Reconocimiento de expresiones faciales en imágenes digitales.

por

Ernesto Cabrera Solis

Tesis sometida como requisito parcial para
obtener el grado de

Maestro en Computación Óptica

en la

Universidad Politécnica de Tulancingo
4 de octubre de 2016

Tulancingo de Bravo, Hidalgo

Supervisada por:

Dra. Carina Toxqui Quitl

Índice general

1. Introducción **1**

1.1. Introducción 1
1.2. Planteamiento del problema 2
1.3. Estado del arte 2
1.4. Objetivos 4
1.5. Justificación 4
1.6. Propuesta de solución 5
1.7. Aportaciones de la tesis 6
1.8. Trabajos derivados de la tesis 7
1.9. Organizacion de la tesis 8

2. Extracción de caracteristicas **11**

2.1. Introducción 11
2.2. Momentos de una imagen 13
2.3. Momentos Geométricos 13
2.3.1. Momentos Invariantes a la rotación, escala y posición 14
2.4. Momentos ortogonales en un circulo 16
2.4.1. Momentos de Zernike 18
2.4.2. Momentos de Jacobi-Fourier 20
2.4.3. Reconstrucción de Imágenes 23
2.4.4. NIRE 23
2.5. Detección de Esquinas 27
2.5.1. Métodos de Harris, Shi y Tomasi 27
2.6. Entrenamiento y clasificación 30
2.7. Proceso de clasificación supervisada 31
2.8. Clasificadores supervisados 31
2.8.1. Vecinos más cercanos (KNN) 32
2.8.2. Clasificador AdaBoost 33
2.8.3. Clasificador Lineal Discriminante 35

3. Técnica propuesta y bases de datos **41**

3.1. Introducción 41
3.2. The Japanese Female Facial Expression (JAFFE) Database 42
3.3. Robot Nao 43
3.4. Técnica propuesta para el reconocimiento de emociones en imágenes digitales . 46

3.4.1. Detección del rostro 46
3.4.2. Procesado de la imagen 48
3.4.3. Extracción de Características 49
3.4.4. Entrenamiento y clasificación 50
3.5. Software en Matlab 51
3.5.1. Funciones de Matlab 54

4. Resultados y conclusiones **63**
4.1. Introducción 63
4.2. Resultados con descriptores basados en momentos de Zernike 63
4.2.1. Clasificación de 3 emociones 63
4.2.2. Clasificación de 4 emociones 66
4.2.3. Clasificación de 5 emociones 68
4.2.4. Clasificación de 6 emociones 70
4.3. Resultados con descriptores basados en momentos de Jacobi-Fourier 72
4.3.1. Clasificación de 3 emociones 72
4.3.2. Clasificación de 4 emociones 74
4.3.3. Clasificación de 5 emociones 77
4.3.4. Clasificación de 6 emociones 79
4.4. Análisis general de los resultados 81
4.5. Conclusiones 82

Capítulo 1

Introducción

1.1. Introducción

Las emociones humanas pueden expresarse de muchas formas, una de estas son las expresiones faciales. Casi todas las personas son capaces de detectar en la mayoría de los casos las emociones presentes en un rostro humano, y esto aunque la apariencia de estas expresiones faciales pueden variar entre cada individuo. La expresión facial juega un papel muy importante dentro de las emociones humanas, las expresiones faciales básicas pueden decirnos si una persona está feliz, triste, tiene miedo, está enojada o disgustada.

Uno de los problemas en este tema es cuando se necesita que un sistema computacional sea capaz de determinar la emoción de un humano a través de la voz, imágenes o video. El reconocimiento de las emociones puede ser abordado de diferentes maneras en las cuales se pueden tomar en cuentas las distintas formas de obtener los datos, algunas de las formas más comunes son:

- Lectura de señales biométricas.
- Expresiones faciales a través de imágenes o videos.
- Señal de la voz.

Una de las técnicas más estudiadas en el reconocimiento de emociones, es utilizando imágenes digitales, es decir tratar de reconocer a través de algún algoritmo o conjunto de algoritmos, la emoción presente en el rostro de una persona capturado en una imagen digital, el principal problema de esto es que las expresiones faciales pueden variar entre cada individuo.

El estudio del reconocimiento de emociones a través de sistemas computacionales se empezó a estudiar en años recientes debido a la necesidad de que estos sistemas puedan entender lo que el usuario siente y dice cuando interactúa con ellos. Las aplicaciones de este tema pueden estar enfocadas en la interacción humano-computadora, en el área de marketing, área de la psicología y psiquiatría, en la detección de mentiras, entre otras aplicaciones.

1.2. Planteamiento del problema

Pensar en una computadora capaz de comunicarse a través de lenguaje natural, de entender a los humanos y de sentir empatía, sonaba a una novela de ciencia ficción. Por esta razón, las emociones no se incluían en la interacción humano-computadora, lo que derivó en sistemas de cómputo: fríos, frustrantes y con muy poca capacidad de comunicación con los usuarios. En respuesta a la demanda de sistemas adaptables a las necesidades humanas, capaces de entender lo que un usuario está diciendo y sintiendo, y con base en la evidencia científica, surgió el interés en entender las emociones para tratar de integrarlas en los sistemas de cómputo. En las primeras investigaciones sobre procesamiento computacional de emociones, el objetivo era razonar con las emociones de los usuarios y responder a estas; sin embargo, no se pensaba aún que las computadoras deberían tener emociones [1].

En general el problema se puede describir como la necesidad de desarrollar algoritmos y/o técnicas que permitan a los sistemas computaciones poder entender las emociones que el usuario presenta al interactuar con ellos. Esto con el objetivo de mejorar la interacción humano-computadora, y poderle ofrecer al usuario una experiencia de uso de acuerdo a la emoción que presenta. Aunque las aplicaciones del reconocimiento de las emociones pueden ser enfocadas a otras muchas áreas, una de las principales es el marketing, en el cual muchas empresas necesitan saber cómo reacciona el público ante algún anuncio publicitario, en este sentido algunas empresas como Affectiva [2] están desarrollando herramientas basadas en el reconocimiento de emociones faciales, con la finalidad de detectar la influencia que puede causar la publicidad en un conjunto de individuos.

1.3. Estado del arte

En 1997, R. Picard en el laboratorio de medios del MIT acuño la frase *computación afectiva* para referirse a la habilidad de una computadora de reconocer, expresar y posiblemente tener emociones . El cómputo afectivo es un campo novedoso de investigación que estudia y utiliza las emociones del ser humano para el diseño de sistemas computacionales. Esta área incluye el desarrollo de algoritmos para reconocer de manera multi-modal gestos de cuerpo y cara, actividad visual y auditiva, estados de ánimo, personalidad y actitud. Asimismo, utiliza diversas medidas fisiológicas para determinar estados anímicos [1].

A través del tiempo mucha gente interesada en el tema de que los sistemas puedan reconocer emociones humanas, han desarrollado diferentes algoritmos y/o técnicas para resolver este problema. Se presentan algunos trabajos anteriores relacionados con el tema del reconocimiento de emociones:

Desarrollo de un avatar animado con expresión de emociones básicas: Con el objetivo de transmitir información a través de las interfaces de usuario en el contexto de las tecnologías de información, surge la inquietud de crear una aplicación multiplataforma que permita modelar en un avatar las expresiones faciales. En http://chopis.com/ se presenta la aplicación Web Chopiface, diseñada para animar un avatar, el cual puede expresar emociones básicas. Se po-

ne a prueba el reconocimiento de las expresiones de ira, miedo, alegría, tristeza y sorpresa en personajes 3D creados con el uso de los estándares MPEG-4 y FACS (Facial Action Coding System) de Ekman, y una estructura de huesos para proporcionar la libertad en la deformación del rostro, permitiendo así poder mover cejas, ojos y boca de forma independiente. El desarrollo final fue evaluado de acuerdo al reconocimiento de las expresiones faciales y se llevó a cabo con un grupo de 96 personas. La aportación de esta aplicación es contribuir como una herramienta educativa gratuita para el entrenamiento del reconocimiento de emociones y mostrar la importancia e influencia de las intensidades y técnicas en el modelado de un avatar [3].

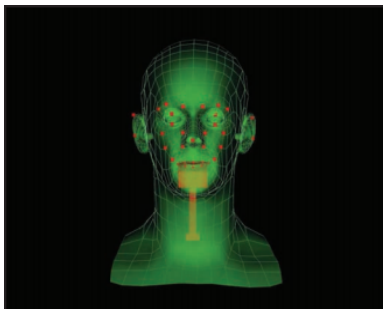


Figura 1.1: **Malla y huesos de uno de los avatares propuestos [3].**

Reconocimiento y procesamiento de emociones en ambientes inteligentes de aprendizaje: En un ambiente tradicional de aprendizaje, el tutor detecta estados emocionales de los estudiantes y adapta la presentación de los materiales con el fin de maximizar el aprendizaje. Conocer el estado emocional del individuo nos brinda información relevante acerca de su estado psicológico y esto le otorga a un sistema de software capacidad de decisión sobre cómo debe responder ante él. El reconocimiento automático de las emociones puede mejorar el desempeño, usabilidad y, en general, la calidad de interacción hombre-computadora, la productividad del aprendizaje de los estudiantes y la atención de un sistema a los usuarios [4].

Reconocimiento de emociones empleando el procesamiento digital de la señal de la voz: Se presenta en este trabajo una metodología para la caracterización de la señal de voz aplicada en el reconocimiento de estados emocionales. Los diferentes estados emocionales de un hablante producen cambios fisiológicos en el aparato fonador, lo que se ve reflejado en la variación de algunos parámetros de la voz. Las técnicas de procesamiento empleadas son: transformadas tiempo-frecuencia, análisis de predicción lineal y raw data [5].

Hablando específicamente de trabajos relacionados con el reconocimiento de emociones utilizando imágenes digitales: Zhang [6] reporto sus experimentos realizados para el reconocimiento de emociones usando dos tipos de características principales, las posiciones geométricas y la transformada Wavelet de Gabor. Por otra parte Panic et.al [7] propuso un sistema automático para el análisis de los gestos faciales a través de imágenes digitales estáticas.

1.4. Objetivos

Objetivo general

Proponer y desarrollar una técnica la cual permita determinar el tipo de emoción expresada en el rostro de un ser humano, esto a partir de una imagen digital.

Objetivos principales

- Determinar el tipo de emociones con las cuales se trabajara en el proyecto.
- Desarrollar e implementar algoritmos que permitan la extracción de características en imágenes digitales con el fin de poder diferenciar entre emociones diferentes.
- Desarrollar e implementar algoritmos para el reconocimiento y clasificación de las emociones.
- Diseñar y desarrollar un software que permita conjuntar los algoritmos de extracción de características y clasificación, con el objetivo de reconocer emociones en imágenes digitales.
- Validar la técnica que se propone como objetivo general, usando para esto el software desarrollado.

1.5. Justificación

El interés en los últimos años de que los sistemas puedan comprender lo que el humano está sintiendo a la hora de interactuar con ellos ha aumentado, ya que son muchas las aplicaciones en el que el reconocimiento de las emociones puede ayudar, algunos ejemplos son:

- Mejorar la interacción humano-computadora.
- Ayuda en el área psicología o psiquiátrica.
- Ayuda en la detección de mentiras.
- Área de marketing.

- Área de ambientes inteligentes.

- Entre otras.

1.6. Propuesta de solución

Esta tesis propone una técnica para el reconocimiento de emociones en rostros presentes en imágenes digitales, para esto se propone el uso de algoritmos de procesamiento de imágenes como son la binarización, la detección de bordes. Así mismo el uso de algoritmos de extracción de características con los momentos de Zernike y los momentos de Jacobi-Fourier, así como algoritmos de clasificación tales como el de vecinos más cercanos (KNN), AdaBoost y clasificador discriminante.

Para la evaluación de la técnica propuesta se debe diseñar y desarrollar un software que permita al usuario conjuntar los algoritmos mencionados, de procesamiento, extracción de características y clasificación. Esto en una herramienta que sea capaz de identificar la emoción presente en el rostro de una persona el cual fue capturado en una imagen digital. En la Figura 1.3 se puede observar el diagrama a bloques de la técnica propuesta.

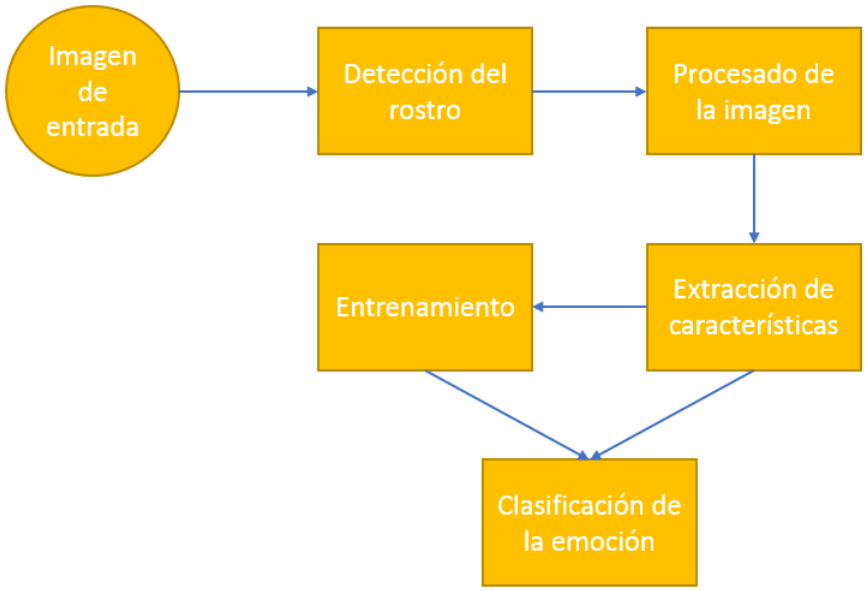


Figura 1.2: **Diagrama general de bloques con los pasos de la técnica propuesta.**

Cabe mencionar que se pretende que la técnica sea capaz de clasificar la emoción en tiempo real, es decir que no solo debe importar el resultado, si no también el tiempo en que tarda en obtener ese mismo resultado. Así que en general, se necesita que el sistema sea capaz de clasificar correctamente una emoción en el menor tiempo posible.

1.7. Aportaciones de la tesis

1. Propuesta de una técnica para el reconocimiento de emociones en imágenes digitales, basada en el procesamiento de la imagen con algoritmos de mejoras de contraste, descriptores con momentos de Zernike y Jacobi-Fourier y algunos clasificadores como el AdaBoost.

2. Desarrollo de un software capaz de reconocer emociones en una o N imágenes digitales, trabajando en tiempo real.

3. Interfaz entre el software creado y un robot NAO, para que desde el software pueda enviar una instrucción al robot, el cual tomara una imagen con alguna de las cámaras con las que cuenta y la enviara al software para la clasificación de la misma.

4. Dentro de una estancia de investigación realizada en la Universidad Politécnica de Valencia en España se desarrolló una web demo donde se utiliza el reconocimiento de emociones como un factor en la detección de la depresión. En conjunto con algunas técnicas tradicionales para la detección de la depresión como lo son los cuestionarios y/o las entrevistas personalizadas se pretende que el reconocimiento de emociones nos sirva como un factor para determinar si el paciente presenta algún tipo de depresión. Para esto en una primera instancia se desarrolló una herramienta que permite grabar a un paciente mediante una webcam mientras este contesta el cuestionario BECK II [8], el cual es una herramienta ya validada para la detección de la depresión. Cada cierto tiempo (3 segundos) se toma una imagen del paciente, esa imagen es procesada para determinar la emoción presente en ella, todos los datos se van guardando en una base de datos para ir generando los resultados de la prueba. Al final los resultados pueden ser visualizados mediante otra pantalla, donde primeramente se muestra la gráfica de la evolución de las emociones, como la que se muestra en la Figura 1.3

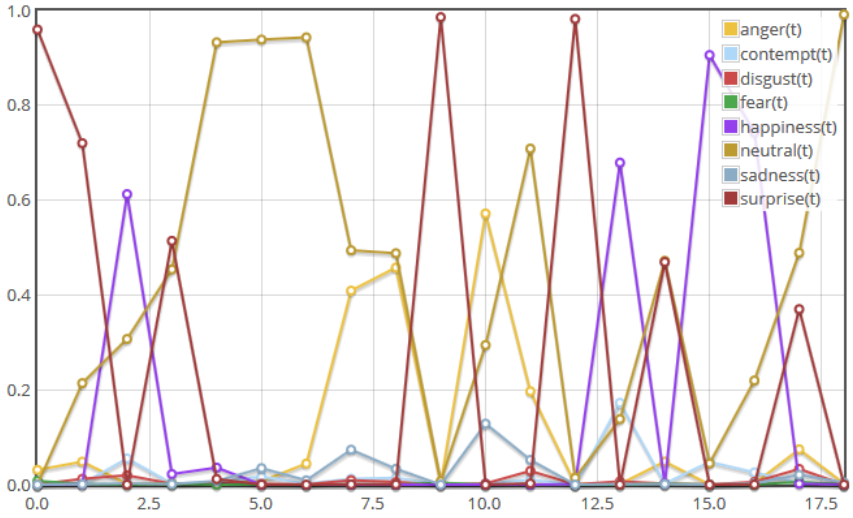
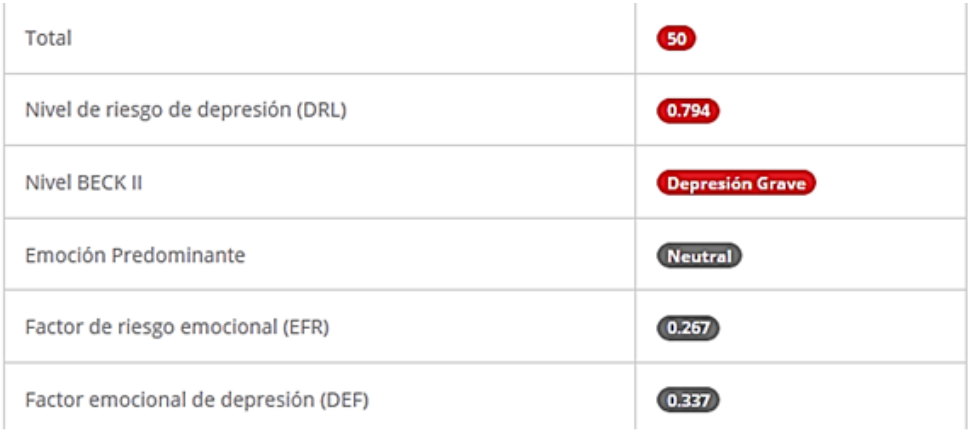


Figura 1.3: **Ejemplo de una gráfica con los resultados del reconocimiento de las emociones en un cierto periodo de tiempo.**

La informacion acerca de los datos proporcionados por las emociones asi como los datos generados mediante el cuestionario BECK II, se muestran en una tabla como la de la Figura 1.4, estos datos deben servir para generar un primer diagnostico de depresión.



Total	50
Nivel de riesgo de depresión (DRL)	0.794
Nivel BECK II	Depresión Grave
Emoción Predominante	Neutral
Factor de riesgo emocional (EFR)	0.267
Factor emocional de depresión (DEF)	0.337

Figura 1.4: **Tabla de resultados de la prueba de reconocimiento de emociones en conjunto con el BECK II.**

La validación de esta herramienta al momento de escribir esto se encuentra en progreso. Se está aplicando la prueba a personas en el ambiente universitario (personal académico, personal administrativo) de la Universidad Politécnica de Valencia, al concluir las pruebas se analizaran los resultados obtenidos, con el fin de validar la herramienta y saber que funciona para el fin que se propuso.

1.8. Trabajos derivados de la tesis

1. Poster presentado en el 8vo encuentro de investigación de la Universidad Politécnica de Tulancingo en el 2015 el cual llevó por título *Reconocimiento de emociones para generar empatía con un robot humanoide*.

2. Poster presentado en el 4º Workshop on Innovation on Information and Communication Technologies (WIICT'16) realizado en el Instituto ITACA de la Universidad Politécnica de Valencia, el cual llevó por título *A facial expression recognition tool to aid in depression assessment*.

3. Articulo aceptado para publicación en memoria de congreso en el 4º Workshop on Innovation on Information and Communication Technologies (WIICT'16) realizado en el Instituto ITACA de la Universidad Politécnica de Valencia, el cual llevó por título *A facial expression recognition tool to aid in depression assessment*.

4. Articulo aceptado para publicación en memoria de congreso en el XXXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica (CASEIB 2016), organizado por el

Centro de Investigación e Innovación en Bioingeniería (Ci2B) de la Universidad Politécnica de Valencia (UPV) y la Sociedad Española de Ingeniería Biomédica (SEIB) el cual lleva por título *Aplicación de la Metodología E-UNIHEALTH a un contexto universitario específico*.

1.9. Organizacion de la tesis

Capitulo 1: En este capítulo se presenta una breve introducción al trabajo realizado en esta tesis, los motivos por el cual se realizó la investigación. Se presentan la hipótesis inicial de la investigación, así como los objetivos que se pretenden alcanzar en la investigación.

Capitulo 2: Se presenta la teoría de los diferentes algoritmos o técnicas usadas para la solución del problema planteado. Teoría matemática de los momentos de Zernike, momentos de Jacobi-Fourier. Así mismo el funcionamiento de los clasificadores KNN, AdaBoost y discriminante.

Capitulo 3: En este capítulo se explica con más detalle la propuesta de solución, se describen las técnicas utilizadas, así como su implementación. Se presenta las bases de datos utilizadas para probar el rendimiento de la técnica propuesta. Se hace la descripción de un software desarrollado como parte de la evaluación de la solución propuesta.

Capitulo 4: Este capítulo está dedicado a los resultados obtenidos al realizar pruebas con el software presentado en el capítulo 3. Se dan las conclusiones de la investigación realizada, así como el trabajo a futuro.

Bibliografía

[1] Yasmín Hernández, Laura Cruz, *Sapiens Piensa*, Komputer Sapiens, Año Volumen II, mayo-agosto (2013).

[2] Affectiva TECHNOLOGY Disponible en: http://www.affectiva.com/technology/

[3] Paula N. Medina, Oleg Starostenko, Octavio Ruiz-Castillo, *Desarrollo de un avatar animado con expresión de emociones básicas*, Komputer Sapiens, Año V Volumen II, mayo-agosto (2013)

[4] María Lucia Barrón Estrada, Ramón Zatarain Cabada, María Yasmíin Hernández Pérez, *Reconocimiento y procesamiento de emociones en ambientes inteligentes de aprendizaje*, Komputer Sapiens, Año V Volumen II, mayo-agosto (2013)

[5] Mauricio Moralez Perez, Julian David Echeverry, *Reconocimiento de emociones empleando el procesamiento digital de la señal de la voz*, Scientia et Technica Año XIII, No 37, Diciembre (2007).

[6] Zhang, Z. *Feature-based facial expression recognition: Sensitive analysis and experiments with a multilayer perceptron*, International Journal of Patter Recognition and Artificial Intelligence 13(6) ,893-911 (1990)

[7] Pantic, M. Rothkrantz, L. J. M, *Facial action recognition for facial expression analysis from static face images*, IEEE Transactions on systems 34(3), (2004)

[8] Beck, A. T., Steer, R. A., y Brown, G. K. Manual. *BDI-II. Inventario de Depresión de Beck-II (Adaptación española: Sanz, J., y Vázquez, C.)*, Madrid, Pearson Educación. (2011).

Capítulo 2

Extracción de caracteristicas

2.1. Introducción

La extracción de características de regiones individuales en una imagen, es una etapa muy importante, ya que de esta dependerá el reconocimiento de un conjunto de objetos, personas, emociones, etc. y en general de cualquier patrón. Esta etapa consiste en extraer la información más relevante de los objetos en la imagen, aunque determinar cuál de la información presente es relevante puede ser un problema. Los descriptores son una herramienta que nos permite extraer características de los objetos presentes en dichas imágenes, algunas de las características pueden ser color, texturas, forma, distancias, entre otras. Existen diferentes técnicas para la extracción de características, algunos de estos pueden verse en el Tabla 2.1.

Tabla 2.1: Algunas técnicas de extracción de características.

Función de representación de formas	**Interrelación de características espaciales**
Distancia al centroide Área de la función	Código Cadena
Transformaciones de forma en otros dominios	**Detectores de esquinas**
Descriptores de Fourier Transformada Wavelet	Algoritmo de Movarec Método de Harris Método de Shi y Tomasi
Detectores de puntos interesantes	**Momentos**
SURF Points BRISK Points	Momentos geométricos Momentos invariantes Momentos de Zernike Momentos de Jacobi-Fourier

Después del proceso de extracción y descripción de características, cada objeto queda representado por un conjunto de descriptores, este conjunto es denominado patrón. Principalmente los patrones se pueden dividir en dos representaciones, patrones vectoriales y patrones estructurados [1][2].

Patrones Vectoriales [2]: representan características cuantitativas de la imagen y se suelen representar como:

$$\xi = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \dots \\ \xi_n \end{bmatrix}, \tag{2.1.1}$$

donde cada componente ξ_i es la *i-ésima* característica y n es el número de características.

Patrones estructurados [2]: codifican relaciones entre componentes o características del objeto. Un ejemplo de esto es el reconocimiento de huellas dactilares, basándose en relaciones entre una serie de rasgos denominados $minucias$ las cuales describen las propiedades de las huellas dactilares. Dentro de estos tipos de patrones se encuentras las cadenas y los árboles. En el caso de las cadenas no es más que una cadena de simbolos que representan una estructura. Estas representaciones generan adecuadamente patrones de objetos cuya estructura se basa en conexiones sencillas, normalmente asociada a formas de contornos, como se puede observar en la Figura 2.1

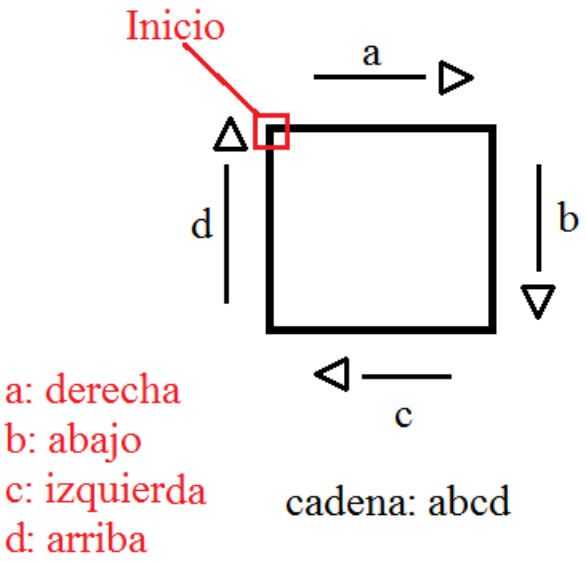


Figura 2.1: **Codificación del contorno de un cuadrado.**

En este capítulo se describen algunas técnicas de extracción de características: momentos y detectores de esquinas, dentro de los momentos específicamente se analizan los momentos de Zernike [6] y de Jacobi-Fourier [7], se evalúa su eficiencia a través de la reconstrucción de imágenes a partir de sus valores. Los detectores de esquinas presentados son el método de Harris [12] y el método de Shi y Tomasi [14] los cuales son muy similares y se usan en el análisis de imágenes y reconocimiento de objetos, entre otros.

2.2. Momentos de una imagen

Los momentos son cantidades escalares que se usan para describir una función y capturar sus características más importantes. Los momentos MG_{pq} de una función $f(x, y)$ donde p, q son enteros positivos y $r = p + q$ es llamado el orden del momento, están definidos como [3]:

$$MG_{pq} = \iint_D P_{pq}(x, y) f(x, y) dxdy, \tag{2.2.1}$$

donde $P_{00}(x, y), P_{10}(x, y), ..., P_{kj}(x, y), ...$ son funciones de base polinomial definidas en D, y dependiendo de la base polinomial usada, se conocen varios sistemas de momentos.

Como una imagen digital es una función discontinua, no podemos aplicar la ecuación (2.2.1) directamente, por lo que debemos de hacer una transformación discreta:

$$MG_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} P_{pq}(x, y) f(x, y), \tag{2.2.2}$$

donde M y N representan el tamaño de la imagen.

2.3. Momentos Geométricos

La opción más común y estándar, es la base de potencia $P_{pq}(x, y) = x^p y^q$, con la que se forman los momentos geométricos [3]:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dxdy. \tag{2.3.1}$$

Los momentos geométricos de orden bajo tienen un significado intuitivo, en una imagen binaria el momento geométrico m_{00} representa el área de un objeto.

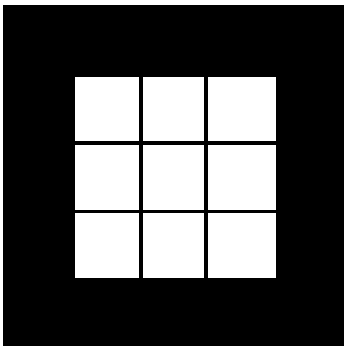


Figura 2.2: **Imagen binaria de 5×5 pixeles, cada cuadro representa un pixel.**

En la Figura 2.2 se presenta una imagen binaria de un cuadrado, si consideramos que cada cuadro interno representa un 1 pixel y sabemos que el área de un cuadrado se calcula LxL donde L es la medida de cada lado del cuadrado, entonces $L = 3$ pixeles y por lo tanto el área sería 3×3 pixeles. Llevando la ecuación (2.3.1) a una transformación discreta como en ecuación (2.2.2) tenemos que:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x, y), \tag{2.3.2}$$

igualando $p = 0, q = 0$, el m_{00} está dado como

$$m_{00} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y), \tag{2.3.3}$$

donde $x^0 = 1, y^0 = 1$, entonces, como observamos en la ecuación (2.3.3) para calcular el m_{00} *(área de la imagen)*, se suman los valores de los pixeles *(en una imagen binaria el negro representa 0 y el blanco 1)*. Si contamos el número de cuadros blancos en la imagen veremos que son 9, con esto comprobamos que el momento m_{00} representa el área en una imagen binaria.

Los momentos m_{10} y m_{01} junto con el momento de orden cero m_{00} determinan el centro de gravedad de los objetos en una imagen, a través de

$$m_{10} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x f(x, y), \tag{2.3.4}$$

$$m_{01} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} y f(x, y), \tag{2.3.5}$$

obtenemos:

$$x_c = \frac{m_{10}}{m_{00}}, \qquad y_c = \frac{m_{01}}{m_{00}}, \tag{2.3.6}$$

de donde, el centroide tiene coordenadas (x_c, y_c).

2.3.1. Momentos Invariantes a la rotación, escala y posición

El reconocimiento de objetos y patrones que fueron deformados de varias maneras han sido el objetivo de muchas investigaciones recientes. Para resolver este problema básicamente existen tres acercamientos: fuerza bruta, normalización de la imagen y los descriptores invariantes [3]. Si se usa la fuerza bruta, se deberían analizar todas las posibles combinaciones de deformaciones que podría tener la imagen, esto llevaría demasiado tiempo de cómputo. En el caso de la normalización requiere resolver problemas inversos.

Por ejemplo si tenemos una imagen con emborronamiento $normalizarla$ significaría quitar el emborronamiento, lo que nos llevaría a otros problemas tal vez un tanto complejos. Los descriptores invariantes aparecen como el acercamiento que más promete para la resolución del problema y su uso ha sido extensivo, la idea básica de esto es poder describir un objeto con un conjunto de valores $invariantes$, los cuales deben ser insensibles a diferentes transformaciones y a la vez contar con el poder de discriminación entre diferentes clases.

Hu [4] introdujo los momentos invariantes a la rotación y escalamiento a partir de los momentos geométricos. Los momentos geométricos pueden hacerse invariantes a la traslación, para ello basta con referirlos al centro de gravedad de los objetos utilizando las ecuaciones (2.3.6). Estos momentos se conocen como momentos centrales y tienen la siguiente forma:

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - x_c)^p (y - y_c)^q f(x, y). \tag{2.3.7}$$

Para conseguir una invariancia a la escala, se definen los momentos centrales normalizados:

$$\eta_{pq} = \frac{\mu^{pq}}{\mu_{00}^{\gamma}}, \tag{2.3.8}$$

donde:

$$\gamma = \frac{p+2}{2} + 1. \tag{2.3.9}$$

De los momentos centrales normalizados de segundo y tercer orden se derivan los siete momentos de Hu invariantes a traslaciones rotaciones y cambios de escala:

$$\begin{aligned}
\phi_1 &= \eta_{20} + \eta_{02} \\
\phi_2 &= (\eta_{20} - \eta_{02})^2 + 4(\eta_{11})^2 \\
\phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\phi_4 &= (\eta_{30} - \eta_{12})^2 + (\eta_{21} - \eta_{03})^2 \\
\phi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] \\
&+ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
\phi_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
&+ 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] \\
&- (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{2.3.10}$$

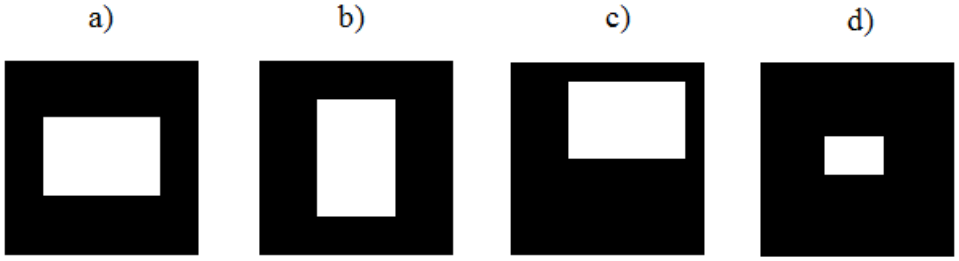


Figura 2.3: **a) Imagen original, b) Imagen rotada 90°, c) Imagen trasladada, d)Imagen escalada**

Se calcularon los 7 momentos invariantes de Hu para la imágenes de la Figura 2.3, cada una de estas es una imagen binaria de tamaño 10×10 pixeles, los resultados obtenidos son los siguientes:

Momentos	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7
Imagen original	0,174	$4{,}823 \times 10^{-3}$	0	0	0	0	0
Imagen rotada 90°	0,174	$4{,}823 \times 10^{-3}$	0	0	0	0	0
Imagen trasladada	0,174	$4{,}823 \times 10^{-3}$	0	0	0	0	0
Imagen escalada	0,153	$4{,}823 \times 10^{-3}$	0	0	0	0	0

Como se puede observar en los resultados, la magnitud de los momentos es igual para las tres primeras imágenes, para la imagen escalada existe una pequeña variación de 0.021 para el momento ϕ_1, si esta variación no influye en la etapa de entrenamiento y clasificación entonces podemos decir que el error es pequeño, con lo que probamos que para este ejemplo los momentos de Hu son invariantes a traslación, rotación y escala.

2.4. Momentos ortogonales en un circulo

En teoría todas las bases polinomiales del mismo grado son equivalentes porque ellas generan el mismo espacio de funciones. Cualquier momento con respecto a cierta base puede ser expresado en términos de momentos de cualquier otra base, desde este punto de vista los momentos ortogonales de cualquier tipo son equivalentes a los momentos geométricos [3]. Sin embargo la diferencia entre los momentos ortogonales y los geométricos aparece cuando se considera la estabilidad y los problemas computaciones en el dominio discreto.

Si la base polinomial $P_{pq}(r)$ es ortogonal, es decir sus elementos satisfacen la condición de ortogonalidad:

$$\iint_0^1 P_{nl}(r) P_{ml}(r) r dr = \frac{1}{2(n+1)} \delta_{mn} \tag{2.4.1}$$

donde el simbolo δ_{mn}, representa el simbolo de Kronecker, definido como:

$$\delta_{mn} = \begin{cases} 0, & m \neq n \\ 1, & m = n \end{cases} \tag{2.4.2}$$

estamos hablando de *momentos ortogonales en el círculo de radio unidad*. Estos están definidos como:

$$M_{nl} = \int_0^{2\pi} \int_0^1 P_{nl}(r) e^{-il\theta} f(r,\theta) r dr d\theta \tag{2.4.3}$$

Conversión a coordenadas polares

Para poder aplicar los momentos ortogonales circulares a una imagen en coordenadas cartesianas es necesario llevarla a coordenadas polares. Una imagen digital $f(x_j, y_k)$ está definida sobre una rejilla rectangular de $M \times N$ donde M y N representan filas y columnas respectivamente. Si consideramos por simplicidad que tenemos imágenes cuadradas de $N \times N$, es (x, y) la representación de un pixel. Podremos hacer una transformación geométrica, donde se mapeará la imagen cuadrada en un círculo unitario [5] [6]. En la Figura 2.4 se muestran los mapeos de coordenadas cartesianas a coordenadas polares.

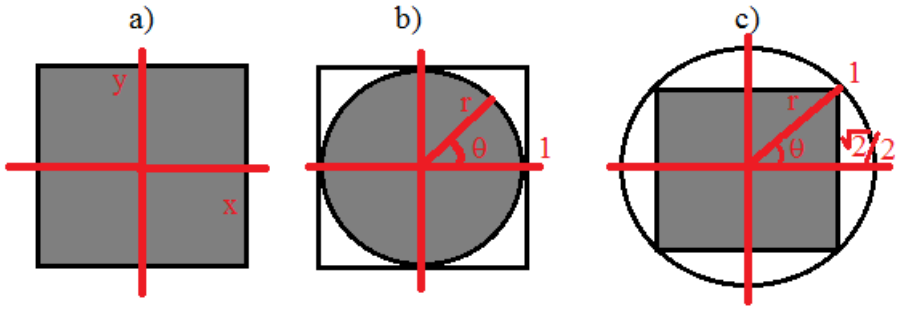


Figura 2.4: **a) Imagen en coordenadas cartesianas b) Imagen mapeada fuera del circulo unitario, c) Imagen mapeada dentro del circulo unitario**

$$x_j = \frac{2j + 1 - N}{D}, \qquad y_k = \frac{2k + 1 - N}{D}, \tag{2.4.4}$$

donde $j, k = 0, 1, ..., N - 1$, y

$$D = \begin{cases} N\sqrt{2}, & \text{para la imagen dentro del circulo} \\ N, & \text{para la imagen fuera del circulo} \end{cases} \tag{2.4.5}$$

Con la ecuación 2.4.4 se generan las coordenadas polares discretas.

$$r_{jk} = \sqrt{x_j^2 + y_k^2}, \qquad \theta_{jk} = arctan\left(\frac{y_k}{x_j}\right). \tag{2.4.6}$$

2.4.1. Momentos de Zernike

Dentro de los momentos ortogonales, podemos encontrar los momentos de Zernike (ZMs). Estos fueron introducidos en el análisis de imágenes en 1980 por Teague [6], quien uso los ZMs para construir invariantes a la rotación. Él uso el hecho de que los ZMs mantienen su magnitud bajo cualquier rotación arbitraria. Teague también mostró que los invariantes de Zernike de segundo y tercer orden son equivalentes a los momentos invariantes de Hu, cuando estos son expresados en términos de momentos geométricos.

Los ZMs de orden n con repetición l están definidos como:

$$Z_{nl} = \frac{n+1}{\pi} \int_0^{2\pi} \int_0^1 V_{nl}^*(r,\theta) f(r,\theta) r dr d\theta, \tag{2.4.7}$$

donde el asterisco (*) denota el complejo conjugado, $n = 0, 1, 2, ...$, $l = -n, -n+2, ..., n$, para $n - |l|$ par. Los polinomios de Zernike están definidos como:

$$V_{nl}(r,\theta) = R_{n,l}(r) e^{il\theta}, \tag{2.4.8}$$

donde la parte radial es:

$$R_{nl}(r) = \sum_{s=0}^{(n-|l|)/2} (-1)^s \frac{(n-s)!}{s!((n+|l|)/2-s)!((n-|l|)/2-s)!} r^{n-2s} \tag{2.4.9}$$

$$= \sum_{k=|l|,|l|+2,...}^{n} C_{nlk} r^k, \tag{2.4.10}$$

los coeficientes C_{nlk} están dados por:

$$C_{nlk} = \frac{(-1)^{(n-k)/2}((n+k)/2)!}{((n-k)/2)!((k+l)/2)!((k-l)/2)!}. \tag{2.4.11}$$

La aproximación discreta de la ecuación 2.4.7 para una imagen $f(x,y)$ de dimensiones $M \times N$ es escrita como:

$$Z_{nl} = \frac{n+1}{\pi} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} f(r_{jk}, \theta_{jk}) R_{nl}^*(r_{jk}) e^{il\theta_{jk}}, \tag{2.4.12}$$

donde $i = \sqrt{-1}$.

En la Figura 2.5 se grafican algunos ordenes de los polinomios radiales de Zernike $R_{nl}(r)$ y en la Figura 2.6 se muestran los primeros 3 órdenes de los polinomios complejos, tanto la parte real como la parte imaginaria.

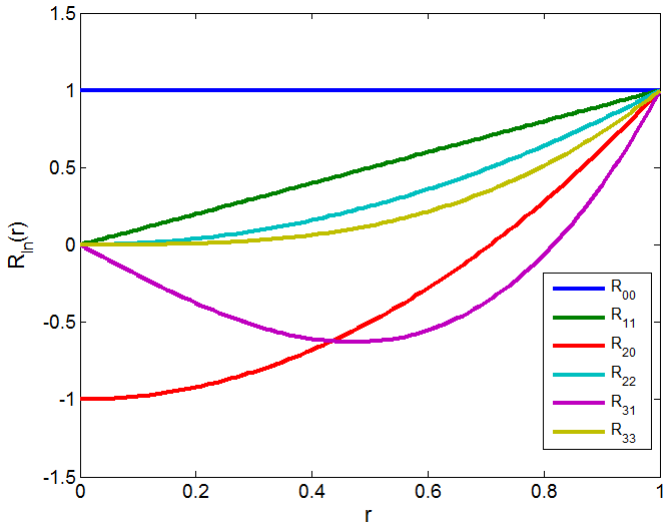


Figura 2.5: **Primeros 6 polinomios radiales de Zernike.**

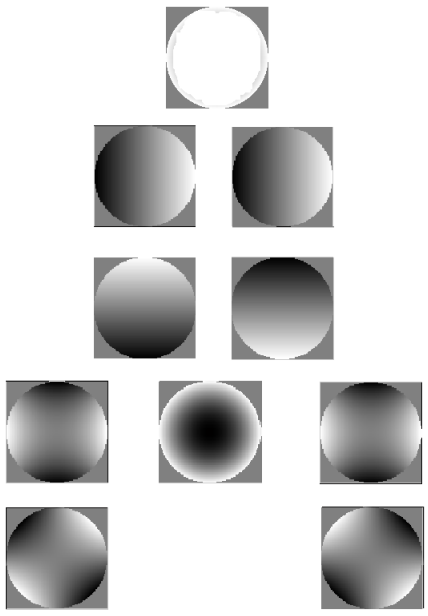


Figura 2.6: **Polinomios complejos de Zernike hasta el orden 2. La parte real: fila 1:** $n = 0, l = 0$**, fila 2:** $n = 1, l = -1, 1$**,fila 4:** $n = 2, l = -2, 0, 2$**. La parte imaginaria en las filas 3 y 5, los índices son los mismos como en la parte real.**

2.4.2. Momentos de Jacobi-Fourier

Los momentos de Jacobi-Fourier (JFMs) [7] de orden n y repetición l para una imagen en coordenadas polares $f(r,\theta)$ están definidos como:

$$\jmath_{nl} = \int_0^{2\pi} \int_0^1 f(r,\theta) P_{nl}(r,\theta) r dr d\theta, \tag{2.4.13}$$

donde $P_{nl}(r,\theta)$ es el kernel de la transformación que consiste de dos conjuntos de funciones separables: los polinomios ortogonales de Jacobi $J_n(\alpha,\beta,r)$ y la exponencial de Fourier $exp(il\theta)$. De esta manera,

$$P_{nl}(r,\theta) = J_n(\alpha,\beta,r) exp(il\theta). \tag{2.4.14}$$

La función generadora de polinomios ortogonales de Jacobi está dada por:

$$J_n(\alpha,\beta,r) = \sqrt{\frac{w(\alpha,\beta,r)}{b_n(\alpha,\beta)}} G_n(\alpha,\beta,r), \tag{2.4.15}$$

donde $G_n(\alpha,\beta,r)$ son los polinomios de Jacobi, $b_n(\alpha,\beta)$ es una constante de normalización y $w(\alpha,\beta,r)$ es la función de peso. Estas expresiones son calculadas de la siguiente manera:

$$G_n(\alpha,\beta,r) = \frac{n!\Gamma(\beta)}{\Gamma(a+n)} \times \sum_{s=0}^{n} (-1)^s \frac{\Gamma(\alpha+n+s)}{(n-s)!s!\Gamma(\beta+s)} r^s, \tag{2.4.16}$$

$$b_n(\alpha,\beta) = \frac{n!\Gamma(\beta)\Gamma(\alpha-\beta+n+1)}{\Gamma(\beta+n)\Gamma(a+n)(a+2n)}, \tag{2.4.17}$$

$$w(\alpha,\beta,r) = (1-r)^{\alpha-\beta} r^{\beta-1}. \tag{2.4.18}$$

El símbolo $\Gamma(*)$ representa la función Gamma la cual extiende el concepto de factorial (!) a los números complejos [8]. Si la parte real de z es positiva, entonces la integral,

$$\Gamma(z) = \int_0^{\infty} t^{z-1} e^{-t} dt \tag{2.4.19}$$

converge absolutamente. Si n es un número positivo entonces,

$$\Gamma(n) = (n-1)! \tag{2.4.20}$$

Calculo recursivo de los momentos de Jacobi-Fourier

Los cálculos de los factoriales y función Gamma $\Gamma(*)$ de la ecuación (2.4.16) aumentan el tiempo de cómputo y sólo es precisa para factoriales menores de 21. También el cálculo de r en altos ordenes, causa inestabilidad numérica en valores cercanos a 1. Camacho et al [9], propone una relación de recurrencia con respecto de n para el cómputo de los polinomios de Jacobi, esta relación está dada por:

$$A_n J_n(\alpha,\beta,r) = (2r - 1 - B_n)J_{n-1}(\alpha,\beta,r) - A_{n-1}J_{n-2}(\alpha,\beta,r) \tag{2.4.21}$$

donde $r \in [0,1], \alpha - \beta > -1, \beta$ y $\alpha > 0$, los coeficientes A_n y B_n son calculados como sigue,

$$A_n = \sqrt{\frac{4n(n+\alpha-\beta)(n+\beta+1)(n+\alpha-1)}{(2n+\alpha-1)^2(2n+\alpha)(2n+\alpha-2)}}, \tag{2.4.22}$$

$$B_n = \frac{(\alpha-1)(2\beta-\alpha-1)}{(2n+\alpha-1)(2(n-1)+\alpha-1)}. \tag{2.4.23}$$

Las condiciones de recurrencia están dadas por:

$$J_0(\alpha,\beta,r) = \sqrt{\frac{w(\alpha,\beta,r)}{b_0(\alpha,\beta)}}, \tag{2.4.24}$$

$$J_1(\alpha,\beta,r) = J_0(\alpha,\beta,r)\sqrt{\frac{(\alpha+2)\beta}{\alpha-\beta+1}}\left(\frac{\alpha+1}{\beta}r - 1\right). \tag{2.4.25}$$

Cabe destacar que dependiendo los valores de α, β podemos formar distintas familias de polinomios ortogonales, como se muestra en la Tabla 2.2

Tabla 2.2: Familias de polinomios de acuerdo a los valores α, β.

α, β	**Familia de polinomios**
$\alpha = 1, \beta = 1$	Legendre
$\alpha = 2, \beta = \frac{3}{2}$	Chebyshev
$\alpha = 2, \beta = 2$	Mellin
$\alpha = 4, \beta = 3$	Pseudo-Jacobi

En la Figura 2.7 se muestran los gráficos de los polinomios radiales de Jacobi usando el método directo 2.4.15 y el método recursivo 2.4.21, como se puede observar para altos ordenes el método directo es inestable para valores cercanos a 1. En la Figura 2.8 se muestran los polinomios radiales de Jacobi hasta orden 3.

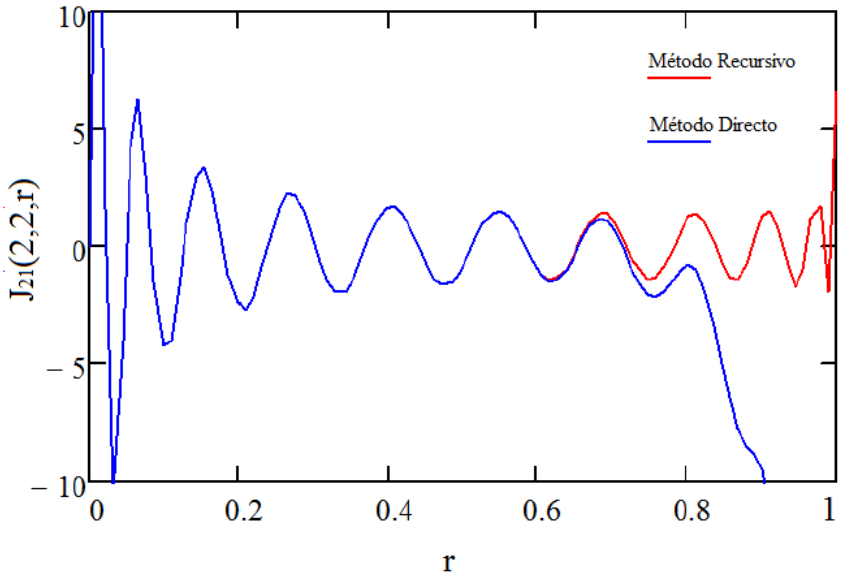


Figura 2.7: **Polinomios radiales de Jacobi $J_{21}(2,2,r)$, calculados usando el método directo y el recursivo.**

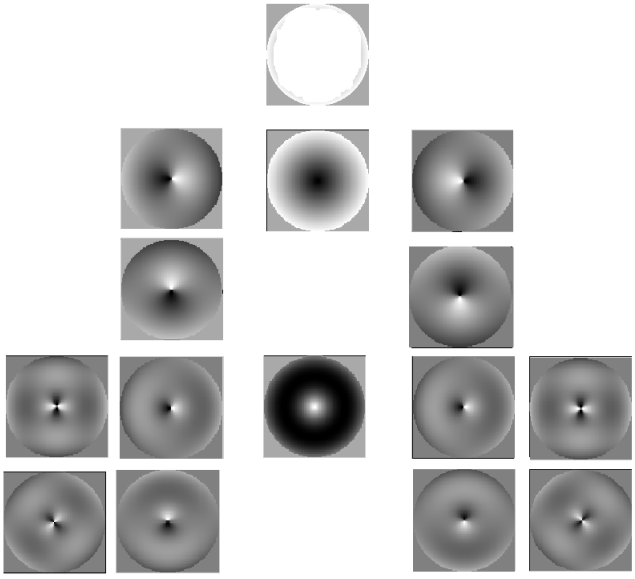


Figura 2.8: **Polinomios de Jacobi hasta el 2 orden. La parte real: fila 1: $n = 0, l = 0$, fila 2: $n = 1, l = -1, 0, 1$,fila 4: $n = 2, l = -2, -1, 0, 1, 2$. La parte imaginaria en las filas 3 y 5, los índices n, l son los mismos como en la parte real.**

2.4.3. Reconstrucción de Imágenes

La reconstrucción de una imagen original a partir de un conjunto de momentos ha sido discutida mucho en la literatura. La reconstrucción está relacionada con el poder de los momentos para el reconocimiento de patrones.

La reconstrucción de imágenes puede ayudarnos a determinar qué tan bien un conjunto de momentos pueden caracterizar una imagen [9]. De acuerdo a la teorías ortogonales una imagen original $f(j,k)$ puede ser reconstruida por un conjunto infinito de momentos de Jacobi-Fourier. La reconstrucción discreta de una imagen utilizando JFMs está dada por:

$$\tilde{f}(j,k) = \sum_{n=0}^{L}\sum_{l=0}^{L} |\jmath_{nl}| J_n(\alpha,\beta,r) exp(-il\theta), \tag{2.4.26}$$

donde $\tilde{f}(j,k)$ es la versión reconstruida de $f(j,k)$, $\jmath_{nl}$ son los momentos de Jacobi-Fourier de orden n y repetición l, $J_n()$ son los polinomios de Jacobi y L es el máximo orden de JFMs usados para la reconstrucción de la imagen.

Para el caso de la reconstrucción con momentos de Zernike la expresión está dada por [3]:

$$\tilde{f}(j,k) = \sum_{n=0}^{\infty}\sum_{l=-n,-n+2\ldots}^{n} Z_{nl} V_{nl}(j,k). \tag{2.4.27}$$

En la práctica la primera sumatoria es truncada a un rango finito.

2.4.4. NIRE

El error de reconstrucción normalizado (NIRE) [9] es usado para analizar el rendimiento de los momentos ortogonales, está definido como la normalización de la diferencia cuadrada entre la imagen de entrada $f(j,k)$ y la imagen reconstruida $\tilde{f}(j,k)$. La forma discreta del NIRE está dada por:

$$NIRE = \frac{\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}[\tilde{f}(j,k) - f(j,k)]^2}{\sum_{j=0}^{M-1}\sum_{k=0}^{N-1} f^2(j,k)}. \tag{2.4.28}$$

Observando la forma de la ecuación del NIRE se puede deducir que entre más parecidas sean las imágenes, menor será el error (NIRE), en caso de que el NIRE sea igual a 0 entonces podremos decir que la reconstrucción fue perfecta y se recuperó la imagen original.

En la Figura 2.9 se muestra el resultado de reconstrucción de una imagen binaria de 20×20 pixeles, usando diferentes órdenes de los momentos de Zernike y en la Figura 2.10 se presenta la gráfica del NIRE, se puede observar que conforme aumenta el orden de los momentos, el error (NIRE) va disminuyendo.

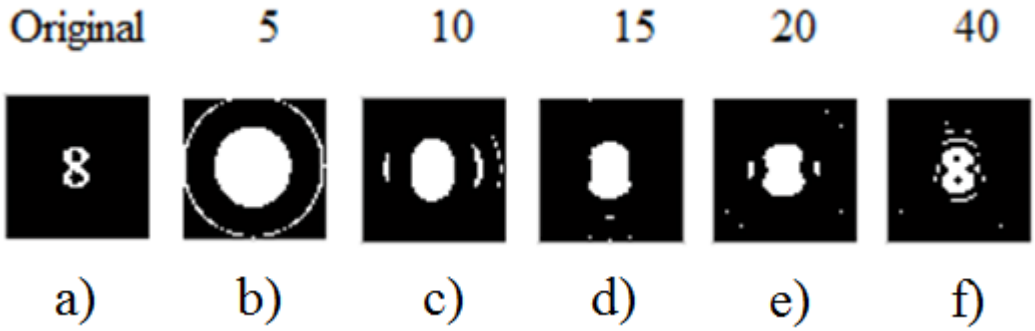


Figura 2.9: **a)Imagen original, b) , c), d), e), f) Reconstrucciones de la imagen original con momentos de Zernike usando ordenes 5,10,15,20,40 respectivamente.**

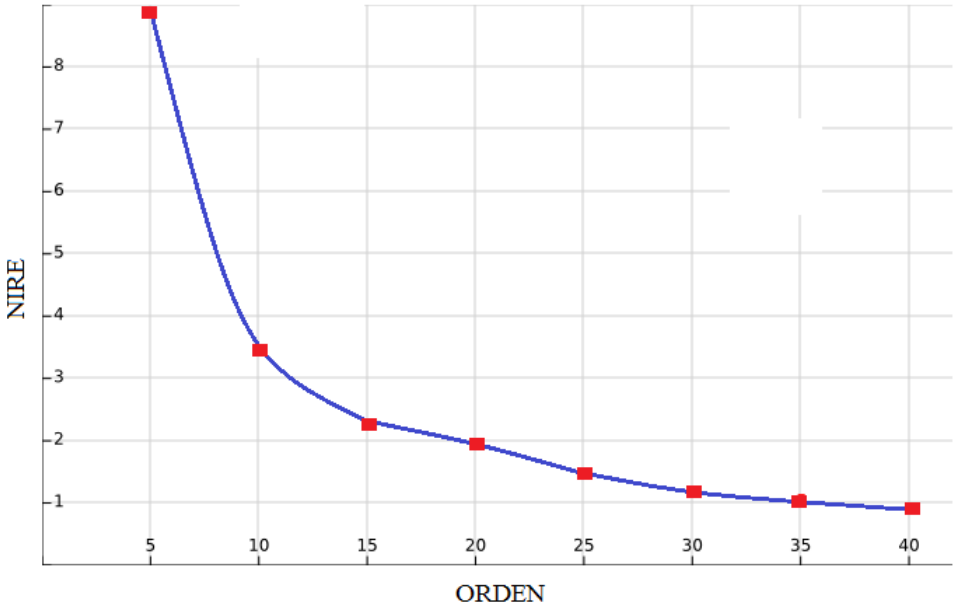


Figura 2.10: **Gráfica del NIRE para las reconstrucciones de la Figura 2.9**

Las Figuras 2.11 y 2.12 muestran reconstrucciones de una imagen binaria para distintos órdenes de JFMs, y su respectivo NIRE. Al igual que las reconstrucciones con momentos de Zernike el NIRE va decreciendo conforme se aumenta el orden de los momentos.

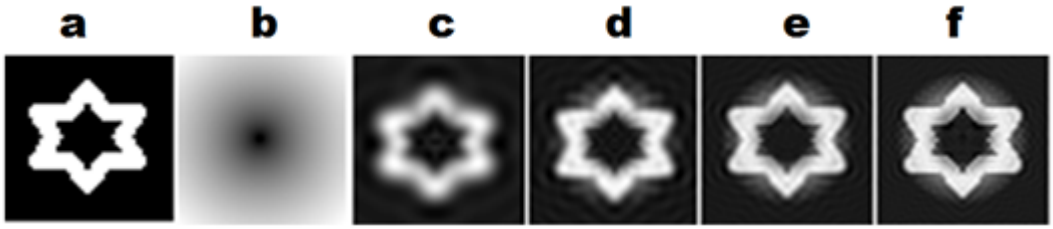


Figura 2.11: **a) Imagen Original, b), c), d), e), f) Reconstrucciones de la imagen original usando JFMs con ordenes 0,10, 20, 30 y 40 respectivamente.**

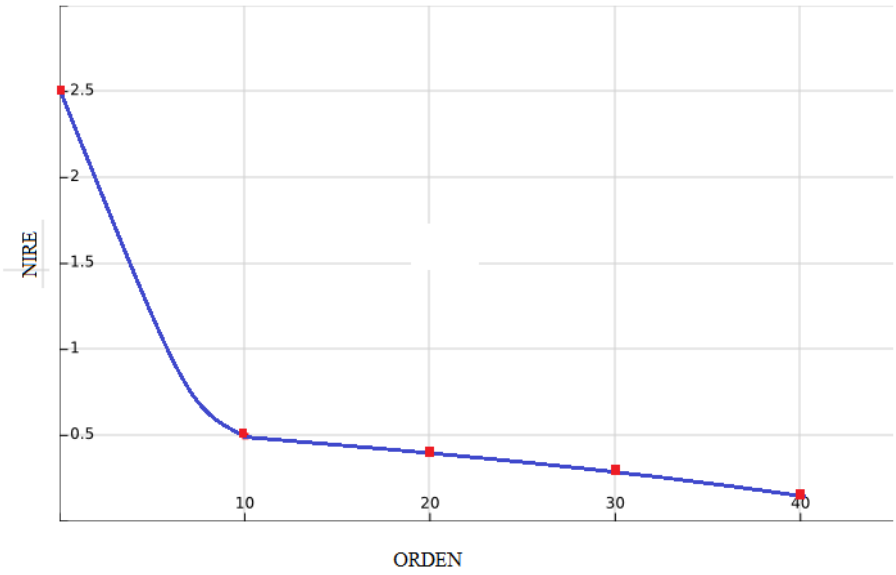


Figura 2.12: **Gráfica del NIRE para las reconstrucciones de la Figura 2.11**

Para el caso de una imagen en escala de grises se realizó la reconstrucción, utilizando los JFMs con distintos órdenes y $\alpha = \beta = 2$. En la Figura 2.13 se presentan los resultados de esta reconstrucción y en la Figura 2.14 se muestra su respectiva gráfica del NIRE.

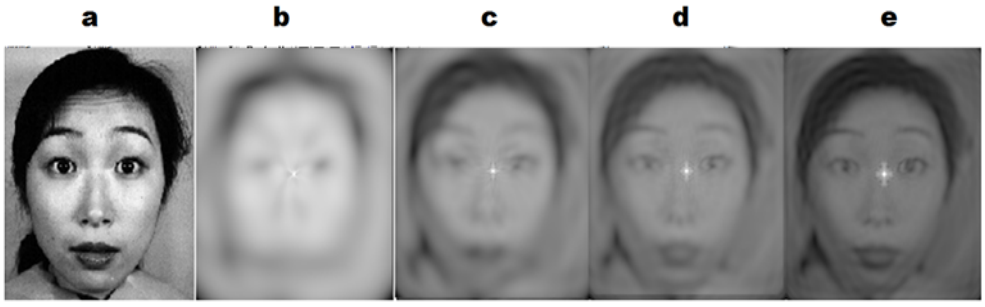


Figura 2.13: **a) Imagen Original, b), c), d), e) reconstrucción de imágenes usando JFMs con ordenes 10, 20, 30 y 40 respectivamente.**

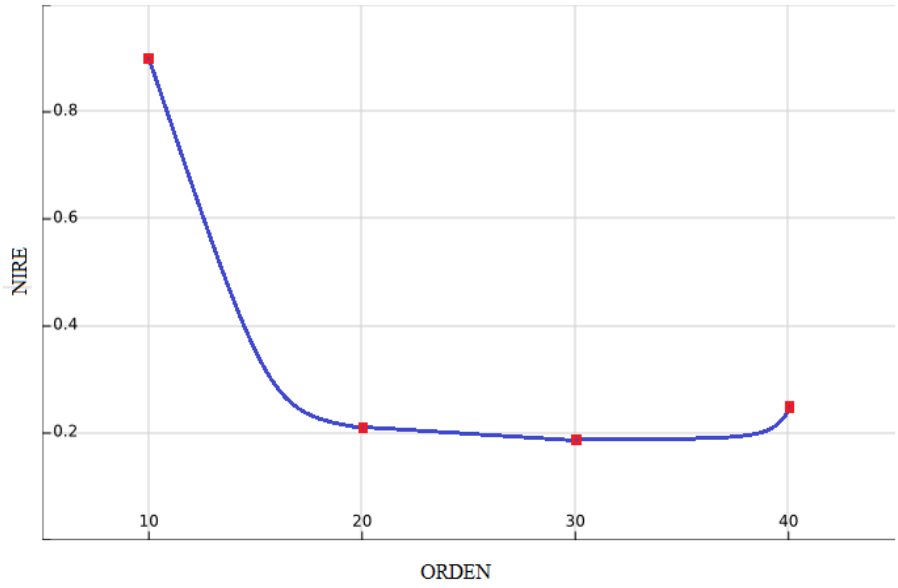


Figura 2.14: **Gráfica del NIRE para las reconstrucciones de la Figura 2.13**

A simple vista podemos ver en la Figura 2.13 que la reconstrucción con orden 40 es la mejor, pero observando la gráfica del NIRE vemos que para estas reconstrucciones la mejor según el NIRE es la reconstrucción utilizando orden 30, esto se debe a que el NIRE mide niveles de intensidad, por lo que al comparar la imagen original con la reconstruida de orden 40 podemos ver mucha diferencia de intensidades (original más clara, reconstruida más obscura).

2.5. Detección de Esquinas

Las esquinas son de las características más fiables para encontrar correspondencias entre imágenes [10] . En la imagen de la Figura 2.15 se muestran tres pixeles, uno dentro del objeto, uno en un borde y otro en una esquina. Si un pixel está dentro del objeto entonces no difiere de sus vecinos más cercanos, si un pixel esta sobre un borde difiere de sus vecinos más cercanos solo en una dirección, si el pixel está en una esquina entonces difiere de sus vecinos más cercanos en varias direcciones.

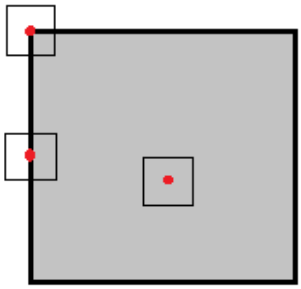


Figura 2.15: **Representación de un objeto con tres pixeles marcados, dentro del objeto, sobre un borde y sobre una esquina**

La detección de esquinas se usa muy frecuentemente en temas como la detección de movimiento, reconocimiento de objetos entre otros. Existen algunos algoritmos que permiten identificar esquinas de objetos en una imagen, el *Método de Harris* y el *Método de Shi y Tomasi* son algunos de estos, ambos métodos dependen de los valores propios.

2.5.1. Métodos de Harris, Shi y Tomasi

Estos métodos están basados en buscar cambios de intensidad para una imagen I usando una pequeña ventana que pueda ser desplazada por toda la imagen en cualquier dirección, como se muestra en la Figura 2.16.

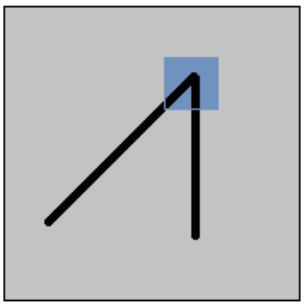


Figura 2.16: **Idea básica del algoritmo de Harris**

Los cambios de intensidad para un desplazamiento $[u, v]$ están dados por [11]:

$$E(u,v) = \sum_{x,y} w(x,y)[f(x+u, y+v) - f(x,y)]^2, \tag{2.5.1}$$

donde $w(x,y)$ es la función de la ventana (Figura 2.17), $f(x,y)$ representa la intensidad en un punto, $f(x+u, y+v)$ representa la intensidad en un punto desplazado.

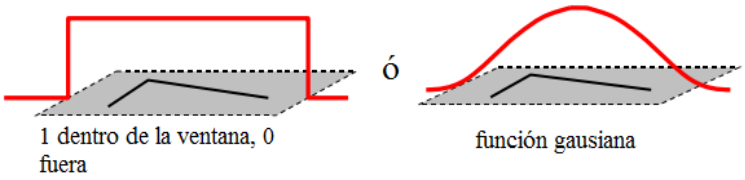


Figura 2.17: **Función ventana** $w(x,y)$

Para pequeños desplazamientos $[u, v]$, se tiene una aproximación bilineal:

$$E(u,v) \simeq [u,v] M \begin{bmatrix} u \\ v \end{bmatrix}, \tag{2.5.2}$$

donde M es una matriz de 2×2 calculada a partir de las derivadas de la imagen:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{bmatrix}, \tag{2.5.3}$$

donde f_x y f_y representan las derivadas parciales respecto a x e y respectivamente.

Una esquina se caracteriza por una variación grande de E en todas las direcciones del vector $[u, v]$. Si se analizan las magnitudes de los valores propios de la matriz M, se puede inferir que:

Si $\lambda_1 \approx 0$ y $\lambda_2 \approx 0$ entonces el pixel (u, v) no es un punto de interés.

Si $\lambda_1 \approx 0$ y λ_2 tiene algún valor positivo grande, entonces existe un borde.

Si λ_1 y λ_2 tiene valores positivos grandes, entonces existe una esquina.

Harris y Stephens [12] se dan cuenta que para calcular los valores propios el costo computacional es alto ya que se requieren calcular raíz cuadrada, por lo que proponen pensar en una función R dada por:

$$R = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 = det(M) - k traza^2(M), \tag{2.5.4}$$

por lo tanto el algoritmo no tiene que calcular la descomposición en valores propios de la matriz M, solo tiene que evaluar el determinante y la traza de M para encontrar las esquinas. El valor de k se determina empíricamente y regularmente se usan valores entre $0{,}04 \leq k \leq 0{,}15$.

La traza de una matriz cuadrada A de $N \times N$ está definida como la suma de los elementos de su diagonal principal [13], es decir:

$$traza(A) = A_{11} + A_{22} + ... + A_{nn} \tag{2.5.5}$$

En la Figura 2.19 se puede observar un ejemplo de la traza de una matriz.

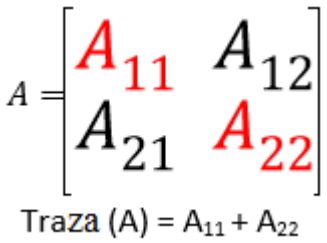


Figura 2.18: **Traza de una matriz**

Por otro lado el detector de esquinas de Shi y Tomasi [14] calcula $R = min(\lambda_1, \lambda_2)$, ya que bajo ciertas suposiciones, las esquinas son más estables para rastrear. Si R es mayor que un valor predefinido, este puede ser marcado como una esquina. Shi y Tomasi demostraron experimentalmente que su criterio era mejor que el de Harris.

En la Figura 2.19 se muestra una comparación de los resultados obtenidos al aplicar el método de Harris y el método de Shi y Tomasi a la misma imagen.

Figura 2.19: **Aplicación del método de Harris y del método de Shi y Tomasi para la detección de esquinas a la imagen del** *cameraman***.**

2.6. Entrenamiento y clasificación

Acabada la etapa de la extracción de características de las imágenes, tendremos un conjunto de patrones que representan a cada imagen, estos patrones nos sirven para el entrenamiento y clasificación, los patrones pueden ser cuantitativos, es decir números.

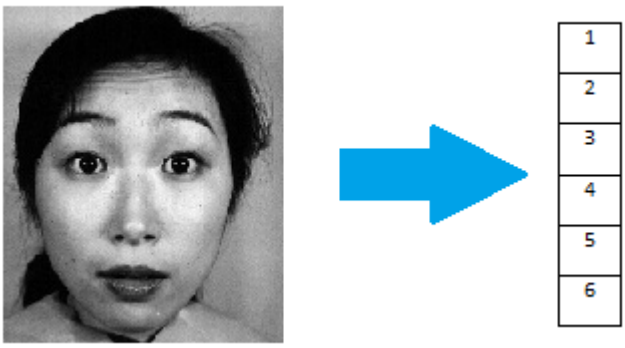


Figura 2.20: **A partir de una imagen generamos un patrón, que describe a la imagen.**

La etapa de clasificación consiste asignar el patrón de características a un grupo o clase, basándose en las características o conjunto de atributos del patrón. Regularmente para esta etapa se utiliza el aprendizaje automático en el cual se desarrollan técnicas las cuales deben hacer que un algoritmo sea capaz de ir aprendiendo. De acuerdo al número de patrones de los cuales se conozca la clase a la que pertenecen y los cuales permitan a nuestro sistema aprender, la clasificación puede ser: **supervisada, parcialmente supervisada o no supervisada**

Clasificación supervisada

En este tipo de clasificación se parte de un conjunto de patrones para los cuales se conoce la clase a la que pertenecen, estos patrones sirven para el entrenamiento del clasificador y deben ser discriminatorios para la clasificación. Algunos clasificadores dentro de este grupo son: Redes neuronales artificiales (RNA) y vecinos más cercanos (KNN).

Clasificación parcialmente supervisada

En este tipo de clasificación el clasificador es entrenado con patrones conocidos sólo para algunas clases. Los algoritmos EM y Assemble pueden entrar en este grupo.

Clasificacion no supervisada

El clasificador sólo necesita información del patrón a clasificar y algunos parámetros que sirvan para limitar el número de clases, aquí los patrones más próximos se van agrupando para formar las clases. Algunos de los clasificadores que pertenecen a este grupo son: c-means e ISODATA.

2.7. Proceso de clasificación supervisada

El proceso de clasificación supervisada, independientemente del tipo de clasificador utilizado, consta de una serie de pasos que se describen a continuación [15]:

1. Se eligen muestras de objetos para los cuales la clase es conocida. Se extraen las características del objeto para crear el patrón característico.
2. Los patrones sirven para el entrenamiento del clasificador, con esto se calculan las fronteras entre cada clase.
3. Se extraen características de objetos desconocidos los cuales se quieran clasificar.
4. El clasificador utiliza las fronteras calculadas durante el entrenamiento, para decidir a qué clase pertenece el objeto a clasificar.

En la etapa de entrenamiento y del cálculo de las fronteras entre cada clase, dependiendo de cómo el clasificador separe las clases, se pueden dividir en dos grupos: si el clasificador usa hiperplanos para separar las clases, entonces se denomina clasificador lineal, si el clasificador utiliza otra superficie arbitraria entonces se denomina clasificador no lineal, ver Figura 2.21.

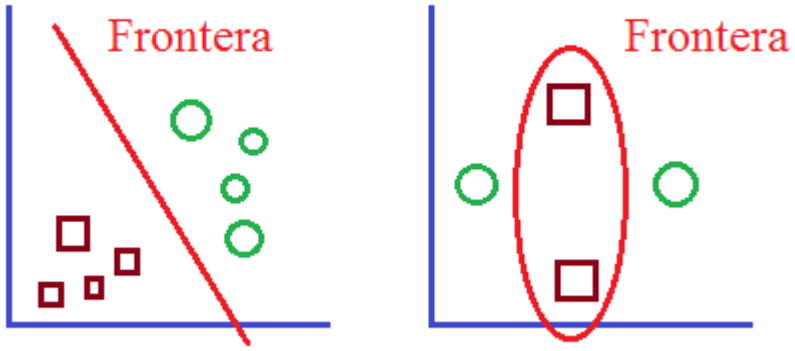


Figura 2.21: **Clasificación lineal (izquierda) y clasificación no lineal (derecha).**

2.8. Clasificadores supervisados

Los clasificadores supervisados operan usualmente sobre la información suministrada por un conjunto de objetos, instancias y/o prototipos de entrenamiento que poseen una etiqueta de clase previamente asignada [16] . A este conjunto de objetos etiquetados se le llama conjunto de entrenamiento y la información que ellos proporcionan es utilizada para la clasificación de nuevos objetos.

El objetivo principal de los clasificadores es determinar a qué clase de las que ya tiene conocimiento mediante el entrenamiento, debe pertenecer una nueva muestra, esto con la información que puede obtener del conjunto de entrenamiento.

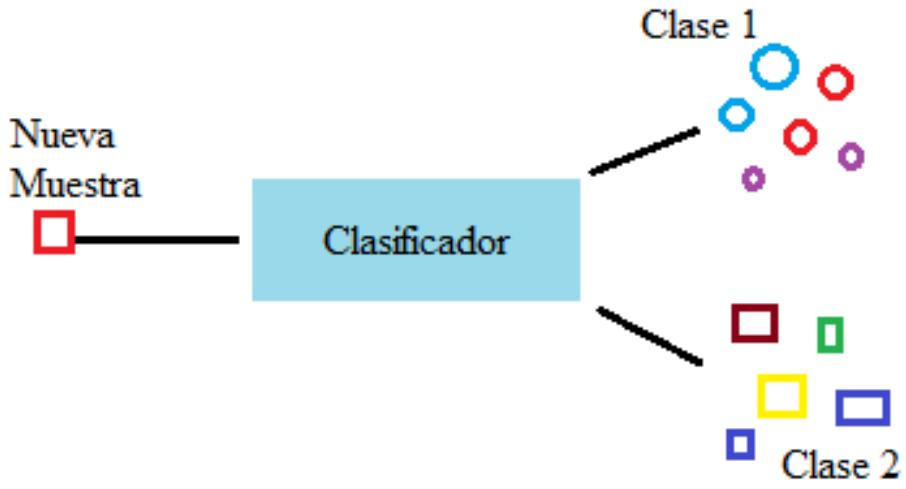


Figura 2.22: **Esquema básico de un clasificador supervisado.**

Entre las aplicaciones que se les puede dar a estos clasificadores están [17]:

1. Reconocimiento de patrones.
2. Detección de transacciones bancarias fraudulentas.
3. Sistemas para la toma de decisiones médicas.
4. Estrategias de marketing.
5. Reconocimiento de rostros.

2.8.1. Vecinos más cercanos (KNN)

Es un método para clasificar casos basándose en su parecido a otros casos. En el aprendizaje automático se desarrolla como una forma de reconocer patrones sin la necesidad de una coincidencia exacta con los patrones de entrenamiento. Los casos parecidos están próximos y los que no están alejados entre sí. Por lo que la distancia entre dos casos puede ser una medida de disimilaridad [18].

Los casos próximos entre sí se denominan $vecinos$. Cuando se presenta un nuevo caso, se calcula su distancia con respecto a los casos del modelo. En su versión más sencilla, el nuevo caso se clasificará con la clase de su vecino más cercano.

Se puede también especificar el número de vecinos más próximos que deben de examinarse, este valor se denomina k. En la Figura 2.23 se muestra como se clasificaría un nuevo patrón utilizando dos valores diferentes para k. Cuando $k = 3$ el patrón se clasificaría como clase 1 ya que la mayoría de vecinos más próximos pertenecen a la clase 1. Cuando $k = 5$, el patrón se incluye en la clase 2 porque la mayoría de vecinos más próximos pertenece a la clase 2.

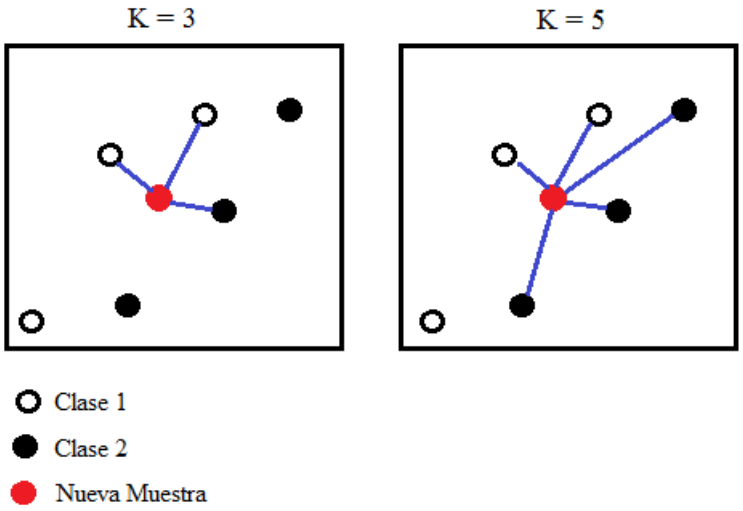


Figura 2.23: **Efectos de cambio de k en la clasificación.**

Como se mencionó anteriormente la clasificación se basa en encontrar distancias con sus vecinos más próximos, pero la distancia se puede determinar mediante varias métricas: distancia euclidiana, distancia de Mahalanobis, métrica de City block, distancia de Chebychev, etc. En su versión más común este clasificador utiliza la distancia euclidiana.

En general la distancia euclidiana [19] entre dos puntos $P = (p_1, p_2, ...p_n)$ y $Q = (q_1, q_2, ..., q_n)$ del espacio euclidiano $n - dimensional$, se define como:

$$d_E(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}. \quad (2.8.1)$$

2.8.2. Clasificador AdaBoost

Boosting es una palabra la cual hace referencia a un tipo de algoritmos cuya finalidad es encontrar una hipótesis fuerte a partir de hipótesis simples y débiles. El algoritmo de clasificación AdaBoost fue propuesto por Freud and Schapire [20] y la idea básica de éste es aprender de un número simple de clasificadores débiles.

Adaboost es un clasificador muy empleado en la detección de objetos, como el caso de Viola y Jones [21] o para el reconocimiento de caras donde fue usado por Michaels y Jones [22].

Este clasificador entrena iterativamente una serie de clasificadores débiles, durante cada ronda de entrenamiento del algoritmo se añade un nuevo clasificador débil que se centra en los ejemplos que fueron mal clasificados en las rondas anteriores, de tal modo que al combinar todos se obtenga un clasificador de elevadas prestaciones.

En la Figura 2.24 se puede observar un diagrama de flujo acerca del funcionamiento del Adaboost [23] y en la Tabla 2.3 se muestran las ventajas y desventajas de Adaboost [25].

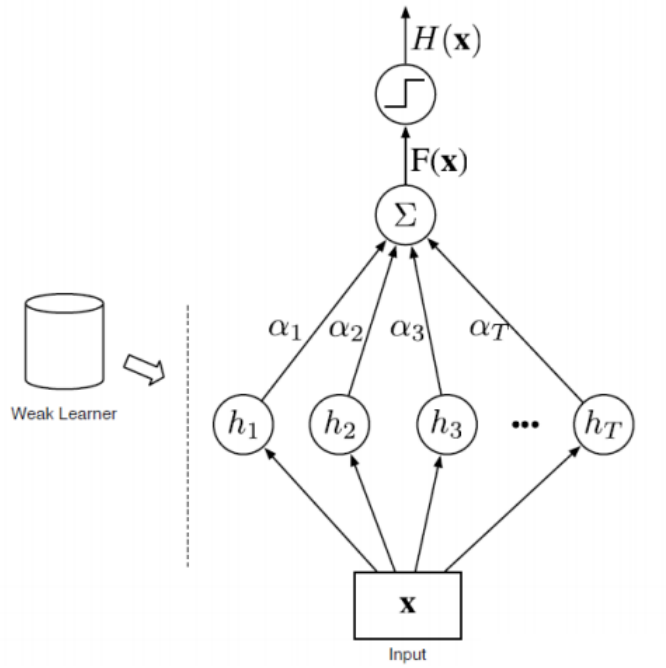


Figura 2.24: **Diagrama del funcionamiento del clasificador Adaboost [23].**

Tabla 2.3: Adaboost: ventajas y desventajas

Ventajas	**Desventajas**
Rapido	Clasificadores débiles complejos pueden causar overfitting
Simple y fácil de programar	Clasificadores débiles simples pueden causar overfitting
No necesita parámetros excepto las iteraciones	Es vulnerable al ruido principalmente en datos empíricos
No se necesita conocimiento previo acerca del clasificador débil	
Efectividad demostrada	
Es versátil	

Descripción técnica del Adaboost
Dado un conjunto de entrenamiento $(x_1, y_1), ..., (x_m, y_m)$, $y_i \in -1, +1$ es la clasificación, $x_i \in X$ es el objeto o instancia.

Inicializar $D_i = 1/m$ para $i = 1, ..., m$
Para $t = 1, ..., T$
Entrenar un clasificador débil usando D_i
Obtener una hipótesis débil $h_t : X \to -1, +1$
La meta es obtener un h_t con un error bajo. $\epsilon_t = Pr_{iD_t}[h_t(x_i) \neq y_i]$
Calcular el peso del clasificador débil actual. $\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t})$
Actualizar la distribución D_t según los errores de la etapa previa. $D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
Fin

Donde Z_t es un factor de normalización, la hipótesis final se expresa como:

$$H(x) = sing(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{2.8.2}$$

Lo que se tiene al final es una sumatoria de todas las hipótesis débiles calculadas para poder obtener una hipótesis fuerte [23].

2.8.3. Clasificador Lineal Discriminante

Es una técnica de aprendizaje supervisado. La idea principal de éste es obtener una proyección de los datos en un espacio menor o incluso de igual dimensión que los datos entrantes, con el fin de que la separabilidad de las clases sea la mayor posible.

Una de las implementaciones de este algoritmo es la propuesta por Sir R. A. Fisher [26], una explicación simple del problema es la siguiente [27]:

Encontrar un vector w de proyección, que proyecte los datos a un espacio uni-dimensional de manera de obtener la mayor separabilidad entre sus clases.

Formalizando este problema, tenemos un conjunto de vectores de prueba $x_i, x_2, ..., x_n$ etiquetados en c clases. Cada clase cuenta con N_c patrones. Se busca w, para obtener $y_i = w^T x_i$ proyecciones uni-dimensionales de los patrones.

La idea del algoritmo de Fisher es maximizar la siguiente función:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}, \tag{2.8.3}$$

donde S_B es la matriz de dispersión inter-clase y S_W es la matriz de dispersión intra-clase. Siendo más precisos

$$S_B = \sum_c N_c(\mu_c - \mu)(\mu_c - \mu)^T, \tag{2.8.4}$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T, \tag{2.8.5}$$

donde μ_c es la media de cada clase, μ la media de todos los datos, N_c la cantidad de patrones de la clase c.

El algoritmo de Fisher busca encontrar el vector w de proyección que maximice el cociente entre la matriz S_B y S_W.

Haciendo algunas operaciones se puede ver que el w que maximiza la función objetivo debe cumplir:

$$S_B w = \lambda S_W w. \tag{2.8.6}$$

Si S_W es no singular se puede resolver el problema de valores propios para una matriz $S_W^{-1} S_B$:

$$S_W^{-1} S_B w = \lambda w. \tag{2.8.7}$$

Si se sustituye la solución en la ecuación 2.8.3 se obtiene lo siguiente:

$$J(w) = \frac{w^T S_B w}{w^T S_W w} = \lambda_k \frac{w_k^T S_B w_k}{w_k^T S_W w_k} = \lambda_k, \tag{2.8.8}$$

con $k = 1..d$, siendo w_k vector propio k de valor propio λ_k.

En consecuencia, para maximizar la solución se debe considerar el vector propio con mayor valor propio asociado.

Bibliografía

[1] Dourado Lema, Juan Manuel y CALVO ESTÉVEZ, Rosa María. *Reconocimiento de objetos*, Grupo de visión artificial y reconocimiento de patrones (VARPA), Universidad de A Coruña, (2007).

[2] Juan Rodríguez Povedano, *Anotación Automática de Imágenes*, Universidad Carlos III de Madrid, Madrid, Octubre (2010).

[3] Jan Flusser,et al. *Moments and Moment Invariants in Pattern Recognition*, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic,Prague, Czech Republic

[4] Hu, M.K. *Visual pattern recognition by moments invariants*, IRE Trans. on Information Theory, Vol. 8. Pp 179-187, (1962)

[5] R. Upneja, C. Singh, *Fast computation of Jacobi-Fourier moments for invariant image recognition*, India, (2015).

[6] A.P.Vinanco, A.M.Ramirez and F.G.Agustin, *Digital image reconstruction by using Zernike moments*, In Proc. of SPIE pp. 281-289, Sept. (2003).

[7] M. R. Teague, *Image analysis via the general theory of moments*, J. Opt. Soc. Amer., vol. 70, pp. 920-930, Aug (1980).

[8] Ping et al, *Generic orthogonal moments: Jacobi-Fourier moments for invariant image description*, Pattern Recognition, vol. 40, no. 4, pp. 1245–1254, (2007)

[9] Davis, Philip J, *Leonhard Euler's Integral: A Historical Profile of the Gamma Function*, IRE Trans. on Information Theory, Vol. 8. Pp 179-187, (1962)

[10] C. Camacho-Bello, C. Toxqui-Quitl and A. Padilla-Vivanco, *Generic orthogonal moments and applications. Moments and Moment Invariants - Theory and Applications*, chapter 8,

(2014)

[11] *Corner Detection*, Disponible en: http://www.mathworks.com/help/images/corner-detection.html

[12] Darya Frolova, Denis Simakov, *Matching with Invariant Features*, The Weizmann Institute of Science,March (2004)

[13] Harris, C., and M. Stephens, *A Combined Corner and Edge Detector*, Proceedings of the 4th Alvey Vision Conference, pp. 147-151, August (1988).

[14] Hazewinkel, Michiel,*Trace of a square matrix*, Encyclopaedia of Mathematics, Springer, ISBN 978-1556080104 (2001)

[15] J. Shi and C. Tomasi,*Good Features to Track*, 9th IEEE Conference on Computer Vision and Pattern Recognition (1994)

[16] Manuel Garrido Satue, *Reconocimiento de señales de trafico para un sistema de ayuda a la conducción*,Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, (2013).

[17] J Ruiz-Shulcloper. *Pattern recognition with mixed and incomplete data*, Pattern Recognition and Image Analysis, 18(4):563-576,ISSN 1054-6618. doi: 10.1134/S1054661808040044, (2008).

[18] Octavio Loyola Gonzalez, José Francisco Martínez Trinidad, Milton García Borroto, *Clasificadores Supervisados basados en Patrones Emergentes para Bases de Datos con Clases Desbalanceadas* , Reporte Tecnico No. CCC-14-004, INAOE (2014)

[19] *Análisis vecino más cercano*, IBM Knowledge Center, Disponible web en: http://www.ibm.com/support/knowledgecenter/SSLVMB_21.0.0 /com.ibm.spss.statistics.help/idh_idd_knn_variables.htm?lang=es

[20] Ayala, Dominguez, Quintero, *Elementos de la Topología General*, Addison-Wesley Iberoamericana, (1997).

[21] Y. Freund and R. Schapire, *A decision-theoristic generalization of on-line learning and an application to boosting* Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119-139, (2000).

[22] P. Viola and M. Jones, *Rapid object detection using a boosted cascade of simple features*, in Proc. of IEEE Conf. on Computer Vision and Pattern Recognition Kauai, Hawaii: (2001).

[23] J. Michael and P. Viola, *Face recognition using boosted local features*, in Proc. of International Conference on Computer Vision (ICCV) (2003).

[24] Alberto Alejandro Morales Sánchez, *Uso de características no lineales para identificar llantos de recién nacidos con un conjunto clasificador*, Tesis profesional, Universidad de las Américas Puebla, Cholula, Puebla, México (2015)

[25] Eric Emer, *Boosting (Adaboost Algorithm)*, Disponible web en: http://math.mit.edu/ rothvoss/18.304.3PM/Presentations/1-Eric-Boosting304FinalRpdf.pdf (2012)

[26] Fisher, R. A., *The Use of Multiple Measurements in Taxonomic Problems.* Annals of Eugenics, Vol. 7, pp. 179-188, (1936)

[27] Delbracio M., Mateu M., *Trabajo Final de Reconocimiento de Patrones: Identifiación utilizando PCA, ICA y LDA* Disponible web en: http://iie.fing.edu.uy/investigacion/grupos/biometria/proyectos/patrones/RecPat_MM.pdf (2006)

Capítulo 3

Técnica propuesta y bases de datos

3.1. Introducción

Los gestos faciales son una representación muy específica de la emoción que presenta una persona. Estos gestos pueden ser capturados en una imagen, y dada la gran información geométrica presente en la imagen de un rostro, se hace posible crear algoritmos para la detección de la emoción. Sin embargo las diferencias entre individuos, los cambios entre expresiones faciales, las oclusiones en alguna parte del rostro en la imagen, etc., pueden ser problemas que contribuyan a aumentar la complejidad del método de detección del tipo de emoción.

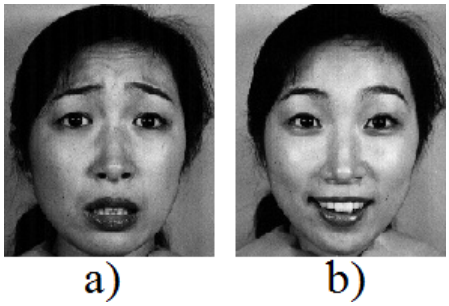


Figura 3.1: **Ejemplo de gestos faciales para dos emociones diferentes. a) Expresión de miedo y b) expresión de felicidad.**

Cuando se proponen técnicas para el reconocimiento de las emociones, es necesario contar con bases de datos que ayuden a determinar el rendimiento de la técnica, estas bases de datos suelen contener imágenes, videos, audios, etc. Para este trabajo se utilizo la ”The Japanese Female Facial Expression”(JAFFE) Database [1]. Esta base de datos se utiliza en muchos trabajos relacionados con el tema, para probar la eficiencia de las técnicas que se proponen en dichos trabajos.

En este capítulo se describen las características principales de las bases de imágenes utilizadas para evaluar el rendimiento de la técnica propuesta, se hace una descripción del robot con el que se capturaron las imágenes de la base de datos propia, se describe la técnica propuesta para la detección de emociones y se describe un software desarrollado en Matlab para la evaluación

probar el rendimiento de la técnica propuesta.

3.2. The Japanese Female Facial Expression (JAFFE) Database

La base de datos JAFFE [1] contiene 213 imágenes de 7 expresiones faciales diferentes, para la cual se utilizaron 10 mujeres japonesas como modelos. La base de datos fue planificada y realizada por Michael Lyons, Miyuki Kamachi y Jiro Gyoba. Las imágenes fueron capturadas en el departamento de Psicología de la Universidad de Kyushu.

Las expresiones faciales con la que cuenta esta base de datos son:

1. Felicidad

2. Tristeza

3. Sorpresa

4. Enojo

5. Disgusto

6. Miedo

7. Neutral

Las imágenes están en formato tiff, sin compresión y de tamaño 256×256 pixeles. Algunas imágenes de esta base de datos se pueden observar en la Figura 3.2

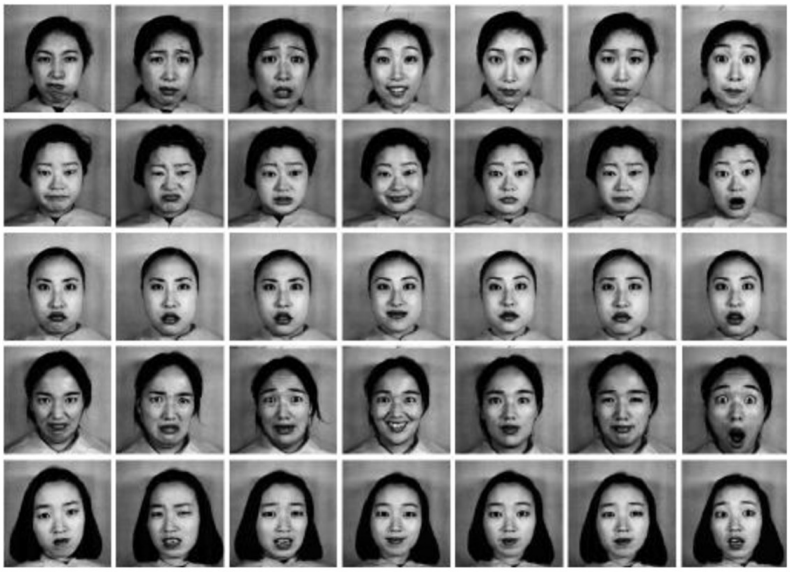


Figura 3.2: **Ejemplo de las imágenes de la base de datos JAFEE.**

La base de datos JAFEE está disponible de forma gratuita para investigaciones no comerciales.

3.3. Robot Nao

Para la adquisición de imágenes de prueba, se utilizó el robot NAO [2], el cual es un robot humanoide de 58 cm. de altura, que tiene capacidad de autonomía, desarrollado por la empresa francesa Aldebaran. La primera versión de este robot salió en el año 2006, actualmente está en su quinta versión. Las principales características del robot se pueden ver en el Cuadro 3.1.

Figura 3.3: **Robot NAO.**

Tabla 3.1: Características principales del robot NAO.

Grados de libertad	25
CPU	ATOM Z530 1.6 GHz
Memoria	256 MB SDRAM
Sensores	Inercia con giroscopio y acelerómetro de 3 ejes.
Conectividad	Ethernet RJ-45, Wifi
Camaras	2x (960p@30fps).
Altavoces	2x con síntesis vocal multi-idioma
Micrófonos	4x con reconocimiento de voz multi-idioma

Como se menciona en el Cuadro 3.1 el robot NAO cuenta con dos cámaras situadas en la cabeza, estas cámaras cuentan con un ángulo de visión vertical de 47.64° y horizontal de 60.97°. Se puede ver el esquema de estos ángulos en las Figuras 3.4 y 3.5

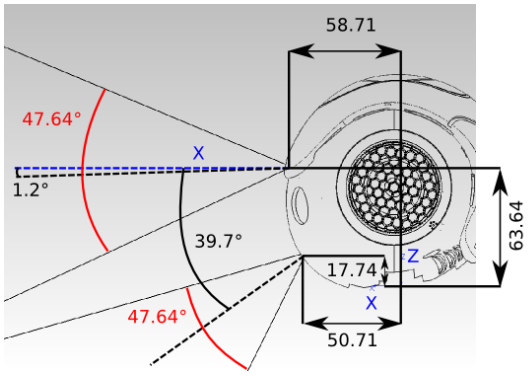


Figura 3.4: **Ángulo de visión vertical de las cámaras del robot NAO.**

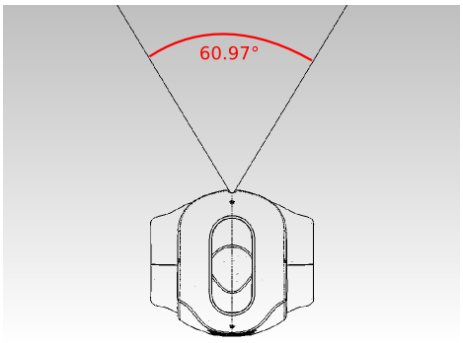


Figura 3.5: **Ángulo de visión horizontal de las cámaras del robot NAO.**

Algunas de las especificaciones técnicas de las cámaras con las que cuenta el NAO se pueden observar en el cuadro 3.2

Tabla 3.2: Especificaciones técnicas de las cámaras del robot NAO

Modelo	MT9M114
Resolución	1.22 Mp
Pixeles Activos	1288×968
Tamaño de pixel	$1{,}9\mu m \times 1{,}9\mu m$
FPS	30
Rango Focal	$30cm - \infty$
Campo de visión	72.6°DFOV (60.9°HFOV,47.6°VFOV)

Las cámaras cuentan con una serie de parámetros los cuales pueden ser modificados, en el Cuadro 3.3 se muestran los principales. Además las resoluciones soportadas por las cámaras se muestran en el Cuadro 3.4

Tabla 3.3: Lista de los principales parámetros modificables en las cámaras del Robot NAO.

Parámetro	**Mínimo**	**Máximo**	**Default**
Brillo	0	255	55
Contraste	16	64	32
Saturación	0	255	128
Resolución	QVGA (320x240)	4VGA(1280x960)	QVGA(320x240)
FPS	1	30	5
Tiempo de exposición (tiempo en ms = valor/10)	1	2500(250 ms)	NA

Tabla 3.4: Resoluciones compatibles en las cámaras del Robot NAO.

QQQQVGA	Imagen de 40 x 30px
QQQVGA	Imagen de 80 x 60px
QQVGA	Imagen de 160 x 120px
QVGA	Imagen de 320 x 240px
VGA	Imagen de 640 x 480px
4VGA	Imagen de 1280 x 960px

El esquema para la toma de imágenes, se presenta en la Figura 3.6. Se da la orden desde la computadora, para que el robot capture una imagen mediante alguna de las dos cámaras con la que cuenta, la imagen es enviada a la computadora y es guardada para después realizar el procesado, descripción y clasificación. Las imágenes capturadas son en calidad VGA (640x480), espacio de color RGB.

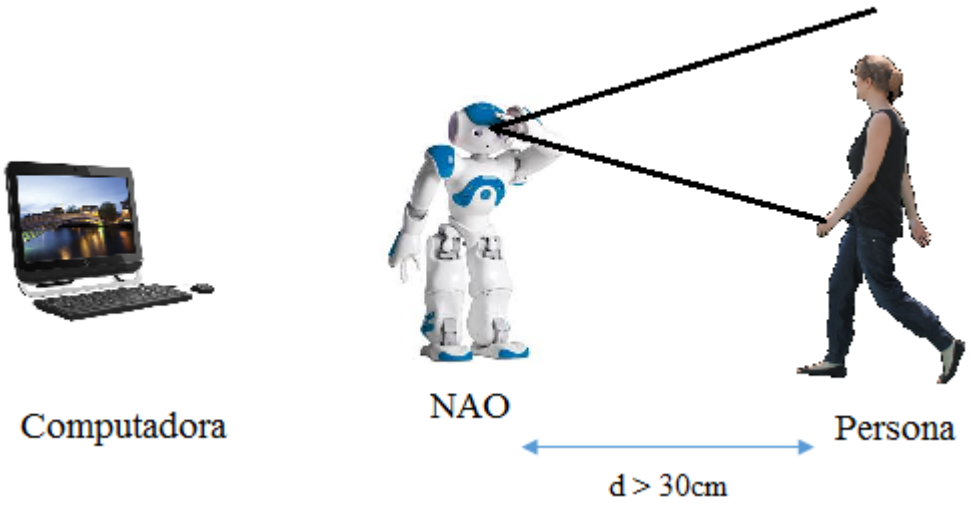


Figura 3.6: **Esquema de la toma de imágenes con el robot NAO.**

3.4. Técnica propuesta para el reconocimiento de emociones en imágenes digitales

En este trabajo de investigación se propone una técnica para la reconocimiento de emociones, basada en el procesamiento de imágenes digitales de rostros de personas, descriptores y clasificadores. La técnica se divide principalmente en 5 etapas: la **detección del rostro** para tener una imagen solo con la información necesaria, del rostro en este caso; el **procesado** de la imagen para mejorar la calidad de la imagen y/o quitar o agregar información con el objetivo de que las etapas siguientes tengan un mejor rendimiento; la **extracción de características** para crear patrones de cada imagen, los cuales servirán tanto para el **entrenamiento** de un clasificador, y para la **clasificación** de la emoción.

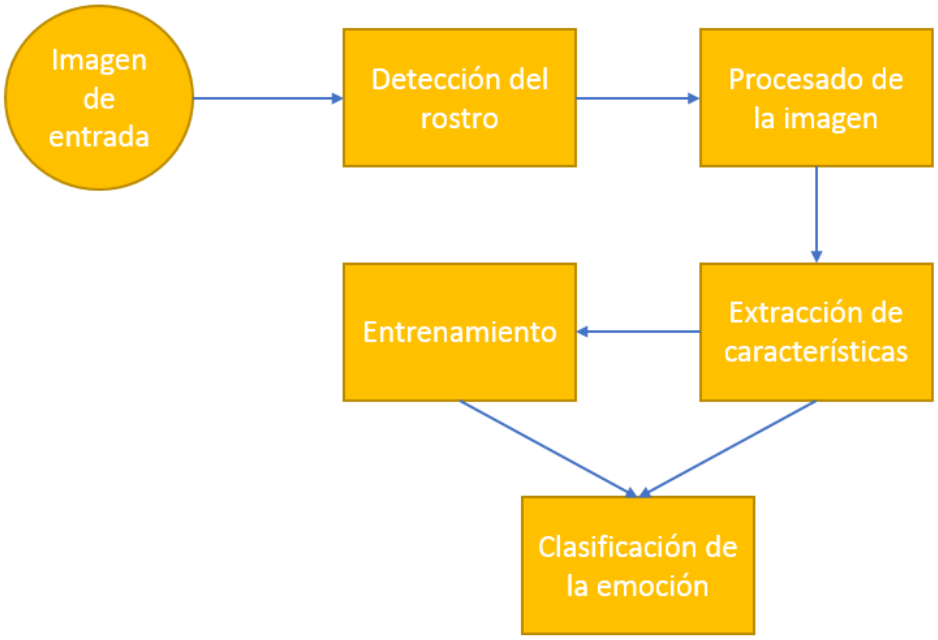


Figura 3.7: **Diagrama de la técnica propuesta.**

3.4.1. Detección del rostro

La detección del rostro se puede catalogar como un caso específico del reconocimiento de objetos, en donde la tarea principal es encontrar la localización y el tamaño de todos los objetos en una imagen que correspondan a cierta clase. En el caso de la detección de rostros la tarea se centra en ubicar los rostros presentes en una imagen o en un video.

Existen diversos algoritmos para realizar la tarea de la detección del rostro, uno de estos es el algoritmo de Viola-Jones [3], este algoritmo provoco una revolución en el campo de la detección de rostros mediante una computadora, el algoritmo puede detectar rostros en tiempo real ya que es bastante rápido y sencillo. En la Figura 3.8 se muestra un ejemplo de la implementación del algoritmo de Viola-Jones de OPENCV para Android.

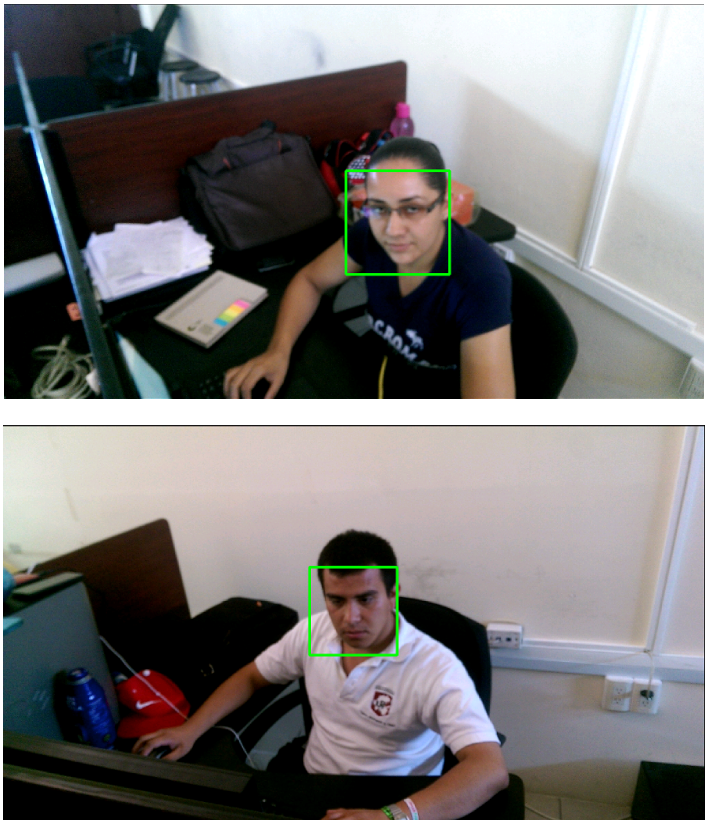


Figura 3.8: **Ejemplos de detección de rostros utilizando OPENCV para Android.**

El truco principal de este algoritmo es que se centraron en ignorar un problema mucho más difícil que es el reconocimiento de rostros y se enfocaron solo en la detección, también se enfocaron en imágenes de rostros vistos de frente. Para formular su algoritmo se dieron cuenta que el puente vertical de la nariz es más brillante que los ojos, y que la región de los ojos es más oscura que la región de las mejillas superiores, así que su algoritmo busca primero banda luminosas verticales, las cuales podrían ser narices, luego busca bandas obscuras horizontales que podrían ser los ojos y después busca otros patrones generales que pueden estar asociados a la cara. Detectadas por si mismas ninguna de las características anteriores podría decirnos que existe una cara en la imagen, pero si se detectan una después de la otra entonces esto nos da una buena indicación de que en la imagen hay un rostro presente. Y como estas pruebas son muy fáciles de realizar el algoritmo puede trabajar rápidamente en tiempo real [4].

En la Figura 3.9 se muestran los resultados obtenidos al aplicar el algoritmo de Viola-Jones a algunas de las imágenes de prueba, utilizando la implementacion del algoritmo de Matlab, la cual es bastante rapida y eficiente.

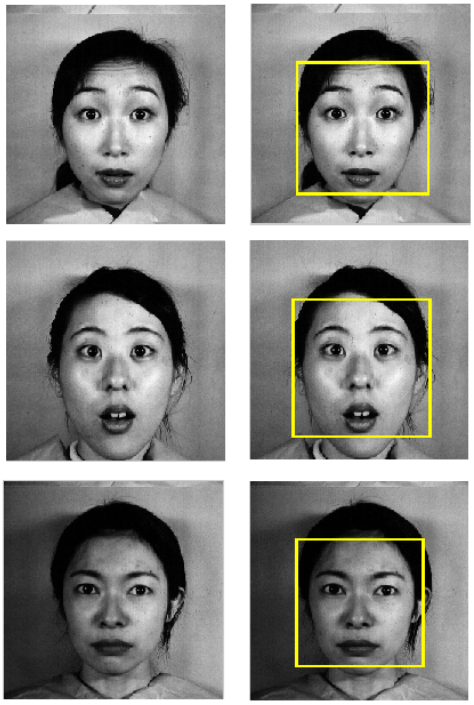


Figura 3.9: **Resultado de aplicar el algoritmo de Viola-Jones a algunas imágenes de la base de datos JAFEE.**

3.4.2. Procesado de la imagen

La función de las técnicas de procesado es mejorar las propiedades espaciales y en intensidades de una imagen, para que en las etapas siguientes como extracción de características o la clasificación, éstas tengan un mejor desempeño. Entre los métodos de procesado se encuentran: realce del contraste, eliminación de ruido, detección de bordes, etc.

La técnica propuesta hace uso del realce de contraste, con este se pretende aumentar el contaste de la imagen, las principales causas por lo que se aplican estos métodos es la falta de iluminación uniforme. El método utilizado para el realce de contraste es el $sharperning$ [6], este método hace que los bordes aparezcan más definidos oscureciendo los pixeles más oscuros y abrillantando los pixeles más brillantes, esto crea un borde nítido entre la luz y las zonas oscuras de la imagen, dándole más contraste, ver Figura 3.10.

Figura 3.10: **Imagen original (izquierda), imagen realzada con** $sharpening$ **(derecha).**

Después de aplicar el metodo de $sharpening$, tendremos una imagen con los bordes más definidos, lo cual es una ventaja, ya que la técnica propuesta, está basada en obtener una imagen binaria donde se permitan identificar los contornos de ojos, cejas, nariz y boca, para esto se realiza una binarización, la cual consiste en reducir la información de una imagen a dos valores: negro y blanco, este procedimiento va comparando el valor de cada pixel de la imagen con un umbral, dependiendo del valor del pixel respecto al umbral (más grande o mas pequeño) se dará un nuevo valor a la nueva imagen (blanco o negro). En la Figura 3.11 podemos observar el resultado de aplicar la binarización a la imagen después del aumento de contraste.

Figura 3.11: **Binarización a la imagen de la Figura 3.10**

La definición matemática de la binarización es:

$$g(x,y) = \begin{cases} 1 \text{ si } f(x,y) > T \\ 0 \text{ si } f(x,y) \leq T \end{cases} \tag{3.4.1}$$

Para calcular el valor del umbral T se puede usar el siguiente algoritmo iterativo [7]:

1. Se propone un valor para T.
2. Se calcula la media de los valores por debajo de T (m_1) y sobre T (m_2).
3. Se calcula el nuevo valor del umbral $T = \frac{1}{2}(m_1 + m_2)$
4. Se repite el paso dos hasta que el valor del nuevo umbral sea igual al valor del umbral anterior.

3.4.3. Extracción de Características

Para este etapa de la técnica propuesta, se necesita una imagen como la de la Figura 3.11, a esta imagen se le aplicaran las técnicas descritas en el capítulo 2 sobre la extracción de características. Se utilizan los momentos ortogonales de Zernike y los momentos de Jacobi-Fourier como descriptores de la imagen, el objetivo en esta etapa es que a partir de la imagen, la cual puede tener un procesamiento, obtengamos un vector de momentos (descriptores) los cuales caractericen a la imagen. Este vector de descriptores se le conoce como patrón.

Esta etapa es muy importante dentro de toda la técnica, ya que de está depende el rendimiento del clasificador a utilizar, si en esta etapa los patrones obtenidos no sirven para caracterizar correctamente a las imágenes, por más que tengamos un clasificador muy bueno los resultados no van a ser correctos.

Figura 3.12: **En la etapa de extracción de características a partir de la imagen binaria obtenemos un vector de descriptores.**

3.4.4. Entrenamiento y clasificación

Estas etapas tienen como objetivo clasificar una imagen, es decir asignarla a una emoción de las disponibles. Para poder clasificar la imagen dentro de alguna emoción es necesario definir fronteras para cada emoción, en la mayoría de los casos las fronteras entre emociones (clases en general) se calculan mediante la etapa de entrenamiento, en el cual el clasificador usa los patrones de clases conocidas para formar los prototipos de cada clase. Terminado el entrenamiento para la clasificación de las emociones desconocidas el clasificador asignará una emoción a cada imagen de entrada, en la cual las características usadas en el entrenamiento son las más parecidas con las características de la imagen.

Para la técnica se propone el uso de tres diferentes clasificadores: Vecinos más cercanos KNN, Adaboost y un clasificador discriminante. Para cada uno de estos clasificadores la primera etapa es el entrenamiento, para lo cual se utilizan patrones de imágenes las cuales se conoce la emoción presente en la rostro de cada una. El entrenamiento es supervisado. Cuando el clasificador ha sido entrenado con la información suficiente, entonces se puede pasar a la etapa de clasificación que es la etapa final de la técnica y donde como entrada se tiene un vector de descriptores que caracteriza una imagen, y la salida será la emoción presente en el rostro de la imagen, suponiendo que existe un rostro.

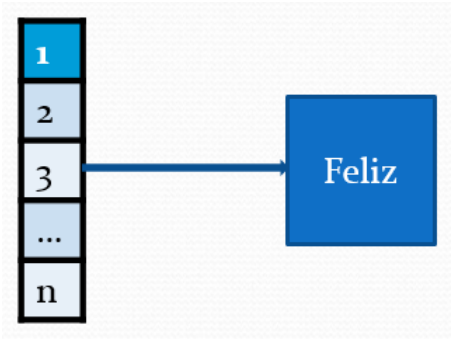


Figura 3.13: **A partir de un vector de descriptores, un clasificador debería ser capaz de reconocer la emoción.**

3.5. Software en Matlab

Se desarrolló en Matlab un software en el cual se conjuntan todos los pasos de la sección 3.4, con la intención de evaluar la técnica propuesta. En la Figura 3.14 se muestra el esquema de los procesos realizados por este software.

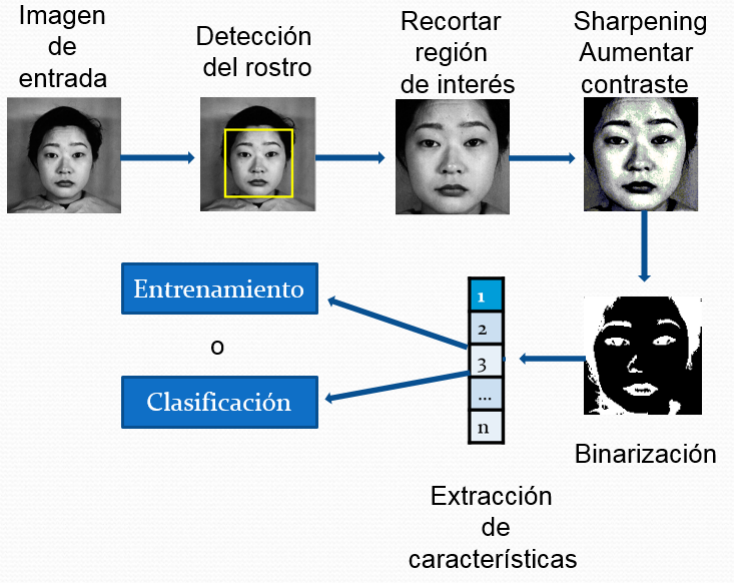


Figura 3.14: **Diagrama de la secuencia de operaciones del software desarrollado.**

El software permite al usuario seleccionar y entrenar un clasificador de los disponibles (KNN, AdaBoost y Discriminante), para esto permite abrir N cantidad de imágenes guardadas en el disco duro o en algún dispositivo externo conectado a la computadora donde se ejecuta el software, a cada una de estas imágenes se aplicará todo el proceso mostrado en la Figura 3.14. En la Figura 3.15 se muestran las opciones principales al que el usuario tiene acceso en el software.

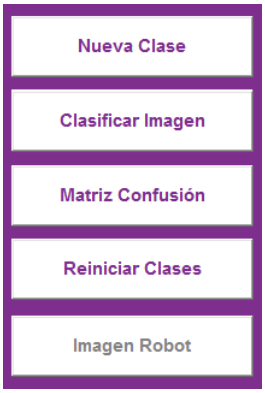


Figura 3.15: **Opciones principales de la interfaz usuario del software desarrollado.**

Nueva Clase: Esta opción permite al usuario generar los vectores descriptores de N cantidad de imágenes, los cuales servirán para entrenar alguno de los clasificadores disponibles. Al seleccionar la opción se le pedirá al usuario ingresar el nombre de la clase es decir el nombre de la emoción, ver Figura 3.16. Seguido se abrirá una ventana donde se permitirá seleccionar las imágenes pertenecientes a esa emoción, ver Figura 3.17. A cada una de las imágenes seleccionadas se le realizaran los procesos de detección del rostro, procesamiento, y extracción de características hasta generar su vector descriptor el cual es almacenado en un archivo, para su utilización en el entrenamiento de algún clasificador.

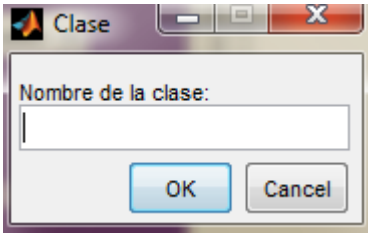


Figura 3.16: **Ventana para nombrar la clase a entrenar dentro del software.**

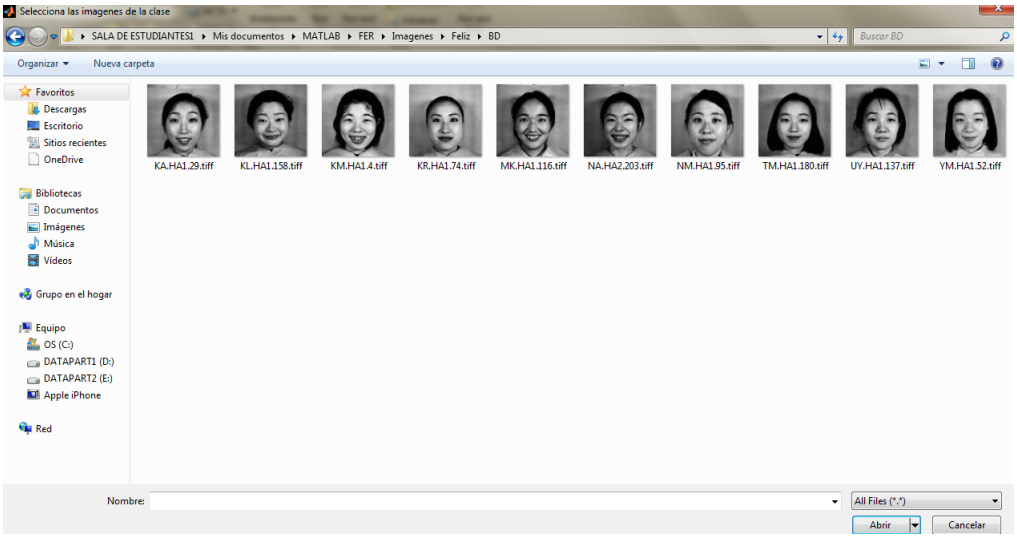


Figura 3.17: **Ventana para seleccionar las imágenes de X clase en la etapa extracción de características del software.**

Cuando se carguen los datos para tres emociones diferentes, el software permitirá al usuario la selección de alguno de los clasificadores disponibles, ver Figura 3.18. Al seleccionar alguno de los clasificadores se realizara el proceso de entrenamiento, usando los datos de las clases cargados anteriormente mediante la opción **Nueva Clase**. Para la clasificación se requiere haber cargado datos de al menos tres clases diferentes y haber seleccionado el clasificador a usar. Cabe mencionar que al cambiar de clasificador no es necesario cargar los datos de las clases nuevamente, ya que estos se encuentran almacenados en un archivo el cual el software puede consultar en cualquier momento.

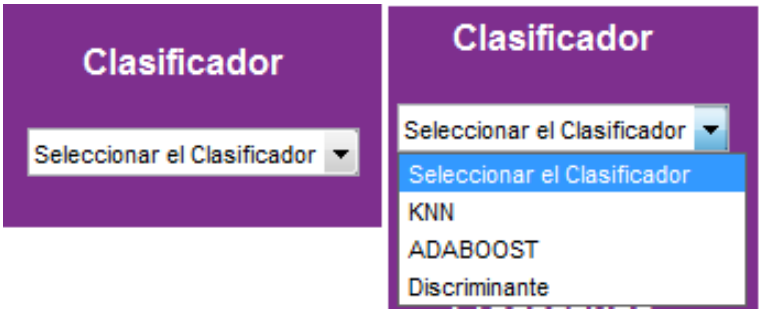


Figura 3.18: **Parte de la interfaz de usuario donde se realiza la selección del clasificador en el software.**

Clasificar Imagen: esta opción permite al usuario seleccionar un conjunto de imágenes guardadas en la computadora donde se esté ejecutando el software, para esto se abrirá una ventana parecida a la de la Figura 3.17 donde el usuario podrá elegir las imágenes a clasificar, a cada una de las imágenes seleccionadas se aplicarán los procesos de detección del rostro, procesamiento y extracción de características hasta obtener el vector descriptor o patrón, el cual se meterá al clasificador seleccionado para que este clasifique la imagen y la asigne a alguna de las emociones con las que fue entrenado. El resultado será mostrado en una pequeña ventana, donde para cada una de las imágenes seleccionadas se mostrará la emoción clasificada, ver Figura 3.19.

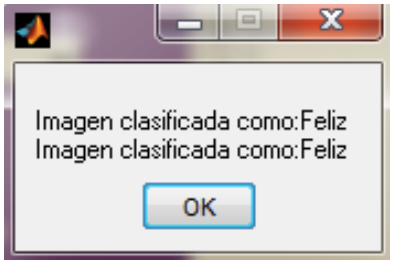


Figura 3.19: **Ejemplo del resultado de clasificación para 2 imágenes.**

Matriz de Confusión: permite generar la matriz de confusión con los resultados de clasificación, ver Figura 3.20. Esta es una matriz de $N \times N$ donde N es el número de clases con el que se entrenó el clasificador.

	Enojado	Feliz	Sorpresa
Enojado	10	0	0
Feliz	0	10	0
Sorpresa	0	0	10

Figura 3.20: **Matriz de confusión generada por el software.**

Acerca de la matriz de confusión [8] se puede decir que ordena todos los casos en clases, determinando si el valor de la predicción coincide con el valor real, para esto se cuentan todos los casos de cada clase y los totales se muestran en la matriz. Esta matriz es una herramienta estándar de evaluación de modelos estadísticos. La matriz está formada por tanto filas y columnas como clases hay. El número de instancias clasificadas correctamente es la suma de la diagonal principal, y el resto son instancias clasificadas incorrectamente.

Reiniciar Clases: esta opción permite al usuario eliminar permanentemente todos los datos de las clases de entrenamiento generadas mediante la opción **Nueva Clase**. Cuando se active está opción y el usuario requiera clasificar alguna imagen, esté deberá volver a cargar los datos de las clases y seleccionar nuevamente el clasificador a entrenar.

Imagen Robot: esta opción solo estará disponible cuando al iniciar el software, esté detecta al robot NAO, para esto se creó una interfaz entre el software y el robot, la cual permite una conexión del robot con el software. Si la detección del robot es exitosa entonces el software permitirá al usuario mandar una instrucción al robot con la cual éste capturará una imagen con alguna de las cámaras con las que cuenta. El robot enviara la imagen al software, en donde se procesará y se clasificará, el resultado será regresado al robot, el cual mediante la API de voz con la que cuenta ”dirá”la emoción detectada. Para que la imagen capturada con el robot se pueda clasificar es necesario, haber seleccionado algún clasificador de los disponibles, así como haberlo entrenado con al menos tres clases diferentes.

3.5.1. Funciones de Matlab

Durante el desarrollo del software en Matlab se utilizarón distintas funciones para cada etapa de la técnica propuesta, en esta sección se explican las funciones utilizadas más importantes así como su implementación.

Detección de Rostro

Para la detección del rostro en la imagen se utilizó la función *CascadeObjectDetector* [9] de Matlab, la cual usa el algoritmo de Viola-Jones para detectar caras humanas, nariz, ojos, boca o la parte superior del cuerpo. Para detectar alguna característica con esta función se tiene que:

1. Definir y configurar el $CascadeObjectDetector$ utilizando el constructor.

2. Llamar a la función *step* pasando como parámetros el $CascadeObjectDetector$ y la imagen de entrada I. Esta función devolverá una matriz de $Mx4$ donde M son las características encontradas, y para cada una de estas características las coordenadas (x, y) que delimitan un cuadro sobre cada una de ellas.

Ejemplo para la detección de rostro en una imagen I:

```
faceDetect = vision.CascadeObjectDetector();
bbox = step(faceDetect,I);
```

Si el detector `faceDetect` encuentra algún rostro en la imagen I entonces `bbox` contendrá un par de coordenadas (x, y) para formar un cuadrado que delimite el rostro, ver Figura 3.21.

```
bbox =

    51    67   160   160
```

Figura 3.21: **Ejemplo de la respuesta de CascadeObjectDetector, sobre una imagen de prueba.**

La matriz de coordenadas `bbox` nos sirve para delimitar la región de interés (ROI) es decir las características encontradas por el $CascadeObjectDetector$, esta matriz la podemos usar para delimitar la ROI o para recortarla, ver Figura 3.22

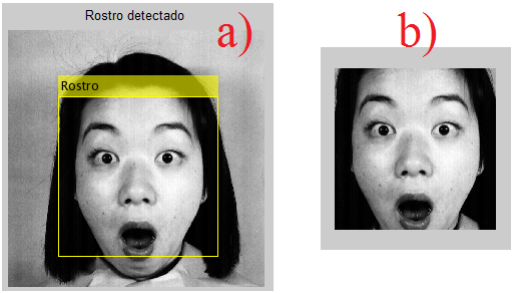


Figura 3.22: **a) Delimitar o b) recortar la región de interés con el** `bbox`**.**

El software realiza el corte de la ROI, para esto utiliza la función `imcrop` [10]. Esta función recibe como parámetro la imagen a recortar y una matriz con las coordenadas de la zona a recortar, para este caso el `bbox`, ejemplo:

```
Irecortada = imcrop(I,bbox);
```

Procesado

El tamaño de la ROI puede variar dependiendo de la imagen de entrada, por esto teniendo la imagen recortada de la ROI (`Irecortada`), el software realiza un ajuste de tamaño de la imagen, esto con el objetivo de que todas las imágenes tengan el mismo tamaño, para esto se utiliza la función `imresize` [11], esta función recibe como parámetros la imagen a ajustar y un vector de dos posiciones con el tamaño de la imagen de salida `[filas, columnas]`, ejemplo:

```
Iajustada = imresize(Irecortada,[256 256]);
```

Con la línea anterior el software ajustara la imagen `Irecortada` a un tamaño de $256x256$.

Ahora que se tiene una imagen ajustada a un cierto tamaño, el software realiza el realce de contraste usando la función `imsharpen` [12], esta función en su versión más simple recibe como parámetro la imagen a mejorar y devuelve una imagen mejorada de la imagen de entrada, donde las características como los bordes se hacen más nítidos. Esta función utiliza la técnica $unsharpmask$, la cual proviene de un proceso editorial en el cual una imagen es mejorada restando una versión borrosa de la misma. La siguiente línea es un ejemplo de la versión más simple de esta función:

```
Imejorada = imsharpen(Iajustada);
```

`Imejorada` contiene una versión mejorada de la imagen `Iajustada`. Al terminar este proceso de mejora, el software realiza la binarización de la imagen, para esto se utiliza la definición matemática de la binarización de la ecuación 3.4.1, el siguiente codigo es un ejemplo de la binarización de la imagen `Imejorada`:

```
%Calcular el umbral T
T = 100;
m1 = mean(mean((Imejorada <T).*Imejorada));
m2 = mean(mean((Imejorada >= T).*Imejorada));
Ttemp = 1/2*(m1 + m2);
while T ~= Ttemp
     T = Ttemp;
     m1 = mean(mean((Imejorada <T).*Imejorada));
     m2 = mean(mean((Imejorada >= T).*Imejorada));
     Ttemp = 1/2*(m1 + m2);
end

%Binarizar la imagen Imejorada
Ibinarizada = Imejorada >T;
```

La función `mean` [13] calcula el promedio del vector que recibe como parámetro. La variable `Ibinarizada` contiene la versión binaria de la imagen `Imejorada`.

Extracción de características

Cuando el software termina la binarización, es decir se tiene la imagen `Ibinarizada`, es momento de la extracción de características. Para esto se programaron los momentos de Zernike y Jacobi-Fourier. Todo el proceso para generar el vector descriptor de cada imagen está divido en varias partes, la primera de estas es llevar a la imagen a coordenadas polares, para esto se utiliza el siguiente código:

```
[M,N] = size(Ibinarizada);
x = 1:M; y = 1:N;
[X,Y] = meshgrid(x,y);
R = sqrt((2.*X-M-1).^2+(2.*Y-N-1).^2)/N;
Theta = atan2((N-12.*Y+2),(2.*X-N+1-2));
R = (R<=1).*R;
```

La variable `R` contiene una matriz de radios y `Theta` una matriz de ángulos, correspondientes a la imagen `Ibinarizada`. Para el caso de los momentos de Zernike el siguiente paso es calcular los polinomios radiales, para esto se emplea el código siguiente:

```
rad = zeros(size(R));
for s = 0:(n-abs(l))/2
    c = (-1)^s*factorial(1+n-s)/(factorial(1+s)*...
    factorial(1+((n+abs(l))/2-s))*...
    factorial(1+((n-abs(l))/2-s)));
    rad = rad + c*R.^(n-2*s);
end
```

Del código anterior `n, l` representan el orden y repetición respectivamente de los momentos de Zernike. La variable `rad` contiene el polinomio radial de Zernike. El siguiente paso es calcular los momentos de Zernike para esto se utiliza el siguiente código:

```
P = Ibinarizada(x,y).*rad.*exp(1i*l*Theta);
Z = sum(P(:));
Z = (n+1)*Z/pi;
Zm = abs(Z);
```

La variable `Zm` contiene el valor del momento de Zernike de orden `n` y repetición `l`. Haciendo que `n, l` tomen diferentes valores podemos obtener una matriz o un vector de momentos, para el caso del software se calculan los momentos de Zernike para orden y repetición de 0 a 20.

En los códigos anteriores se utilizan algunas funciones como `size` [14] la cual devuelve el tamaño de un vector o matriz, `zeros` [15] la cual genera una matriz de ceros, `abs` [16] que calcula el valor absoluto, `sqrt` [17] calcula la raíz cuadrada, `atan2` [18] calcula la tangente inversa, `sum` [19] hace la sumatoria de los valores de un vector.

Clasificadores

Matlab cuenta con la implementación de varios clasificadores, incluidos KNN, Adaboost y el clasificador discriminante. Empezando por el clasificador KNN, la función `fitcknn(X,Y)` [20] regresa un modelo de clasificación basado en las variables de entrada X y las de salida (respuesta) Y.

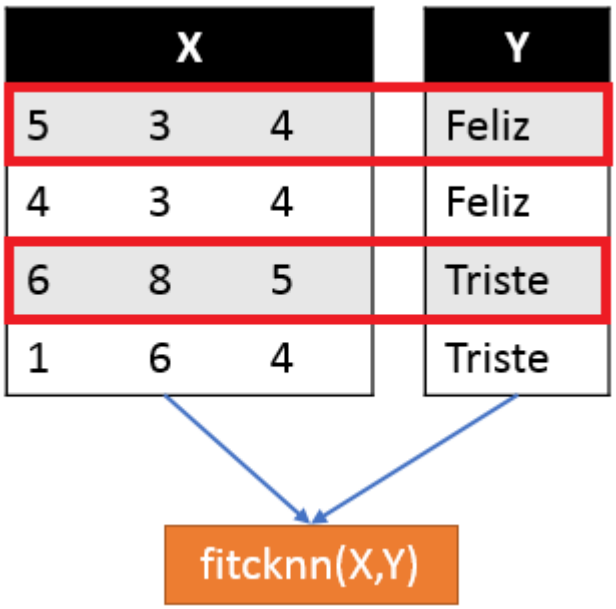


Figura 3.23: **Forma de los parametros que recibe la funcion fitcknn(X,Y)**

La función `fitensemble(X,Y,Method,NLearn,Learners)` [21] crea un modelo ensamblado para predecir una respuesta con los datos de entrada. X, Y pueden tener la misma forma que los que se utilizan en `fitcknn(X,Y)`, ver Figura 3.23. `Method` es el clasificador a usar, en el caso del software desarrollado se está utilizando $AdaBoostM2$ que es la implementación del algoritmo AdaBoost para clasificar más de dos clases diferentes. `Nlearn` es un número entero positivo que representa los ciclos de aprendizaje del clasificador. `Learners` es el tipo de clasificadores débiles que se entrenaran para generar un clasificador con mejor rendimiento, para el caso del software se está utilizando $Tree$.

El siguiente código muestra un ejemplo de implementación de la función `fitensemble`, para crear un modelo de clasificación de la función lógica AND, generando un modelo basado en el algoritmo $AdaBoostM1$ el cual permite clasificar dos clases diferentes, entrenando clasificadores débiles del tipo Discriminante:

```
X = [0 0; 0 1; 1 0; 1 1];
Y = [0; 0; 0; 1];
FAND = fitensemble(X,Y,'AdaBoostM1',100,'Discriminant');
```

Para el caso del clasificador discriminante la función `fitcdiscr(X,Y)` [22] regresa un análisis discriminante basado en las variables de entrada X y la salida Y, estas variables pueden tener la forma de las usadas por la función `fitcknn(X,Y)`, ver Figura 3.23.

Para la clasificación de emociones en el software se está usando la función `predict(obj,X)` [23], donde obj es el clasificador construido con alguna de las funciones anteriores (`fitcknn, fitensemble, fitcdiscr`) y X es una matriz donde cada fila representa una observación a clasificar. El siguiente código muestra un ejemplo para clasificar el vector [0 0] usando el clasificador generado en el código anterior `FAND`:

```
predict(FAND,[0 0]);
```

La respuesta generada con la línea de código anterior es: 0.

Bibliografía

[1] Michael J. Lyons, Shigeru Akemastu, Miyuki Kamachi, Jiro Gyoba, *Coding Facial Expressions with Gabor Wavelets*, 3rd IEEE International Conference on Automatic Face and Gesture Recognition, pp. 200-205 (1998).

[2] *NAO - Video camera* Disponible en: http://doc.aldebaran.com/2-1/family/robots/video_robot.html

[3] Viola, Paul A. and Jones, Michael J. *Rapid Object Detection using a Boosted Cascade of Simple Features*, IEEE CVPR, (2001).

[4] *The Face Detection Algorithm Set to Revolutionize Image Search* Disponible en: https://www.technologyreview.com/s/535201/the-face-detection-algorithm-set-to-revolutionize-image-search

[5] *vision.CascadeObjectDetector System object* Disponible en: http://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html

[6] Jeff Schewe, Jeff Schewe, *Real World Image Sharpening with Adobe Photoshop, Camera Raw, and Lightroom*, Editorial Peachpit Press, ISBN:0321637550 9780321637550, (2009).

[7] Daniel Nieto Bonilla, *Sistema de reconocimiento de kanjis japoneses basado en procesamiento digital de imágenes aplicado a dispositivos móviles*, Tesis profesional, Universidad de las Américas Puebla, (2010)

[8] MSDN Library, *Matriz de clasificación (Analysis Services - Minería de datos)*, Disponible en: https://msdn.microsoft.com/es-es/library/ms174811.aspx

[9] Matlab Documentation, *vision.CascadeObjectDetector System object*, Disponible en: http://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html

[10] Matlab Documentation, *imcrop*, Disponible en: http://www.mathworks.com/help/images/ref/imcrop.html

[11] Matlab Documentation, *imresize*, Disponible en: http://www.mathworks.com/help/images/ref/imresize.html

[12] Matlab Documentation, *imsharpen*, Disponible en: http://www.mathworks.com/help/images/ref/imsharpen.html

[13] Matlab Documentation, *mean*, Disponible en: http://www.mathworks.com/help/matlab/ref/mean.html

[14] Matlab Documentation, *size*, Disponible en: http://www.mathworks.com/help/matlab/ref/size.html

[15] Matlab Documentation, *zeros*, Disponible en: http://www.mathworks.com/help/matlab/ref/zeros.html

[16] Matlab Documentation, *abs*, Disponible en: http://www.mathworks.com/help/matlab/ref/abs.html

[17] Matlab Documentation, *sqrt*, Disponible en: http://www.mathworks.com/help/matlab/ref/sqrt.html

[18] Matlab Documentation, *atan2*, Disponible en: http://www.mathworks.com/help/matlab/ref/atan2.html

[19] Matlab Documentation, *sum*, Disponible en: http://www.mathworks.com/help/matlab/ref/sum.html

[20] Matlab Documentation, *fitcknn*, Disponible en: http://www.mathworks.com/help/stats/fitcknn.html

[21] Matlab Documentation, *fitensemble*, Disponible en: http://www.mathworks.com/help/stats/fitensemble.html

[22] Matlab Documentation, *fitcdiscr*, Disponible en: http://www.mathworks.com/help/stats/fitcdiscr.html

[23] Matlab Documentation, *predict*, Disponible en: http://www.mathworks.com/help/search.html?qdoc=predict

Capítulo 4

Resultados y conclusiones

4.1. Introducción

En este capítulo se presentan los resultados principales obtenidos al evaluar la técnica propuesta para el reconocimiento de emociones en imágenes digitales. Estos resultados corresponden a la evaluación de la técnica mediante el uso del software desarrollado en Matlab, se realizaron ocho pruebas principales, cuatro usando los momentos de Zernike y cuatro usando los momentos de Jacobi-Fourier, como descriptores de las imágenes, y clasificando en cada una de estas pruebas con los clasificadores KNN, AdaBoost y Lineal Discriminante. Para cada una de las ocho pruebas se presentas tablas con los parámetros de las pruebas, las matrices de confusión con los resultados obtenidos y un análisis de estos resultados. Al final del capítulo se dan las conclusiones sobre todo el trabajo realizado y se presenta el trabajo a futuro.

4.2. Resultados con descriptores basados en momentos de Zernike

Como se mencionó en el tercer capítulo de estas Tesis, se desarrolló un software en Matlab con el objetivo de evaluar el rendimiento de las técnica propuesta usando como imágenes de entrada las procedentes de la base de datos JAFFE y como descriptores los momentos de Zernike y los momentos de Jacobi-Fourier, se evaluó la eficiencia de tres clasificadores KNN, AdaBoost y Lineal Discriminante, todo esto con el objetivo de determinar cuáles son los descriptores y clasificadores más adecuados a utilizar para tener un desempeño eficiente a la hora de reconocer emociones en imágenes digitales utilizando esta técnica propuesta.

4.2.1. Clasificación de 3 emociones

En una primera prueba realizada, se evalúa la técnica para la clasificación de tres emociones: enojo, felicidad y sorpresa, usando como descriptores los momentos de Zernike y clasificando con los tres clasificadores mencionados anteriormente. En el Tabla 4.1 se muestran los principales parámetros con los que se realizó esta prueba, en las siguientes tres Tablas 4.2, 4.3, 4.4 se muestran las matrices de confusión con los resultados obtenidos para cada uno de los cla-

sificadores. Para todas las pruebas realizadas se sigue este mismo concepto de descripción de parámetros de la prueba así como sus respectivas matrices de confusión.

Tabla 4.1: Descripción de los parámetros de la prueba 1.

Prueba	**1**
Descriptores	Momentos de Zernike
Emociones a clasificar	enojo felicidad sorpresa
Imágenes Totales	91
No. de imágenes a clasificar	enojo: 30 felicidad: 31 sorpresa: 30
No. de imágenes de entrenamiento	10 por emoción

Tabla 4.2: Matriz de confusión con los resultados de la primera prueba usando el clasificador KNN

	Enojo	**Felicidad**	**Sorpresa**
Enojo	26	4	0
Felicidad	5	24	2
Sorpresa	4	5	21

Tabla 4.3: Matriz de confusión con los resultados de la primera prueba usando el clasificador AdaBoost

	Enojo	**Felicidad**	**Sorpresa**
Enojo	25	4	1
Felicidad	1	25	5
Sorpresa	0	1	29

Tabla 4.4: Matriz de confusión con los resultados de la primera prueba usando el clasificador Discriminante

	Enojo	**Felicidad**	**Sorpresa**
Enojo	30	0	0
Felicidad	0	30	1
Sorpresa	0	0	30

Análisis de resultados de la prueba 1.

En la prueba 1 se clasificarón 91 imágenes diferentes pertenecientes a 3 emociones: enojo, felicidad y sorpresa. Los descriptores utilizados fueron los momentos de Zernike, para el entrenamiento de los 3 clasificadores se utilizaron 30 imágenes (10 imágenes por emoción). En el Tabla 4.5 se muestran los porcentajes de clasificación correcta para cada uno de los tres clasificadores.

Tabla 4.5: Porcentajes de clasificación correcta de la prueba 1

Clasificador	**Porcentaje de clasificación correcta**
KNN	78.02 %
Adaboost	86.81 %
Discriminante	98.9 %

Como se puede observar en el Tabla 4.5, el clasificador Discriminante es el que mayor porcentaje de clasificación correcta presenta con un 98.9 %, lo que representa que solo una de 91 imágenes clasificadas fue incorrecta. Los otros dos clasificadores KNN y Adaboost presentan una clasificación de 78.02 % y 86.81 % respectivamente lo que representa que acertaron en 71 y 79 imágenes de 91 totales. Los resultados obtenidos en esta primera prueba muestran la eficacia del clasificador discriminante que en conjunto con los momentos de Zernike lograron casi el 100 % de clasificación correcta.

4.2.2. Clasificación de 4 emociones

Para la segunda prueba se aumentó en uno las emociones a reconocer, ahora los clasificadores deberían reconocer entre enojo, felicidad, sorpresa y miedo. En el Tabla 4.6 se pueden observar las especificaciones para esta prueba.

Tabla 4.6: Descripción de los parámetros de la prueba 2.

Prueba	**2**
Descriptores	Momentos de Zernike
Emociones a clasificar	enojo felicidad sorpresa miedo
Imágenes Totales	123
No. de imágenes a clasificar	enojo: 30 felicidad: 31 sorpresa: 30 miedo:32
No. de imágenes de entrenamiento	10 por emoción

Tabla 4.7: Matriz de confusión con los resultados de la segunda prueba usando el clasificador KNN

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**
Enojo	27	0	0	3
Felicidad	5	19	2	5
Sorpresa	2	2	18	8
Miedo	6	1	3	22

Tabla 4.8: Matriz de confusión con los resultados de la segunda prueba usando el clasificador Adaboost

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**
Enojo	20	5	1	4
Felicidad	0	21	5	5
Sorpresa	0	1	24	5
Miedo	0	6	3	23

Tabla 4.9: Matriz de confusión con los resultados de la segunda prueba usando el clasificador Discriminante

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**
Enojo	28	1	0	1
Felicidad	0	29	1	1
Sorpresa	0	0	28	2
Miedo	0	0	2	30

Análisis de resultados de la prueba 2.

Para la prueba 2 se clasificaron 123 imágenes diferentes pertenecientes a 4 emociones: enojo, felicidad, sorpresa y miedo, los descriptores utilizados fueron los momentos de Zernike y para el entrenamiento de los 3 clasificadores se utilizaron 40 imágenes (10 imágenes por emoción). En el Tabla 4.10 se muestran los porcentajes de clasificación correcta para cada uno de los tres clasificadores.

Tabla 4.10: Porcentajes de clasificación correcta de la prueba 2

Clasificador	**Porcentaje de clasificación correcta**
KNN	69.91 %
Adaboost	71.54 %
Discriminante	93.49 %

En el Tabla 4.10 se puede observar nuevamente que el clasificador Discriminante es el que presenta mayor porcentaje de clasificación correcta con un 93.49 %, pero comparado con la

prueba 1 el porcentaje ha disminuido casi 6 %, se debe a si aumenta el número de emociones a clasificar el problema se vuelve más complejo. En cuanto a KNN y Adaboost el porcentaje de clasificación correcta también disminuyó. En general, con el clasificador Discriminante más los momentos de Zernike como descriptores, los resultados de clasificación correcta son muy buenos para diferenciar entre 4 emociones diferentes.

4.2.3. Clasificación de 5 emociones

Para una tercera prueba se agregó a la clasificación la emoción de tristeza, esto con la intención de aumentar la complejidad del problema a la hora de la clasificación. En el Tabla 4.11 se pueden observar los parámetros con los que se realizó la tercera prueba.

Tabla 4.11: Descripción de los parámetros de la prueba 3.

Prueba	**3**
Descriptores	Momentos de Zernike
Emociones a clasificar	enojo felicidad sorpresa miedo tristeza
Imágenes Totales	154
No. de imágenes a clasificar	enojo: 30 felicidad: 31 sorpresa: 30 miedo:32 tristeza:31
No. de imágenes de entrenamiento	10 por emoción

Tabla 4.12: Matriz de confusión con los resultados de la tercera prueba usando el clasificador KNN

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**
Enojo	25	0	1	3	1
Felicidad	0	24	0	4	3
Sorpresa	1	1	25	2	1
Miedo	3	0	3	23	3
Tristeza	0	3	3	2	30

Tabla 4.13: Matriz de confusión con los resultados de la tercera prueba usando el clasificador Adaboost

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**
Enojo	26	0	0	3	1
Felicidad	1	24	2	3	1
Sorpresa	1	2	24	3	0
Miedo	2	1	2	25	2
Tristeza	0	2	1	2	26

Tabla 4.14: Matriz de confusión con los resultados de la tercera prueba usando el clasificador Discriminante

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**
Enojo	28	0	0	1	1
Felicidad	0	29	0	2	0
Sorpresa	0	1	27	1	1
Miedo	1	0	1	29	1
Tristeza	0	1	2	1	27

Análisis de resultados de la prueba 3.

En la prueba 3 se clasificaron 154 imágenes diferentes pertenecientes a 5 emociones: enojo, felicidad, sorpresa, miedo y tristeza. Los descriptores utilizados fueron los momentos de Zernike y para el entrenamiento de los 3 clasificadores se utilizaron 50 imágenes (10 imágenes por emoción). En el Tabla 4.15 se muestran los porcentajes de clasificación correcta para cada uno de los tres clasificadores.

Tabla 4.15: Porcentajes de clasificación correcta de la prueba 3

Clasificador	**Porcentaje de clasificación correcta**
KNN	77.92 %
Adaboost	81.16 %
Discriminante	90.90 %

El Tabla 4.15 muestra que para la tercera prueba el clasificador Discriminante es el que presenta mayor porcentaje de clasificación correcta con un 90.90 %, el clasificador Adaboost mejoró el porcentaje de clasificación respecto a la segunda prueba con un 81.16 % al igual que el KNN con un 77.92 %.

4.2.4. Clasificación de 6 emociones

Para una cuarta prueba se agregó a la clasificación una emoción neutral, en el Tabla 4.16 se pueden observar los parámetros con los que se realizó la cuarta prueba.

Tabla 4.16: Descripción de los parámetros de la prueba 4.

Prueba	**4**
Descriptores	Momentos de Zernike
Emociones a clasificar	enojo, felicidad, sorpresa, miedo, tristeza, neutral
Imágenes Totales	185
No. de imágenes a clasificar	enojo: 30, felicidad: 31, sorpresa: 30, miedo:32, tristeza:31, neutral:31
No. de imágenes de entrenamiento	10 por emoción

Tabla 4.17: Matriz de confusión con los resultados de la cuarta prueba usando el clasificador KNN

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**	**Neutral**
Enojo	20	1	1	2	2	4
Felicidad	4	17	3	2	1	4
Sorpresa	0	7	15	8	0	0
Miedo	5	2	4	18	0	3
Tristeza	6	4	0	2	12	7
Neutral	3	5	2	1	1	19

Tabla 4.18: Matriz de confusión con los resultados de la cuarta prueba usando el clasificador AdaBoost

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**	**Neutral**
Enojo	18	3	3	1	5	0
Felicidad	0	18	7	0	6	0
Sorpresa	0	4	27	0	0	0
Miedo	0	3	7	14	7	1
Tristeza	0	2	2	3	24	0
Neutral	0	6	1	2	6	16

Tabla 4.19: Matriz de confusión con los resultados de la cuarta prueba usando el clasificador Discriminante

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**	**Neutral**
Enojo	27	0	0	1	2	0
Felicidad	0	26	2	0	2	1
Sorpresa	0	1	27	1	0	1
Miedo	0	1	1	27	1	2
Tristeza	0	1	1	1	26	2
Neutral	0	1	2	1	2	25

Análisis de resultados de la prueba 4.

En la prueba 4 se clasificaron 185 imágenes diferentes pertenecientes a 6 emociones: enojo, felicidad, sorpresa, miedo, tristeza y neutral. Los descriptores utilizados fueron los momentos de Zernike y para el entrenamiento de los 3 clasificadores se utilizaron 60 imágenes (10 imágenes por emoción). En el Tabla 4.20 se muestran los porcentajes de clasificación correcta para cada uno de los tres clasificadores.

Tabla 4.20: Porcentajes de clasificación correcta de la prueba 4

Clasificador	**Porcentaje de clasificación correcta**
KNN	54.59 %
Adaboost	63.24 %
Discriminante	85.40 %

En el Tabla 4.20 se puede observar que el mejor clasificador en cuanto a clasificación correcta es el Discriminante con un 85.40 %, los clasificadores AdaBoost y KNN presentan un desempeño bajo con un 63.24 % y 54.59 % respectivamente. Analizando las 4 pruebas realizadas utilizando los momentos de Zernike se puede decir que el clasificador con un mayor porcentaje de clasificación correcta es el Discriminante, el segundo pero con una amplia diferencia está el AdaBoost y por último el que tiene el desempeño más bajo es el clasificador KNN.

4.3. Resultados con descriptores basados en momentos de Jacobi-Fourier

4.3.1. Clasificación de 3 emociones

Para las siguientes 4 pruebas se cambiaron los descriptores, ahora en lugar de usar los momentos de Zernike, se utilizaron los momentos de Jacobi-Fourier. Los datos de la prueba 5 se presentan en el Tabla 4.21.

Tabla 4.21: Descripción de los parámetros de la prueba 5.

Prueba	**5**
Descriptores	Momentos de Jacobi-Fourier
Emociones a clasificar	enojo, felicidad, sorpresa
Imágenes Totales	91
No. de imágenes a clasificar	enojo: 30, felicidad: 31, sorpresa: 30
No. de imágenes de entrenamiento	10 por emoción

Tabla 4.22: Matriz de confusión con los resultados de la quinta prueba usando el clasificador KNN

	Enojo	**Felicidad**	**Sorpresa**
Enojo	24	2	4
Felicidad	6	21	4
Sorpresa	3	1	26

Tabla 4.23: Matriz de confusión con los resultados de la quinta prueba usando el clasificador AdaBoost

	Enojo	**Felicidad**	**Sorpresa**
Enojo	25	2	3
Felicidad	0	29	2
Sorpresa	0	3	27

Tabla 4.24: Matriz de confusión con los resultados de la quinta prueba usando el clasificador Discriminante

	Enojo	**Felicidad**	**Sorpresa**
Enojo	29	1	0
Felicidad	0	31	0
Sorpresa	0	1	29

Análisis de resultados de la prueba 5.

En la prueba 5 se clasificaron 91 imágenes diferentes pertenecientes a 3 emociones: enojo, felicidad y sorpresa. Los descriptores utilizados fueron los momentos de Jacobi-Fourier y para el entrenamiento de los 3 clasificadores se utilizaron 30 imágenes (10 imágenes por emoción). En el Tabla 4.25 se muestran los porcentajes de clasificación correcta para cada uno de los tres clasificadores.

Tabla 4.25: Porcentajes de clasificación correcta de la prueba 5

Clasificador	**Porcentaje de clasificación correcta**
KNN	78.02 %
Adaboost	89.01 %
Discriminante	97.8 %

La prueba 5 muestra que el clasificador Discriminante presenta el mejor porcentaje de clasificación correcta con un 97.8 %. Los clasificadores KNN y Adaboost presentan porcentajes de 78.02 % y 89.01 % respectivamente. El clasificador discriminante sólo fallo en 2 de 91 imágenes totales.

En comparación con la prueba 1 el clasificador Adaboost mejoró el porcentaje de clasificación, en caso contrario el clasificador Discriminante disminuyó, ya que en la prueba 1 solo fallo 1 de las 91 imágenes.

4.3.2. Clasificación de 4 emociones

Para la prueba número 6, se utilizaron los momentos de Jacobi-Fourier como descriptores para tratar de clasificar las emociones enojo, felicidad, sorpresa y miedo, en el Tabla 4.26 se pueden observar las especificaciones completas para esta prueba.

Tabla 4.26: Descripción de los parámetros de la prueba 6.

Prueba	**6**
Descriptores	Momentos de Jacobi-Fourier
Emociones a clasificar	enojo felicidad sorpresa miedo
Imágenes Totales	123
No. de imágenes a clasificar	enojo: 30 felicidad: 31 sorpresa: 30 miedo:32
No. de imágenes de entrenamiento	10 por emoción

Tabla 4.27: Matriz de confusión con los resultados de la sexta prueba usando el clasificador KNN

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**
Enojo	22	3	1	4
Felicidad	2	21	1	7
Sorpresa	2	2	18	8
Miedo	3	3	1	25

Tabla 4.28: Matriz de confusión con los resultados de la sexta prueba usando el clasificador Adaboost

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**
Enojo	20	1	7	2
Felicidad	2	25	4	0
Sorpresa	1	3	26	0
Miedo	7	4	5	16

Tabla 4.29: Matriz de confusión con los resultados de la sexta prueba usando el clasificador Discriminante

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**
Enojo	27	0	2	1
Felicidad	0	29	1	1
Sorpresa	0	1	27	2
Miedo	0	1	2	29

Análisis de resultados de la prueba 6.

Para la prueba 6 se clasificaron 123 imágenes diferentes pertenecientes a 4 emociones: enojo, felicidad, sorpresa y miedo. Los descriptores utilizados fueron los momentos de Jacobi-Fourier y para el entrenamiento de los 3 clasificadores se utilizaron 40 imágenes (10 imágenes por emoción). En el Tabla 4.30 se muestran los porcentajes de clasificación correcta para cada uno de los tres clasificadores. El clasificador Discriminante fue el de mejor desempeño con una clasificación correcta de 91.05 %.

Tabla 4.30: Porcentajes de clasificación correcta de la prueba 6

Clasificador	**Porcentaje de clasificación correcta**
KNN	69.91 %
Adaboost	70.73 %
Discriminante	91.05 %

4.3.3. Clasificación de 5 emociones

Para una séptima prueba se agregó a la clasificación la emoción de tristeza. En el Tabla 4.31 se pueden observar los parámetros con los que se realizó la séptima prueba.

Tabla 4.31: Descripción de los parámetros de la prueba 7.

Prueba	**7**
Descriptores	Momentos de Jacobi-Fourier
Emociones a clasificar	enojo, felicidad, sorpresa, miedo, tristeza
Imágenes Totales	154
No. de imágenes a clasificar	enojo: 30 felicidad: 31 sorpresa: 30 miedo:32 tristeza:31
No. de imágenes de entrenamiento	10 por emoción

Tabla 4.32: Matriz de confusión con los resultados de la séptima prueba usando el clasificador KNN

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**
Enojo	22	1	0	2	5
Felicidad	3	15	0	7	6
Sorpresa	3	1	19	3	4
Miedo	2	2	2	23	3
Tristeza	3	2	1	7	18

Tabla 4.33: Matriz de confusión con los resultados de la séptima prueba usando el clasificador Adaboost

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**
Enojo	22	0	3	3	2
Felicidad	0	19	0	0	12
Sorpresa	0	3	26	0	1
Miedo	0	2	3	18	9
Tristeza	1	4	1	3	22

Tabla 4.34: Matriz de confusión con los resultados de la séptima prueba usando el clasificador Discriminante

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**
Enojo	27	0	0	2	1
Felicidad	1	26	0	1	3
Sorpresa	0	0	28	1	1
Miedo	2	2	1	27	0
Tristeza	1	2	0	2	26

Análisis de resultados de la prueba 7.

En la prueba 7 se clasificarón 154 imágenes diferentes pertenecientes a 5 emociones: enojo, felicidad, sorpresa, miedo y tristeza. Los descriptores utilizados fueron los momentos de Jacobi-Fourier y para el entrenamiento de los 3 clasificadores se utilizaron 50 imágenes (10 imágenes por emoción). En el Tabla 4.15 se muestran los porcentajes de clasificación correcta para cada uno de los tres clasificadores.

Tabla 4.35: Porcentajes de clasificación correcta de la prueba 7

Clasificador	**Porcentaje de clasificación correcta**
KNN	62.98 %
Adaboost	69.48 %
Discriminante	87.01 %

4.3.4. Clasificación de 6 emociones

Para una octava prueba se agregó a la clasificación una emoción neutral, en el Tabla 4.36 se pueden observar los parámetros con los que se realizó la octava prueba.

Tabla 4.36: Descripción de los parámetros de la prueba 8.

Prueba	**8**
Descriptores	Momentos de Zernike
Emociones a clasificar	enojo, felicidad, sorpresa, miedo, tristeza, neutral
Imágenes Totales	185
No. de imágenes a clasificar	enojo: 30, felicidad: 31, sorpresa: 30, miedo:32, tristeza:31, neutral:31
No. de imágenes de entrenamiento	10 por emoción

Tabla 4.37: Matriz de confusión con los resultados de la octava prueba usando el clasificador KNN

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**	**Neutral**
Enojo	25	0	0	2	3	0
Felicidad	9	14	0	2	4	2
Sorpresa	4	4	17	2	2	1
Miedo	2	2	2	23	2	1
Tristeza	10	0	0	2	18	1
Neutral	5	4	0	4	2	16

Tabla 4.38: Matriz de confusión con los resultados de la octava prueba usando el clasificador AdaBoost

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**	**Neutral**
Enojo	22	1	0	4	2	1
Felicidad	0	19	0	0	12	0
Sorpresa	0	7	21	0	1	1
Miedo	2	3	2	18	7	0
Tristeza	1	4	1	3	22	0
Neutral	0	10	0	1	8	12

Tabla 4.39: Matriz de confusión con los resultados de la octava prueba usando el clasificador Discriminante

	Enojo	**Felicidad**	**Sorpresa**	**Miedo**	**Tristeza**	**Neutral**
Enojo	27	0	0	1	2	0
Felicidad	1	25	0	1	3	1
Sorpresa	0	1	27	0	1	1
Miedo	2	1	2	25	1	1
Tristeza	1	3	0	4	22	1
Neutral	0	0	1	0	2	28

Análisis de resultados de la prueba 8.

En la prueba 8 se clasificarón 185 imágenes diferentes pertenecientes a 6 emociones: enojo, felicidad, sorpresa, miedo, tristeza y neutral. Los descriptores utilizados fueron los momentos de Jacobi-Fourier y para el entrenamiento de los 3 clasificadores se utilizaron 60 imágenes (10 imágenes por emoción). En el Tabla 4.40 se muestran los porcentajes de clasificación correcta para cada uno de los tres clasificadores.

Tabla 4.40: Porcentajes de clasificación correcta de la prueba 8

Clasificador	**Porcentaje de clasificación correcta**
KNN	61.08 %
Adaboost	61.62 %
Discriminante	83.24 %

En el Tabla 4.40 se puede observar que el mejor clasificador en cuanto a clasificación correcta es el Discriminante con un 83.24 %, los clasificadores AdaBoost y KNN presentan un desempeño bajo con un 61.08 % y 61.62 % respectivamente.

4.4. Análisis general de los resultados

Con el software desarrollado se realizarón ocho pruebas principales, cuatro utilizando los momentos de Zernike y cuatro utilizando los momentos de Jacobi-Fourier, como descriptores de las imágenes, las cuales procedían de la base de datos JAFFE. Para la clasificación fueron probados tres clasificadores diferentes, KNN, AdaBoost y Discriminante. Para cada una de estas ocho pruebas se formarón las matrices de confusión con los resultados obtenidos, se calculó además el porcentaje de clasificación correcta.

La Tabla 4.41 muestra la comparación de los parámetros utilizados para los momentos de Zernike y los de Jacobi-Fourier. Como se puede observar se utilizaron en total 121 momentos, lo que representa un orden 20 en los momentos de Zernike y un orden 11 en los momentos de Jacobi-Fourier. El motivo de este número de momentos (121) resulta de algunas pruebas donde se buscó un número adecuado para que al realizar la clasificación de una imagen la relación tiempo/resultado fueran adecuados. Para el α y β usados en los momentos de Jacobi-Fourier se eligió el numero Φ o número de oro, el cual es un numero irracional con un valor aproximado a 1.61803398874988.

Tabla 4.41: Comparacion entre parametros de los momentos de Zernike y Jacobi-Fourier utilizados para las pruebas.

	α **y** β	**Orden**	**No. de Momentos**
Momentos de Zernike	N/A	20	121
Momentos de Jabobi-Fourier	$\Phi \simeq 1{,}61803398874988$	11	121

Tomando en cuenta solo las pruebas cuatro y ocho, en las cuales se intentan clasificar seis emociones diferentes y las cuales representan mayor complejidad, se puede decir que el clasificador Discriminante es el que presenta un mejor desempeño en cuanto a clasificación correcta de las emociones, con un 85.40 % utilizando como descriptores los momentos de Zernike y un 83.24 % usando como descriptores los momentos de Jacobi-Fourier.

Tabla 4.42: Comparación de porcentajes de clasificación correcta de la pruebas 4 y 8

Clasificador	**Momentos de Zernike**	**Momentos de Jacobi-Fourier**
KNN	54.59 %	61.08 %
Adaboost	63.24 %	61.62 %
Discriminante	85.40 %	83.24 %

De acuerdo al Tabla 4.42, el clasificador KNN presenta un mayor porcentaje de clasificación correcta cuando los descriptores son los momentos de Jacobi-Fourier, caso contrario a los clasificadores AdaBoost y Discrimínate para los cuales los mayores porcentajes de clasificación correcta son cuando se utilizan los momentos de Zernike como descriptores de las imágenes, pero solo el clasificador Discriminante es el que en todas las pruebas presenta porcentajes de clasificacion correcta altos. En general para la clasificación de emociones en todas las pruebas realizadas la combinación de los momentos de Zernike como descriptores y el clasificador Discriminante son los que presentan el mejor desempeño para la clasificación de las emociones.

4.5. Conclusiones

Se propuso una técnica para el reconocimiento de las emociones en imágenes digitales, basado en el procesamiento de las imágenes usando métodos como el $sharpening$ y la binarización, la extracción de características usando para estos los momentos de Jacobi-Fourier y los momentos de Zernike y la clasificación para la cual se propone el uso del clasificador Lineal Discriminante.

Para la evaluación de la técnica propuesta se desarrolló un software en Matlab. Este software permite al usuario entrenar un clasificador utilizando imágenes de las cuales se conozca previamente la emoción presente, con el objetivo de poder determinar la emoción presente en imágenes de las cuales no tengan datos, estas imágenes pueden estar guardadas en la PC donde se esté corriendo el programa o también pueden ser capturadas mediante un robot NAO, para el cual se creó una interfaz de comunicación con el Software. La elección de Matlab se debió a las capacidades de esta herramienta para la manipulación de matrices, y los diferentes paquetes que incluye, como el de los clasificadores y el de imágenes el cual ya trae precargado diferentes algoritmos de procesado de imágenes, los cuales fueron muy útiles para este trabajo.

Fueron implementados los algoritmos de los momentos de Zernike y Jacobi-Fourier en Matlab ya que este no cuenta con una implementación propia. Para probar que la implementa-

ción fue la correcta se graficaron los primeros seis polinomios tanto de Zernike como de Jacobi y se compararon con los vistos en la literatura, y para probar la eficacia de descripción de los mismos se realizaron pruebas como la reconstrucción de imágenes, y el cálculo del error NIRE para cada una de las reconstrucciones.

Se investigó y evaluó el desempeño de clasificación para cada uno de los clasificadores propuestos, para esto se utilizaron datos que trae Matlab por default de algunas bases de datos conocidas, al hacer la comparación con estos datos los tres clasificadores presentaban porcentajes de clasificación muy parecida, pero a la hora de trabajar con los datos obtenidos con esta técnica propuesta se vio que algunos clasificadores tenían mucho mejor desempeño en clasificar correctamente los datos.

En cuanto a la evaluación del desempeño de la técnica propuesta con el software desarrollado, se realizaron ocho pruebas principales, en cuatro de ellas se utilizaron los momentos de Zernike como descriptores y en otras cuatro se utilizaron los momentos de Jacobi-Fourier, la diferencia entre cada una de las pruebas es el número de emociones a reconocer (tres, cuatro, cinco y seis) y los descriptores utilizados. Con estas pruebas se pudo comprobar la eficacia de los momentos tanto de Zernike como de Jacobi-Fourier para la descripcion de imagenes, con las ventajas de que estos momentos son invariantes a algunas transformaciones como la escala, la rotación, el desplazamiento, etc.

Para la selección del número de momentos a utilizar, se tomó mucho en cuenta el tiempo de computo, ya que se necesitaba que la técnica propuesta trabajara casi en tiempo real, por esto en las pruebas realizadas se buscó un numero de momentos para el cual tanto la calidad de descripción de la imagen así como el tiempo de computo proporcionaran buenos resultados en la clasificación. En la Tabla 4.43 se presentan los tiempos de computo de algunas pruebas realizadas.

Tabla 4.43: Cuadro comparativo de tiempos de computo entre momentos de Zernike y momentos de Jacobi-Fourier, los valores estan en segundos.

	Entrenamiento de 10 imágenes		**Clasificación de 2 imágenes**	
	Zernike	**Jacobi**	**Zernike**	**Jacobi**
Felicidad	10,9181	14,0549	2,0861	2,7513
Enojo	10,7944	14,2938	1,9795	2,7873
Sorpresa	10,3727	14,088	2,1543	2,7328

Según los resultados obtenidos el clasificador Discriminante es el que presenta un mejor desempeño de clasificación correcta, ya que en todas las pruebas fue el que obtuvo un porcentaje mayor, logrando para la clasificación de tres hasta cinco emociones porcentajes de más de 90 % de clasificación correcta, para el caso del reconocimiento de seis emociones el cual es el que presenta mayor complejidad, los porcentajes son del 85 % aproximadamente (83.24 %

usando los momentos de Jacobi-Forier y 85.40 % usando los momentos de Zernike). Al hacer el análisis de los resultados se pudo concluir que en general el mejor desempeño de la técnica para la clasificación de las emociones es cuando se utilizan los momentos de Zernike como descriptores de las imágenes y el clasificador Discriminante.

Se logró el objetivo principal de la tesis que era la parte de desarrollar e implementar algoritmos que fueran capaces de reconocer emociones en imágenes digitales, siguiendo esta línea de investigación la cual tiene muchas aplicaciones como las interacción humano-computadora, aplicaciones en áreas como la mercadotecnia o marketing donde empresas ya están utilizando tecnología de reconocimiento de emociones, analizando a través de la webcam que emociones transmiten los productos en algunos potenciales consumidores. También pueden ayudar en sistemas de detección de mentiras, en programas de ayuda psicológica o psiquiátrica, etc.

En cuanto al trabajo a futuro, se pretende la creación de una base de datos propia con la cual se pueda evaluar la técnica, para la captura de las imágenes se utilizaran las cámaras del robot NAO. Se utilizaran algunas de las imágenes para el proceso de entrenamiento de los clasificadores y las demás se usaran como imágenes de prueba. Con los resultados obtenidos con esta base de datos se podrá aumentar la valides de la técnica en el reconocimiento de emociones.

Si la validación de la técnica con la base de datos propia es satisfactoria se buscara alguna aplicación, como por ejemplo la que se trabajó en la Universidad Politécnica de Valencia (ver Capitulo 1), donde se utilizó el reconocimiento de emociones como un parámetro en la detección de la depresión, usándolo en conjunto con otros métodos de ayuda en la detección como los cuestionarios o las entrevistas personalizadas.

Índice de figuras

1.1. **Malla y huesos de uno de los avatares propuestos [3].** . . . 3
1.2. **Diagrama general de bloques con los pasos de la técnica propuesta.** . . . 5
1.3. **Ejemplo de una gráfica con los resultados del reconocimiento de las emociones en un cierto periodo de tiempo.** . . . 6
1.4. **Tabla de resultados de la prueba de reconocimiento de emociones en conjunto con el BECK II.** . . . 7

2.1. **Codificación del contorno de un cuadrado.** . . . 12
2.2. **Imagen binaria de 5×5 pixeles, cada cuadro representa un pixel.** . . . 13
2.3. **a) Imagen original, b) Imagen rotada 90°, c) Imagen trasladada, d)Imagen escalada** . . . 16
2.4. **a) Imagen en coordenadas cartesianas b) Imagen mapeada fuera del circulo unitario, c) Imagen mapeada dentro del circulo unitario** . . . 17
2.5. **Primeros 6 polinomios radiales de Zernike.** . . . 19
2.6. **Polinomios complejos de Zernike hasta el orden 2. La parte real: fila 1: $n = 0, l = 0$, fila 2: $n = 1, l = -1, 1$,fila 4: $n = 2, l = -2, 0, 2$. La parte imaginaria en las filas 3 y 5, los índices son los mismos como en la parte real.** . . . 19
2.7. **Polinomios radiales de Jacobi $J_{21}(2, 2, r)$, calculados usando el método directo y el recursivo.** 22
2.8. **Polinomios de Jacobi hasta el 2 orden. La parte real: fila 1: $n = 0, l = 0$, fila 2: $n = 1, l = -1, 0, 1$,fila 4: $n = 2, l = -2, -1, 0, 1, 2$. La parte imaginaria en las filas 3 y 5, los índices n, l son los mismos como en la parte real.** . . . 22
2.9. **a)Imagen original, b) , c), d), e), f) Reconstrucciones de la imagen original con momentos de Zernike usando ordenes 5,10,15,20,40 respectivamente.** . . . 24
2.10. **Gráfica del NIRE para las reconstrucciones de la Figura 2.9** . . . 24
2.11. **a) Imagen Original, b), c), d), e), f) Reconstrucciones de la imagen original usando JFMs con ordenes 0,10, 20, 30 y 40 respectivamente.** . . . 25
2.12. **Gráfica del NIRE para las reconstrucciones de la Figura 2.11** . . . 25
2.13. **a) Imagen Original, b), c), d), e) reconstrucción de imágenes usando JFMs con ordenes 10, 20, 30 y 40 respectivamente.** . . . 26
2.14. **Gráfica del NIRE para las reconstrucciones de la Figura 2.13** . . . 26
2.15. **Representación de un objeto con tres pixeles marcados, dentro del objeto, sobre un borde y sobre una esquina** . . . 27
2.16. **Idea básica del algoritmo de Harris** . . . 27
2.17. **Función ventana $w(x, y)$** . . . 28
2.18. **Traza de una matriz** . . . 29

2.19. **Aplicación del método de Harris y del método de Shi y Tomasi para la detección de esquinas a la imagen del $cameraman$.** . . . 29
2.20. **A partir de una imagen generamos un patrón, que describe a la imagen.** . . . 30
2.21. **Clasificación lineal (izquierda) y clasificación no lineal (derecha).** . . . 31
2.22. **Esquema básico de un clasificador supervisado.** . . . 32
2.23. **Efectos de cambio de k en la clasificación.** . . . 33
2.24. **Diagrama del funcionamiento del clasificador Adaboost [23].** . . . 34

3.1. **Ejemplo de gestos faciales para dos emociones diferentes. a) Expresión de miedo y b) expresión de felicidad.** . . . 41
3.2. **Ejemplo de las imágenes de la base de datos JAFEE.** . . . 42
3.3. **Robot NAO.** . . . 43
3.4. **Ángulo de visión vertical de las cámaras del robot NAO.** . . . 44
3.5. **Ángulo de visión horizontal de las cámaras del robot NAO.** . . . 44
3.6. **Esquema de la toma de imágenes con el robot NAO.** . . . 45
3.7. **Diagrama de la técnica propuesta.** . . . 46
3.8. **Ejemplos de detección de rostros utilizando OPENCV para Android.** . . . 47
3.9. **Resultado de aplicar el algoritmo de Viola-Jones a algunas imágenes de la base de datos JAFEE.** . . . 48
3.10. **Imagen original (izquierda), imagen realzada con $sharpening$ (derecha).** . . . 48
3.11. **Binarización a la imagen de la Figura 3.10** . . . 49
3.12. **En la etapa de extracción de características a partir de la imagen binaria obtenemos un vector de descriptores.** . . . 50
3.13. **A partir de un vector de descriptores, un clasificador debería ser capaz de reconocer la emoción.** . . . 50
3.14. **Diagrama de la secuencia de operaciones del software desarrollado.** . . . 51
3.15. **Opciones principales de la interfaz usuario del software desarrollado.** . . . 51
3.16. **Ventana para nombrar la clase a entrenar dentro del software.** . . . 52
3.17. **Ventana para seleccionar las imágenes de X clase en la etapa extracción de características del software.** . . . 52
3.18. **Parte de la interfaz de usuario donde se realiza la selección del clasificador en el software.** . . . 53
3.19. **Ejemplo del resultado de clasificación para 2 imágenes.** . . . 53
3.20. **Matriz de confusión generada por el software.** . . . 53
3.21. **Ejemplo de la respuesta de CascadeObjectDetector, sobre una imagen de prueba.** . . . 55
3.22. **a) Delimitar o b) recortar la región de interés con el `bbox`.** . . . 55
3.23. **Forma de los parametros que recibe la funcion fitcknn(X,Y)** . . . 58

Índice de tablas

2.1. Algunas técnicas de extracción de características. 11
2.2. Familias de polinomios de acuerdo a los valores α, β. 21
2.3. Adaboost: ventajas y desventajas 34

3.1. Características principales del robot NAO. 43
3.2. Especificaciones técnicas de las cámaras del robot NAO 44
3.3. Lista de los principales parámetros modificables en las cámaras del Robot NAO. 45
3.4. Resoluciones compatibles en las cámaras del Robot NAO. 45

4.1. Descripción de los parámetros de la prueba 1. 64
4.2. Matriz de confusión con los resultados de la primera prueba usando el clasificador KNN 64
4.3. Matriz de confusión con los resultados de la primera prueba usando el clasificador AdaBoost 65
4.4. Matriz de confusión con los resultados de la primera prueba usando el clasificador Discriminante 65
4.5. Porcentajes de clasificación correcta de la prueba 1 65
4.6. Descripción de los parámetros de la prueba 2. 66
4.7. Matriz de confusión con los resultados de la segunda prueba usando el clasificador KNN 66
4.8. Matriz de confusión con los resultados de la segunda prueba usando el clasificador Adaboost 67
4.9. Matriz de confusión con los resultados de la segunda prueba usando el clasificador Discriminante 67
4.10. Porcentajes de clasificación correcta de la prueba 2 67
4.11. Descripción de los parámetros de la prueba 3. 68
4.12. Matriz de confusión con los resultados de la tercera prueba usando el clasificador KNN 69
4.13. Matriz de confusión con los resultados de la tercera prueba usando el clasificador Adaboost 69
4.14. Matriz de confusión con los resultados de la tercera prueba usando el clasificador Discriminante 69
4.15. Porcentajes de clasificación correcta de la prueba 3 70
4.16. Descripción de los parámetros de la prueba 4. 70

4.17. Matriz de confusión con los resultados de la cuarta prueba usando el clasificador KNN 71
4.18. Matriz de confusión con los resultados de la cuarta prueba usando el clasificador AdaBoost 71
4.19. Matriz de confusión con los resultados de la cuarta prueba usando el clasificador Discriminante 71
4.20. Porcentajes de clasificación correcta de la prueba 4 72
4.21. Descripción de los parámetros de la prueba 5. 73
4.22. Matriz de confusión con los resultados de la quinta prueba usando el clasificador KNN 73
4.23. Matriz de confusión con los resultados de la quinta prueba usando el clasificador AdaBoost 73
4.24. Matriz de confusión con los resultados de la quinta prueba usando el clasificador Discriminante 74
4.25. Porcentajes de clasificación correcta de la prueba 5 74
4.26. Descripción de los parámetros de la prueba 6. 75
4.27. Matriz de confusión con los resultados de la sexta prueba usando el clasificador KNN 75
4.28. Matriz de confusión con los resultados de la sexta prueba usando el clasificador Adaboost 76
4.29. Matriz de confusión con los resultados de la sexta prueba usando el clasificador Discriminante 76
4.30. Porcentajes de clasificación correcta de la prueba 6 76
4.31. Descripción de los parámetros de la prueba 7. 77
4.32. Matriz de confusión con los resultados de la séptima prueba usando el clasificador KNN 77
4.33. Matriz de confusión con los resultados de la séptima prueba usando el clasificador Adaboost 78
4.34. Matriz de confusión con los resultados de la séptima prueba usando el clasificador Discriminante 78
4.35. Porcentajes de clasificación correcta de la prueba 7 78
4.36. Descripción de los parámetros de la prueba 8. 79
4.37. Matriz de confusión con los resultados de la octava prueba usando el clasificador KNN 79
4.38. Matriz de confusión con los resultados de la octava prueba usando el clasificador AdaBoost 80
4.39. Matriz de confusión con los resultados de la octava prueba usando el clasificador Discriminante 80
4.40. Porcentajes de clasificación correcta de la prueba 8 81
4.41. Comparacion entre parametros de los momentos de Zernike y Jacobi-Fourier utilizados para las pruebas. 81
4.42. Comparación de porcentajes de clasificación correcta de la pruebas 4 y 8 82
4.43. Cuadro comparativo de tiempos de computo entre momentos de Zernike y momentos de Jacobi-Fourier, los valores estan en segundos. 83