funcionamiento de Adaboost para 4 iteraciones del algoritmo.

rísticas de los descriptores del conjunto de entrenamiento, y seleccionando las que menor error generen para cierto clasificador débil utilizado, lo que permite crear un clasificador que no utilice los descriptores en su totalidad, sino sólo parte de ellos.

Uno de los métodos más utilizados de boosting es el algoritmo Adaboost (*adaptative boosting*) propuesto por Freund y Sachapire [39]. En Adaboost la acumulación de clasificadores débiles y la selección de las características más representativas, se realiza asignado pesos a las muestras del conjunto de entrenamiento dependientes del error que éstas generen. Con esto se asignan pesos mayores a las muestras mal clasificadas, forzando al algoritmo a generar clasificadores débiles que disminuyan el error de clasificación en estas muestras. Este proceso se repite M veces, según la tasa de acierto que se requiera para la aplicación, con lo que se genera un clasificador fuerte que utiliza las M características que mejor clasifican los descriptores.

El proceso de creación de un clasificador fuerte se ilustra en la Figura 2.16, donde se muestran las primeras cuatro iteraciones del algoritmo para un conjunto de datos de ejemplo con distribución gaussiana. En ésta se puede ver la progresión del clasificador fuerte, que finalmente genera una función de decisión en base a distintas regiones con diferentes pesos asociado a cada una de ellas. Así, el signo de la sumatoria de los pesos de cada clasificador débil, indicados por las líneas rojas en Figura 2.16, indica la pertenencia de dicha región a una u otra clase.

En este trabajo se utilizó una variación de Adaboost, llamada Gentle Adaboost [40]. En esta variación se utiliza minimización de pasos de Newton para la actualización de los pesos en cada iteración, esto asegura que éstos sean los óptimos para la siguiente iteración del algoritmo. A diferencia de otras implementaciones de Adaboost, en Gentle Adaboost es necesario agregar la información sobre el error durante la creación de los clasificadores débiles. Ya que el algoritmo los añade al clasificador fuerte sin realizar una ponderación de ellos según el error de clasificación obtenido.

En el Algoritmo 5 se describe el proceso realizado durante el entrenamiento de Gentle Adaboost. Para esto se obtiene un conjunto de entrenamiento $(x_1, y_1)\ \ldots\ (x_N, y_N)$, donde x_i corresponde al vector de características de dimensión N e y_i, con valores entre $(+1, -1)$, corresponde a la clase asociada al i-ésimo vector de características. Se inicializan todos los pesos ω_i con el mismo valor $(1/N)$, esto ya que a priori no se conoce el aporte que realiza cada características al proceso de clasificación. Posteriormente, para cada una de las características se construye un clasificador débil y se calcula el error cuadrático correspondiente a cada una de ellas

$$J_{wse} = \sum_{i=1}^{N} w_i (y_i - f_m(x_i))^2, \tag{2.9}$$

donde J_{wse} corresponde al error cuadrático ponderado por los pesos w, w_i al peso asignado al i-ésimo elemento dentro del conjunto de entrenamiento, y_i a la clase del i-ésimo elemento del conjunto de entrenamiento, f_m corresponde a un clasificador débil de la iteración m del algoritmo. En base a lo anterior se escoge la característica que genere el menor error de clasificación. Además, el clasificador débil correspondiente es agregado al clasificador fuerte

$$F(x) \leftarrow F(x) + f_m(x), \tag{2.10}$$

donde $F(x)$ corresponde al clasificador fuerte y $f_m(x)$ al clasificador débil. Luego todos los pesos son actualizados según:

$$w_i = w_i \cdot e^{-y_i \cdot f_m(x_i)}, \qquad i = 1, \ldots, N, \tag{2.11}$$

donde w_i es el peso asignado al i-ésimo termino del conjunto de entrenamiento, $f_m(x_i)$ al clasificador débil asignado al mismo termino e y_i a la clase de pertenencia del i-ésimo termino. Finalmente los pesos son normalizados, haciendo que la suma total de los pesos sea igual a uno

Algoritmo 5 Gentle Adaboost

1: Obtener un conjunto de entrenamiento de N elementos $(x_1, y_1)\ \ldots\ (x_N, y_N)$.
2: Comenzar con pesos $w_i = 1/N, i = 1, \ldots, N$ y clasificador final $F(x) = 0$.
3: **for** $m = 1, 2, \ldots, M$ **do**
4: Encontrar el clasificador débil $f_m(x)$ que utilizando sólo una característica produzca el menor error cuadrático ponderado (Ecuación 2.9).
5: Actualizar el clasificador fuerte añadiendo el clasificador débil y la característica que éste utiliza (Ecuación 2.10).
6: Actualizar los pesos (Ecuación 2.11) y normalizarlos ($\sum w_i = 1$).
7: **end for**
8: El clasificador final estará definido por la combinación de los M clasificadores débiles (Ecuación 2.12).

($\sum w_i = 1$). El proceso anterior se repite M veces, generando el clasificador fuerte

$$F(x) = sign\left(\sum_{m=1}^{M} f_m(x)\right), \tag{2.12}$$

donde $F(x)$ es el clasificador final de la característica X, $fm(x)$ es un clasificador débil de la misma característica y M es el número de características utilizadas.

Para realizar el proceso de clasificación utilizando Gentle Adaboost, se generan sólo las características seleccionadas durante el entrenamiento de los elementos de prueba. Luego se calcula la sumatoria de la salida de los clasificadores débiles respectivos a cada característica. Finalmente, la clasificación se realiza según el signo de la sumatoria. Esto quiere decir que si el valor es mayor o igual a cero, la muestra se clasifica dentro de la clase $+1$, en caso contrario se clasifica como -1. Si bien en el algoritmo se define el valor cero como limite entre las clases, es posible modificar este umbral para cambiar la cantidad de falsos positivos y falsos negativos. Aumentando o disminuyendo la cantidad de clasificadores débiles que deben clasificar dentro de una u otra clase a la muestra.

Una parte importante del entrenamiento de Adaboost es la generación de los clasificadores débiles. Éstos son llamados débiles ya que al ser utilizados por si solos no son capaces de realizar una buena clasificación de los datos. Comúnmente se utilizan umbrales de decisión como clasificadores débiles en Adaboost. El diseño de estos clasificadores es simple, ya que sólo se requiere encontrar el valor y sentido de la inecuación que menor error genera durante la clasificación. Sin embargo al utilizar Gentle Adaboost no se incluye información de los errores generados en el algoritmo, por lo que es necesario incluir información de ello en los clasificadores débiles. Por

esto, se crearon clasificadores débiles basados en árboles de decisión de dos ramas [41]. Donde en lugar de utilizar umbrales, se divide el rango de valores posibles en un número arbitrario de segmentos, asignando un peso a cada uno de ellos. Así la salida del clasificador es:

$$f_m(x^k) = \begin{cases} a_{+1} & \text{si } p \cdot x^k > p \cdot u \\ a_{-1} & \text{en otro caso} \end{cases}, \tag{2.13}$$

donde x^k corresponde a la k-ésima característica de vector x, p al sentido de la inecuación y u al umbral del clasificador débil. Además los términos a_{+1} y a_{-1} se definen como:

$$a_{+1} = \frac{\sum_i^N w_i \cdot y_i \cdot h(x_i^k)}{\sum_i^N w_i \cdot h(x_i^k)}, \tag{2.14}$$

$$a_{-1} = \frac{\sum_i^N w_i \cdot y_i \cdot (1 - h(x_i^k))}{\sum_i^N w_i \cdot (1 - h(x_i^k))}, \tag{2.15}$$

donde N es número de elementos en el conjunto de entrenamiento, w_i corresponde al i-ésimo peso, y_i a la clase del i-ésimo elemento y x_i^k corresponde a la k-ésima característica del i-ésimo elemento del conjunto de entrenamiento. Además $h(x)$ corresponde indicador de pertencia a cada clase, dado por:

$$h(x) = \delta(p \cdot x > p \cdot u), \tag{2.16}$$

donde $\delta(x)$ corresponde a una función *booleana*, en la que si la condición x es verdadera, la salida de la función es 1, en caso contrario la salida es 0.

La salida del clasificador para una característica, dada por los términos a_{+1} y a_{-1}, corresponde a la sumatoria normalizada de los pesos de las muestras bien clasificadas en el caso de a_{+1} y mal clasificadas en a_{-1}. Esto implica que mientras más muestras tengan la clasificación correcta, mayor será el valor entregado, lo que causará una mayor relevancia del clasificador débil en la clasificación final.

Los clasificadores débiles basados en umbrales de decisión sólo pueden ser utilizados en algoritmos donde los valores de las características sean métricos. Esto se refiere a que exista una relación entre un valor en particular y su vecindario. En el caso particular de MB-LBP, las características toman valores entre 0 y 255, donde cada uno de éstos representa un tipo de textura en particular y sólo son utilizados como etiquetas de identificación. Por esto, no es

posible utilizar los clasificadores débiles descritos anteriormente. Por lo que se utiliza un árbol de decisión multirama como clasificador débil para Gentle Adaboost [7], dado por:

$$f_m(x^k) = \begin{cases} a_0 & \text{si } x^k = 0 \\ & \dots \\ a_j & \text{si } x^k = j \\ & \dots \\ a_{255} & \text{si } x^k = 255 \end{cases}, \tag{2.17}$$

donde x^k corresponde a la k-ésima característica del vector x y a_j corresponde al valor asignado a la rama j del árbol de decisión. Los valores de cada una de las 256 ramas se obtienen, en base a la cantidad de ocurrencias del patrón j en la característica k ponderada por los pesos w asignados a cada elemento, lo que corresponde a:

$$a_j = \frac{\sum_i^N w_i \cdot y_i \cdot \delta(x_i^k == j)}{\sum_i^N w_i \cdot \delta(x_i^k == j)}, \tag{2.18}$$

donde N es número de elementos en el conjunto de entrenamiento, w_i e y_i corresponden al peso y clase respectivamente del i-ésimo elemento y x_i^k corresponde a la k-ésima característica del i-ésimo elemento del conjunto de entrenamiento. Además $\delta(x)$ corresponde a una función booleana cuya salida es 1 si la condición del argumento x es verdadera, y 0 en caso contrario.

Máquina de vectores de soporte, SVM

Las máquinas de vectores de soporte, fueron desarrolladas por Vapnik y Lerner [42] y corresponden a un algoritmo de aprendizaje supervisado utilizado en problemas de clasificación y regresión. En SVM se busca dividir el espacio de los datos de entrenamiento con un hiperplano que minimice el error de clasificación y maximice el margen entre el hiperplano y los datos de ambas clases, esto se ejemplifica en la Figura 2.17.

El clasificador SVM es binario y la correspondencia a una u otra clase, durante el proceso de clasificación, se define según la posición del elemento a clasificar respecto al hiperplano de decisión. En el caso más simple, donde los datos de entrenamiento son linealmente separables, la salida del clasificador corresponde al signo de la distancia, no normalizada, entre el elemento

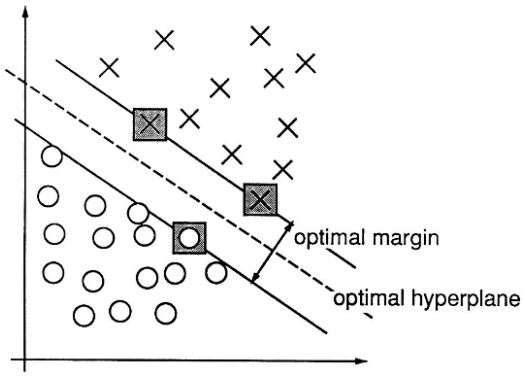


Figura 2.17: Ejemplo de separación en espacio de 2 dimensiones, las muestras marcadas en con rectángulos grises definen el margen máximo de separación entre las dos clases y la línea segmentada el hiperplano óptimo calculado, imagen de [2].

a clasificar y el hiperplano dado por:

$$f(x) = sign(w^T \cdot x + b), \tag{2.19}$$

donde x corresponde al elemento a clasificar, w al vector normal al hiperplano de decisión y b el desplazamiento del hiperplano respecto al origen.

Para que SVM funcione apropiadamente es necesario que las clases sean separables por el hiperplano. Como esto no ocurre en todos los casos, es posible utilizar funciones de *kernel* no lineales [43]. Con esto se proyectan los datos a un espacio de características de mayor dimensionalidad, donde si pueden ser separados por un hiperplano. El uso de *kernel* no lineales, si bien posibilita un mejor desempeño del clasificador, requiere de más operaciones durante la etapa de entrenamiento y de clasificación, ya que es necesario el cálculo de estas funciones sobre los datos. Los *kernel* más utilizados en SVM son el *kernel* lineal

$$K(x_i, x) = x_i^T \cdot x, \tag{2.20}$$

el *kernel* polinomial

$$K(x_i, x) = (x_i^T \cdot x + 1)^d, \tag{2.21}$$

y el *kernel* de base radial gaussiana

$$K(x_i, x) = \exp(-\gamma \|x_i - x\|^2), \tag{2.22}$$

donde, en todos los casos, x_i es el i-ésimo vector de soporte, x es el elemento a proyectar, d y γ son parámetros de los respectivos *kernels*.

Con el uso de *kernels* la función de decisión se generaliza a

$$f(x) = \sum_{i=1}^{m} \alpha_i K(x_i, x) + b, \tag{2.23}$$

donde x_i es el i-ésimo vector de soporte, a_i es el peso asociado a x_i, m es el número de vectores de soporte, $K(x_i, x)$ corresponde al *kernel* utilizado y b al desplazamiento del hiperplano de decisión respecto al origen. Donde si se reemplaza el *kernel* lineal se obtiene la Ecuación 2.19, donde el vector w normal al hiperplano equivale a $\sum a_i \cdot x_i$.

2.3. Reconocimiento de rostros

El proceso de reconocimiento de patrones consiste en dos etapas fundamentales, la extracción de características y la clasificación. La extracción de características es realizada para obtener una descripción de la imagen, conservando sólo la información importante que ésta presenta. La clasificación compara la descripción generada con la generada previamente para un conjunto de referencia, que corresponde a la base de datos de los patrones.

La extracción de características puede ser realizada con distintos algoritmos. Éstos se dividen en dos tipos, los métodos holísticos y los basados en características locales. Los métodos holísticos consideran la imagen a clasificar como un todo, intentando obtener la mayor cantidad de información de éstas para generar el descriptor. Entre los métodos más conocido de este tipo se encuentran PCA y LDA. Los métodos basados en características locales extraen información espacial de cada imagen, generando un descriptor de forma similar a como un humano lo haría. En el caso del reconocimiento de rostro, el descriptor generado tiene información separada de cada área de interés, como los ojos, nariz, boca o vello facial. Uno de los métodos más conocidos de este tipo es LBP.

La etapa de clasificación puede ser realizada con casi cualquier tipo de clasificador, como

redes neuronales o clasificadores bayesianos. También es posible utilizar un clasificador más simple, como la clasificación por vecino más cercano.

2.3.1. Extracción de características

Análisis de componentes principales, PCA

El análisis de componentes principales (PCA) es la técnica más común de reducción de dimensionalidad. Es un método lineal no supervisado, que reduce la redundancia y maximiza la varianza entre las muestras. Esto genera un subespacio de igual o menor dimensión al espacio original, al que son proyectados los datos como combinaciones lineales de los vectores de proyección generados durante el entrenamiento del sistema.

El entrenamiento del sistema se realiza para calcular los coeficientes de la matriz de proyección. Para esto es necesario calcular la matriz de covarianza

$$S_T = \sum_{i=1}^{N} (x_i - \mu)(x_i - mu)^T, \tag{2.24}$$

donde μ corresponde a la media de todos los datos de entrenamiento, N a la cantidad total de elementos en el conjunto de entrenamiento y x_i al i-ésimo elemento de los datos de entrenamiento centrados, lo que corresponde a restarles la media de todo el conjunto.

Como la proyección es lineal, la matriz de covarianza en el subespacio proyectado estará dada por:

$$S_{T_y} = W^T \cdot S_T \cdot W, \tag{2.25}$$

donde W corresponde a la matriz de proyección lineal y S_T a la matriz de covarianza en el espacio original.

En PCA la matriz de covarianza proyectada es calculada para maximizar la varianza de las muestras. Para esto se debe resolver el problema de optimización y encontrar

$$W_{opt} = argmax|W^T S_T W|, \tag{2.26}$$

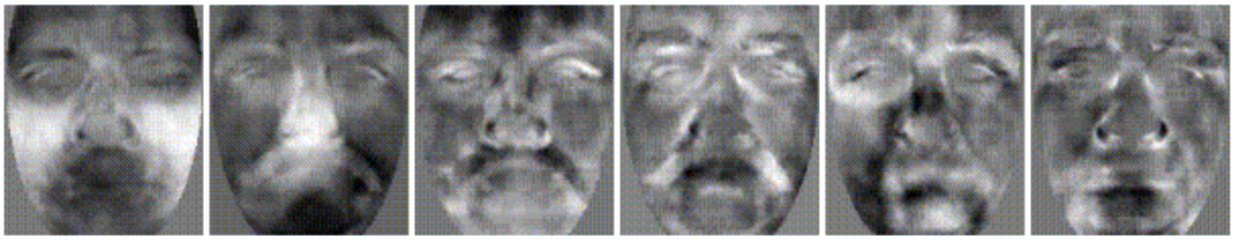


Figura 2.18: Ejemplo de Eigenfaces [3].

donde W corresponde a una matriz de proyección lineal a un subespacio de menor dimensión, y W_{opt} corresponde a la matriz de proyección óptima.

El resultado de W_{opt} es la matriz de proyección formada por los vectores propios con mayores valores propios asociados [44], cuya dimensión es definida según la representatividad deseada para la aplicación.

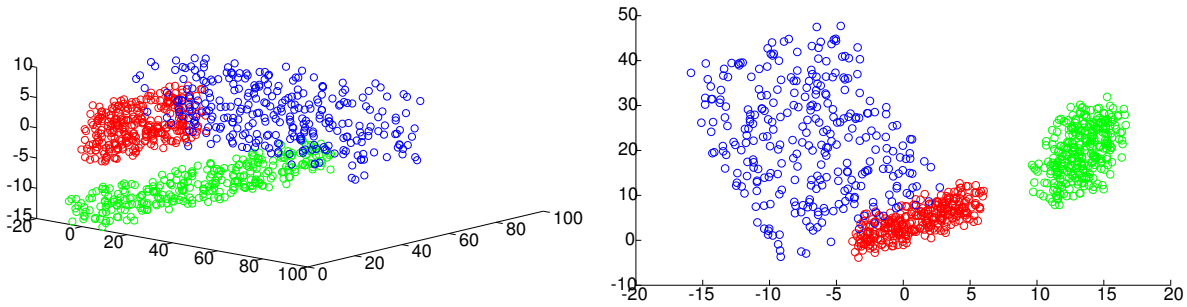
Finalmente, cada nuevo dato es proyectado al subespacio lineal creado para poder clasificarlo. Así, considerando un conjunto de imágenes $x_0,\ x_2 \dots x_N$, donde cada x_i es de dimensión $1 \times N^2$ correspondiente a una imagen de $N \times N$ píxeles. Se tiene que la proyección lineal y_i será:

$$y_i = W^T \cdot x_i \qquad i = 0, 2, \dots, N, \tag{2.27}$$

donde y_i corresponde a la i-ésima proyección lineal asociada al i-ésimo elemento de los datos de prueba x_i, N corresponde al número de imágenes en el conjunto de prueba, y W corresponde a la matriz de proyección lineal.

El uso de PCA en reconocimiento de rostros es conocido como Eigenfaces [45]. Un ejemplo de esto se muestra en la Figura 2.18, donde cada imagen corresponde a un vector de proyección perteneciente a la matriz de proyección generada durante el entrenamiento. Cada vector tiene una dimensión de $1 \times N^2$, correspondientes a una imagen de $N \times N$ píxeles.

El algoritmo 6 muestra el funcionamiento de PCA, describiendo las etapas de entrenamiento y prueba del algoritmo. En la etapa de entrenamiento se calcula la matriz de covarianza (S_T) de los datos del conjunto de entrenamiento, con la que se resuelve el problema de optimización obteniendo la matriz de proyección W_{opt} al subespacio lineal que maximiza la varianza entre las muestras. Los datos del conjunto de entrenamiento son proyectados a este subespacio, escogiendo a lo menos un elemento por clase para generar el conjunto de referencia. En la etapa de prueba los datos son proyectados al nuevo subespacio lineal, realizando el producto punto entre éstos



(a) Espacio original 3-D

(b) Proyección lineal en nuevo subespacio 2-D

Figura 2.19: Ejemplo de proyección LDA para tres clases.

y la matriz de proyección. Posteriormente es posible clasificar las muestras.

Análisis de discriminante lineal, LDA

El análisis de discriminante lineal (LDA), al igual que PCA, realiza una proyección en un subespacio de menor dimensión, pero a diferencia de PCA, LDA es un método supervisado, lo que quiere decir que durante el entrenamiento del algoritmo la clase de cada imagen es conocida. Gracias a esto, LDA además de maximizar la varianza entre clases, minimiza la varianza dentro de cada clase, proyectando los datos a un espacio que maximiza la correcta clasificación.

El procedimiento de este algoritmo es similar a PCA. Se define una matriz de covarianza

Algoritmo 6 Análisis de componentes principales (PCA)

1: Obtener un conjunto de entrenamiento desde la base de datos.
2: Centrar los datos restando la media del conjunto de entrenamiento.
3: Calcular la matriz de covarianza S_T (Ecuación 2.24)
4: Obtener los vectores y valores propios de la matriz de covarianza.
5: Escoger los vectores propios con mayores valores propios asociados, el número a escoger corresponderá a la nueva dimensión de las muestras.
6: Proyectar el conjunto de entrenamiento al nuevo subespacio y escoger al menos un elemento por clase para el conjunto de referencia.
7: Proyectar el conjunto de prueba al nuevo subespacio y clasificar las muestras.

entre clases

$$S_B = \sum_{j=1}^{C} n_j(\mu_j - \mu)(\mu_j - \mu)^T \tag{2.28}$$

y una matriz de covarianza al interior de las clases

$$S_W = \sum_{j=1}^{C} \sum_{i=1}^{n_j} (x_{ij} - \mu_j)(x_{ij} - \mu_j)^T, \tag{2.29}$$

donde

$$\mu_j = \frac{1}{n_j} \sum_{k=1}^{n_j} x_k \tag{2.30}$$

es la media de todos los datos del conjunto de entrenamiento, μ_j es la media de los datos pertenecientes a la clase j, n_j es la cantidad de datos de la clase j en el conjunto de entrenamiento, x_{ij} es el i-ésimo elemento de la clase j, y C es la cantidad de clases utilizadas para el entrenamiento del algoritmo.

Como el objetivo de LDA es maximizar la varianza entre las diferentes clases (S_B) y minimizar la varianza dentro de cada clase (S_W), la matriz de proyección al nuevo subespacio corresponde a la solución del problema de optimización

$$W_{opt} = argmax \frac{|W^T S_B W|}{|W^T S_W W|}, \tag{2.31}$$

donde S_B y S_W corresponden a las matrices de covarianza entre clases y al interior de las clases respectivamente; W corresponde a la matriz de proyección lineal y W_{opt} a la matriz de proyección óptima, que maximiza la separación entre clases y la minimiza al interior de las clases.

La solución de este problema de optimización, al igual que en caso de PCA, corresponde a la matriz formada por los vectores propios con mayores valores propios asociados de la matriz S_B/S_W.

El Algoritmo 7 describe el funcionamiento de LDA, para las etapas de entrenamiento y prueba. En el entrenamiento, se calculan las matrices de covarianza entre clases S_B y dentro de las clases S_W. Luego se resuelve el problema de optimización, con lo que se genera la matriz de proyección W_{opt} que maximiza la varianza entre diferentes clases y la minimiza al interior de cada clases. Posteriormente, se realiza la proyección de los datos de entrenamiento, escogiendo

a lo menos un elemento por clase para generar el conjunto de referencia. En la etapa de prueba los datos son proyectados al subespacio lineal, tal como en PCA, realizando el producto punto entre éstos y la matriz de proyección W_{opt}. La clasificación se realiza comparando los datos proyectados con los del conjunto de referencia.

En la mayoría de los casos, la cantidad de imágenes es menor a la cantidad de píxeles de cada imagen, por lo que la matriz de covarianza S_W es singular y la solución planteada no existe. Por esto, es necesario aplicar PCA con anterioridad, para reducir la dimensionalidad hasta, a lo más, la cantidad de las imágenes del conjunto de entrenamiento menos la cantidad de clases. Tras esto se aplica LDA, con el que se pueden obtener proyecciones de a lo más el número de clases menos uno. Finalmente la matriz de proyección óptima

$$W_{opt}^T = W_{LDA}^T W_{PCA}^T \tag{2.32}$$

estará dada por la multiplicación entre las matrices de proyección de PCA

$$W_{PCA} = argmax|W^T\ S_T\ W| \tag{2.33}$$

y la matriz de proyección de LDA

$$W_{LDA} = argmax\frac{|W^T\ W_{PCA}^T\ S_B\ W_{PCA}\ W|}{|W^T\ W_{PCA}^T\ S_W\ W_{PCA}\ W|}, \tag{2.34}$$

donde W_{PCA} es la matriz de proyección generada durante el entrenamiento de PCA y que tiene hasta $N - C$ vectores de proyección, con N imágenes y C clases en el entrenamiento; S_T es la matriz de covarianza del conjunto de entrenamiento; S_B y S_W son las matrices de

Algoritmo 7 Análisis de discriminante lineal (LDA)

1: Obtener un conjunto de entrenamiento desde la base de datos.
2: Centrar los datos restando la media del conjunto de entrenamiento.
3: Calcular la matriz de covarianza entre clases (Ecuación 2.28) y dentro de cada clase (Ecuación 2.29).
4: Resolver el problema de optimización (Ecuación 2.31), obteniendo los vectores propios con mayores valores propios asociados de la matriz S_B/S_W.
5: Proyectar el conjunto de entrenamiento al nuevo subespacio y escoger al menos un elemento por clase para el conjunto de referencia.
6: Proyectar el conjunto de prueba al nuevo subespacio y realizar la clasificación.

Figura 2.20: Ejemplo de Fisherfaces [4].

covarianza entre clases y al interior de las clases respectivamente; W corresponde a la matriz de proyección lineal y W_{opt} a la matriz de proyección óptima. Aplicar la matriz de proyección final W_{opt}, corresponde a proyectar los datos al subespacio generado por PCA y esto proyectarlo al subespacio de LDA.

La aplicación de LDA en reconocimiento de rostros se denomina Fisherfaces. Un ejemplo de esto se muestra en la Figura 2.20, donde cada imagen corresponde a un vector de proyección de la matriz W_{opt} generada en el entrenamiento del algoritmo. Cada uno de éstos tiene una dimensión de $1 \times N^2$, los que corresponden a una imagen de $N \times N$ píxeles.

Algoritmo Ahonen

El algoritmo Ahonen utiliza LBP para realizar la extracción de características en reconocimiento de rostros [27]. Como se muestra en la Figura 2.21, el algoritmo primero aplica LBP a una imagen facial, creando una nueva imagen donde cada píxel corresponde al patrón LBP respectivo. La imagen resultante es dividida en regiones rectangulares no solapadas, y se calcula un histograma descriptor por cada una de éstas. Todos los histogramas son concatenados

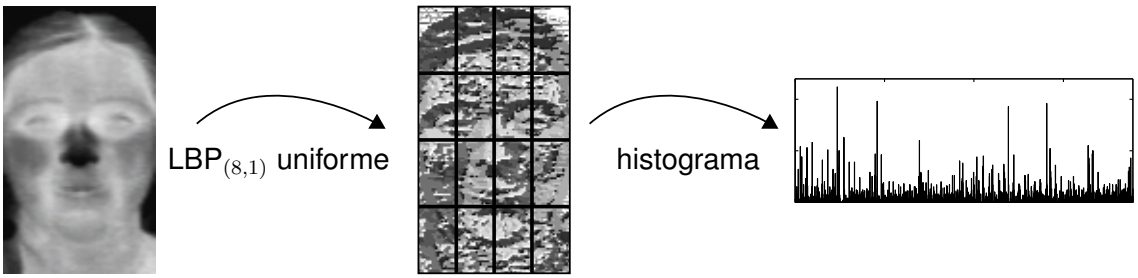


Figura 2.21: Algoritmo Ahonen.

Algoritmo 8 Algoritmo Ahonen

1: Seleccionar el conjunto de referencia.
2: Aplicar LBP uniforme a todas las imágenes del conjunto de referencia.
3: Dividir las imágenes en regiones no solapadas.
4: Calcular el *histograma espacialmente mejorado* de cada imagen.
5: Escoger al menos un elemento por clase para el conjunto de referencia.
6: Realizar el mismo proceso anterior para todas las imágenes de prueba y realizar la clasificación.

generando un *histograma espacialmente mejorado* (*spatially enhanced histogram*), que contiene información espacial de las texturas extraídas por LBP. El vector generado presenta una alta dimensionalidad, la que depende de la cantidad de regiones en que se divide la imagen y de la cantidad de etiquetas posibles para el LBP utilizado. Sin embargo, es independiente del tamaño de la imagen. Por ejemplo, para una imagen dividida en 64 regiones (8×8) y usando $\text{LBP}_{(8,1)}$ uniforme, la dimensión del histograma generado es 3,776.

El algoritmo 8 describe el proceso de clasificación utilizando el algoritmo Ahonen. Éste, a diferencia de los dos métodos anteriores, no necesita de una etapa de entrenamiento, ya que el conjunto de referencia se genera sólo aplicando el algoritmo a las imágenes seleccionadas. Esto corresponde a aplicar LBP uniforme a todas las imágenes del conjunto de referencia, generar un histograma descriptor para cada una de ellas y escoger al menos uno de éstos por clase. En la etapa de prueba se realiza el mismo procedimiento, obteniendo un histograma descriptor para cada imagen, con el que se realiza la comparación con el conjunto de referencia para clasificar cada imagen de prueba.

2.3.2. Clasificación

La clasificación puede abordarse de muchas formas [46], utilizando funciones discriminantes, redes neuronales, enfoques bayesianos, máquinas de soporte vectorial (Support Vector Machine, SVM) o métodos no lineales. En este trabajo se optó por métodos más simples, como la clasificación utilizando proyecciones LDA y por vecino más cercano utilizando dos medidas de similitud.

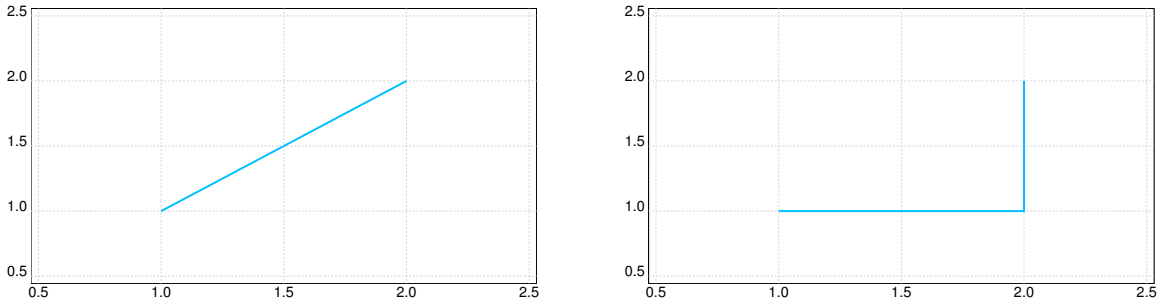
Proyección LDA

Como se explicó en la Subsección 2.3.1, LDA corresponde a un método de reducción de dimensionalidad, que proyecta los datos a un subespacio de características diseñado para optimizar la correcta clasificación. Por esto es posible utilizar LDA sobre los descriptores creados utilizando otros algoritmos. Esto se realiza considerando los descriptores como los datos entrenamiento y prueba en LDA, y generando la matriz de proyección a partir de ellos. La sola salida de LDA no entrega información acerca de la correspondencia a una u otra clase, por lo que en conjunto a la proyección, se debe utilizar otro método de clasificación para obtener la clase correspondiente.

Vecino más cercano

La clasificación por vecino más cercano consiste en evaluar la distancia entre un descriptor y los descriptores de la base de datos de referencia, generalmente generadas durante el entrenamiento del sistema. Luego, el elemento más cercano indica a qué clase corresponde el descriptor. No todas las medidas de similitud presentan los mismos resultados, pues son utilizadas para comparar distintos tipos de datos, y presentan distintos tipos de complejidad en su cálculo y posterior implementación en hardware.

En este trabajo se consideraron dos medidas de similitud, la distancia Euclidiana y la distancia Manhattan. Si bien existen otras medidas de similitud [47] como la distancia Chi-Square, que corresponde a la distancia Euclidiana ponderada; o la distancia Minkowski, que corresponde a una generalización de la distancia Euclidiana y Manhattan. Éstas son más difíciles de implementar en hardware por las operaciones de división que requieren.



(a) Distancia Euclidiana (b) Distancia Manhattan

Figura 2.22: Ejemplo de distancias Euclidiana y Manhattan entre dos puntos (1,1) y (2,2).

La comparación con la base de datos de referencia, utilizando el método de vecino más cercano, puede realizarse de dos formas. La primera consiste en comparar cada imagen nueva con todos los elementos utilizados en el entrenamiento del sistema. La segunda corresponde a considerar la media por clase del conjunto de entrenamiento, y calcular la distancia hacia ese promedio. La última opción presenta claras ventajas, reduciendo el tiempo de clasificación y la cantidad de memoria necesaria para almacenar la base de datos de referencia. Sin embargo, presenta la desventaja de poder reducir la tasa de acierto obtenida en casos donde los elementos de cada clase estén muy distantes entre sí.

Distancia Euclidiana

La distancia Euclidiana es la medida de similitud más usada para calcular la distancia entre dos puntos de n-dimensiones. Ésta se deduce a partir del teorema de Pitágoras por el cálculo de la hipotenusa de un triángulo, y corresponde a una distancia invariante a cambios de escala [47]. La distancia Euclidiana entre dos puntos P y Q de dimensión D, estará dada por:

$$D(P,Q) = \sqrt{\sum_{i=1}^{D} (p_i - q_i)^2}, \tag{2.35}$$

donde p_i y q_i corresponden a la i-ésima componente de los puntos P y Q respectivamentne-wenvironmente. Por ejemplo la distancia Euclidiana entre los puntos de la Figura 2.22 es de $\sqrt{2}$.

Distancia Manhattan

Es la distancia con menor dificultad de implementar, de las consideradas en este trabajo, ya que sólo se necesita una operación de resta y el cálculo del valor absoluto, a diferencia de la distancia Euclidiana. La distancia Manhattan entre los dos puntos P y Q de dimensión D está dada por:

$$d(P,Q) = \sum_{i=1}^{D} |p_i - q_i|, \tag{2.36}$$

donde p_i y q_i corresponden a la i-ésima componente de los puntos P y Q respectivamente. Por ejemplo la distancia Manhattan entre los puntos de la Figura 2.22 es de 2.

Distancia Chi-Square

La distancia Chi-Square es una distancia ponderada por el valor de las muestras, esto permite que sé de la misma relevancia a diferencias en muestras con pocas ocurrencias que a aquellas con muchas ocurrencias. Esta distancia es principalmente utilizada para comparar histogramas, puesto que en éstos son más relevantes las diferencias entre las muestras pequeñas [48]. La distancia Chi-Square entre dos punto P y Q de dimensión D, se define como:

$$d(P,Q) = \sum_{i=1}^{D} \frac{(p_i - q_i)^2}{p_i + q_i}, \tag{2.37}$$

donde p_i y q_i corresponden a la i-ésima componente de los puntos P y Q respectivamente.

Capítulo 3. Metodología y pruebas en software

3.1. Introducción

La selección del algoritmo a implementar para resolver el problema de detección y reconocimiento de rostros, se realizó en base a pruebas en software para evaluar las tasas de acierto y número de operaciones. Las pruebas se realizaron sobre los algoritmos descritos en el Capítulo 2 utilizando las bases de datos mencionadas en el Anexo A. Todas las pruebas se realizaron utilizando el software de programación de alto nivel Julia [49], en el caso particular del uso de SVM se utilizó la biblioteca de *machine learning* sklearn [50].

En este capítulo se describe la metodología utilizada para realizar las pruebas en software de los distintos algoritmos. Además, se presentan las tasas de acierto obtenidas y el análisis de complejidad correspondientes a cada algoritmo.

3.2. Metodología

La metodología utilizada corresponde a la descripción de los parámetros con los que se realizaron las pruebas en software para cada algoritmo, indicando que bases de datos se utilizaron y la forma en que se realizó el entrenamiento de los clasificadores.

3.2.1. Detección de rostros

Para la evaluación de los algoritmos de detección de rostros en imágenes infrarrojas de onda larga, se utilizó la base de datos LWIR de la Universidad de Concepción. Sobre esta base de datos se realizaron pruebas en software con los algoritmos de características Haar, características MB-LBP, histogramas LBP, histogramas HOG y la versión modificada del HOG (mHOG).

Para realizar las pruebas en software, en primer lugar, se adaptaron las imágenes de la base de datos, recortando los rostros a un tamaño de 120×120 píxeles. Posteriormente todos los rostros se redimensionaron a un tamaño de 24×24 píxeles, con lo que se generó el conjunto de rostros para el entrenamiento y prueba de los algoritmos. El conjunto de imágenes sin rostros

se obtuvo recortando secciones al azar de las imágenes donde no hubiesen rostros. Extrayendo secciones a varias escalas y redimensionandolas a 24×24 píxeles. Con lo anterior se obtuvieron 612 imágenes de rostros y 636 imágenes sin la presencia de éstos.

El tamaño de imágenes escogido permite que se puedan encontrar rostros de 24×42 píxeles o más dentro de una imagen. Con este tamaño se generan 162.336 características Haar, 8.464 características MB-LBP e histogramas de 203 muestras para LBP. Para los histogramas HOG y mHOG se decidió utilizar el tamaño de celdas original de 8×8 píxeles descrito en la Sección 2.2, que genera histogramas de 144 muestras. Además se utilizaron celdas de 4×4 píxeles, lo que genera histogramas de 900 muestras. Debido al tamaño de los vectores de características generados para los algoritmos Haar y MB-LBP, sólo se utilizó Adaboost para su clasificación. Además, para obtener resultados comparables con todos los algoritmos de extracción de características, las tasas de acierto presentadas utilizado Adaboost, corresponden a las obtenidas con las primeras 100 características seleccionadas para cada descriptor.

El entrenamiento de los algoritmos se realizó dividiendo la base de datos en una parte de entrenamiento y otra de prueba, con una proporción de 60 % y 40 % respectivamente. Para evitar resultados dependientes del conjunto de entrenamiento, se realizaron 5 iteraciones de división de los conjuntos de forma aleatoria (validación cruzada). Dividiendo las imágenes de rostros y no rostros de forma igual, es decir, considerando siempre el 60 % y 40 % para las imágenes con y sin rostros.

3.2.2. Reconocimiento de rostros

Para la evaluación del algoritmo de reconocimiento de rostros en espectro infrarrojo de onda larga, se utilizaron las tres bases de datos descritas en el Anexo A. Éstas son: la base de datos de rostros LWIR de la Universidad de Concepción, la base de datos Equinox de Equinox Corporation y la base de datos UCHThermalFace de la Universidad de Chile. En el caso de ésta última los conjuntos de interior (Indoor) y exterior (Outdoor) fueron entrenados en conjunto. Para los tres casos se utilizaron los algoritmos Eigenfaces (PCA), Fisherfaces (LDA), Ahonen (LBP) y la combinación de LBP con LDA (LBP+LDA).

La aplicación del operador LBP se realizó utilizando $\text{LBP}_{(8,1)}$ uniforme y el histograma descriptor se creó utilizando el algoritmo Ahonen. Para lo cual cada imagen con LBP aplicado fue dividida en 64 (8×8), 36 (6×6) y 16 (4×4) regiones no solapadas, cuyos histogramas individuales fueron concatenados, generando un histograma de 3.776, 2.124 y 944 elementos para

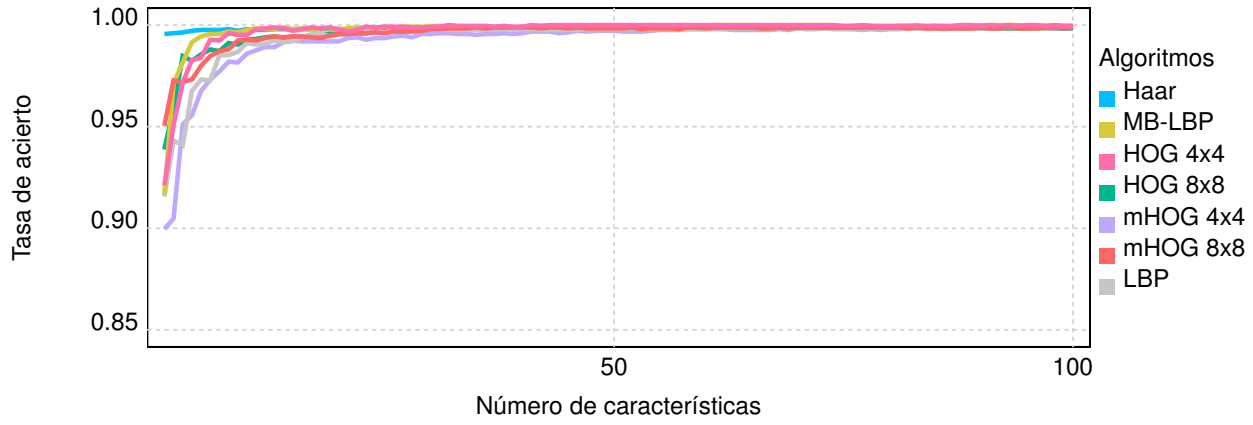


Figura 3.1: Tasa de acierto vs número de caracteríticas utilizadas en Adaboost para los distintos algoritmos considerando en este trabajo.

cada número de regiones respectivamente. Para los algoritmos que utilizan métodos de reducción de dimensionalidad como PCA, LDA y LBP + LDA, se muestran los resultados utilizando el número máximo de vectores de proyección LDA por base de datos. Esto corresponde a 101 para la base de datos LWIR UdeC, 19 para Equinox y 52 para UCHThermalFace. En los algoritmos PCA y LDA las imágenes fueron redimensionadas a un 30 % de su tamaño original, debido a la alta dimensionalidad que presentaban los datos originales y al alto costo computacional que requiere el entrenamiento de estos algoritmos.

El entrenamiento de todos los algoritmos se realizó con el 60 % de la base de datos y la clasificación se realizó con el método del vecino más cercano, utilizando distancia Manhattan y distancia Euclidiana. La selección del 60 % de la base de datos para la etapa de entrenamiento del algoritmo se realizó de forma aleatoria, pero considerando las mismas imágenes seleccionadas para todos los sujetos. Por ejemplo si aleatoriamente se seleccionaron las imágenes 1, 2 y 6 para un sujeto, éstas son las que se seleccionan en el entrenamiento para cada los sujetos.

3.3. Pruebas en software

Para los algoritmos presentados en el Capítulo 2 se realizaron implementaciones en el lenguaje de programación de alto nivel Julia, con el fin de obtener resultados para comparar la efectividad de cada uno de los algoritmos. Para lo anterior se siguió la metodología descrita en la sección anterior.

Tabla 3.1: Tasas de acierto para 5 iteraciones de algoritmos de detección de rostros.

Algoritmo	**Clasificador**	**Tasa de acierto por iteración (%)**					
		1	2	3	4	5	Media
Haar	Adaboost	100	99.4	100	100	100	99.88
MB-LBP	Adaboost	100	100	100	99.8	100	99.96
LBP	Adaboost	99.8	99.8	100	100	99.8	99.88
	SVM	99.6	99.6	99.8	99.2	99.4	99.52
HOG 4×4	Adaboost	100	100	99.8	100	100	99.96
	SVM	100	100	100	99.8	100	99.96
HOG 8×8	Adaboost	99.8	99.4	100	100	100	99.83
	SVM	98.8	99.2	99.2	99.6	99.0	99.16
mHOG 4×4	Adaboost	100	99.8	99.8	100	99.8	99.88
	SVM	100	100	100	99.8	100	99.96
mHOG 8×8	Adaboost	99.8	99.6	100	100	100	99.88
	SVM	99.4	99.6	99.8	99.6	100	99.68

3.3.1. Detección de rostros

En la Tabla 3.1 se muestran las tasas de acierto obtenidas con los distintos algoritmos de la Sección 2.2 y el Capítulo 2 para la base de datos LWIR UdeC. Las tasas de acierto dependen en gran medida de la forma en que se crearon los conjuntos de entrenamiento y prueba. En particular para la base de datos utilizada, la diferencia entre los rostros y el resto de la imagen es clara. Además no se presentan sectores con características similares a rostros en otros lugares de la imagen. Lo anterior favorece la obtención de tasas de acierto altas, ya que la diferencia entre las clases es notoria.

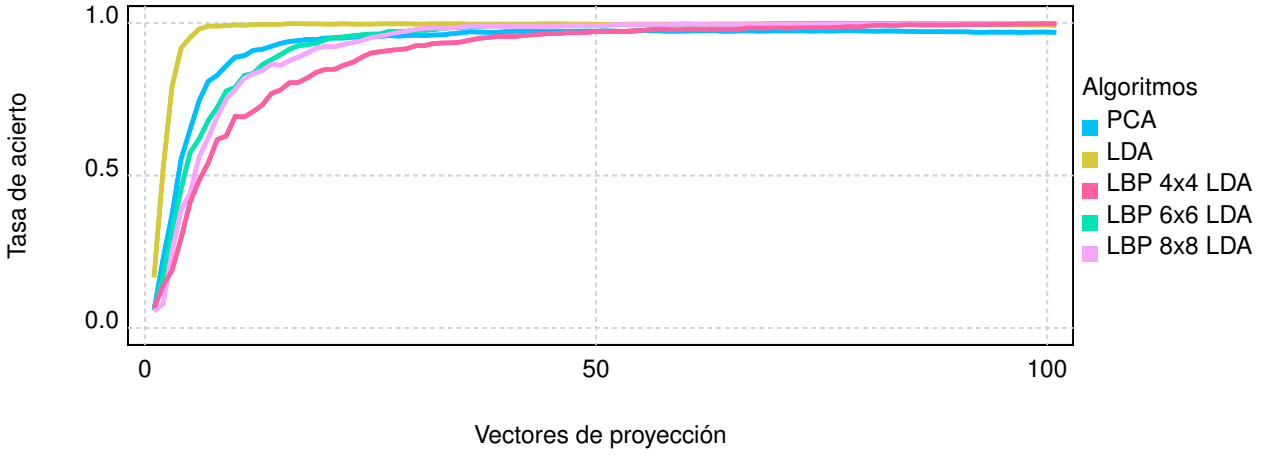
Los algoritmos con mejor desempeño fueron HOG con celdas de 4×4 píxeles utilizando Adaboost y SVM, mHOG con celdas de 4×4 píxeles utilizando SVM y MB-LBP utilizando Adaboost. En los cuatro casos se obtuvo una tasa de acierto de 99.96 % promedio. En la Figura 3.1, como comparativa del uso de Adaboost en los algoritmos de extracción de características, se muestra la evolución en las tasas de acierto respecto al número de características seleccionadas por el algoritmo.

3.3.2. Reconocimiento de rostros

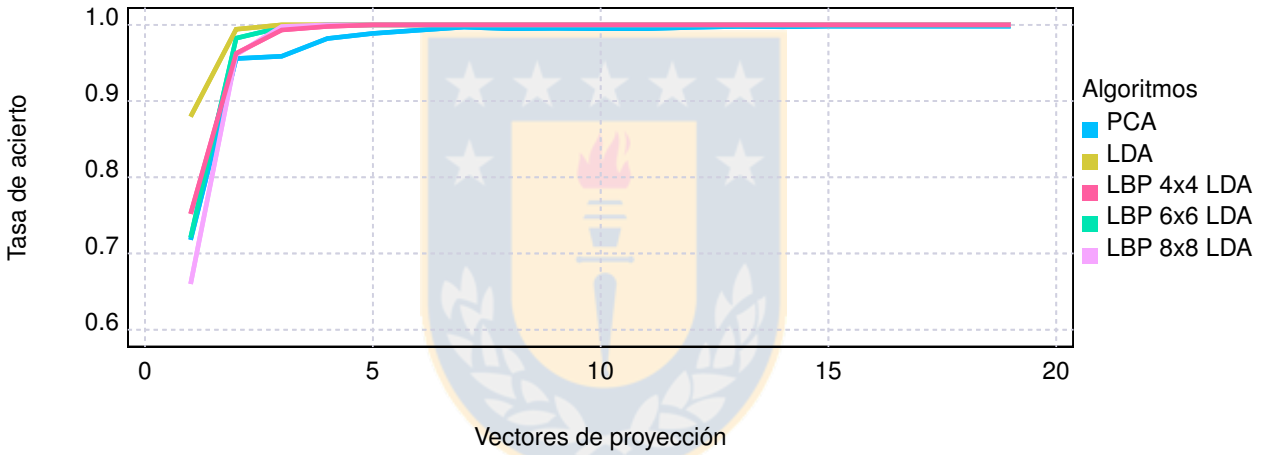
Los resultados de los algoritmos para la base de datos LWIR UdeC se muestran en la Tabla 3.2. En ésta se puede ver que la tasa de acierto para todos los algoritmos es muy alta, sobre 96 % en promedio, con un máximo de 99.92 % para el algoritmo LBP+LDA con 8×8 y 6×6 regiones utilizando distancia Euclidiana. Los buenos resultados obtenidos pueden explicarse por la baja cantidad de imágenes por sujeto presentes en la base de datos, las que no presentan una gran variación entre ellas (Figura A.2).

Los resultados obtenidos en la base de datos Equinox se muestran la Tabla 3.3. Las tasas de acierto obtenidas están dentro de lo esperado para esta base de datos, ya que de las 60 imágenes disponibles por sujeto, la mayoría corresponden a una secuencia de imágenes capturadas a 10 cuadros por segundo durante cuatro segundos. Esto último produce que la variación entre las distintas imágenes por sujeto sea pequeña (Figura A.4), obteniendo muy buenos resultados en la clasificación. Los mejores resultados para esta base de datos se obtuvieron con los algoritmos LDA y LBP+LDA, con los que se obtuvo un 100 % de acierto. Además en la Figura 3.2 se puede ver que la tasa de acierto máxima se alcanza sólo con 5 vectores de proyección.

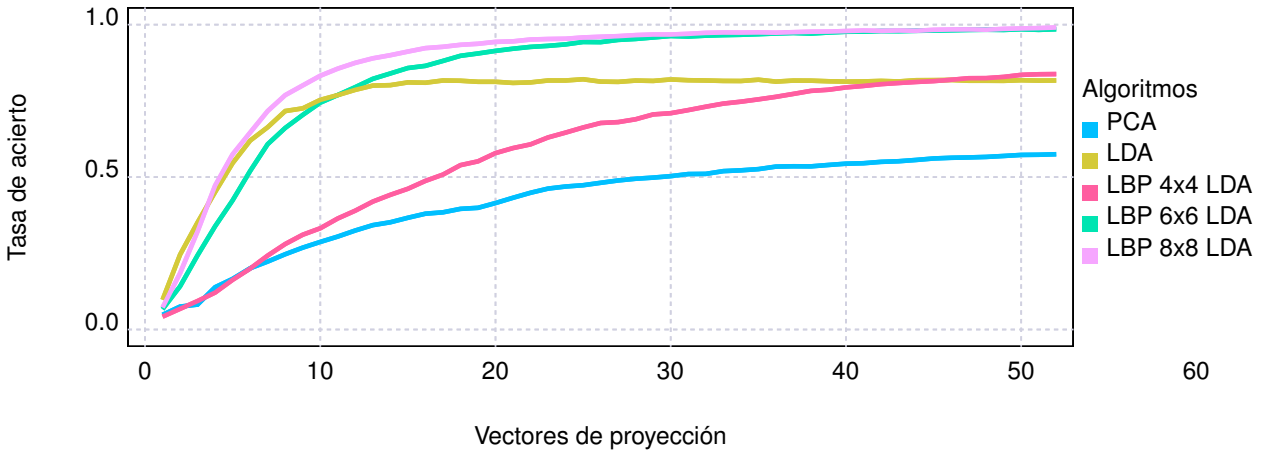
En la base de datos UCHThermalFace es posible ver de mejor forma como actúan los distintos algoritmos, ya que ésta presenta imágenes con mayor variación para un mismo sujeto. Entre éstas se incluyen rotaciones horizontales, distintos gestos faciales y diferentes ambientes de captura (Figura A.6). Esto se puede ver en las tasas de acierto obtenidas, donde los algoritmos PCA y LDA obtienen un bajo desempeño, con un acierto inferior al 60 % para PCA e inferior al 85 % para LDA. En el caso de LBP se obtienen tasas de acierto inferiores al 80 % utilizando distancia Euclidiana. Sin embargo, al utilizar distancia Manhattan, éstos se incrementan llegando hasta un 94.25 % y un 94.85 % con distancia Chi-Square, en ambos casos para LBP con 8×8 regiones. Los mejores resultados se obtuvieron al usar el algoritmo LBP+LDA con 8×8 regiones con distancia Euclidiana para la clasificación por el método de vecino más cercano. También con el algoritmo LBP+LDA con 6×6 regiones se obtuvieron altas tasas de acierto, las que en el caso de la distancia Euclidiana son sólo un 0.2 % inferior, en promedio, que al utilizar 8×8 regiones. Con esto se presenta una buena alternativa, la que si bien no entrega la mayor tasa de acierto, requiere de sólo un 56 % de la memoria necesaria al utilizar LBP+LDA con 8×8 regiones.



(a) LWIR UdeC



(b) Equinox



(c) UCHThermalFace

Figura 3.2: Número de vectores de proyección vs tasas de acierto para las tres bases de datos utilizadas, usando distancia Manhattan.

Tabla 3.2: Tasas de acierto para 5 iteraciones de algoritmos de reconocimiento de rostros sobre la base de datos LWIR UdeC.

Algoritmo	**Distancia**	**Tasa de acierto por iteración (%)**					
		1	2	3	4	5	Media
PCA	Euclidiana	99.02	97.06	87.58	99.35	98.37	96.27
	Manhattan	98.69	99.02	88.24	99.35	99.35	96.93
LDA	Euclidiana	100	100	96.41	100	100	99.28
	Manhattan	100	100	96.41	100	99.67	99.22
LBP 4×4	Euclidiana	99.67	100	95.75	99.67	100	99.02
	Manhattan	100	100	97.06	100	100	99.41
	Chi-Square	100	100	100	100	99.02	99.80
LBP 6×6	Euclidiana	99.67	99.67	95.42	99.67	99.67	98.82
	Manhattan	100	100	97.39	100	100	99.48
	Chi-Square	100	100	100	100	99.34	99.87
LBP 8×8	Euclidiana	100	100	95.42	99.67	100	99.02
	Manhattan	100	100	97.71	100	100	99.53
	Chi-Square	100	100	100	100	99.02	99.80
LBP 4×4 + LDA	Euclidiana	100	100	99.02	100	100	99.8
	Manhattan	100	100	98.69	100	100	99.74
LBP 6×6 + LDA	Euclidiana	100	100	99.67	100	100	99.92
	Manhattan	100	100	99.35	100	100	99.87
LBP 8×8 + LDA	Euclidiana	100	100	99.67	100	100	99.92
	Manhattan	100	100	99.35	100	100	99.87

3.4. Análisis de complejidad

Para la decisión de implementar uno u otro algoritmo, no basta con la información que entregan las tasas de acierto obtenidas en la subsección anterior. Además, es necesario analizar la complejidad que presenta cada algoritmo durante su etapa de prueba, que es la que será implementada en hardware.

En las Tablas 3.5 y 3.7, se muestran el número de operaciones necesarias para realizar los cálculos de histogramas o características, proyección a subespacios y clasificación por los

Tabla 3.3: Tasas de acierto para 5 iteraciones de algoritmos de reconocimiento de rostros sobre la base de datos Equinox.

Algoritmo	**Distancia**	**Tasa de acierto por iteración (%)**					
		1	2	3	4	5	Media
PCA	Euclidiana	99.79	99.58	100	99.58	100	99.79
	Manhattan	100	99.58	100	99.58	100	99.83
LDA	Euclidiana	100	100	100	100	100	100
	Manhattan	100	100	100	100	100	100
LBP 4×4	Euclidiana	98.96	99.17	98.96	98.96	99.17	99.03
	Manhattan	99.58	99.79	99.79	99.79	99.79	99.75
	Chi-Square	99.38	100	99.79	99.79	99.79	99.75
LBP 6×6	Euclidiana	99.79	99.79	99.79	99.58	99.79	99.75
	Manhattan	100	100	100	100	100	100
	Chi-Square	99.58	100	100	100	99.79	99.88
LBP 8×8	Euclidiana	99.17	99.38	99.38	99.17	99.17	99.25
	Manhattan	99.38	99.58	99.79	99.38	99.79	99.58
	Chi-Square	99.38	100	99.79	99.79	99.58	99.71
LBP 4×4 + LDA	Euclidiana	100	100	100	100	100	100
	Manhattan	100	100	100	100	100	100
LBP 6×6 + LDA	Euclidiana	100	100	100	100	100	100
	Manhattan	100	100	100	100	100	100
LBP 8×8 + LDA	Euclidiana	100	100	100	100	100	100
	Manhattan	100	100	100	100	100	100

diversos métodos estudiados. En todos los casos el número de operaciones se muestra en función del número de píxeles N_p, número de clases N_c, número de características N_f y número de regiones N_{reg} según corresponde en cada caso.

3.4.1. Detección de rostros

Entre los algoritmos de detección se tiene que las características Haar y características MB-LBP requieren generar sumas de áreas de píxeles. Si bien la cantidad de píxeles a sumar depende de las características seleccionadas durante el entrenamiento, se asume que se construyen imáge-

Tabla 3.4: Tasas de acierto para 5 iteraciones de algoritmos de reconocimiento de rostros sobre la base de datos UCHThermalFace.

Algoritmo	**Distancia**	**Tasa de acierto por iteración (%)**					
		1	2	3	4	5	Media
PCA	Euclidiana	35.53	41.19	40.72	39.31	34.75	38.3
	Manhattan	53.62	61.79	63.36	59.75	48.9	57.48
LDA	Euclidiana	81.6	87.74	87.74	84.91	76.73	83.74
	Manhattan	78.77	84.43	86.95	83.17	75.16	81.69
LBP 4×4	Euclidiana	69.5	59.75	64.62	66.03	55.82	63.13
	Manhattan	87.26	83.65	87.42	87.11	77.66	84.61
	Chi-Square	88.05	81.92	86.64	84.91	77.52	83.81
LBP 6×6	Euclidiana	77.2	71.7	75.79	74.83	65.88	73.08
	Manhattan	91.97	90.25	93.08	93.71	87.11	91.23
	Chi-Square	92.30	89.47	94.03	94.65	85.69	91.23
LBP 8×8	Euclidiana	78.14	73.9	80.97	81.92	72.01	77.39
	Manhattan	94.65	93.55	95.91	96.7	90.41	94.25
	Chi-Square	96.38	93.71	96.07	96.70	91.35	94.84
LBP 4×4 + LDA	Euclidiana	91.51	85.38	87.26	86.32	89.14	87.92
	Manhattan	86.79	81.6	82.69	82.55	85.22	83.77
LBP 6×6 + LDA	Euclidiana	99.37	98.74	98.42	99.21	98.74	98.9
	Manhattan	98.9	98.42	98.27	98.42	98.42	98.49
LBP 8×8 + LDA	Euclidiana	99.53	99.06	98.74	99.06	99.21	99.11
	Manhattan	99.21	98.9	99.06	99.06	99.06	99.06

nes integrales, por lo que se requieren de N_p sumas para generarlas. Posteriormente se necesitan 4 sumas para obtener cada área de las características. En caso de Haar se requieren entre 2 y 3 áreas, mientras que MB-LBP siempre se requieren de 9. Considerando el peor caso para generar el vector descriptor en Haar se requieren de $N_p + 3 \cdot 4 \cdot N_f$ operaciones de suma, mientras que en MB-LBP se necesitan $N_p + 9 \cdot 4 \cdot N_f$. Además en ambos casos se necesita de N_f operaciones de suma para calcular la salida del clasificador Adaboost. En el caso de LBP, HOG y mHOG la generación de los histogramas se hace de forma similar, debido a que las regiones centrales se contabilizan cuatro veces. Asumiendo que todos los píxeles son contabilizados cuatro veces y considerando que ambos operadores tienen una salida de un píxel menos de resolución por eje, se tiene que son necesarias $4 \cdot (N_p - 1)$ sumas para construir los histogramas. En el caso de LBP,

Tabla 3.5: Número de operaciones para algoritmos de detección de rostros para una ventana de análisis.

Algoritmo	**Clasificador**	**Multiplicación**	**Suma/Resta**
Haar	Adaboost	0	$N_p + 13 \cdot N_f$
MBLBP	Adaboost	0	$N_p + 37 \cdot N_f$
LBP	Adaboost	0	$5 \cdot (N_p - 1) + N_f$
	SVM	$16 \cdot N_{reg} + 59$	$5 \cdot (N_p - 1) + 16 \cdot N_{reg} + 59$
HOG	Adaboost	$(N_p - 1)$	$5 \cdot (N_p - 1) + N_f$
	SVM	$(N_p - 1) + 36 \cdot N_{reg}$	$5 \cdot (N_p - 1) + 36 \cdot N_{reg}$
mHOG	Adaboost	$(N_p - 1)$	$5 \cdot (N_p - 1) + N_f$
	SVM	$(N_p - 1) + 36 \cdot N_{reg}$	$5 \cdot (N_p - 1) + 36 \cdot N_{reg}$

además se construye un histograma general, utilizando $5 \cdot (N_p - 1)$ sumas en total. Para HOG y mHOG, previo al cálculo del histograma, se deben calcular los ángulos y módulos de cada píxel. Para esto es necesario calcular las imágenes gradientes, que requieren de $N_p - 1$ sumas cada una. Luego, el cálculo del módulo necesita de $N_p - 1$ operaciones de resta-multiplicación y el ángulo, en caso de ser implementado con una tabla de traducción, no necesita operaciones aritméticas. Finalmente, la clasificación realizada con Adaboost requiere de N_f operaciones de suma para ambos algoritmos. En el caso de la clasificación con SVM, se necesitan multiplicar los vectores de peso por los descriptores, y sumar todos los resultados. Así, para el caso de LBP el descriptor tiene un tamaño de $59 + 16 \cdot N_{reg}$, donde 59 corresponde al número de patrones $\text{LBP}_{(8,1)}$ uniformes y 16 al número de patrones $\text{LBP}_{(4,1)}$, por lo que se requieren $59 + 16 \cdot N_{reg}$ operaciones de multiplicación-suma. En HOG y mHOG el tamaño del histograma descriptor corresponde al número de bloques dentro de la imagen, los cuales están compuestos de cuatro celdas, para cada una de las cuales se genera un histograma de 9 muestras. Con esto el tamaño del descriptor es $9 \cdot 4 \cdot N_{reg}$, necesitando $9 \cdot 4 \cdot N_{reg}$ operaciones de multiplicación-suma. En la Tabla 3.5 se muestra el total de operaciones por algoritmo.

Todos los números de operaciones descritos corresponden a los necesarios para una ventana de análisis, por lo que para obtener el total por imagen, estos resultados se deben multiplicar por el número de ventanas presentes en una imagen. A modo de ejemplo se realiza una comparativa numérica en la Tabla 3.6, considerando una búsqueda de rostros en una imagen infrarroja de 640×512 píxeles, con ventanas de análisis de 24×24 píxeles, sin considerar otros tamaños. Con esto, las variables con las que se indicó previamente el número de píxeles serán: $N_p = 576$, $N_f = 100$ y $N_{reg} = 5$. Además, para una imagen de 640×512, existen 300608 regiones posibles

Tabla 3.6: Ejemplo de número de operaciones para algoritmos de detección de rostros.

Algoritmo	**Clasificador**	**Operaciones por ventana**		**Operaciones por imagen**	
		Multiplicación	Suma/Resta	Multiplicación	Suma/Resta
Haar	AdaBoost	0	1876	0	563.94M
MBLBP	AdaBoost	0	4276	0	1285.40M
LBP	AdaBoost	0	2975	0	894.31M
	SVM	203	3078	61.02M	925.27M
HOG	AdaBoost	575	2975	172.85M	894.31M
	SVM	899	3199	270.25M	961.64M
mHOG	AdaBoost	575	2975	10.80M	55.89M
	SVM	899	3199	16.89M	60.10M

usando búsqueda basada en ventana deslizante, lo que disminuye a 18788 en mHOG con celdas iniciales de 4 píxeles.

3.4.2. Reconocimiento de rostros

En los algoritmos de reconocimiento se tiene que, para Eigenfaces (PCA) y Fisherfaces (LDA) la etapa de proyección de los píxeles de entrada requiere de N_p operaciones de multiplicación y suma en acumuladores. Posteriormente, los datos proyectados son clasificados con el método del vecino más cercano. Esto, considerando que se utilizan $N_c - 1$ vectores de proyección, requiere del tamaño de los vectores por el número de sujetos en la base de datos, lo que equivale a $(N_c - 1) \cdot N_c$ operaciones de resta-multiplicación para distancia Euclidiana, y sólo de resta para distancia Manhattan. En el caso de LBP se requieren de N_p sumas para generar los histogramas. El tamaño de estos histogramas corresponde al número de patrones usados (59) por el número de regiones N_{reg} en que se divide la imagen. Así, al clasificar utilizando sólo el histograma, se requieren de $59 \cdot N_{reg} \cdot N_c$ operaciones resta-multiplicación para distancia Euclidiana y sólo de resta para distancia Manhattan. Al utilizar distancia Chi-Square, considerando que la división se implementa como una operación de multiplicación por el inverso del denominador, se requieren del mismo número de operaciones que para la distancia Euclidiana, más una operación de suma-multiplicación por cada muestra del histograma. En caso de aplicar LDA a los histogramas LBP (LBP+LDA), a las operaciones necesarias para calcular el histograma, se añaden las utilizadas durante la proyección. Esto corresponde al tamaño de los histogramas $(59 \cdot N_{reg})$ por el número

Tabla 3.7: Número de operaciones para algoritmos de reconocimiento de rostros.

Algoritmo	**Distancia**	**Multiplicación**	**Suma/Resta**
PCA	Euclidiana	$(N_c - 1)(N_p + N_c)$	$(N_c - 1)(N_p + N_c)$
	Manhattan	$(N_c - 1) \cdot N_p$	$(N_c - 1)(N_p + N_c)$
LDA	Euclidiana	$(N_c - 1)(N_p + N_c)$	$(N_c - 1)(N_p + N_c)$
	Manhattan	$(N_c - 1) \cdot N_p$	$(N_c - 1)(N_p + N_c)$
LBP	Euclidiana	$59 \cdot N_{reg} \cdot N_c$	$N_p + 59 \cdot N_{reg} \cdot N_c$
	Manhattan	0	$N_p + 59 \cdot N_{reg} \cdot N_c$
	Chi-Square	$2 \cdot 59 \cdot N_{reg} \cdot N_c$	$N_p + 2 \cdot 59 \cdot N_{reg} \cdot N_c$
LBP + LDA	Euclidiana	$(59 \cdot N_{reg} + N_c) \cdot (N_c - 1)$	$N_p + (59 \cdot N_{reg} + N_c) \cdot (N_c - 1)$
	Manhattan	$59 \cdot N_{reg} \cdot (N_c - 1)$	$N_p + (59 \cdot N_{reg} + N_c) \cdot (N_c - 1)$

de vectores de proyección $(N_c - 1)$. Posteriormente, la clasificación requiere del mismo número de operaciones que los algoritmos PCA y LDA, lo que es $(N_c - 1) \cdot N_c$. El total de operaciones, que corresponde a la suma de las requeridas para cada etapa de cada algoritmo, se expone en la Tabla 3.7.

Para facilitar la comparación del número de operaciones, en la Tabla 3.8 se muestra, a modo de ejemplo, la cantidad de operaciones necesarias para un rostro de 81×150 píxeles, divido en 64 regiones para el análisis y considerando una base de datos con 53 sujetos. Así, los valores de las variables con que se realizó el análisis serán: $N_p = 12150$, $N_c = 53$ y $N_{reg} = 64$.

3.5. Discusión

Tras el análisis de la complejidad de los algoritmos, basado en el número de operaciones, se puede ver que las modificaciones propuestas al algoritmo HOG, reducen efectivamente el número de operaciones necesarias para la búsqueda de rostros. Como se ve en la Tabla 3.6, el algoritmo modificado, requiere del 6.2 % de las operaciones del algoritmo sin modificar, para ambos clasificadores. Lo anterior corresponde a la reducción considerando la búsqueda en sólo un tamaño de región, en caso de incluir el análisis en diferentes tamaños de región, el porcentaje disminuiría aún más.

El análisis de complejidad para los distintos algoritmos de reconocimiento de rostros, presenta

Tabla 3.8: Ejemplo de número de operaciones para algoritmos de reconocimiento de rostros.

Algoritmo	**Distancia**	**Multiplicación**	**Suma/Resta**
PCA	Euclidiana	634.56K	634.56K
	Manhattan	631.80K	634.56K
LDA	Euclidiana	634.56K	634.56K
	Manhattan	631.80K	634.56K
LBP	Euclidiana	200.13K	212.28K
	Manhattan	0	212.28K
	Chi-Square	412.40K	400.25K
LBP+LDA	Euclidiana	199.11K	211.26K
	Manhattan	196.35K	211.26K

menos diferencias que en el caso de la detección. En este caso se ve que el algoritmo que menos operaciones requiere es LBP con distancia Manhattan, ya que no se utilizan operaciones de multiplicación. El algoritmo LBP+LDA presenta el menor uso de operaciones de sumas y es el segundo con menores operaciones de multiplicación. Es importante ver que la diferencia en el número de operaciones de multiplicación, entre los dos tipos de distancia para el algoritmo LBP+LDA es menor, por lo que las diferencias en una implementación en hardware serían mínimas.

En definitiva, en base a los resultados de las pruebas en software y al número de operaciones necesarias para cada algoritmo, es posible afirmar que el algoritmo HOG, con las modificaciones propuestas, es la mejor opción para implementar, pues presenta un buen resultado en las pruebas en software, con una reducción considerable del número de operaciones necesarias. En el caso del reconocimiento, se puede ver que los mejores resultados se obtienen con el algoritmo LBP+LDA con distancia Euclidiana, sin presentar un número elevado de operaciones sobre el resto de los algoritmos.

Capítulo 4. Arquitectura de sistema

4.1. Introducción

La arquitectura del sistema se divide en tres partes fundamentales, el detector facial, el clasificador facial y los periféricos necesarios para la comunicación con elementos externos y comunicación entre ambos módulos de procesamiento.

En este capítulo se detalla el desarrollo de todos los módulos de procesamiento realizados en este trabajo. Incluyendo las arquitecturas internas y las interfaces utilizadas para la comunicación interna y externa. En la Figura 4.1 se muestra una vista completa de la distribución física de los bloques de procesamiento sobre el SoC.

4.2. Arquitectura general

La arquitectura general del sistema se desarrolló de forma mixta, realizando parte del procesamiento en software y parte en hardware, como se ve en la Figura 4.1. En particular toda la etapa de detección de rostros se realizó en el procesador presente en chip y la etapa de reconocimiento de rostros se realizó en lógica programable.

Muchos de los datos necesarios para el funcionamiento del sistema de reconocimiento de rostros son recibidos vía Ethernet desde un computador y almacenados, por el procesador, en

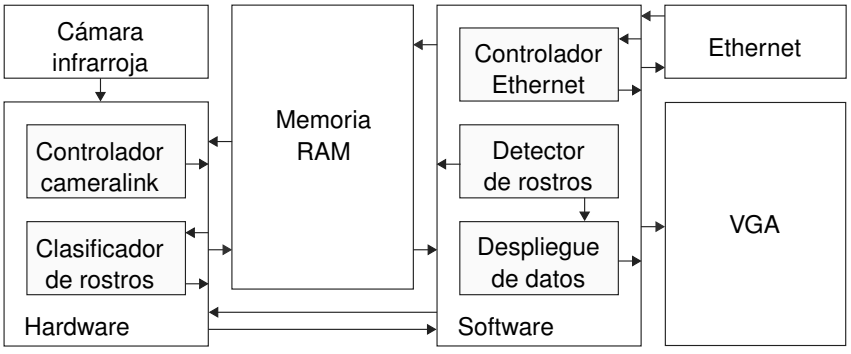


Figura 4.1: Distribución del sistema en SoC.

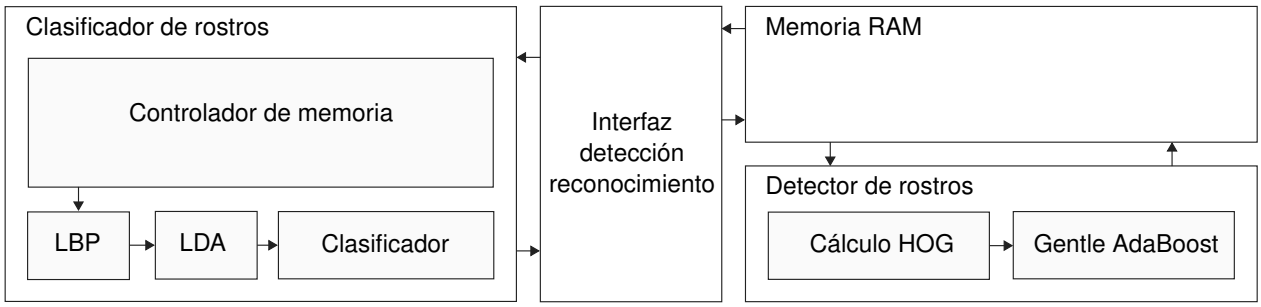


Figura 4.2: Arquitectura general del sistema.

una posición específica de la memoria RAM. Por esto se reservaron 256MBytes de RAM, desde la dirección 0x10000000 hasta 0x1FFFFFFF, para uso de datos compartidos entre el procesador y la lógica programable. En este espacio de memoria se almacenó la imagen obtenida desde la cámara infrarroja, los coeficientes de proyección LDA, la salida obtenida por el detector de rostros y la salida del clasificador de rostros. De esta manera todas las transferencias de datos entre los módulos de procesamiento se realizan con transacciones en la memoria RAM, además, todas las señales necesarias para indicar para indicar el comienzo o fin se ciertos procesos, fueron implementadas como interrupciones en el caso de señales de lógica a procesador, y como escritura de registros en el caso de señales del procesador a lógica programable.

4.3. Arquitectura detector de rostros

La etapa de detección implementa el algoritmo de extracción de características HOG con un clasificador Gentle Adaboost, ambos vistos en el Capítulo 2. El diseño general de la arquitectura del sistema de detección se divide en la implementación de ambos algoritmos en software, sobre un sistema embebido.

El sistema de detección facial requiere de una imagen presente en memoria RAM con la que trabajar. En el funcionamiento normal del sistema esta imagen es obtenida desde una cámara infrarroja conectada por interfaz CameraLink, en el modo de desarrollo la imagen puede ser recibida desde un computador por Ethernet. Cada vez que se recibe una nueva imagen comienza el procesamiento del detector de rostros, cuya primera etapa consiste en generar un matriz de histogramas de gradientes (HOG). Posteriormente es necesario generar descriptores para ventanas de distinto tamaño y posición dentro de la imagen y aplicar un criterio de clasificación

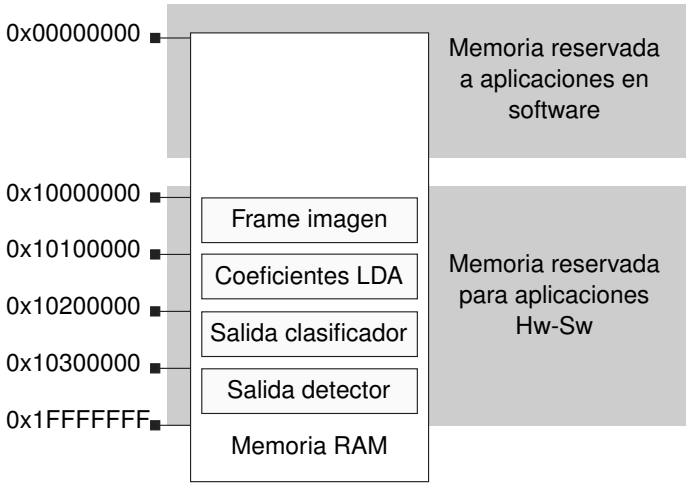


Figura 4.3: Distribución de datos compartidos en memoria RAM.

a cada uno ellos, para indicar si corresponde o no a una cara, a este último proceso se le incorpora un filtro por coincidencias de área para eliminar detecciones múltiples de un mismo rostro.

Como se indicó anteriormente, el sistema de detección facial puede ser alimentado con imágenes obtenida desde una cámara infrarroja o enviada por Ethernet desde un computador, en ambos casos éstas son almacenadas en una posición específica de la memoria RAM para ser procesadas por el detector facial. Una vez realizada la detección en todas las escalas y posiciones posibles es necesario reducir el número de rostros detectados, ya que para cada rostro muchas regiones son clasificadas de forma positiva. Finalmente el detector genera una lista sólo con las regiones que además de ser clasificadas como positivas, no están contenidas dentro de otra región de mayor valor de clasificación. Esta lista es almacenada en una posición fija de memoria RAM, como se muestra en la Figura 4.3, desde donde es leída en hardware para comenzar la etapa de reconocimiento. En el algoritmo 9 se muestra el funcionamiento general del módulo de detección.

4.3.1. Cálculo HOG

El cálculo de histogramas de gradientes orientados (HOG) requiere generar imágenes gradientes de forma horizontal y vertical, para obtener las matrices de magnitud y ángulo de cada píxel. Con ambas matrices se generan histogramas para distintas divisiones dentro de la imagen, las que posteriormente son concatenadas y normalizadas formando bloques, con los que

Algoritmo 9 Funcionamiento de módulo de detección de rostros

```
1: Cálculo de gradientes de imagen.
2: Generación de histogramas de celdas.
3: for all Tamaños de celda do
4:     Genera celdas para tamaño actual en base a las generadas en la interación anterior.
5:     Genera bloques de 4 × 4 celdas para tamaño actual.
6:     for all Posiciones dentro de la imagen do
7:         Clasifica cada grupo de bloques como cara o no cara.
8:         if Un rostro existente de menor valor de clasificación está incluido dentro del rostro
           actual then
9:             Borra rostro anterior
10:        else
11:            Añade nuevo rostro a la lista
12:        end if
13:    end for
14: end for
15: Genera salida de detector
```

finalmente se generan los descriptores de cada región. Este proceso se debe repetir para cada ventana análisis, implicando un gran costo computacional al utilizar búsqueda por ventanas móviles, por esto, como se explicó en el Capítulo 2, la implementación se realizó utilizando una búsqueda basada en desplazamiento de celdas. Además el cálculo de gradientes se realiza de forma inmediata al leer el vecindario de un píxel, sin necesidad de crear imágenes o matrices intermedias, reduciendo así los requerimientos de memoria.

El módulo de procesamiento de HOG requiere como entrada una imagen presente en memoria RAM, ésta es accedida de forma secuencial generando gradientes para cada píxel de la imagen, excluyendo las filas y columnas de los extremos. Luego la imagen es dividida en celdas de 4×4 píxeles, generando un histograma de los gradientes correspondientes a cada una de ellas. Cada histograma generado posee nueve muestras posibles de ángulos en valor absoluto, esto quiere decir que cada muestra contiene un rango de $20°$ cuya ocurrencia se genera según la magnitud del gradiente respectivo. El proceso de creación de la matriz de histogramas por celda se realiza sólo una vez por imagen, ya que, si bien es necesario realizarlo para cada tamaño de análisis, se puede realizar de forma iterativa, manteniendo siempre la matriz original almacenada y sumando al histograma de cada celda los histogramas de las celdas aledañas. Esto genera celdas que representan áreas de mayor tamaño de la imagen, pero que mantienen la separación espacial en sólo 4 píxeles, que corresponde al tamaño de representación original de las celdas. Al realizar esto se evita la construcción de imágenes piramidales que requieren

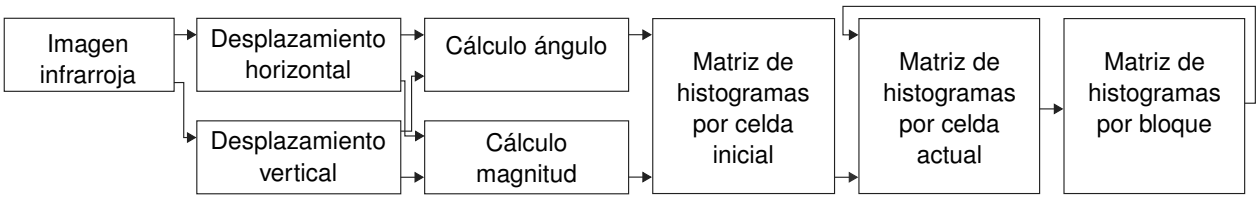


Figura 4.4: Diagrama de flujo de implementación de algoritmo HOG en software.

de algoritmos de redimensionado, reemplazándolos por sumas de histogramas y manteniendo una baja granularidad en la búsqueda en diferentes posiciones. Así, el cambio de tamaño en la ventana de análisis queda restringido a aumentar en el tamaño de celda original cada celda. Posteriormente, y para cada tamaño de análisis, es necesario construir bloques de 2×2 celdas, esto corresponde a un problema de concatenación y normalización de histogramas, en el que se requiere crear una nueva matriz de bloques, cuyas dimensiones son el número de celdas menos uno. Para crear la matriz de bloques es necesario indexar de forma correcta las celdas utilizadas en cada bloque, ya que, en tamaños distintos al original, las celdas contiguas no son parte del mismo bloque. Por esto, para construir los bloques se necesita conocer el factor de escala, respecto al tamaño inicial, en que se realiza el análisis, ya que en esa cantidad de celdas está separada la siguiente celda perteneciente al bloque. Una vez realizada la copia de datos desde la matriz de celdas a la de bloques, cada bloque es normalizado, expresando el resultado en punto fijo, con 15 bits para la parte decimal y uno para la parte entera. Con esto se generan todos los bloques posibles para un tamaño de ventana determinado en la imagen, posteriormente estos bloques deben ser concatenados para poder clasificarlos.

Todo el proceso anterior se muestra en la Figura 4.4 donde se indica la entrada de datos, el cálculo de los gradientes de cada píxel, en base a los desplazamientos horizontales y verticales respecto a su vecindario, y el proceso de creación de bloques, que corresponde a la salida de este módulo de procesamiento.

4.3.2. Clasificador Gentle AdaBoost

El clasificador Gentle AdaBoost está encargado de indicar si un vector de características, generado por el módulo de cálculo HOG para una región en particular, presenta o no condiciones necesarias para representar un rostro. Para esto, AdaBoost combina la salida de distintos clasificadores débiles para distintas muestras del descriptor, generando un valor de clasificación

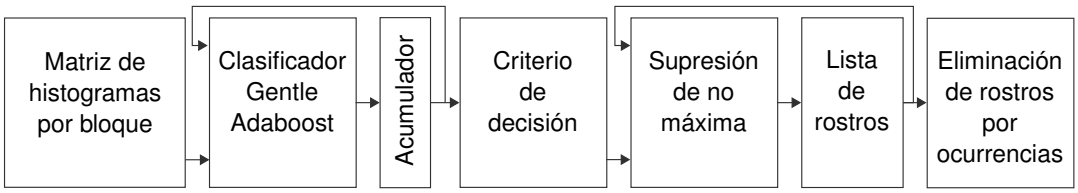


Figura 4.5: Diagrama de flujo de implementación de algoritmo Gentle AdaBoost en software.

que debe ser comparado con un umbral para determinar la pertenencia a una u otra clase.

Como se mencionó en la sección anterior, el algoritmo HOG genera descriptores para todas las posibles regiones, en tamaño y posición, presentes en la imagen, enviando cada una de ellas al clasificador. Por esto, el clasificador recibe como entrada un descriptor de una región perteneciente a la imagen de análisis, en este caso particular este descriptor corresponde a una concatenación de bloques generados con el algoritmo HOG. El descriptor de cada región es analizado, leyendo todas las muestras que fueron seleccionadas como las que menor error de clasificación generaban durante el entrenamiento del sistema, como se ve en el Capítulo 2, cada clasificador débil, asociado a una muestra del descriptor, entrega un valor de clasificación proporcional al acierto producido por éste en el conjunto de entrenamiento. Finalmente, la sumatoria de todas las salidas de los clasificadores débiles, se compara con un umbral para decidir si corresponde o no a un rostro. Este proceso es repetido para cada región posible en un tamaño determinado, lo que implica que la entrada del clasificador se desplaza dentro de la matriz de bloques generados anteriormente. Cada región con clasificación positiva se añade a una lista de salida, pero en caso de que una región previamente añadida esté contenida por la región actual y posea un valor de clasificación menor, la región actual remplaza a la región anterior en la lista, reduciendo la cantidad de salidas y de comparaciones necesarias para eliminar las zonas múltiples de clasificación positiva, actuando como un supresor de no máximos. Una vez realizada la detección a una escala es necesario generar las celdas y bloques de histogramas para un tamaño de región mayor, volver a clasificar cada uno de los descriptores y agregar o remplazar los rostros encontrados en la lista. Finalmente, una vez completado el análisis de toda la imagen, sólo aquellas regiones que hayan sido detectadas más de tres veces, es decir hayan sido reemplazadas por otras regiones en al menos tres oportunidades, son almacenadas para ser enviadas a la etapa de reconocimiento de rostros. El proceso completo se resume en el diagrama de flujo presente en la Figura 4.5, indicando la recurrencia en la generación del valor clasificación y en la supresión de no máximos.

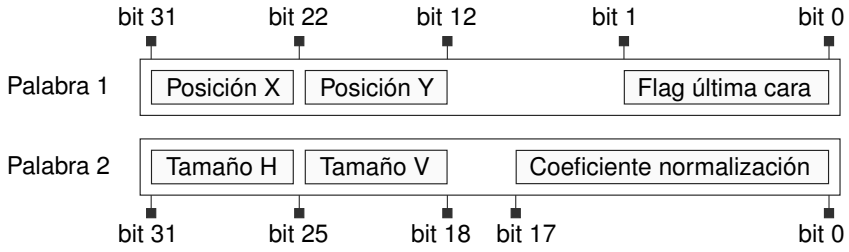


Figura 4.6: Distribución de almacenamiento de salida de detector en memoria.

El resultado de este módulo se almacena en la posición 0x10300000 de memoria RAM, donde, para cada rostro, se guarda la posición en dentro de la imagen; el tamaño de la región divido por el número de regiones que se utilizarán duran el reconocimiento; un *flag* que indica si el rostro corresponde o no al último detectado; y el factor de normalización, que corresponde al inverso multiplicativo del tamaño de la región. Todos estos datos requieren un total de 53 bits, por lo que se utilizan 2 palabras de 32 bits para almacenarlos en RAM, según la distribución presentada en la Figura 4.6.

4.3.3. Discusión

La implementación del detector de rostros en software representa un retardo considerable en el procesamiento del sistema en general, ya que ninguna de sus etapas es paralelizable por la plataforma en la que fueron implementadas. Una implementación en hardware dedicado podría reducir considerablemente el tiempo necesario requerido para procesar un cuadro de video.

Considerando un flujo de datos similar al descrito en las Figuras 4.4 y 4.5, el cálculo de los desplazamientos, ángulos, magnitudes e histogramas iniciales por celda, se puede realizar en línea con la lectura de píxeles, añadiendo latencia pero sin reducir el flujo de salida.

Los bloques de cálculo de histogramas para las regiones del tamaño correspondiente, la generación de histogramas por bloque y la clasificación son procesos que requieren necesariamente de las salidas anteriores, pero pueden ser ejecutadas en paralelo. Por ejemplo, mientras se generan los histogramas de bloques para un tamaño de análisis, es posible calcular los histogramas de celdas para el tamaño siguiente. Lo mismo ocurre con el clasificador, en el que cada etapa puede ejecutarse en paralelo con otras de otro tipo. Con esto es posible generar un *pipeline*, con todos los bloques de procesamiento. Si bien esto añade latencia inicial, no genera ningún

retraso acumulable en un sistema cuya operación es periódica, como en el caso de un sistema de detección de rostros embebido alimentado directamente por una cámara.

4.4. Arquitectura clasificador de rostros

El clasificador facial fue implementado íntegramente en hardware y consta de tres módulos para la implementación de los algoritmos de extracción de características y clasificación. Todos los módulos fueron diseñados para funcionar en paralelo con una arquitectura segmentada. Como criterio de diseño se minimizo el uso de recursos, aumentando el tiempo de procesamiento, pero reduciendo drásticamente el uso de recursos de almacenamiento. La arquitectura diseñada implementa etapas de:

1. Extracción de texturas con $\text{LBPu}_{(8,1)}$.
2. Cálculo de histograma y proyección LDA.
3. Centrado de datos y normalización.
4. Clasificación con distancia Euclidiana.

El módulo de reconocimiento recibe las coordenadas, tamaños y el coeficiente de normalización para el rostro en cuestión. Naturalmente cada rostro se lee desde memoria como un flujo de píxeles horizontales. Sin embargo, como la imagen se divide en regiones para crear el histograma descriptor con LBP, es posible realizar la lectura del rostro por regiones, disminuyendo así la cantidad de coeficientes LDA almacenados en memoria interna a sólo los necesarios para realizar la proyección de una región. Para lograr esto se diseñó, además de los módulos de procesamiento para implementar el algoritmo, un controlador de memoria para realizar la correcta lectura de los píxeles de cada región de la imagen y de los coeficientes LDA correspondientes a dicha región. El módulo de reconocimiento de rostros entrega como salida la identidad con la que fue clasificado el rostro de entrada, en caso de que el rostro no esté lo suficientemente cerca de alguna de las clases en la base de datos el módulo entrega un cero como salida.

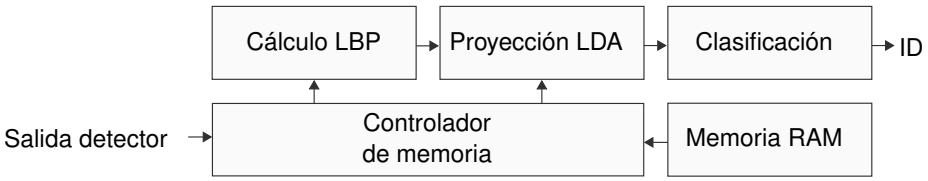


Figura 4.7: Diagrama de flujo de clasificador de rostros.

4.4.1. Cálculo LBP uniforme

El módulo de cálculo del operador $LBP_{(8,1)}$ uniforme realiza comparaciones de todos los píxeles con su vecindario de 3×3 píxeles, obteniendo una etiqueta que representa una micro textura asociada a cada vecindario. Esto genera una imagen de un píxel menos por extremo, cuyo histograma puede ser utilizado para representar la imagen original. Para lo que la imagen es dividida en regiones no solapadas, concatenando los histogramas de cada una de ellas. En el enfoque propuesto en este trabajo, el procesamiento se realiza en cada región por separado, esto es transparente para el cálculo del operador LBP, ya que sólo se debe indicar la cantidad de filas y columnas que posee la región que se está procesando y cuidar que la entrada de píxeles sea correcta, con una señal de entrada válida.

El módulo diseñado recibe un flujo de píxeles en escala de grises de 16 bits, obtenidos desde una cámara infrarroja. Con éstos que se genera una salida de 6 bits, los que corresponden a la precisión necesaria para representar cualquiera de los 59 patrones uniformes en $LBP_{(8,1)}$. Por el enfoque de diseño de este módulo, el tamaño de cada región procesada debe ser variable, ya que las regiones del borde de una imagen son de menor tamaño, además el clasificador debe ser capaz de identificar rostros de distinto tamaño, por esto, el módulo de procesamiento de LBP posee como entrada, además de los datos, el tamaño de cada región a procesar con una precisión de 10 bits.

Como se ve en la Figura 4.8, el procesamiento del operador LBP se realiza con dos *buffers* de línea que almacenan los píxeles de entrada necesarios para el cálculo del operador LBP. Uno de estos buffers es alimento directamente desde la entrada del módulo, el otro se alimenta con la salida del primer buffer. Así, los buffers almacenan siempre las últimas dos lineas de la imagen. También se utilizaron nueve registros que representan el vecindario de 3×3 píxeles con el que se calcula el operador LBP, esto se muestra en la parte central de la Figura 4.8. La salida de estos registros alimenta ocho comparadores que generan el patrón LBP, éste es utilizado para

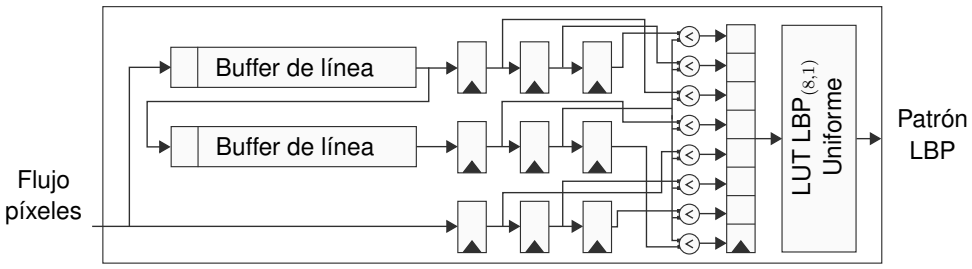


Figura 4.8: Diagrama módulo cálculo operador LBP uniforme.

indexar una memoria Block RAM de 256 entradas y 59 salidas, con la que se traduce el patrón LBP al patrón uniforme asociado a éste.

En el comienzo de cada región los valores de intensidad de cada píxel son almacenados en los dos *buffers* de línea descritos anteriormente y en la última línea del bloque de 3×3 registros, además la salida de cada *buffer* se almacena en su respectiva fila del arreglo de nueve registros, según se ve en la Figura 4.8. Esto se repite sin generar resultados hasta que se recibe el tercer píxel de la tercera línea de datos de entrada, con la cual se completa el vecindario de 3×3 píxeles por primera vez, generando un patrón correspondiente a ese vecindario. Las comparaciones entre el vecindario se realizan durante toda la ejecución del módulo, pero sólo en cuando todos los valores presentes en el arreglo de registros son válidos, se generan patrones válidos, que son traducidos a patrones LBP uniformes y representan la salida del módulo. Todos los patrones generados que corresponden a una salida válida son acompañados por una señal que indica esta condición. El módulo además genera una salida de fin de región cuando se genera el último patrón asociado a una región, con éste se le indica al módulo de proyección LDA que los siguientes patrones corresponderán a la próxima región de procesamiento.

4.4.2. Proyección LDA

Tras realizar el cálculo del operador LBP para cada píxel de una región, se debe generar el histograma de ésta y luego realizar la proyección LDA. En este trabajo ambas etapas se implementaron en un mismo módulo, ya que, la construcción de un histograma y su posterior proyección, pueden reducirse a una acumulación de sumas de los coeficientes de proyección. El realizar esta consideración reduce los recursos de almacenamiento necesarios para el histograma y el uso de multiplicadores dedicados. Sin embargo, al realizar la proyección de esta forma, no es posible centrar los datos restando la media del conjunto de entrenamiento, pero como

la proyección es lineal, esta operación se puede postergar y ser realizada en el subespacio de clasificación, lo que reduce la memoria necesaria para almacenar la media del conjunto de entrenamiento. Tal como se indica a continuación:

$$y = W^T \cdot (x - \mu) = W^T \cdot x - W^T \cdot \mu \tag{4.1}$$

donde y es el vector proyectado, x el histograma descriptor, μ la media del conjunto de entrenamiento y W la matriz de proyección al subespacio LDA.

Considerando lo anterior, el proceso de proyección LDA consiste en realizar una correcta lectura de los coeficientes de proyección de la zona que está siendo procesada y acumular el valor de estos coeficientes. Esto se debe repetir hasta completar el proceso para el total la imagen.

El módulo de proyección LDA tiene como entrada patrones LBP uniformes generados por el módulo de cálculo de LBP, éstos tienen un ancho máximo de 6 bits y son utilizados para indexar las memorias de coeficientes. Otra entrada del módulo es la de los valores de los coeficientes LDA, que son enviados para reemplazar a los de una región previa, reduciendo los recursos de memoria necesarios. Éstos se reciben a través de un bus de comunicación de 32 bits que transmite los valores de los coeficientes; una entrada de 6 bits que indica a que patrón corresponden los coeficientes que se reciben; y una entrada de 13 bits que indica que memorias de coeficientes se deben escribir. El ancho del último bus mencionado es parametrizable y depende del número de proyecciones utilizadas en el sistema. La única salida del módulo corresponde al vector de características proyectado, éste corresponde a un arreglo de registros de 20 bits con signo, que es transmitido al módulo de clasificación de forma secuencial una sola vez por imagen.

Para cada una de las proyecciones se utilizó un doble *buffer* para las memorias de coeficientes, con el fin de minimizar el tiempo de espera entre una región y otra. El primero de estos *buffers* se utiliza para procesar los patrones LBP de la región actual, mientras el segundo almacena los coeficientes necesarios para procesar la siguiente región. Cada uno de estos *buffers* tiene un tamaño de 59 elementos, que corresponde a la cantidad de patrones LBP uniformes, y almacena, en 8 bits, la parte decimal de los coeficientes de proyección de una región. Cada *buffer* fue implementado en memoria distribuida ya que debido a su bajo tamaño una implementación en Block RAM resultaría en un consumo demasiado elevado de este tipo de recurso. La salida del doble *buffer* es acumulada en un registro de 25 bits, cuyos 20 bits más significativos corresponden a una de las salidas del módulo. Todo lo anterior se realiza para cada proyección utilizada en el sistema, lo que es parametrizable durante la configuración de éste. En la Figura 4.9 se ven dos

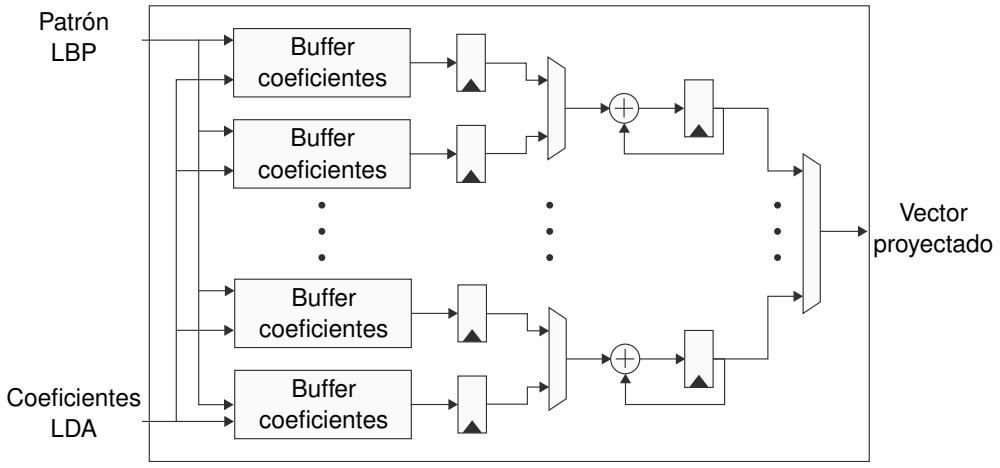


Figura 4.9: Diagrama módulo proyección LDA.

instancias del doble *buffer* y de sus registros y acumuladores correspondientes.

El procesamiento del módulo de proyección LDA comienza al recibir una señal de *start* proveniente del controlador de memoria. Con esta señal se indica que comenzará el procesamiento de una imagen, sin embargo esto sólo se realiza cuando llegan patrones válidos desde el módulo cálculo LBP. Todos los patrones recibidos son procesados, obteniendo su coeficiente respectivo y sumándolos al acumulador, a la vez el módulo recibe los coeficientes LDA asociados a la siguiente región de procesamiento. Cuando se detecta un fin de región, indicado por una señal del módulo de cálculo LBP, se cambia la selección de los multiplexores presentes en la Figura 4.9 y se espera una nueva señal de *start* para comenzar el procesamiento de la siguiente región. En la próxima región los *buffers* intercambian su función, así el que contenía los coeficientes de la región anterior es sobrescrito, almacenando en él los coeficientes de la siguiente región. Este proceso se repite para todas las regiones en las que se divide un rostro, generando el vector proyectado sin centrar, que es enviado posteriormente, de forma secuencial, al módulo de clasificación para centrarlo y asociarlo a una de las clases presentes en el conjunto de entrenamiento del sistema.

4.4.3. Clasificador por vecino más cercano

La última etapa del proceso de reconocimiento de rostros, con el algoritmo LBP+LDA, corresponde a la clasificación de los vectores en el subespacio LDA. Este proceso se implementó utilizando el método del vecino más cercano con distancia Euclidiana, para esto se requiere

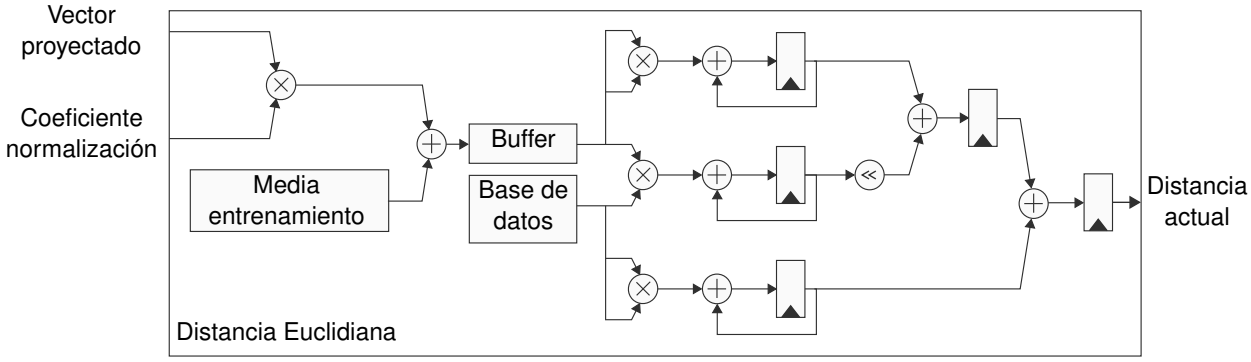


Figura 4.10: Diagrama módulo clasificador, cálculo distancia Euclidiana.

comparar el vector proyectado con los centroides de cada clase presente en la base de datos, calculando su distancia a cada uno de ellos. Posteriormente, y sólo si la distancia es menor a un umbral definido, se selecciona la clase a la que es más próximo el vector de análisis y se entrega como salida del módulo y del núcleo de reconocimiento facial. Por la forma en que se diseño la arquitectura del sistema de reconocimiento facial, el módulo de clasificación, además de realizar el proceso de identificación, debe centrar el vector proyectado antes de realizar la búsqueda del vecino más cercano.

Para implementar el cálculo de la distancia Euclidiana, ésta se separó en tres componentes, considerando que:

$$d(P,Q) = \sqrt{\sum (p_i - q_i)^2} = \sqrt{\sum p_i^2 - 2\sum p_i \cdot q_i + \sum q_i^2} \tag{4.2}$$

donde $d(P,Q)$ representa la distancia Euclidiana entre los vectores P y Q con componentes p_i y q_i respectivamente. Además, considerando que se requiere calcular la proximidad entre dos vectores y no la distancia en sí, es posible utilizar $d^2(P,Q)$ como medida de similitud. Así, el módulo de clasificación por vecino más cercano implementa la distancia Euclidiana como:

$$d^2(P,Q) = \sum p_i^2 - 2\sum p_i \cdot q_i + \sum q_i^2 \tag{4.3}$$

Este módulo recibe datos provenientes del módulo de proyección LDA, los que tienen un ancho de 20 bits y corresponden a valores con signo. La única salida de datos del módulo corresponde a la clase con la que fue identificada el rostro de entrada, para la que se utilizaron 8 bits.

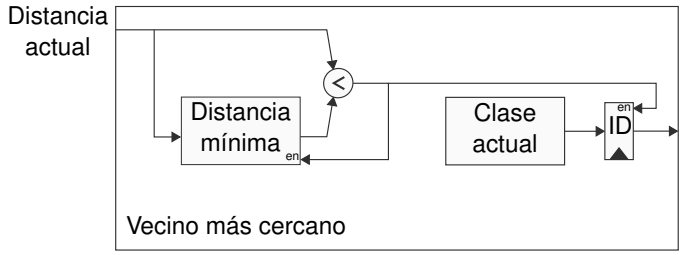


Figura 4.11: Diagrama módulo clasificador, cálculo del vecino más cercano.

Como se muestra en la Figura 4.10 se utilizó un *buffer* de entrada para almacenar el vector proyectado en el módulo LDA, pero con cada una de las muestras centradas. La dimensión de este *buffer* es igual al número de proyecciones utilizadas y cada uno de los elemento es de 26 bits con signo. El cálculo de la distancia Euclidiana (Ecuación. 4.3) se realizó utilizando tres acumuladores de 48 bits con signo, indicados en la parte central de la Figura 4.10, dos sumadores y un desplazador binario. Para el proceso de búsqueda del vecino más cercano al descriptor actual, ilustrado en la Figura 4.11, sólo se necesitó un comparador y un registro para almacenar la distancia mínima entre las calculadas para todas las clases. La base de datos de referencia se almacenó en una Block RAM de sólo lectura de 16 bits, con un número de elementos equivalente al número de proyecciones por el número de clases.

El módulo de clasificación, usando el método del vecino más cercano, comienza cuando se recibe el vector proyectado desde el módulo LDA. Durante la recepción del vector cada muestra es centrada, restando la media proyectada de los elementos de la base de datos, y normalizada, según el tamaño del rostro. El resultado de esta operación, que corresponde a una multiplicación suma, es almacenado en el *buffer* de entrada de la Figura 4.10. Una vez terminado lo anterior comienza el cálculo de la distancia Euclidiana entre el vector almacenado y el centroide de la primera clase de la base de datos. Para esto, cada uno de los componentes en que se dividió la distancia Euclidiana (Ecuación 4.3) es calculado en paralelo, esto realiza las operaciones $\sum p_i^2$, $\sum q_i^2$ y $\sum p_i \cdot q_i$, utilizando tres unidades de multiplicación con acumulador de salida. Posteriormente al resultado del primer acumulador $(\sum p_i^2)$ se le resta el segundo $(\sum p_i \cdot q_i)$ desplazado en un bit hacia la izquierda, al resultado de esto se le suma el tercer acumulador $(\sum q_i^2)$ obteniendo el resultado final de la distancia Euclidiana al cuadrado. Luego esta distancia es comparada con la distancia mínima almacenada hasta el momento, la cual se inicializa en el valor de umbral de pertenencia a una clase. Así, en caso de que la distancia calculada sea menor al umbral, la distancia mínima es reemplazada por la actual y la clase correspondiente es copiada en el registro que indica la clase más cercana. Este proceso se muestra en la Figura 4.11

y es repetido para todas las clases encontrando la distancia mínima en línea con el cálculo de éstas. Finalmente, una vez que se analizó la proximidad a todas las clases de la base de datos, la clase asociada a la menor distancia es enviada fuera del módulo de reconocimiento de rostros, junto a una señal que indica su validez.

4.4.4. Controlador de acceso a memoria

El controlador de acceso a memoria, diseñado para el módulo de reconocimiento de rostros, está encargado de leer de forma correcta los píxeles pertenecientes a una de las regiones del rostro, y el valor de los coeficientes de proyección asociados a esa región, ambos almacenados en RAM. Además está encargado de enviar estos datos a los módulos LBP y LDA respectivamente.

Para realizar lo anterior se utilizaron dos Block RAMs para almacenar los datos leídos desde memoria RAM, y dos máquinas de estado para controlar la lectura de ambos tipos de datos. Cada de una de estas máquinas cuenta con contadores de filas, columnas y regiones, para la modificación de la cantidad de datos a solicitar en el caso de la lectura de píxeles, y para determinar cuando se llegó a la última región de la imagen en ambas lecturas.

El controlador de memoria comienza su funcionamiento cuando se reciben los datos de posición y tamaño de cada región de un rostro. En ese momento las dos máquinas de estado inician la solicitud de píxeles de la primera región de la imagen, y de los valores de coeficientes LDA de la segunda región.

La lectura de píxeles se realiza línea a línea, ya que la imagen obtenida desde la cámara es almacenada como un conjunto de filas en memoria RAM. Por esto, para realizar la lectura completa de una región es necesario realizar tantas solicitudes de datos como filas tenga la región, lo que implica un tiempo de lectura variable, ya que no existe un tiempo fijo de respuesta para las solicitudes a RAM. Asi, como se indicó en la arquitectura del módulo LBP, los píxeles son enviados por fila y de forma no continua, indicando con una señal cuales píxeles son válidos. La cantidad de líneas a enviar y el tamaño de cada una de ellas no es fijo, ya que varia según la región a enviar, las regiones de los extremos tienen un píxel menos que las regiones interiores, por lo que la máquina de estado que controla la lectura y envío de píxeles cambia el tamaño de las regiones de lectura de la imagen original, para que las regiones con LBP aplicado tengan el tamaño adecuado. Una vez que se envía completamente una región, la máquina de estado espera la finalización de la lectura de los coeficientes LDA, con el fin mantener la sincronización entre ambas entradas del módulo de reconocimiento facial. El proceso de lectura, envío y actualización

de los tamaños de región, se realiza para cada una de las regiones presentes en cada imagen hasta completar el procesamiento de todo el rostro.

La lectura de los coeficientes de proyección LDA se realiza siempre para la región siguiente a la actual, ya que al enviar las intensidades de los píxeles a al módulo LBP, éstos son procesados y enviados al módulo LDA para ser proyectados, por lo que los coeficientes de la región actual deben estar previamente almacenados, debido a esto, como se explicó anteriormente, este módulo se implemento con un doble *buffer*. La lectura de los coeficientes es más simple que la de los píxeles, ya que todos los valores de una región se encuentran contiguos en memoria RAM, por lo que sólo se requiere de una solicitud para realizar la lectura. El envío de datos al módulo LDA se realiza enviando todos los coeficientes de todas las proyecciones para un sólo patrón, repitiendo el proceso para todos los patrones. Al finalizar, al igual que en la máquina de estado que maneja la lectura de píxeles, se espera el fin de la lectura y envío de píxeles, ya que en ningún caso se puede determinar el tiempo exacto de ejecución, por lo que es imperativo mantener la sincronización entre ambos procesos de lectura.

Considerando el uso de 52 proyecciones, de 8 bits cada una, para 59 patrones, se requiere de 767 ciclos para transmitir todos los valores de una región al módulo LDA, lo que equivale a 7,67 μS con un reloj de 100 MHz. Esto significa que, sin considerar los retardos en la lectura desde RAM, una región de hasta 27×27 píxeles tardará menos tiempo en ser enviada (7,29 μS) al módulo LBP que los coeficientes al módulo LDA. Para imágenes de 28×28 píxeles o más el envío de los coeficientes tardará menos tiempo, por esto la sincronización entre ambas máquinas de estado es suma importancia y en ambos casos el fin de el procesamiento es seguido de una etapa de espera de la finalización del procesamiento de la otra máquina, ya que ninguna puede comenzar sin que la otra termine la lectura y envío de los datos al módulo respectivo.

En base a lo anterior, es posible afirmar que el tiempo mínimo necesario para el procesamiento de LDA, considerando que cada imagen es dividida en 64 regiones (8×8), es de 490 μS, con esto es posible tener hasta 876 clases en la base de datos sin modificar el enfoque planteado.

4.5. Periféricos

Para que los módulos de procesamiento principales funcionen se requiere de controladores, interfaces y sistemas de despliegue de datos, con el fin alimentar, recibir, desplegar y transmitir datos entre el exterior del sistema y los módulos de detección y reconocimiento de rostros. Todos

los módulos desarrollados para este propósito son conocidos como periféricos, pues si bien son imprescindibles para el funcionamiento del sistema completo, no son el objeto del desarrollo principal del sistema.

4.5.1. Interfaz detección-reconocimiento

La transferencia de datos entre el módulo de detección y el de reconocimiento de rostros se realizó a través de datos compartidos en memoria RAM. Éstos fueron almacenados según la distribución de la Figura 4.6, guardando la posición dentro de la imagen, el tamaño y el coeficiente de normalización asociado a cada rostro. Además el último rostro almacenado en la lista de salida del detector contiene un *flag* que indica que debe ser el último en ser procesado. En base a esto, una de las tareas principales de la interfaz entre el módulo de detección y el de clasificación, es realizar la lectura, desde RAM, de los datos de todos los rostros detectados y enviarlos uno por uno al módulo de reconocimiento, esperando en cada uno de ellos la finalización del proceso de clasificación para enviar el siguiente, repitiendo esto hasta encontrar el flag que indica el último rostro de la lista de detecciones.

La otra tarea del módulo de interfaz entre el detector y el clasificador de rostros, consiste en almacenar todas las salidas válidas del clasificador y, una vez recibida la última, escribirlas en la memoria RAM compartida, como un arreglo de identidades de 8 bits cada una y en el mismo orden en que fueron almacenadas las salidas del detector. Una vez que la escritura es realizada, en la dirección 0x10200000 como se ve en la Figura 4.3, se envía una interrupción indicando que el proceso de reconocimiento finalizó y que las identidades de cada rostro se encuentran en RAM.

4.5.2. Recepción de parámetros

Gran parte de los parámetros utilizados en el funcionamiento del sistema son recibidos a través de Ethernet y almacenados en memoria RAM según la distribución de la figure 4.3. Para realizar esto se desarrolló un programa en el procesador del SoC capaz de leer los distintos tipos de parámetros identificados por un código inicial, y copiarlos a la posición de memoria correspondiente, para su posterior uso en los módulos de detección y de reconocimiento. Los parámetros recibidos por Ethernet son: las características, umbrales y valores de peso del clasificador Gentle AdaBoost para detección, cuya dirección de almacenamiento no se muestra

en la Figura 4.3 porque no son de uso compartido con la lógica programable; y los coeficientes de proyección al subspacio LDA para el reconocimiento que se almacenan en la posición 0x10100000.

4.5.3. Despliegue de resultados

Una vez que el reconocimiento de los rostros encontrados durante la detección es finalizado, la lista de identidades es almacenada en RAM y una señal de término es enviada al procesador del SoC, esta señal es atendida como una interrupción, por lo que las identidades encontradas son desplegadas inmediatamente cuando se finaliza el reconocimiento, lo que se realizó mediante el envío de los datos por serial para su posterior análisis.

Los resultados de la etapa de detección son desplegados en pantalla por la salida VGA. Cada rostros detectado es encasillado en un cuadrado de color verde, en la posición y tamaño correspondiente a la salida del filtrada del clasificador. Cada cuadro es dibujado en hardware, los datos de posición y tamaño son enviados por el bus de comunicación AXI desde el procesador a la lógica que realiza el dibujo en pantalla. Además del encasillamiento, el despliegue de resultados entrega la cantidad de rostros presentes en la escena, con un módulo en hardware de funcionamiento similar al de dibujo de cuadrados.

4.6. Flujo de datos del sistema

Como se ha visto a lo largo de este capítulo, de diseñó una arquitectura mixta para implementar los algoritmos de detección HOG con clasificador Gentle AdaBoost para detección, y el algoritmo LBP+LDA con clasificación por el método del vecino más cercano en reconocimiento. El funcionamiento del sistema, a grandes rasgos, consiste en la búsqueda de rostros dentro de una imagen almacenada en RAM, y la posterior clasificación de estos rostros para conocer su identidad respecto a una base de datos.

El flujo general del sistema se ilustra en la Figura 4.12, donde se pueden ver las distintas etapas del detector y del clasificador de rostros. Todo el proceso comienza con una señal de *start*, la que, en funcionamiento normal, corresponde a la señal del fin de recepción de un cuadro de video. Una vez que se recibe la señal de *start*, se calculan los histogramas de gradientes orientados (HOG), para cada una de las mínimas celdas de la imagen, como se explicó anteriormente, esto se

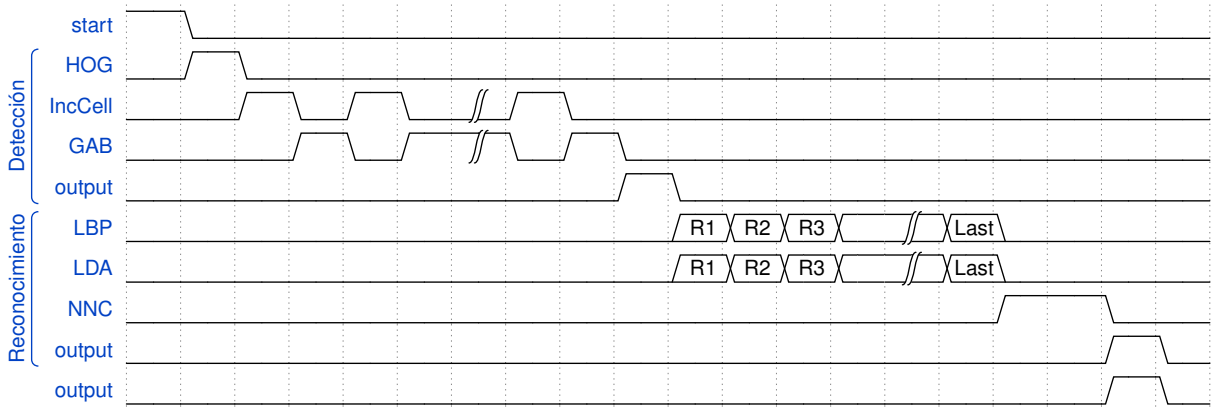


Figura 4.12: Diagrama de flujo de datos para el sistema.

realiza sólo una vez por cuadro. Luego las celdas son incrementadas en tamaño para representar un mayor área de la imagen, lo que se realiza sumando los histogramas por celda generados en la etapa anterior. Las nuevas celdas son concatenadas y normalizadas, formado bloques, los que son utilizados como descriptores de regiones de la imagen y a los que se aplica el clasificador Gentle AdaBoost. Lo anterior se repite para todos los tamaños posibles de regiones, que en el caso del sistema diseñado corresponden a 23, con regiones desde 40×40 hasta 500×500 píxeles, con un incremento de 20 píxeles en cada etapa. Tras realizar la clasificación en todos los tamaños y posiciones posibles, se genera la salida del detector, la cual es almacenada en memoria RAM y leída por sistema de reconocimiento facial. Toda la etapa de detección se implementó en software, por lo que su ejecución es secuencial y no se pueden realizar operaciones en paralelo, tal como se ve en la Figura 4.12.

Tras ser generada la salida del detector, el módulo de reconocimiento comienza la lectura de datos de desde RAM y el envío de éstos a los módulos de procesamiento. El primero módulo en ser utilizado es el del cálculosles y genera un patrón asociado a estos. Cada patrón generado es enviado al módulo de proyección LDA. Si bien el procesamiento de ambos módulos no ocurre al instante, la salida del módulo LBP alimenta directamente al de proyección LDA, por lo que durante la mayor parte del tiempo ambos funcionan en paralelo, tal como se ve en la Figura 4.12, donde se indica que el procesamiento se realiza para todas las regiones presentes en cada imagen. Una vez que la proyección se completa para todas las regiones comienza el proceso de clasificación por el método del vecino más cercano (NNC), con la cual se busca la clase a la que el descriptor generado presenta menor distancia Euclidiana. Finalmente la salida del clasificador presenta el resultado final de todo el módulo de reconocimiento y, en caso de corresponder al último rostro detectado, el fin de todo el procesamiento del sistema.

Capítulo 5. Resultados

5.1. Introducción

La arquitectura diseñada y presentada en el Capítulo 4 se implementó en un *System on Chip* Zynq-7000, que integra un procesador ARM con un FPGA. Como la arquitectura diseñada es mixta se desarrolló en dos ambientes de trabajo distintos. La parte implementada en hardware se realizó en lenguaje Verilog HDL y fue sintetizado y mapeado al FPGA con el software Vivado Design Suite de Xilinx. La parte implementada en software fue escriba en C con el software Xilinx Software Development Kit de Xilinx.

En este capítulo se describe brevemente el hardware utilizado; se detalla la configuración con la que se llevaron a cabo los experimentos de validación del sistema; y se presenta la metodología y los resultados obtenidos por la implementación realizada.

5.2. Descripción de hardware

Para implementar la arquitectura diseñada se utilizó una placa de desarrollo ZedBoard, la cual integra un SoC Zynq-7000 XC7Z020-CLG484. La placa desarrollo presenta todos los periféricos necesarios para el funcionamiento de este trabajo, como son: interfaz de comunicación Ethernet, memoria RAM externa, salida de video VGA y la posibilidad de añadir una entrada CameraLink a través del puerto de comunicación FMC. Los principales componentes que se integran en la tarjeta de desarrollo son:

- SoC Zynq-7000 XC7Z020-CLG484.
- 512 MB de Memoria RAM DDR3.
- Interfaz Gigabit Ethernet.
- Salida VGA de 8 bits y HDMI de hasta 1080p.
- Puerto I/O de alta velocidad FMC.
- 8 switch, 8 LEDs y 5 botones para uso de usuario.

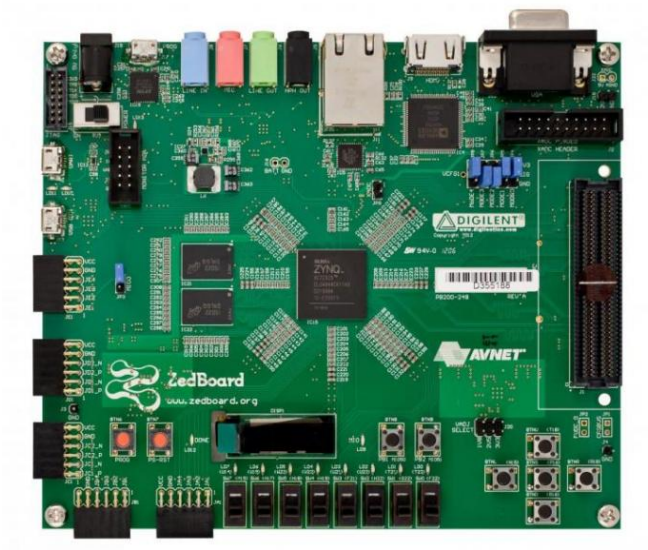


Figura 5.1: Tarjeta de desarrollo ZedBoard [5].

La lógica programable incluida en el SoC Zynq-7000 XC7Z020-CLG484 corresponde a la serie 7 de Xilinx y es equivalente a la incorporada en la familia Artix-7. Ésta cuenta con las siguientes características:

- 53.200 LUTs.
- 106.400 Flip-Flops.
- 560 KB de Block RAMs, distribuidas en a 140 Block RAMs.
- 220 DSP.

El procesador incorporado en el SoC corresponde a un ARM Cortex-A9, el cual posee una arquitectura ARMv7 y funciona a 1 GHz. La comunicación entre el procesador y la lógica programable se realiza utilizando los buses de comunicación AXI, que corresponden a un estándar de comunicación en procesadores ARM.

5.3. Metodología de prueba

La evaluación del sistema se realizó de forma similar a la realizada durante las pruebas en software, pero, considerando la función para la que fue diseñado el sistema, no es posible

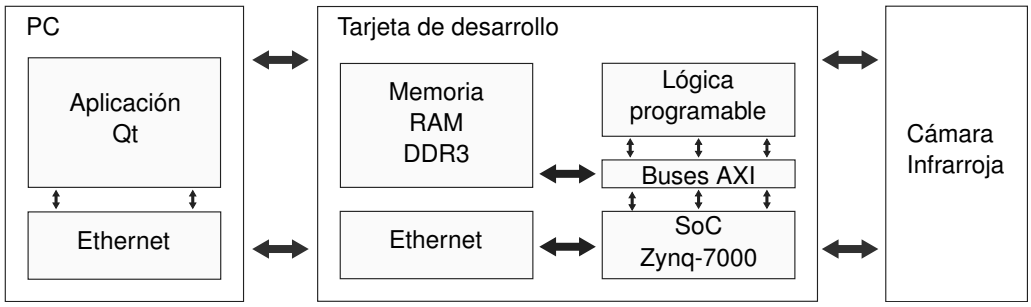


Figura 5.2: Diagrama de conexión entre tarjeta de desarrollo y computador.

obtener directamente las tasas de acierto desde el sistema embebido. Por esto se realizó una comparación numérica, con lo que se aseguró que los resultados obtenidos en el sistema embebido son equivalentes a los obtenidos en una implementación en software utilizando las bases de datos presentadas en el Anexo A.

A diferencia de las pruebas en software para la etapa de detección, la implementación se realizó para detectar rostros entre 40×40 y 500×500 píxeles. Esto se realizó por la baja superficie real que presenta una región de 24×24 píxeles, como las utilizadas para comparar los algoritmos en el Capítulo 3, dentro de una imagen de 640×512 píxeles, como las obtenidas por la cámara infrarroja. En resumen, las pruebas de detección realizadas para el sistema embebido se realizaron con la misma base de datos generada para las pruebas en software del Capítulo 3, pero con imágenes de rostros de 40×40 píxeles. Si bien el entrenamiento se realizó sólo con imágenes de la base de datow LWIR UdeC, también se realizaron pruebas con imágenes obtenidas desde de la cámara infrarroja y con imágenes creadas de forma artificial con rostros de las otras bases de datos utilizadas en reconocimiento.

En el caso del reconocimiento de rostros se utilizó la base de datos UCHThermalFace, pero todas las imágenes debieron ser modificadas, ya que la salida del detector entrega regiones cuadradas. Por esto los resultados se obtuvieron con rostros de 81×81 píxeles, en lugar de los 81×150 originales.

En la Figura 5.2 se ilustra el *setup* utilizado para realizar las pruebas del prototipo. En ésta se diferencia el computador que provee los parámetros de entrenamiento, la cámara que se utiliza como fuente de imágenes y la tarjeta de desarrollo. Además se destaca el flujo de datos durante el funcionamiento normal, donde los parámetros son enviados desde un computador a la tarjeta de desarrollo a través del interfaz Ethernet, éstos son recibidos por el procesador y almacenados

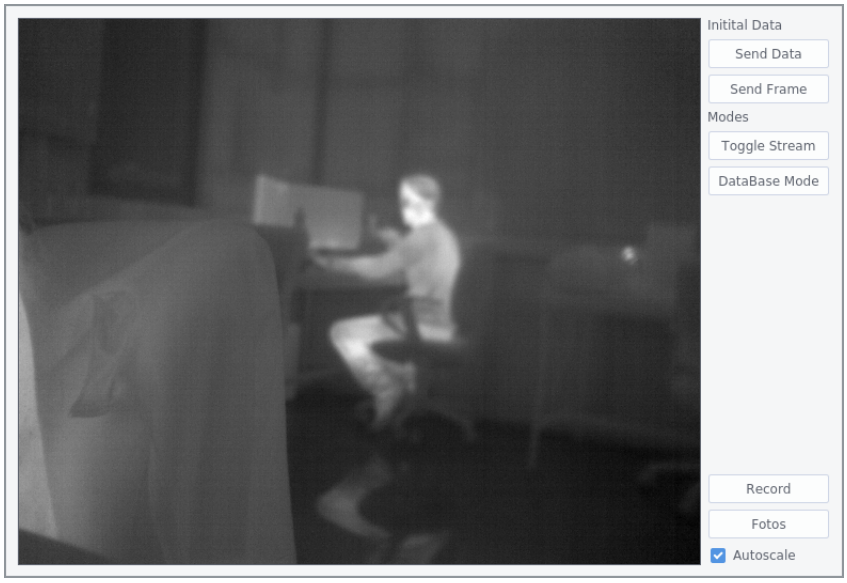


Figura 5.3: Aplicación diseñada para envío de parámetros a tarjeta de desarrollo.

en la memoria RAM, para poder ser accedidos desde la lógica programable usando los buses de comunicación AXIs. Los datos provenientes de la cámara infrarroja son recibidos por un controlador de CameraLink implementado en hardware, éstos son almacenados en RAM, para posteriormente ser analizados por el detector facial en el procesador. Una vez que se produce la salida del detector se inicia el proceso de reconocimiento, cuya salida es almacenada en RAM y enviada por comunicación serial al computador donde pueden ser analizados. El sistema embebido en funcionamiento se puede ver en la Figura 5.4.

Como se mencionó en el Capítulo 4, los parámetros del clasificador Gentle AdaBoost y los coeficientes de proyección LDA, deben ser enviados a la tarjeta de desarrollo por la interfaz Ethernet. Para este fin se desarrolló una aplicación usando las bibliotecas Qt [51], cuya interfaz de usuario se ve en la Figura 5.3. La aplicación desarrollada también permite el envío de imágenes, a la tarjeta de desarrollo, posibilitando el análisis de imágenes artificiales o capturadas con anterioridad.

El reporte de utilización de recursos corresponde al generado por el software de desarrollo Vivado Design Suite, tras la ejecución de la etapa de *place and route*, donde el software decide a que recursos físicos del FPGA corresponderá cada elemento del diseño realizado en la descripción de hardware en Verilog HDL.

El análisis de potencia se obtuvo con la herramienta XPower Analyzer de Xilinx incorporada en Vivado. Esta herramienta realiza simulaciones, considerando un funcionamiento a temperatu-

(a) Conexión entre cámara infrarroja y tarjeta de desarrollo.

(b) Funcionamiento del sistema completo.

Figura 5.4: Configuración de prueba utilizada.

Tabla 5.1: Tasas de acierto del sistema para la base de datos LWIR UdeC en detección y UCHThermalFace en reconocimiento.

Etapa	**Tasa de acierto**	
	Pruebas software	Implementación
Detección	100 %	99.8 %
Reconocimiento	99.53 %	99.21 %

ra ambiente y una entrada de datos estándar, para lo que se genera una estimación de potencia, separable en módulos de procesamiento y recursos lógicos.

5.4. Resultados en SoC

Los resultados de aciertos obtenidos por el sistema se muestran en la Tabla 5.1, donde se utilizaron los conjuntos de entrenamiento que mejores resultados presentaron en las pruebas realizadas en el Capítulo 3, pero con metodología mencionada en la sección anterior.

El uso del sistema en condiciones reales, con imágenes obtenidas por la cámara infrarroja, presenta resultados buenos, considerando que el sistema fue entrenado sólo con imágenes de otra cámara. En la Figura 5.5 se muestran las tres salidas del detector de rostros, aplicando el sistema de descarte de regiones explicado en el Capítulo 4, para eliminar regiones que representan a un mismo rostro. En la Figura 5.6 se muestra la salida generada por el detector para 12 imágenes de prueba, nueve tomadas con la cámara infrarroja, una de la base de datos LWIR UdeC, base

(a) Salida directa del clasificador.

(b) Salida del clasificador tras eliminación de regiones concentricas.

(c) Salida final del clasificador tras eliminación de regiones sin repeticiones.

Figura 5.5: Salidas en distintas etapas del detector.

de datos con la que se entrenó el sistema; una con rostros de la base de datos UCHThermalFace generada de forma artificial; y una con imágenes de la base de datos UCHThermalFace, LWIR UdeC e imágenes tomadas con la cámara infrarroja. En los resultados obtenidos se puede ver que mayoritariamente sólo se detectan rostros verdaderos, pero también se producen falsos positivos, particularmente en regiones que representan hombros, codos y espaldas.

El reporte de recursos del sistema completo se muestra en la Tabla 5.2, donde se puede observar que la mayor parte de recursos se utilizan para realizar el despliegue de datos por VGA, seguido por la lógica necesaria para realizar las interconexiones entre los módulos que utilizan el bus de comunicación AXIs. El módulo de reconocimiento facial sólo supera a los anteriores en el uso de Slice LUTs y Slices, presentando un uso de recursos reducido en general. Las tablas LUTs utilizadas como memoria, son el recurso más utilizado por el sistema con un 52.3 % del total disponible en el chip; le siguen los LUTs con un 26.0 %; Slice Registers con un 18.2 %; Slice con un 10.5 %; y Block RAMs y DSPs con un 8.2 % en ambos casos.

El uso de recursos para núcleo de reconocimiento de rostros se individualiza en la Tabla 5.3, donde se separa la utilización para cada uno de los módulos interiores, cuyas arquitecturas se expusieron en el Capítulo 4. En todos los módulos se puede ver la coherencia entre el diseño realizado y el reporte de recursos entregado por el software de desarrollo de Xilinx.

En el módulo LBP, cuya principal unidad utilizada corresponde a los *buffers* de línea, se implementó uno de los *buffers* en una Block RAM y el otro en memoria distribuida, ya que para el primer *buffer* se requiere que la lectura se realice en un ciclo, para no retrasar el flujo de datos de entrada, esto impide su implementación en Block RAM ya que no se cumple con la estructura requerida. En el reporte se puede ver el uso de 0.5 Block RAMs, equivalente a la mitad de una

Figura 5.6: Salida del detector para 12 imágenes diferentes, nueve obtenidas por la cámara infrarroja, una correspondiente a la base de datos LWIR UdeC, y dos generadas de forma artificial para contener rostros de la base de datos UCHThermalFace y de distintas bases de datos. Se pueden ver falsos positivos generados por el clasificador en distintas regiones.

Tabla 5.2: Uso de recursos del sistema completo.

Sección	**Slice LUTs**	**Slice Regs**	**Slice**	**LUT Mem**	**BRAM**	**DSPs**
Clasificador facial	4732	2817	1349	1320	2.5	6
Despliegue de datos	4382	11307	3813	6	1.5	12
Interconexión AXI	4295	4588	1571	495	6.0	0
Interfaces RAM	361	467	151	0	0.0	0
Interfaz Det-Rec	46	132	46	0	1.0	0
Interfaz CameraLink	18	85	22	0	0.5	0
Total utilizado	13834	19396	6952	1821	11.5	18
Disponible	53200	106400	13300	17400	140	220
Porcentaje utilizado	26.0 %	18.2 %	52.3 %	10.5 %	8.2 %	8.2 %

Block RAM de 36 Kb disponibles en el FPGA de serie 7, que pueden ser configurada como una Block RAM dual de 18 Kb. Esta Block RAM se utiliza para almacenar los datos de uno de los *buffers* de línea, mientras el otro se almacena en memoria distribuida, utilizando 23 LUTs, ya que cada una de ellas puede almacenar 64 bits y cada *buffer* tiene 90 elementos de 16 bits.

El módulo de proyección LDA no utiliza Block RAMs para almacenar los coeficientes, ya que por cada proyección se necesitan dos memorias de 59 elementos, las que son muy pequeñas como para implementarlas utilizando un recurso tan escaso en el chip. Por esto, las memorias son implementadas en LUTs, con primitivas RAM64Mx3, que corresponden a memorias distribuidas de 64 elementos de 3 bits, las cuales requieren de 4 LUTs para ser implementadas. Así, cada memoria de 59 elementos de 8 bits requiere de tres memorias RAM64Mx3, lo que equivale 12 LUTs. Si se utilizan 52 proyecciones, se necesitan de 624 LUTs para almacenar los coeficientes de una región y, como se utiliza un doble *buffer* en este módulo, se requiere de un total de 1248 LUTs, lo que equivale al reporte indicado en la Tabla 5.3. Una característica importante de este módulo, es el bajo consumo de recursos que presenta, debido al diseño planteado, con el que se eliminó el uso de Block RAMs y de multiplicadores dedicados (DSP), marcando una clara diferencia en implementaciones completamente almacenadas en el chip [25].

El módulo de clasificación por el método del vecino más cercano, utiliza una Block RAM para almacenar la base de datos de referencia. La media de los datos de entrenamiento y el *buffer* de entrada del módulo, se implementaron en memoria distribuida con las mismas primitivas de las memorias de coeficientes LDA. La media de los datos de entrenamiento tiene un número de elementos igual al número de proyecciones, 52 en este caso, donde cada elemento es de

Tabla 5.3: Uso de recursos del módulo de reconocimiento de rostros.

Sección	**Slice LUTs**	**Slice Regs**	**Slice**	**LUT Mem**	**BRAM**	**DSPs**
Módulo LBP	164	106	66	24	0.5	0
Módulo LDA	2891	2195	780	1248	0.0	0
Módulo Clasificación	1450	331	415	48	0.5	4
Controlador Memoria	227	185	88	0	1.5	2
Total utilizado	4732	2817	1349	1320	2.5	6
Disponible	53200	106400	13300	17400	140	220
Porcentaje utilizado	8.9 %	2.6 %	10.1 %	7.6 %	1.8 %	2.7 %

16 bits, para lo que son necesarias 5 memorias de 64 elementos de 3 bits, necesitando 20 LUTs para su implementación. El *buffer* de entrada del módulo también tiene 52 elementos, pero con un tamaño de 25 bits por elemento, con lo que se necesitan 7 memorias RAM64Mx3, siendo necesarias 28 LUTs, con lo que finalmente se requieren 48 LUTs para memoria distribuida. El módulo de clasificación además utiliza 4 multiplicadores dedicados o DSP, los que se pueden ver en la Figura 4.10, tres de estos multiplicadores se utilizan para realizar el cálculo de la distancia Euclidiana, el DSP restante se usó para normalizar los valores de entrada del módulo.

El consumo de potencia del sistema, mostrado en la Tabla 5.4, fue estimado con la herramienta XPower Analyzer, con una frecuencia de funcionamiento de 100 MHz y una temperatura de 25°C. El consumo estimado total del sistema es de 1.88 W, dentro del cual el procesador representa el 81.8 % del gasto energético total con 1.54 W. En contraparte, el núcleo de reconocimiento facial consume sólo 72 mW representando un 3.8 % del consumo total del sistema. El realizar parte del procesamiento en software produce un consumo energético alto, equivalente al total del procesador, porque no es posible separar las tareas que se realizan en él para obtener el consumo sólo de los bloques de procesamiento de interés. A simple vista se puede advertir la notable diferencia en consumo de potencia entre el procesamiento en software y en hardware. Aun así, el consumo reportado dista mucho del de una implementación similar en un computador de escritorio, lo que representa una de las principales ventajas de desarrollar una aplicación en un sistema embebido.

El tiempo de ejecución del sistema está compuesto por: el tiempo que tarda el detector en realizar la búsqueda de rostros sobre toda la imagen, y el tiempo de reconocimiento de los rostros detectados. La ejecución del detector es la gran limitante del sistema, pues, al estar implementado en software, no es posible realizar ningún tipo de ejecución paralela de las etapas.

Tabla 5.4: Consumo de potencia según para distintos.

Sección	**Consumo [mW]**
Procesador	1540
Clasificador facial	72
Despliegue de datos	221
Interconexión AXI	39
Interfaces RAM	3
Interfaz Detección-Reconocimiento	6
Interfaz CameraLink	1
Total	1882

Esto se ve reflejado en el tiempo obtenido en la evaluación del sistema, donde el detector tarda 1.01 S por cuadro. El tiempo necesario para la etapa de reconocimiento, corresponde al total de ciclos que necesita la arquitectura para procesar los datos. Como ésta fue implementada en hardware con una arquitectura paralela, sólo se añade una latencia inicial y un tiempo de procesamiento igual tiempo necesario enviar los datos al módulo de reconocimiento. Como se mencionó en el Capítulo 4, el tiempo mínimo sin considerar el retardo de la memoria RAM, utilizando un reloj de 100 MHz, es de 490 μS, llegando a un máximo teórico de 2500 μS para rostros de 500×500 píxeles, el tamaño máximo de rostro a detectar por el sistema. En la práctica se obtuvo un tiempo mínimo de 1037.8 μS, para rostros de hasta 100×100 píxeles. Las diferencias presentes se explican por los tiempos de trasferencia no deterministicos desde RAM a Block RAMs internas.

Según lo anterior el sistema alcanza una tasa de procesamiento de un cuadro por segundo, pudiendo reconocer hasta 936 rostros por segundo. Además según el reporte obtenido por la herramienta Time Analyzer de Xilinx, el sistema diseñado puede alcanzar un frecuencia máxima de 127.83 MHz, lo que permitiría reconocer hasta 1196 rostros por segundo,.

5.5. Discusión

En los resultados entregados puede observar la baja cantidad de cuadros por segundo que puede procesar el detector de rostros, lo que se da, principalmente, por que fue implementado en software. Un diseño en hardware dedicado permitiría aumentar notablemente la velocidad

de procesamiento, ya que, como se indicó en el Capitulo 4, es posible ejecutar en paralelo todas las etapas de procesamiento del detector, generando un *pipeline* que sólo añadirá una latencia inicial, permitiendo un trabajo en línea. Si se realiza esta implementación con un reloj de 100 MHz, el sistema podria procesar 305 cuadros por segundo a una resolución de 640×512 píxeles.

En el caso del sistema de reconocimiento de rostros, la cantidad de rostros que pueden ser procesados está limitada por las gran cantidad de transferencias de memoria que se deben realizar. Para cada imagen se deben traer desde memoria RAM 3068 coeficientes de 64 regiones de 8 bits cada uno, lo que equivale a 191.75 KBytes de memoria. En caso de disponer de suficiente memoria interna de alta velocidad, se podrían almacenar todos los coeficientes en memorias Block RAM, con lo que el número de rostros a clasificar aumentaría a un máximo de 62500 para rostros de 40×40 píxeles, con el circuito funcionando a 100 MHz.

Capítulo 6. Conclusiones y trabajo futuro

En este trabajo se presentó el prototipo de un sistema de embebido de detección y reconocimiento de rostros, utilizando una versión simplificada del algoritmo HOG con un clasificador Gentle AdaBoost para la detección, y el algoritmo LBP+LDA con clasificación por método del vecino más cercano para el reconocimiento.

Los algoritmos utilizados requieren de una etapa de entrenamiento, la cual debe repetirse ante cualquier modificación de alguna de las bases de datos. El entrenamiento del sistema representa un alto costo computacional, ya que se requiere realizar muchas operaciones de forma iterativa para el detector, y operaciones con matrices de grandes dimensiones en el caso del sistema de reconocimiento. Por esto, el prototipo sólo implementa las etapas de prueba del algoritmo. Los parámetros del entrenamiento, necesarios para el funcionamiento, son enviados al sistema a través de Ethernet y, los de menor tamaño, son almacenados durante la configuración de la lógica programable.

El sistema de detección de rostros, alimentado directamente por la salida de una cámara infrarroja, divide una imagen en celdas de 8×8 píxeles y calcula histogramas de los gradientes (HOG) para cada una de las celdas. Con 4×4 celdas se generan bloques, que son agrupados en 5×5 y son utilizados para representar distintas regiones dentro de la imagen. Cada representación de una región es clasificada con el algoritmo Gentle AdaBoost, generando una salida que indica la pertenencia a una u otra clase. Este proceso se repite para todas las posibles posiciones y tamaños dentro de la imagen.

El sistema de reconocimiento de rostros, que recibe la salida generada por el detector, divide las imágenes de rostros en 64 regiones (8×8) y aplica el operador $\text{LBP}_{(8,1)}$ generando histogramas para cada una de las regiones. La concatenación de los histogramas de todas las regiones se utiliza como descriptor de los rostros. El cual, posteriormente es proyectado a un subespacio LDA, en el que se realiza la clasificación por el método del vecino más cercano, usando distancia Euclidiana como medida de similitud.

El prototipo desarrollado en este trabajo realiza una clasificación correcta del 99.8 % de las 503 imágenes de prueba, extraídas de la base de datos LWIR UdeC, con las que fue validado en la etapa de detección. Y un 99.21 % sobre las 636 imágenes de prueba, de la base de datos UCHThermalFace, con las que fue validada la etapa de reconocimiento. El sistema fue implementado en un SoC Zynq-7000 de Xilinx, con una arquitectura mixta hardware-software, con

la que se puede realizar la detección a un cuadro por segundo, permitiendo la clasificación de hasta 936 rostros por segundo funcionando a 100 MHz. El sistema completo tiene un consumo de potencia de 1.88 W, donde la etapa de reconocimiento sólo consume un 3.8 %, correspondiente a 72 mW.

El prototipo no puede trabajar directamente en línea con una cámara infrarroja, ya que se alcanzó una tasa de procesamiento de un cuadro por segundo para el detector, lo que limita notablemente el sistema. Aun así, la etapa de reconocimiento puede clasificar hasta 936 rostros por segundo. De ser utilizado en una aplicación en línea, que recibe y detecta rostro a 30 cuadros por segundo, sería capaz de detectar hasta 32 rostros por cuadro. Sin embargo, dada la baja utilización de recursos en el diseño, menos de un 11 % en todos los tipos de recursos, es posible utilizar varias instancias del módulo en un mismo sistema, aumentando la cantidad de rostros clasificados de forma lineal. En caso de disponer de suficiente memorias Block RAMs, todos los coeficientes de entrenamiento podrían ser incluidos en el chip, aumentando enormemente el uso de recursos de memoria y la cantidad de rostros clasificados por segundo.

En el desarrollo de este trabajo se pudo comprobar el correcto funcionamiento de algoritmos de detección y reconocimiento de rostros para espectro visible, en imágenes infrarrojas. Confirmando el correcto desempeño de algoritmos basados en extracción de gradientes y texturas en el área de detección y clasificación de objetos, particularmente en rostros. Adicionalmente, se compró que las simplificaciones realizadas al algoritmo HOG no tuvieron un efecto significativamente negativo, sólo un 0.2 % menos en la tasa de acierto del detector, pero permitieron reducir notablemente el número de operaciones necesarias. Además, el hecho de utilizar imágenes cuadradas para el reconocimiento, en lugar de las rectangulares utilizadas en las pruebas en software, significó una reducción de sólo un 0.32 % en la tasa de acierto sobre la base de datos.

El sistema presenta una alta tasa de acierto, incluso cuando se prueba con imágenes obtenidas con una cámara infrarroja diferente a las usadas en el entrenamiento de detección. Esto lo convierte en una buena opción a escoger para aplicaciones donde no se requiere un procesamiento en todo instante, como sistemas seguridad o de control de acceso a lugares restringidos, lo que está dado principalmente por el muy bajo consumo de potencia del sistema embebido. Además, es posible utilizar el sistema para la detección y clasificación de otros objetos, ya que el diseño se realizó de forma genérica y parametrizable, por lo que sólo es necesario cambiar los coeficientes del entrenamiento para utilizarlo en otra aplicación.

La continuación inmediata de este trabajo pasa directamente por realizar la implementación del detector de rostros en hardware dedicado, lo que, por razones de tiempo, no pudo ser

incorporado. Sin embargo se realizaron avances en este sentido, ya que el detector de rostros se diseñó con un flujo de datos similar al que podría implementarse en hardware. Usando como base el trabajo actual, un circuito dedicado del detector de rostros, podría procesar rostros a 305 cuadros por segundo para imágenes de 640×512 píxeles a 100 Mhz, posibilitado el funcionamiento en línea con la cámara infrarroja.

Capítulo A. Bases de datos

Para evaluar los algoritmos descritos se utilizaron tres bases de datos, LWIR Udec, Equinox y UCHThermalFace. El primer conjuntos consta de imágenes frontales de sujetos con espacio al rededor de ellos y puede ser utilizada para detección y reconocimiento de rostros. Los otros dos conjuntos sólo tienen imágenes de rostros recortadas por lo que sólo fueron utilizadas para reconocimiento de rostros.

A.1. LWIR UdeC

La base de datos LWIR UdeC contiene imágenes frontales de rostro en espectro infrarrojo dentro de un entorno controlado en interior, presentando distintas inclinaciones y distintos gestos faciales. Esta base de datos fue capturada utilizando una cámara Cedip Jade UC33.

La base de datos presenta 612 imágenes de 102 sujetos, donde cada sujeto tiene 6 imágenes, 3 correspondientes a diferentes inclinaciones del rostro y 3 a diferentes gestos faciales. El conjunto original de datos presenta imágenes a una resolución de 320×240 píxeles. Cada una de éstas contiene sólo un sujeto cuyo rostro fue capturado de forma frontal, lo que se ejemplifica en la Figura A.1. En este trabajo ésta es la única base de datos utilizada para verificar los algoritmos de detección presentados anteriormente, ya que es la única disponible que presenta rostros dentro de un entorno sin recortar.

Figura A.1: Imágenes de base de datos LWIR UdeC.

Figura A.2: Todas las imágenes de un sujeto de la base de datos LWIR UdeC.

Con el fin de utilizar esta misma base de datos para reconocimiento de rostros, se recortaron los rostros de las 612 imágenes. Obteniendo imágenes de 120×120 píxeles de resolución. Todas las imágenes de un sujeto presentes en la base de datos se muestran en la Figura A.2.

A.2. Equinox

La base datos Equinox de Equinox Corporation [52] es una base de datos multiespectral de rostros. La que contiene imágenes en espectro visible, infrarrojo de onda corta (SWIR), infrarrojo de onda media (MWIR) e infrarrojo de onda larga (LWIR).

La base de datos consta de 340 sujetos, 43 imágenes por sujeto con 3 diferentes tipos de ilumiaciones, por lo que para cada espectro se tienen 43.860 imágenes con un total de 175.440 imágenes. De las 43 imágenes tomadas por sujeto, 40 de ellas fueron capturadas mientras cada sujeto decía las vocales con un tiempo de 4 segundo de duración y a una tasa de captura de 10 cuadros por segundo. Las 3 imágenes restantes fueron capturadas de forma estática y presentan gestos faciales de sonrisa, sorpresa y desaprobación (fruncir el ceño). Todas las imágenes fueron capturadas en condiciones de interior. Las imágenes LWIR tienen una resolución de 240×320 píxeles y fueron capturadas utilizando una cámara termal Indigo Merlin de FLIR [53].

De esta base de datos sólo se tiene acceso a la versión pública, la que está compuesta por 1.200 imágenes LWIR, que se dividen en 60 imágenes de 20 sujetos. En la Figura A.3 se presenta un ejemplo de la base de datos y en la Figura A.4 se muestran todas las imágenes de un sujeto.

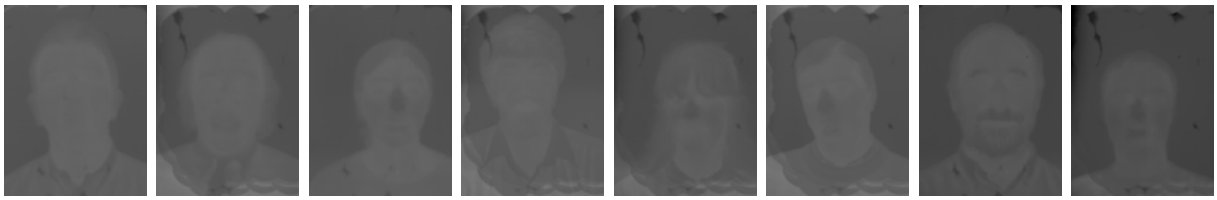


Figura A.3: Imágenes de base de datos Equinox.

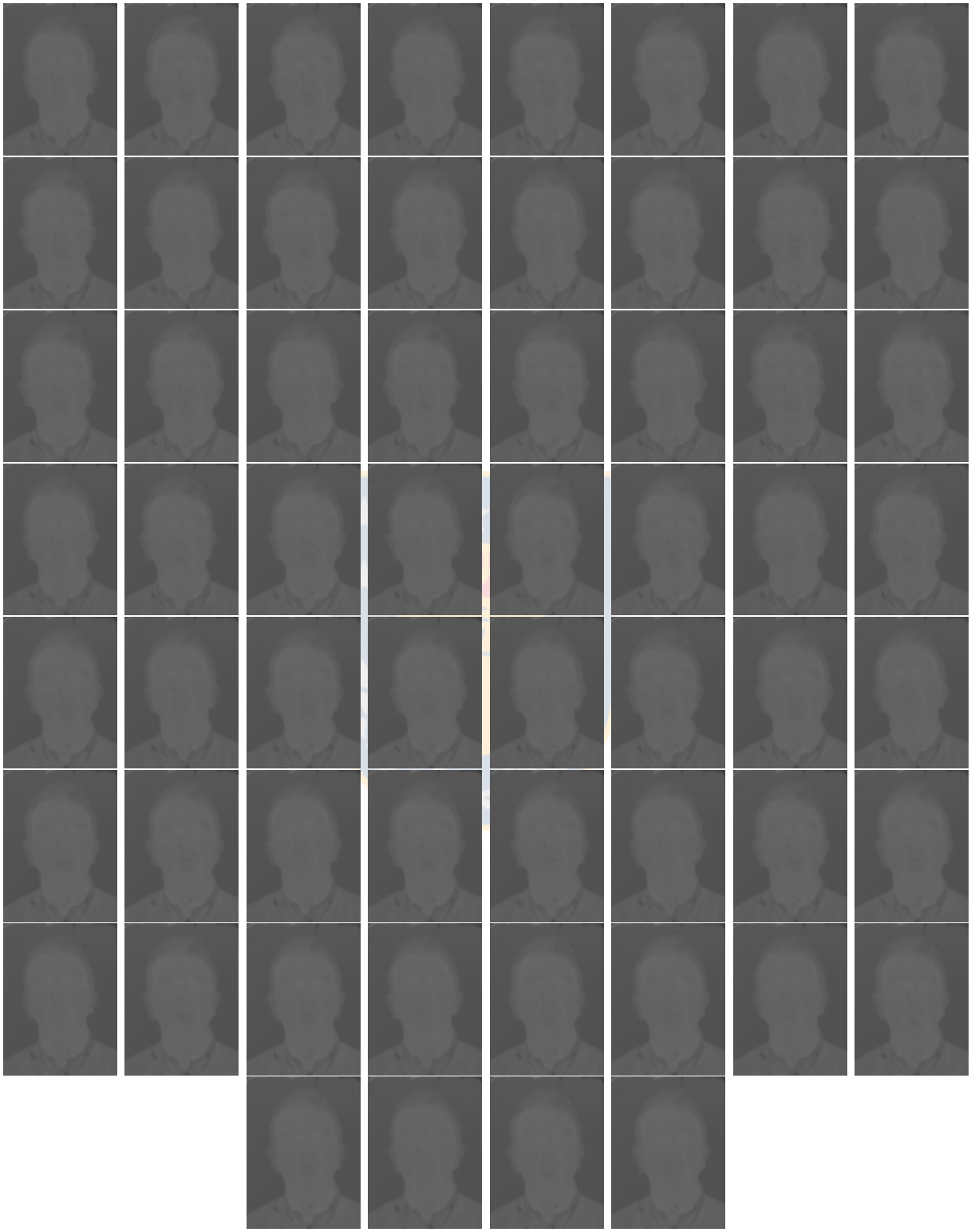


Figura A.4: Todas las imágenes de un sujeto de la base de datos Equinox.

(a) Indoor

(b) Outdoor

Figura A.5: Imágenes de base de datos UCHThermalFace.

A.3. UCHTHermalFace

La base de datos UCHThermalFace de la Universidad de Chile [54] contiene imágenes de rostros en espectro infrarrojo, las que se presentan en condiciones de interior (Indoor) y exterior (Outdoor), con diversas variaciones en cuanto a rotaciones y expresiones faciales.

La base de datos consta de 53 personas con 28 imágenes por cada una, 14 en condiciones controladas y 14 en condiciones no controladas. En cada condición 11 imágenes presentan diferentes rotaciones (R1 a R11) y 3 presentan diferentes gestos faciales (S1 a S3). Además, se presentan otros dos conjuntos con las mismas imágenes, el primero incorpora regiones de oclusión artificial en la imagen y el segundo incorpora distintos niveles de ruido. Todos los conjuntos anteriores se encuentran en tres resoluciones 81×150, 100×185 y 125×225 píxeles, con una intensidad de 8 bits por píxel. Todas las imágenes fueron adquiridas con la cámara térmica FLIR 320 TAU [55].

En este trabajo se utilizó el primer conjunto descrito, en 81×150 píxeles de resolución, el que corresponde a imágenes normales. No se consideraron los otros conjunto, pues incluyen modificaciones artificiales a las imágenes, como la oclusión que añade un sector negro a la imagen. En la Figura A.5 se presentan imágenes en ambas condiciones para 16 personas. En la Figura A.6 se muestran las 28 imágenes diferentes para un mismo sujeto, 14 en condiciones de interior y 14 en condiciones de exterior.

(a) Indoor

(b) Outdoor

Figura A.6: Todas las imágenes de un sujeto de la base de datos UCHThermalFace.

Referencias

[1] Timo Ahonen, Jiří Matas, Chu He, and Matti Pietikäinen. Rotation invariant image description with local binary pattern histogram fourier features. In Arnt-Børre Salberg, JonYngve Hardeberg, and Robert Jenssen, editors, *Image Analysis*, volume 5575 of *Lecture Notes in Computer Science*, pages 61–70. Springer Berlin Heidelberg, 2009.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[3] X. Chen, P.J. Flynn, and K.W. Bowyer. Pca-based face recognition in infrared imagery: baseline and comparative studies. In *Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003. IEEE International Workshop on*, pages 127–134, Oct 2003.

[4] Face recognition with OpenCV. In *OpenCV 2.4.9.0 documentation*, url: docs.opencv.org.

[5] Zedboard product page. In *ZedBoard*, url: zedboard.org/product/zedboard.

[6] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[7] Lun Zhang, Rufeng Chu, Shiming Xiang, Shengcai Liao, and StanZ. Li. Face detection based on multi-block LBP representation. In Seong-Whan Lee and StanZ. Li, editors, *Advances in Biometrics*, volume 4642 of *Lecture Notes in Computer Science*, pages 11–18. Springer Berlin Heidelberg, 2007.

[8] A. Hadid, M. Pietikainen, and T. Ahonen. A discriminative feature space for detecting and recognizing faces. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–797–II–804 Vol.2, June 2004.

[9] A. Hadid and M. Pietikainen. A hybrid approach to face detection under unconstrained environments. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 227–230, 2006.

[10] Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 2879–2886, Washington, DC, USA, 2012. IEEE Computer Society.

[11] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *ECCV*, 2014.

[12] Sachin Sudhakar Farfade, Mohammad J. Saberian, and Li-Jia Li. Multi-view face detection using deep convolutional neural networks. *CoRR*, abs/1502.02766, 2015.

[13] Gil Friedrich and Yehezkel Yeshurun. Seeing people in the dark: Face recognition in infrared images. In HeinrichH. Bülthoff, Christian Wallraven, Seong-Whan Lee, and TomasoA. Poggio, editors, *Biologically Motivated Computer Vision*, volume 2525 of *Lecture Notes in Computer Science*, pages 348–359. Springer Berlin Heidelberg, 2002.

[14] Chieh-Li Chen and Bo-Lin Jian. Infrared thermal facial image sequence registration analysis and verification. *Infrared Physics & Technology*, 69:1 – 6, 2015.

[15] Adel Elmaghraby Kristopher Reese, Yufeng Zheng. A comparison of face detection algorithms in visible and thermal spectrums. In *2012 International Conference on Advances in Computer Science and Application*, pages 49–53, 2012.

[16] S.J. Krotosky, S.Y. Cheng, and M.M. Trivedi. Face detection and head tracking using stereo and thermal infrared cameras for "smart.ªirbags: a comparative analysis. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 17–22, Oct 2004.

[17] Marcelo Vergara, Alejandro Wolf, and Miguel Figueroa. A texture-based architecture for face detection in IR images on an FPGA. *Proc. SPIE*, 9249:92490L–92490L–12, 2014.

[18] Junguk Cho, Shahnam Mirzaei, Jason Oberg, and Ryan Kastner. Fpga-based face detection system using haar classifiers. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '09, pages 103–112, New York, NY, USA, 2009. ACM.

[19] Changjian Gao and Shih-Lien Lu. Novel fpga based haar classifier face detection algorithm acceleration. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 373–378, Sept 2008.

[20] L. Acasandrei, A. Barriga, M. Quintero, and A. Ruiz. Face identification implementation in a standalone embedded system. In *Industrial Electronics (ISIE), 2014 IEEE 23rd International Symposium on*, pages 1938–1943, June 2014.

[21] L. Acasandrei and A. Barriga. Embedded face detection application based on local binary patterns. In *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS), 2014 IEEE Intl Conf on*, pages 641–644, Aug 2014.

[22] O. Bilaniuk, E. Fazl-Ersi, R. Laganiere, C. Xu, D. Laroche, and C. Moulder. Fast LBP face detection on low-power simd architectures. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 630–636, June 2014.

[23] M.A Turk and AP. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591, Jun 1991.

[24] P.N. Belhumeur, J.P. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, Jul 1997.

[25] Javier E. Soto and Miguel Figueroa. An embedded face-classification system for infrared images on an FPGA. *Proc. SPIE*, 9249:92490K–92490K–12, 2014.

[26] Daniel Maturana, Domingo Mery, and Álvaro Soto. Face recognition with local binary patterns, spatial pyramid histograms and naive bayes nearest neighbor classification. In *Proceedings of the 2009 International*

Conference of the Chilean Computer Science Society, SCCC '09, pages 125–132, Washington, DC, USA, 2009. IEEE Computer Society.

[27] T. Ahonen, A Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2037–2041, Dec 2006.

[28] Shengcai Liao, Xiangxin Zhu, Zhen Lei, Lun Zhang, and StanZ. Li. Learning multi-scale block local binary patterns for face recognition. In Seong-Whan Lee and StanZ. Li, editors, *Advances in Biometrics*, volume 4642 of *Lecture Notes in Computer Science*, pages 828–837. Springer Berlin Heidelberg, 2007.

[29] Y. Taigman, Ming Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1701–1708, June 2014.

[30] X. Chen, P.J. Flynn, and K.W. Bowyer. PCA-based face recognition in infrared imagery: baseline and comparative studies. In *Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003. IEEE International Workshop on*, pages 127–134, Oct 2003.

[31] Moulay A. Akhloufi and Abdelhakim Bendada. Infrared face recognition using texture descriptors. *Proc. SPIE*, 7661:766109–766109–10, 2010.

[32] Heydi Méndez, Cesar San Martín, Josef Kittler, Yenisel Plasencia, and Edel García-Reyes. Face recognition with LWIR imagery using local binary patterns. In *Proceedings of the Third International Conference on Advances in Biometrics*, ICB '09, pages 327–336, Berlin, Heidelberg, 2009. Springer-Verlag.

[33] M. Saquib Sarfraz and Rainer Stiefelhagen. Deep perceptual mapping for thermal to visible face recognition. *CoRR*, abs/1507.02879, 2015.

[34] Pablo Pizarro and Miguel Figueroa. Subspace-based face recognition on an FPGA. In Lazaros Iliadis and Chrisina Jayne, editors, *Engineering Applications of Neural Networks*, volume 363 of *IFIP Advances in Information and Communication Technology*, pages 84–89. Springer Berlin Heidelberg, 2011.

[35] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51 – 59, 1996.

[36] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, Jul 2002.

[37] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.

[38] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto. Architectural study of hog feature extraction processor for real-time object detection. In *2012 IEEE Workshop on Signal Processing Systems*, pages 197–202, Oct 2012.

[39] Yoav Freund and Robert E. Schapire. Experiments with a New Boosting Algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[40] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Ann. Statist.*, 28(2):337–407, 04 2000.

[41] Bo Wu, Haizhou Ai, Chang Huang, and Shihong Lao. Fast rotation invariant multi-view face detection based on real adaboost. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 79–84, May 2004.

[42] V. Vapnik and A. Lerner. Pattern Recognition using Generalized Portrait Method. *Automation and Remote Control*, 24, 1963.

[43] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[44] P.N. Belhumeur, J.P. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, Jul 1997.

[45] M.A Turk and AP. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591, Jun 1991.

[46] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[47] Carles M. Cuadras. Distancias estadísticas. In *Estadistica Española*, volume 30, pages 295–378, 1989.

[48] Ofir Pele and Michael Werman. *The Quadratic-Chi Histogram Distance Family*, pages 749–762. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[49] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. November 2014.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[51] Qt home page. In *Qt*, url: qt.io.

[52] LawrenceB. Wolff, DiegoA. Socolinsky, and ChristopherK. Eveland. Face recognition in the thermal infrared. In Bir Bhanu and Ioannis Pavlidis, editors, *Computer Vision Beyond the Visible Spectrum*, Advances in Pattern Recognition, pages 167–191. Springer London, 2005.

[53] FLIR. *Cámara Térmica FLIR Indigo Merlin*, url: www.flir.com/legacy/view/?id=51541.

[54] Gabriel Hermosilla, Javier Ruiz-del-Solar, Rodrigo Verschae, and Mauricio Correa. A comparative study of thermal face recognition methods in unconstrained environments. *Pattern Recognition*, 45(0):2445 – 2459, Jul 2012.

[55] FLIR. *Cámara Térmica FLIR TAU 320*, url: www.flir.com/cvs/cores/uncooled/products/tau.