UNIVERSIDAD DEL MAR
CAMPUS PUERTO ESCONDIDO

CLASIFICACIÓN DE OBJETOS RÍGIDOS A PARTIR DE IMÁGENES DIGITALES, EMPLEANDO LOS MOMENTOS INVARIANTES DE HU

T E S I S

QUE PARA OBTENER EL TÍTULO DE
LICENCIADO EN INFORMÁTICA

PRESENTA
CARLOS PÉREZ LARA

DIRECTOR DE TESIS
M. EN C. JORGE OCHOA SOMUANO

PUERTO ESCONDIDO, OAXACA 2012

Dedicatoria

A mi mamá,

Porque sin usted no lo hubiera hecho posible, por haberme enseñado a mirar más allá que cualquier otra persona, por enseñarme a trabajar sin importar el agotamiento, por eso y muchísimo más, este libro es para usted mamá.

A mis hermanas Eli y Soli.

Por su amor, cariño, comprensión, hermandad y ayuda en este trabajo.

A mi hermosa novia, ***Yane****, quien me alienta y cree en mí, por demostrarme que puedo contar con su ayuda en cualquier momento, y sobre todo, por haber hecho que disfrutara más la última parte de esta etapa de mi vida.*

A ***Sol****, por los momentos agradables que convivimos.*

Agradecimientos

*A la **Universidad del Mar** por brindarme la facilidad de adquirir estos conocimientos y aplicarlos en este trabajo de tesis y por prepararme en la entrada de mi vida profesional.*

*A la **Honorable comisión de becas**, por su apoyo con la beca alimenticia en algunos semestres de mi carrera.*

*Al **Programa Nacional de Becas**, por bridarme la beca de Servicio Social la cual me fue de gran ayuda en un momento oportuno.*

*A **mi mamá**, por su ayuda incondicional a lo largo de toda mi trayectoria académica, la única persona que más admiraré en toda mi vida, por ser la madre más atenta con sus hijos y darles lo mejor sin pensar antes en ella, por ayudarme a pensar que nada es gratis y a cumplir mis ambiciones, por eso y más le viviré agradecido toda mi existencia.*

*Al **Maestro en Ciencias Jorge Ochoa Somuano**, mi director de tesis y maestro, por aceptarme como tesista, guiarme en cada momento, tenerme paciencia y hacerme un espacio en su apretada agenda a lo largo de este trabajo de tesis; y sobre todo por ser un excelente maestro.*

*A los **revisores de tesis** que participaron con su tiempo y profesión en la corrección y aprobación de esta tesis.*

*A la **Maestra en Administración Mabel Rodríguez de la Torre**, por darme sus consejos para mejorar mi calidad académica, por alentarme en cada momento para aspirar a cosas grandes.*

*A mi hermana **Eli**, por su orientación académica a lo largo de mi carrera y por su cuidado de toda mi vida.*

*A mi hermanita **Soli**, por despertarme en las mañanas cada vez que se me hacía tarde y por hacer desaparecer mi mal humor, causa de mis desvelos, con sus pláticas.*

*A **Alejandro Torres García**, **Anayeli Pérez Clemente,** a mis compañeros de grupo: **Anel, Ali, Farith, Iván, Erik, José, Evelio, Segifredo y Marquitos,** sin olvidar a **Poly** y **Ayleth,** y a **todos** aquellos que formaron parte de este trabajo.*

Resumen

La inteligencia artificial ha funcionado como un instrumento muy importante en el desarrollo de nuevas tecnologías referentes a visión, voz, señales, todo esto referente para procesarlo por medio de una computadora. Dentro del área de visión, es de suma importancia cuando ésta se aplica se haga la simulación de distinguir un objeto de otros, la diferenciación de estos lo hace un clasificador de objetos. Este trabajo de tesis nombrado "Clasificación de objetos rígidos a partir de imágenes digitales, empleando los momentos invariantes de *Hu*" fue una investigación en la que se realizó un software clasificador de objetos. Para la extracción de las características de los objetos en las imágenes se emplearon lo momentos invariantes de *Hu*, que, como su nombre lo dice, son invariantes a rotación, escala y traslación. En la clasificación de estos mismos objetos se utilizó el clasificador *K-means*. Las pruebas que se realizaron para la validación de este software fueron: números, letras, herramientas, tornillos, figuras geométricas y llaves con monedas. En el resultado de estas pruebas se reportó un porcentaje de certeza de 97.76 % y un porcentaje de error de 2.24 % bajo condiciones controladas del ambiente. De acuerdo a los resultados planteados, estos mismos pueden ser aplicados en la industria como clasificadores de control de calidad, en la medicina como clasificación de células cancerígenas, en la milicia como detección de artillería enemiga, entre otras.

Abstract

Artificial intelligence has been a very important instrument for the development of new technologies related to vision, voice, signals which can be processed using computers. Artificial intelligence is of utmost importance when simulation on its viewing area is programmed to distinguish an object from others, in effect classifying objects. This thesis named "Classification of rigid objects from digital images, using *Hu* invariant moments" was an investigation using object-classification software. In order to features extrction of objects in the images the method *Hu's* invariant moments was used, which, as its name implies, are invariant to rotation, scaling and translation of objects. In the classification of these same objects the *K-means* classifier was used. The tests performed for the validation of this software were: numbers, letters, tools, screws, keys shapes and coins. The results of these tests are reported to a percentage of certainty of 97.76 % and an error rate of 2.24 %. According to the proposed results, the same accuracy variables can be applied in industry for quality control in medicine and classification of cancer cells; in the military as enemy-artillery detection, among many other uses.

CONTENIDO

LISTADO DE FIGURAS

LISTADO DE TABLAS

LISTADO DE ECUACIONES

LISTADO DE ALGORITMOS

LISTADO DE CÓDIGOS

GLOSARIO DE TÉRMINOS

Centroide	Es el objeto que sirve como referencia para agrupar o clasificar los demás objetos en relación a él mismo (Hernández, Ramírez & Ferri 2004).
Clase	Es el resultado de la clasificación de objetos a partir del reconocimiento de patrones (Russel & Norving 2004).
ICA	El análisis de componentes independientes (*Independent Component Analysis*) es una técnica estadística e informática para revelar los factores ocultos que subyacen en conjuntos de variables aleatorias, medidas o señales (Aapo S.F.).
IDE	El entorno de desarrollo integrado (*Integrated Development Environment*), es un conjunto de herramientas que ayudan al desarrollo de aplicaciones. La mayoría de los IDE cuentan con herramientas que le permiten: escribir y editar el código fuente, consultar los errores a medida que escribe, ver la sintaxis del código resaltada, automatizar las tareas repetitivas y compilar el código (Oracle 2011b).
Imagen digital	Representación de una imagen en una matriz de valores en dos dimensiones (Russel & Norving 2004).
Java	Lenguaje de programación orientado a objetos multiplataforma, desarrollado por Sun Microsystem (Oracle 2011a).
K-vecinos	Es el método de aprendizaje de algoritmos como el *K-means*, el cual se basa en instancias, donde cada nueva instancia se compara con las existentes usando una métrica de distancia (Hernández, Ramírez & Ferri 2004).

LDA	El método de análisis de discriminante lineal (*Linear Discriminant Analysis*) consiste en maximizar la proporción de la varianza entre clases dentro de la clase variación en los datos particulares establecidos garantizando la máxima separabilidad (Balakrishnama & Ganapathiraj S.F.).
MATLAB	MATLAB es un lenguaje de alto nivel técnico de computación y un entorno interactivo para desarrollo de algoritmos, visualización de datos, análisis de datos y cálculo numérico (MathWorks 2011).
MLP	*MultilayerPerceptron*. Es la adición de múltiples capas a un perceptron simple, el cual se entrena mediante el algoritmo de retro propagación (Del Brio & Sanz 2007).
PCA	El análisis de componentes principales (*Principal Components Analysis)* es un método que reduce la dimensión de datos al realizar un análisis de covarianza entre los factores (Agilent 2005).
PDI	Procesamiento Digital de Imágenes, es el proceso de la transformación de una imagen digital en una diferente (De la Escalera 2001).
Pixel	*Picture Element*. Cada elemento de una matriz de dos dimensiones (Pajares & De la Cruz 2008).
RBF	*Radial Basis Function*, es una red neural para ver el diseño como un problema de ajuste de curvas (una aproximación) en un espacio dimensional. (Sahin 1997).
RGB	Es un modelo de color donde las imágenes en este modelo se forman por las combinaciones de cada uno de los colores primarios (rojo, verde y azul). (Pajares & De la Cruz 2008).
Sesgo	Es un valor de ajuste que se le aplica a la salida de una red neuronal artificial (Del Brio & Sanz 2007).

SO	El sistema operativo (SO), es un programa especial que se carga en un ordenador tras ser encendido y cuya función es gestionar los demás programas, o aplicaciones, que se ejecutarán en dicho ordenador, como, por ejemplo, un procesador de texto o una hoja de cálculo, o la impresión de un texto en una impresora o una conexión a Internet (Fernández 2001).
SVM	Máquina de vectores de soporte (*Support Vector Machine*), es un algoritmo informático que aprende asignar etiquetas a los objetos (Nature 2006).
Umbral	Es el valor que indica el cambio de intensidad de color en el histograma de una imagen (Pajares & De la Cruz 2008).

CAPÍTULO 1. INTRODUCCIÓN

El trabajo de tesis que se presenta es el desarrollo de un software clasificador de objetos basado en imágenes digitales, utilizando como extractor de características los momentos invariantes de *Hu* y como algoritmo clasificador el *K-means*. El presente software fue desarrollado en Java utilizando como IDE NetBeans 6.9.1.

Problemática

Las últimas tendencias en Inteligencia Artificial se basan en buscar soluciones de inspección visual como verificar la calidad de frutas, verduras u otros comestibles. Por otro lado, también se utilizan en la clasificación de objetos, como pueden ser tornillos, partes de motores automotrices, entre otros.

Algo fundamental para que la inspección visual se pueda llevar a cabo, es la extracción de características de los objetos que estén involucrados en la escena que se desea inspeccionar. Con base en estas características o propiedades, entonces se aplican métodos

para determinar si el objeto en estudio contiene los atributos necesarios para considerarse de buena calidad o en su defecto considerarlo de mala calidad. Por otro lado, también se puede hacer una separación de clases, con base en la similitud que existe entre algunos objetos. Esta última situación es la que se utiliza como enfoque para el desarrollo del presente trabajo de investigación, la clasificación de objetos.

Metodología de solución

Para la resolución de este trabajo de tesis se utilizó principalmente el algoritmo de momentos invariantes de *Hu* como extractor de características y para clasificar dichas características se usó el clasificador *K-means*. También se empleó la metodología **divide y vencerás** del desarrollo de software, en la cual el objetivo general se dividió en submódulos consecutivos, esto con el fin de resolver el problema principal en sus partes más pequeñas. La metodología de divide y vencerás, consiste en descomponer un problema en cierto número de subproblemas más simples del mismo tipo de problema, con el fin analizarlos en forma independiente y combinarlos al final una vez resueltos (Dasgupta, Papadimitriou & Vazirani 2006, Joyanes, Zahonero, Fernández & Sánchez 1999). Si los subproblemas son todavía relativamente grandes se aplicará de nuevo este método hasta alcanzar subproblemas lo suficientemente pequeños para ser solucionados directamente (Guerequeta & Vallecillo 1998).

Organización del documento de tesis

Este documento está basado en cinco capítulos. En este primer capítulo de introducción se mencionó el problema de la tesis, que metodología se utilizó para resolverlo y como está estructurado este documento. El siguiente capítulo presenta las relaciones que tiene este trabajo de tesis con otros ya realizados. Se mencionan seis trabajos de tesis desarrollados en diversos centros de investigación e institutos y dos trabajos relacionados, además se habla acerca del porque realizar esta tesis, que justificación se tomó, que se pretende alcanzar y cuáles son los alcances y límites de este trabajo.

En el capítulo 3 se mencionan las áreas y conocimientos de procesamiento digital de imágenes requeridos para poder llevar a cabo el desarrollo de este trabajo. El capítulo de desarrollo del tema muestra el proceso de la solución del problema de este trabajo de tesis,

primero, se describe el análisis del problema, después se hace el diseño de la solución, una vez acabado el diseño se procede a la implementación en un lenguaje de programación, cuando este ha sido codificado se hacen las pruebas necesarias para validar y verificar el software desarrollado, por último se presentan los resultados obtenidos de las pruebas. Después, en el capítulo 5 se presentan las conclusiones obtenidas a partir de los resultados; se menciona el porcentaje de certeza y error del software, así como de cada prueba realizada, también se hace mención de los posibles trabajos que surgen a partir de éste.

En el primer anexo se describen cada una de las imágenes utilizadas, una descripción de la imagen y que característica fue evaluada. Después, en el anexo B se muestra detalladamente el proceso de su procesamiento, es decir, que procesos fueron necesarios para aplicar el algoritmo de momentos invariantes de *Hu* para un resultado óptimo en la clasificación. El siguiente anexo muestra el código fuente de las principales clases y funciones que hicieron posible el funcionamiento de este sistema, desde conversión a escala de grises, binarización, negativo, momentos invariantes de *Hu* y *K-means*. En el anexo D se presenta el manual de operación, resultado de este trabajo de tesis. La estructura del manual se basa en la descripción de cada función de los módulos desarrollados. El último anexo muestra como están distribuidos todos los archivos necesarios para ejecutar la aplicación desarrollada, además de adjuntar este trabajo en formato digital.

CAPÍTULO 2. ANTECEDENTES

2.1. Estado del arte y trabajos relacionados

Técnicas de extracción de características en imágenes para el reconocimiento de expresiones faciales (Castrillón, Álvarez & López 2008), es una tesis realizada en la Universidad Tecnológica de Pereira (Colombia). Se basó en la recolección de técnicas para la extracción de características de los rostros para la determinación de las seis expresiones fáciles básicas: alegría, enojo, tristeza, sorpresa, miedo y disgusto. Todas estas expresiones están relacionadas y están codificadas por el sistema de codificación de expresiones faciales (FACS, por sus siglas en inglés).

Para realizar este estudio se eligieron técnicas que se dividieron en tres categorías principales. La primera se basa en los movimientos que se realizan a través del flujo óptico, y ésta a su vez se puede calcular de dos formas: calculando los gradientes y realizando la segmentación. La segunda se basa en el análisis del dominio espacial mediante métodos

estadísticos, este proceso lo hace por medio de procesos holísticos espaciales que utilizan por lo general imágenes en niveles de gris. Entre estos procesos se encuentran: PCA (análisis de componentes principales, por sus siglas en inglés) e ICA (análisis de componentes independientes, por sus siglas en inglés) los cuales usan representaciones del rostro encontradas por métodos estadísticos no supervisados, típicamente que encuentran un conjunto de imágenes base y representan los rostros como una combinación lineal de estas imágenes base. La última categoría es un análisis en el dominio espectral mediante transformadas frecuencia-tiempo, entre ellas la transformada de Wavelet de enteros a enteros y la transformada de Gabor. Se mencionaron las diferentes técnicas para la extracción de características faciales correspondientes a movimientos y deformaciones, es difícil comparar los sistemas de reconocimiento de expresiones faciales debido a la forma en que cada técnica presenta los resultados.

La desventaja de este trabajo es que todas las pruebas de las técnicas mencionadas fueron probadas en imágenes de vistas frontales tomadas bajo condiciones controladas lo que representa dificultad al momento de realizar una aplicación real de interfaz hombre-máquina.

Sistema de clasificación de imágenes basado en técnicas de reconocimiento de patrones aplicado en termografía y robótica (Vélez, Erazo & Loaiza 2009), en este artículo se presenta el proceso de diseño de tres clasificadores: uno bayesiano y dos con redes neuronales (perceptron multicapa y redes de base radial) para la clasificación de imágenes, utilizando como características cuatro momentos invariantes estadísticos y siete momentos invariantes de *Hu*. El sistema se valida con dos aplicaciones, la primera en termografía que consiste en un prediagnóstico de la operación normal o anómala de un motor eléctrico a partir del análisis de sus imágenes infrarrojas. La segunda consiste en identificar un objeto de un sistema robótico asignándolo a una de las dos posibles clases existentes: robot u obstáculo.

El sistema se basa en el seguimiento de cinco bloques principales:

1. Adquisición de imágenes. El sistema tiene como entrada una etapa para la adquisición de información desde una cámara. El resultado corresponde a imágenes en formato RGB.

2. Preprocesamiento. En este punto se obtienen sólo los elementos que se utilizarán de la imagen; esto se logra convirtiendo la imagen a escala de grises y después aplicando un proceso de segmentación.
3. Extracción de características. Aquí se caracteriza el objeto para que éste sea identificado por mediciones, cuyos valores sean muy similares entre objetos de la misma clase y muy diferentes para objetos de una clase diferente. En esta etapa se utilizó información que proporcionaron dos extractores de características. En el caracterizador se usaron los cuatro primeros momentos estadísticos y en el segundo los siete momentos invariantes de *Hu*, los cuales son invariantes a rotación, traslación y cambio de escala; quedando una matriz de características de n x 11, siendo n el número de muestras que se procesaron.
4. Ajuste de parámetros de clasificador. En este bloque se utilizan tres tipos de clasificadores: uno estadístico y dos basados en redes neuronales artificiales. El primero se basa en la teoría de decisión bayesiana, el cual es un enfoque estadístico fundamental para resolver problemas que involucran la clasificación de patrones. En lo que respecta a los clasificadores neuronales, en el primero de ellos se utilizó un perceptron multicapa (MultiLayerPerceptron, MLP); la regla de aprendizaje tipo perceptron calcula cambios deseados a los pesos y sesgo de la red, dado un vector de entrada p y el error asociado e. En el segundo clasificador, también basado en redes neuronales fue RBF (redes de base radial, por sus siglas en inglés), para este tipo de redes la expresión para la entrada a sus neuronas es diferente a la que tiene una red MLP. El argumento para la función de activación f de la red es el vector distancia entre su vector de p.
5. Clasificación. La función de este bloque fue asignar un objeto cuya clase de pertenencia es desconocida a una clase que sea parte de la base de conocimiento del sistema, teniendo en cuenta los parámetros previamente ajustados en el clasificador. En el clasificador bayesiano, se evalúan todas las funciones discriminantes del sistema y se asigna el objeto a aquella clase cuya función haya arrojado el valor más alto. Para las redes neuronales, se asignará el objeto a la clase cuya neurona de salida haya obtenido el valor más alto en la función de activación.

En lo que respecta a termografía, ésta se usó para la resolución de problemas en motores. Para ello se hizo una recolección de imágenes a utilizar en esta prueba, se utilizó como motor el prototipo del sistema *Feedback* ES151 y una cámara termográfica FLIR A20. Se obtuvo un total de 120 imágenes, 60 en operación normal del motor y 60 en condiciones de falla. En la mayoría de las pruebas los clasificadores neuronales presentaron un mejor desempeño que los clasificadores estadísticos, lo que permite comprobar la capacidad de generalización de este tipo de máquinas de aprendizaje. En términos de porcentaje, el 90 % de certeza lo obtuvieron los clasificadores neuronales, mientras que un 80 % lo obtuvo el clasificador bayesiano.

El sistema también se aplicó para la clasificación de imágenes en sistema robótico. Para este caso se utilizó una cámara de visión global Sony P73, la cual entrega imágenes digitales en formato RGB con una resolución de 640x480 pixeles. Se definen dos clases: una correspondiente al robot Pio-neer 3DX (clase 1) y otra que comprende un conjunto de obstáculos conformados por cajas de distintos tamaños, recipientes cilíndricos y sillas (clase 2). La base de datos está conformada por 60 imágenes de la clase robot y 75 imágenes de la clase obstáculo. Aquí, el mejor desempeño fue para el perceptron con un promedio de 85 %, mientras que los restantes tuvieron un 75 % de certeza.

El diseño de un clasificador requiere una selección cuidadosa del conjunto de patrones, así como del tipo de extractor de características. Para este trabajo se obtienen resultados satisfactorios utilizando imágenes provenientes de distintas bandas del espectro electromagnético, lo que verifica la robustez del tipo de extractores de características elegidos.

Clasificación de las fases de cicatrización cutánea en conejos empleando algoritmos estadísticos (Vargas 2009), este trabajo realizado en la Universidad del Quindío (Colombia) consiste en la implementación de varios clasificadores, con el propósito de identificar las fases de cicatrización cutánea en conejos, por medio de algoritmos desarrollados en el software MATLAB e imágenes previamente procesadas con algún método de extracción de características.

La cicatrización cutánea en todos los seres vivos es un tema de constante investigación y desarrollos científicos. La cicatrización es un proceso gradual, lento y que

históricamente ha sido descrito principalmente en tres etapas o fases: inflamación, epitelización y remodelación.

Cuando se quiere hacer un diagnóstico, el médico, de acuerdo a la experiencia que tenga dará una respuesta, la cual puede no ser tan acertada. De allí que el análisis y clasificación de las fases de este tipo de heridas, por medio de procesamiento digital de imágenes pueda resultar útil, para cualquier complemento en el diagnóstico, es decir, un posible soporte de consulta para un experto.

Para lograr la implementación de este software se utilizaron fotografías que fueron tomadas cada ocho horas a los conejos que habían sufrido una herida, las imágenes fueron tomadas con la misma cantidad de luz y distancia para una mejor adaptación de los algoritmos. Para el proceso de clasificación se utilizaron los clasificadores estadísticos; entre ellos, el clasificador bayesiano, los procesos gaussianos, la red de base radial probabilística, la máquina de soporte vectorial (SVM), el clasificador *K-means* y se incluyó un perceptron multicapa.

Los clasificadores se probaron sobre tres bases de datos distintas cada una con 463 patrones que equivalen a 463 fotografías procesadas. Los conjuntos de clasificación se dividieron en un 75 % para entrenamiento y 25 % para validación. Para comparar el rendimiento de los algoritmos de clasificación se tiene como marco de referencia el porcentaje de acierto y la velocidad de ejecución.

Los resultados dicen que para la clasificación de fases de cicatrización cutánea en conejos, el mejor algoritmo es el clasificador bayesiano con porcentaje de certeza de 93.43 %, le sigue la red neuronal perceptron con un 87.28 %, después la red de base radial probabilística con 87.5 %, luego la SVM con un 88.75 %, le sigue los procesos gaussianos con un 87.39 % y el *K-means* con un 73.24 % de certeza. En cuanto al tiempo de cómputo de los algoritmos, el mejor fue el clasificador *K-means*, motivo por el cual será considerado para su implementación en el presente trabajo de tesis.

En resumen, para la clasificación de imágenes procesadas de cicatrización, los algoritmos de clasificación probabilísticos obtienen mejores resultados de eficiencia de acierto, que las redes neuronales convencionales, considerando también un mejor costo computacional y mayor rapidez en la convergencia de los algoritmos.

Detección y clasificación de objetos dentro de un salón de clases empleando técnicas de procesamiento digital de imágenes (García 2008), es el título de una tesis que se desarrolló en la Universidad Autónoma Metropolitana (México) en la que se hizo un sistema de reconocimiento de objetos utilizando diferentes técnicas de procesamiento de imágenes y reconocimiento de patrones.

Las pruebas para verificar el funcionamiento de tal sistema se hicieron con objetos pequeños que se encontraron dentro de un salón de clases, limitado a reconocer sólo 10 objetos. La clasificación se hizo mediante el clasificador *K-means*.

La tesis se basó en cinco pasos principales que se exponen a continuación:

1. Adquisición de la imagen. La imagen se obtuvo por medio de una cámara web Genius NB. La resolución que se tuvo fue de 640x480 pixeles en formato BMP. Las imágenes se tomaron en un ambiente controlado, en este caso, en el salón de clases. Se capturaron 24 imágenes por cada objeto a una distancias de 25, 30 y 35 centímetros con rotaciones de 0, 45, 90, 135, 180, 225, 270 y 315 grados. En total se obtuvieron 240 imágenes por los 10 objetos que se utilizaron para realizar el reconocimiento.
2. Procesamiento de la imagen. En esta etapa se realizaron procesamientos como conversión a escala de grises, este proceso se hizo mediante el promedio de los tres componentes del formato RGB.
3. Segmentación. En esta etapa de hizo la segmentación de la imagen, en la que se aplicó el método de Otsu. Este método consiste en separar la imagen principal del fondo, obteniendo como resultado una imagen binaria. También se aplicó la eliminación de ruido por medio de operadores morfológicos.
4. Representación y descripción. En este proceso se aplicaron varias etapas que se mencionan a continuación:
    - Entrenamiento. Las características de las 240 imágenes se almacenaron en la base de conocimiento del sistema para su posterior clasificación.
    - Las características que presentaron las imágenes se mencionan a continuación:
        - Color. Se utilizó el formato RGB. Las fotografías se capturaron dentro de un salón de clase con una iluminación alta. Los problemas

que hubo en la captura de imágenes fue la posición del objeto con respecto a su lado de iluminación.

- Factor de compacidad. Este factor es de gran importancia porque es invariante en desplazamiento, rotación y escala.
- Excentricidad. Es parecido al factor de compacidad y se da por que los objetos se parecen mucho a una elipse. Se define como el cociente entre la distancia focal *c* y el semieje mayor *a* (Ec. 1).

$$e = \frac{c}{a} \tag{1}$$

  - Cuando $e = 0$ se tiene el caso de que $a = b$ y es una circunferencia.
  - Si $e = 1$, el semieje mayor coincide con la distancia focal y el semieje menor es igual a 0. Será el caso de un segmento de recta.

  Donde,

  - e es igual a la excentricidad de la elipse.
  - c es la distancia focal de la elipse.
  - a es el semieje menor de la elipse.
  - b es el semieje mayor de la elipse.

5. Reconocimiento e interpretación. En esta etapa, el método utilizado se describe para la clasificación del objeto, así como los resultados obtenidos.

Para la clasificación se mencionó que se utilizaron tres tipos de características. El primer descriptor que fue utilizado es el color con formato RGB, el segundo descriptor fue el factor de compacidad y la excentricidad y el tercero fue la textura. Los pasos que se usaron en el proceso de clasificación son:

1. Analizar la clasificación para cada característica por separado.
2. Seleccionar las mejores características.

3. Llevar a cabo la clasificación con las características seleccionadas.
4. Si el objeto ya está clasificado termina la clasificación.
5. Llevar a cabo un filtro en la base de conocimientos cuando hay una confusión en la clasificación.
6. Eliminar las características empleadas.
7. Si hay más características que clasificar ir al paso 3, de lo contrario aquí termina.

El *K-means* en la clasificación por color, textura y forma alcanza al final de las pruebas un porcentaje de éxito de 90 % o superior.

Clasificación de grandes conjuntos de datos vía Máquinas de Vectores Soporte y aplicaciones en sistemas biológicos (Cervantes 2009), es el título de una tesis que se realizó en el Instituto Politécnico Nacional (México) en la que se hizo la aplicación de una mejora para la clasificación de datos utilizando máquinas de vectores de soporte (SVM).

La clasificación de datos y objetos es un proceso que crece exponencialmente a medida que la cantidad de datos va aumentando considerablemente. Este clasificador arroja resultados satisfactorios, pero el inconveniente, y donde es necesario poner énfasis, es en el costo computacional que se genera cuando la cantidad de datos u objetos que se quieren clasificar aumentan formidablemente. Entonces, dicho lo anterior, se obtiene que las SVM no sean adecuadas para la clasificación con grandes conjuntos de datos, ya que la matriz del *kernel* crece de forma cuadrática con el tamaño del conjunto de datos, provocando que el entrenamiento de las SVM sobre conjuntos de datos grandes sea un proceso muy lento.

Es por lo mencionado anteriormente que se propuso una mejora a este clasificador presentando tres algoritmos basados en las SVM que reducen considerablemente el tiempo de entrenamiento, recuperando los datos más importantes y eliminando aquellos que no son importantes.

Los algoritmos propuestos usan dos etapas de SVM. Una primera etapa, sobre un conjunto de datos pequeño con el objetivo de obtener un esbozo de la distribución de los vectores de soporte (VS) y recuperar los datos con más probabilidades de ser VS y una segunda etapa de SVM con el objetivo de mejorar el escenario de clasificación obtenido.

Los tres algoritmos para clasificación de grandes conjuntos de datos que son presentados son: SMO, Libsvm y SSVM. La principal novedad de los enfoques propuestos consiste en emplear agrupamiento o técnicas de seccionamiento con el objetivo de reducir el tiempo de entrenamiento de las SVM. Además se propone también un algoritmo de multiclasificación, la principal novedad del algoritmo propuesto con respecto a las tres primeras técnicas consiste en la implementación de un método de búsqueda de candidatos a SV entre el espacio de los SV. El algoritmo reduce aún más el conjunto de candidatos a vectores de soporte debido a que la búsqueda de candidatos a VS es restringida a un pequeño espacio.

La implementación y mejora de este clasificador perfecciona sustancialmente la etapa de entrenamiento, mejorando la principal limitación de las SVM del excesivo costo computacional sobre los enormes conjuntos de datos, no afectando la certeza de su clasificación y manteniendo un promedio de clasificación óptima de 96.81 %.

Extracción de características de ECG basadas en transformaciones no lineales y Wavelets (Montes, Guarin & Castellanos 2005), es el nombre de un artículo de investigación que se desarrolló en la Universidad de Colombia en la que se presentan tres métodos de extracción de características en señales ECG normales y con cardiopatía isquémica. Los métodos para la extracción de características que se utilizaron son: basados en mediciones de diagnóstico, la transformada de Wavelet y el análisis no lineal de componentes principales.

Con el fin de determinar las características que contribuyen de mejor manera con el modelo, se aplicaron dos técnicas de selección efectiva de características empleando métodos estadísticos con: pruebas de independencia estadística y combinación de técnicas estadísticas; esta última metodología conlleva a la aplicación subsecuente de análisis multivariado de varianza, análisis de varianza y análisis de correlación.

Para la evaluación de los extractores de características se utilizaron dos clasificadores: análisis de discriminante lineal (LDA) y máquinas de soporte vectorial (SVM).

Cuando se utilizó el clasificador de análisis de discriminante lineal, el extractor de características más efectivo fue el de análisis no lineal de componentes principales. En este

caso, el error obtenido de clasificación es de hasta el 0.22 %, contra 6.78 % en el caso de las Wavelets y 24.22 % en el caso de las mediciones de diagnóstico.

Con las máquinas de soporte vectorial se obtiene que las características más determinantes se obtienen empleando Wavelets aplicadas al latido, con un error de clasificación hasta del 0.1 %, contra 0.12 % en el caso del análisis no lineal de componentes principales y 5.11 % en el caso de las mediciones de diagnóstico.

Según el autor, las Wavelets son mejores debido a que se basan en técnicas de filtrado, por lo tanto, las componentes de alta frecuencia son suprimidas en el momento de aplicar la transformada, mientras que las técnicas no lineales implementadas actúan directamente sobre la información sin suprimir ningún tipo de componente frecuencial, por lo que se ven afectadas en presencia de ruido. En general los resultados obtenidos mediante LDA y SVM indican que al aplicar la extracción de características a un latido completo hay un mejor desempeño en la identificación de la cardiopatía isquémica.

Detección y seguimiento de objetos invariantes en color en una secuencia de imágenes (Toscano 2009), este trabajo es una tesis desarrollada en la Universidad del Mar campus Puerto Escondido que se basó en la detección de objetos en movimiento a partir de una secuencia de imágenes haciendo su seguimiento en una trayectoria lineal. Para solucionar este problema, al igual que en el presente trabajo de tesis, se utilizaron técnicas de procesamiento digital de imágenes, como escala de grises, filtros para eliminación de ruido y resta de imágenes para determinar la posición de un objeto.

En el desarrollo de esta tesis se aplicaron técnicas de procesamiento de imágenes. Las imágenes que se utilizaron en las pruebas fueron adquiridas en un ambiente de iluminación controlado. La metodología empleada para llevar a cabo la investigación constó de los pasos siguientes:

- Escala de grises. Todas las imágenes estaban en color, por lo tanto hubo necesidad de transformarlas cada imagen en una monocromática para facilitar su manipulación por lo algoritmo de procesamiento.
- Filtro de la mediana. Este algoritmo tomó parte importante en la mejora de las imágenes que no presentaron las características adecuadas para servir como entrada de los algoritmos.

- Filtro de binarización. Para llevar a cabo el objetivo general y poder realizar la resta de imágenes se necesito obtener su respectiva imagen binaria.
- Operadores de Sobel. En este punto se continuó con la eliminación de ruido, en este caso de falsos bordes que llevaran a resultados no esperados.
- Resta de imágenes. Aquí yace la principal solución del objetivo general. A través de esta técnica se puede determinar cuando un objeto ha cambiado o no su posición y por lo tanto seguir su trayectoria de movimiento, esto es, el objeto se detecta y se sigue utilizando la característica de la traslación.

  Cuando el software estuvo realizado y al aplicar pruebas con la base de datos de imágenes obtenidas se obtuvo un 99.7 % de efectividad según lo reportado por el autor.

Reconocimiento de rostros en un ambiente de iluminación controlado no invariantes a expresiones faciales (Pereyra 2011), es una investigación desarrollada en la Universidad del Mar campus Puerto Escondido en la que se desarrolló un software capaz de reconocer rostros de personas invariantes a sus expresiones faciales. Para resolver este problema se utilizó un clasificador basado en una red neuronal tipo perceptron, en este caso la red fue diseñada con cuatro neuronas. Para entrenar la red se utilizaron imágenes de 378x512 pixeles. Estas imágenes se obtuvieron en un ambiente controlado para una mejor aplicación de los algoritmos.

El software se desarrollo en lenguaje C++ utilizando como IDE Borland C++ Builder 5.0 el cual consta de tres módulos:

- Preprocesamiento, en este módulo se mejora la calidad de la imagen, como el suavizado, eliminando en gran medida el ruido y los algoritmos principales actúen en su forma más óptima.
- Aprendizaje, se encarga de entrenar la red neuronal moldeándola a las características de los rostros del entrenamiento. Las características de los rostros que se obtuvieron fueron las medidas de los rostros, la distancia entre los ojos y distancia entre el ojo y la nariz.

- Reconocimiento, aquí se hace el objeto principal de la investigación, con una imagen abierta desde un archivo se muestra el rostro de la persona reconocida. En base a las características utilizadas en el aprendizaje, el reconocimiento de los rostros puede realizarse aunque el rostro no esté en la misma posición, en el mismo lugar o se encuentre más alejado que en otras fotografías.

Al terminar el desarrollo y las pruebas del sistema el autor reporta un 92 % de efectividad en el índice de certeza de la red neuronal. Las pruebas toman en cuenta que las imágenes de los rostros de las personas se hayan tomado en un ambiente controlado y una buena iluminación.

2.2. Justificación

El desarrollo de software de este tipo tiene precios muy elevados, por lo tanto, el costo para la realización de este proyecto será menor a los que pudieran desarrollar grandes empresas dedicadas al desarrollo de software, ya que éste será el resultado de un proyecto de investigación.

El desarrollo de este tipo de proyectos también permitirá la incursión de otros proyectos que tengan relación con la segmentación de imágenes, la clasificación de objetos y sobre todo aquellos proyectos que partan de esta idea y lleven su aplicación a secuencias de video.

La clasificación de objetos y la extracción de características de los mismos objetos son importantes en la implementación de proyectos de investigación orientados a la visión por computadora, porque cuando se tienen este tipo de herramientas, hacer una investigación de las ya mencionadas facilita en gran medida el trabajo a realizar.

2.3. Planteamiento del problema

Una de las últimas tendencias en investigación orientadas a la inteligencia artificial, se enfoca en resolver problemas de inspección visual en imágenes digitales, los cuales a través de diversas técnicas de visión artificial y procesamiento digital de imágenes, verifican la calidad de frutas, verduras u otros comestibles, haciendo que el proceso de control de

calidad se haga de manera automática. Por otro lado también se utilizan en la clasificación de objetos, como pueden ser tornillos, partes de motores automotrices, entre otros.

Algo fundamental para que la inspección visual se pueda llevar a cabo, es la extracción de características de los objetos que estén involucrados en la escena que se desea inspeccionar, estas características pueden ser tamaño del objeto, o que este se encuentre en una posición o lugar diferente. Con base en estas características o propiedades entonces se aplican métodos para determinar si el objeto en estudio contiene los atributos necesarios para considerarse de buena calidad, o en su defecto considerarlo de mala calidad.

También se puede realizar la clasificación de los objetos por sus propiedades físicas y generar grupos o clases que compartan las mismas características. Uno de los principales problemas en este último caso, se da cuando el objeto cambia de tamaño, orientación o posición en la imagen adquirida. Para resolver este tipo de problemas, es necesario diseñar e implementar un sistema informático que permita la extracción de características evitando en lo posible los tres aspectos mencionados anteriormente. La idea anterior es el punto crucial a resolver, puesto que es necesario que se verifique que un sistema extractor de características arroje resultados satisfactorios por medio de un clasificador de objetos. Esto, para un sistema computacional representa, en términos informáticos, complejidad temporal dependiendo de cuán grande es la imagen que contiene los objetos para el tiempo que tarde en extraer las características y clasificar dichos objetos, y complejidad espacial, debido a que si la imagen es muy grande, los algoritmos que realicen esta tarea utilicen mucho espacio en memoria.

2.4. Objetivos

Objetivo general

Diseñar e implementar un sistema informático, que permita, a partir de una imagen, separar los diferentes objetos que se encuentren en ella, extraer sus características con los momentos invariantes de *Hu* y con base en esta información agrupar los objetos en diferentes clases, haciendo uso de un clasificador como el *K-means*.

Objetivos específicos

1. Investigar y documentar proyectos desarrollados con enfoques similares a la extracción de características y a la clasificación de objetos.
2. Obtener las características físicas (rotación, traslación y tamaño) de los objetos presentes en una imagen por medio del algoritmo de momentos invariantes de *Hu*.
3. Verificar los resultados que arroja el algoritmo de momentos invariantes de *Hu*, clasificando dichas características utilizando el clasificador *K-means*.

2.5. Alcances y límites

Alcances

A continuación se menciona cada uno de los objetivos específicos que se lograron desarrollar:

1. Se documentaron seis proyectos relacionados directamente con el desarrollo de la tesis presente, los mismos trabajos se refieren a clasificación de objetos y métodos de extracción de características.
2. Se logró la extracción de características de los objetos implementando el algoritmo de momentos invariantes de *Hu*.
3. Se logró clasificar los objetos contenidos en una imagen por medio de las características extraídas.

Limites

1. No realiza la clasificación de objetos a partir de secuencias de video.
2. El formato de entrada de las imágenes que contienen los objetos a clasificar deben ser jpg, png y gif.
3. El valor de entrada K del algoritmo *K-means* sólo puede proporcionado por el usuario, puesto que no hay un algoritmo que lo calcule automáticamente.
4. Puesto que no se implementó un algoritmo de encadenamiento de bordes, si la imagen de entrada es de mala calidad, al realizar la etapa de preprocesamiento pueden quedar objetos o partes de la imagen aislados.

CAPÍTULO 3. MARCO TEÓRICO

En este capítulo se describe cada uno de los conocimientos necesarios para llevar a cabo este trabajo de tesis. Los temas se definen en el siguiente orden: Escala de grises, Binarización, Negativo de una imagen, Etiquetado de regiones, Momentos invariantes de *Hu* (métodos para extraer las características de las imágenes), y por último el Clasificador *K-means*, utilizado para clasificar los objetos con base en las características extraídas.

3.1. Procesamiento digital de imágenes

El Procesamiento Digital de Imágenes (PDI) consta de un proceso generalizado que se sigue para resolver un problema específico. Este método inicia con la captura del mundo real a través de una cámara para así obtener una imagen digital y termina con la aplicación de interés (Fig. 3.1). La primera etapa se denomina adquisición de las imágenes de una escena tridimensional, hecho lo anterior se puede o no seguir con el proceso de

segmentación de la imagen con el propósito de preparar la imagen y obtener sólo la información útil de la imagen, tal como los bordes o regiones que la forman. Después, con el proceso de descripción se obtienen las propiedades de la imagen a través de sus características. Terminados los pasos anteriores, se obtiene una estructura de la escena tridimensional para seguir con su aplicación.

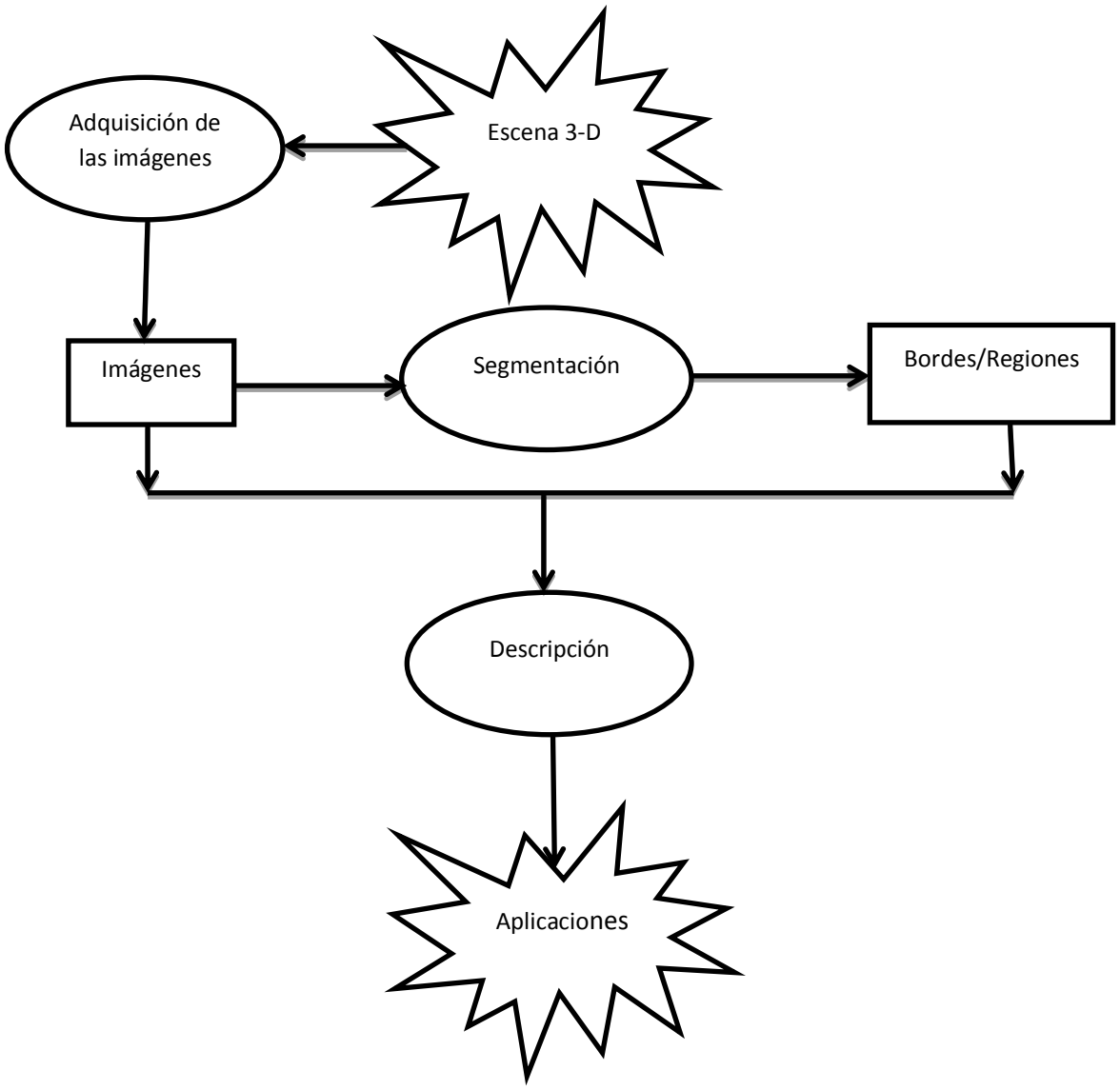
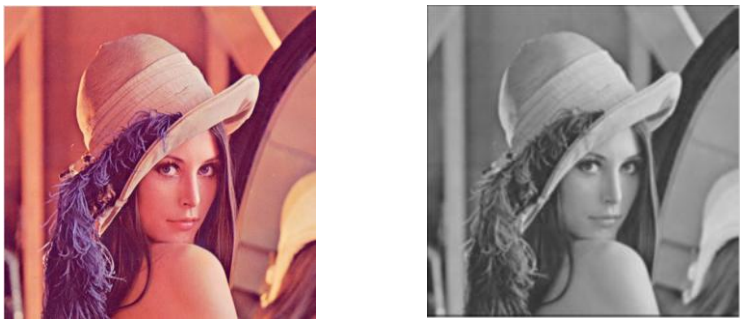


Figura 3.1. Diagrama de bloques mostrando el proceso completo del Procesamiento Digital de Imágenes (Pajares & De la Cruz 2008).

3.2. Escala de grises

Una imagen en escala de grises es el resultado de obtener la media de las tres matrices de colores (rojo, verde y azul) de la imagen, en consecuencia cada pixel en la imagen resultante de esta operación está en el rango de 0 a 255 (Pereyra 2011). "Cada pixel de una imagen en escala de grises tiene un valor de brillo que oscila entre 0 (negro) y 255 (blanco)" (Sánchez-Maza 2001, citado por Toscano 2009).

El motivo de la conversión es tener una versión más compacta de la imagen, en la que sea más fácil de aplicar los algoritmos de PDI, cabe mencionar que este proceso no altera la cantidad de iluminación que posee la imagen (Pereyra 2011). En la figura 3.2 se muestra un ejemplo de una imagen a color convertida a una imagen en escala de grises a través del promedio de sus componentes RGB.



(a) (b)

Figura 3.2. Conversión de una imagen en color a una en escala de grises. (a) Imagen original, (b) Imagen en escala de grises.

3.3. Binarización

Consiste en obtener una imagen de salida binaria, es decir; crear una imagen con sólo dos valores de intensidad de color, blanco y negro (255 y 0, respectivamente). El nivel de transición está dado por el parámetro de entrada *p1* (Ec. 2), el cual se conoce como operador umbral, y es el que especificará hasta que valores de intensidad de los colores se transformarán en ceros y hasta que parte serán 255 (Pajares & De la Cruz 2008). La función de transformación se puede ver en la ecuación 2, en la figura 3 se muestra un ejemplo de esta transformación.

$$q = \begin{cases} 0\ para\ p\ \leq p1 \\ 255\ para\ p\ > p2 \end{cases} \quad (2)$$

Donde:

- p es el valor actual de la intensidad de nivel de gris de la imagen.
- q es el valor de la imagen binarizada.

(a) (b)

Figura 3.3. Binarización de una imagen en escala de grises. (a) Imagen en escala de grises, (b) Imagen binarizada.

3.4. Negativo

Este proceso consiste en crear una imagen de salida que es la inversa de la imagen de entrada (Pajares & De la Cruz 2008). El negativo de una imagen digital se obtiene empleando la función de transformación de la ecuación 3. La idea es invertir el orden de blanco al negro, de forma que la intensidad de la imagen de salida disminuya conforme la intensidad de la imagen de entrada aumente (Gonzalez & Woods 1992, Pajares & De la Cruz 2008). En la figura 3.4 se muestra un ejemplo de esta transformación.

$$q = L - p \quad (3)$$

Donde:

- q es el valor actual del nivel de gris de salida de la imagen.
- L es el número de niveles de gris
- p es el valor actual del nivel de gris de entrada de la imagen.

(a) (b)

Figura 3.4. Negativo de una imagen. (a) Imagen binarizada, (b) Negativo de la imagen.

3.5. Etiquetado de regiones

Cuando los objetos presentes en una imagen se han separado del fondo (segmentación) todos los objetos tienen un mismo nivel de gris, entonces es necesario diferenciarlos unos con otros, este proceso se llama etiquetado de regiones (*labelling*, en inglés) y consiste en que a cada objeto o región de la imagen tenga un nivel de gris diferente (De la Escalera 2001).

El esquema matemático mencionado por Chacón (2007) que representa el conjunto de regiones de la imagen se describe en los cinco puntos siguientes:

1. $\bigcup_{i=1}^{n} I(x,y) = I(x,y)$, que indica que toda la imagen se descompone en regiones distintas.
2. $I_i(x,y)$ para $i = 1, 2, \dots, n$ es una región conectada, cada pixel de una región debe cumplir un criterio de conectividad con los demás pixeles que forman la misma región.
3. $I_i(x,y) \cap I_j(x,y) = \emptyset \; \forall i \; y \; j, i \neq j$, indica regiones disjuntas. Los pixeles en esa región sólo pueden formar parte de esa región.
4. $P(R_i) = VERDADERO \; para, i = 1, 2, \dots, n$, los pixeles dentro de esa región deberán satisfacer el predicado o conjunto de predicados, $P(R_i)$ que se utilizan para distinguir las regiones. Por ejemplo, un predicado puede ser "el valor del pixel está dentro del rango de tonos de gris (l_1, l_2)", donde l_1 y l_2 corresponde al rango menor y mayor de tonos de gris, respectivamente.

5. $P(R_i) = FALSO\ para, i \neq j$, esta condición indica la distinción entre regiones, es decir, la característica o características definidas con el o los predicados de una región no pueden ser iguales en otra región.

Para lograr el proceso mencionado se siguen estos sencillos pasos propuestos por De la Escalera (2001):

1. Se inicia por el pixel superior izquierdo que esté a nivel de gris alto y se le asocia la etiqueta 1, para identificarlos como primer objeto.
2. Se examinan los vecinos para ver si están a nivel alto, si lo están recibirán la etiqueta 1.
3. Cuando un pixel no sea vecino y este a nivel alto se le asociará la etiqueta 2, y así sucesivamente.

Una ventaja de este algoritmo es que es sencillo obtener el número de regiones, porque sólo hay que obtener el histograma de la imagen y ver el nivel de gris más alto.

3.6. Momentos invariantes de *Hu*

Los momentos invariantes pretenden extraer características para poder reconocer los objetos aunque no se encuentren siempre en la misma posición, estén girados o su tamaño sea distinto (De la Escalera 2001). Para ello se parte de los **momentos ordinarios**, donde, para una función continua bidimensional $f(x,y)$, el momento de orden $(p+q)$ está definido en la ecuación 4.

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p\, y^q f(x,y)\, dx\, dy \tag{4}$$

La ecuación 4 representada en forma digital se visualiza en la ecuación 5 (Gonzalez & Woods 1992).

$$m_{pq} = \sum_x \sum_y (x)^p (y)^q f(x,y) \tag{5}$$

Donde,

- La suma de p y q representa el orden del momento calculado.

Para conseguir la invariancia se le resta a las coordenadas de cada punto el centro de gravedad (Ec. 6), lo que da como resultado los **momentos centrales**.

$$\mu_{pq} = \sum_{x}\sum_{y}(x-\bar{x})^{p}(y-\bar{y})^{q}f(x,y) \tag{6}$$

Donde $\bar{x}$ y $\bar{y}$ están definidas en la ecuación 7.

$$\bar{x} = \frac{m_{10}}{m_{00}} \qquad y \qquad \bar{y} = \frac{m_{01}}{m_{00}} \tag{7}$$

Donde:

- $\bar{x}$: Representa el centro de gravedad o centroide del objeto en el eje de las abscisas.
- $\bar{y}$: Representa el centro de gravedad o centroide del objeto en el eje de las ordenadas.
- m_{10}: Representa el momento ordinario de primer orden, cuando p es igual a 1 y q es igual a 0 en l ecuación 7.
- m_{01}: Representa el momento ordinario de primer orden, cuando p es igual a 0 y q es igual a 1 en l ecuación 7.
- m_{00}: Es el momento de orden cero y representa el área del objeto.

Los momentos centrales permiten reconocer figuras dentro de una imagen independientemente de su posición. El momento central de orden 0 es igual al área del objeto (Ec. 8), los momentos centrales de orden 1 son por definición 0 (Ec. 9-10), los de orden 2 forman la matriz de rotación para calcular de ahí el ángulo de rotación y la excentricidad del objeto (Ec. 11-13), y los momentos centrales de orden 3 se utilizan para calcular los momentos invariantes (Ec. 14-17). Entonces, los momentos centrales de hasta orden 3 quedarían como:

$$\mu_{10} = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^0 f(x,y)$$

$$= m_{10} - \frac{m_{10}}{m_{00}}(m_{00})$$

$$= 0$$

$$\mu_{11} = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^1 f(x,y)$$

$$= m_{11} - \frac{m_{10} m_{01}}{m_{00}}$$

$$\mu_{20} = \sum_x \sum_y (x-\bar{x})^2 (y-\bar{y})^0 f(x,y)$$

$$= m_{20} - \frac{2m_{10}^2}{m_{00}} + \frac{m_{10}^2}{m_{00}} = m_{20} - \frac{m_{10}^2}{m_{00}}$$

$$\mu_{02} = \sum_x \sum_y (x-\bar{x})^0 (y-\bar{y})^2 f(x,y)$$

$$= m_{02} - \frac{2m_{01}^2}{m_{00}}$$

$$\mu_{30} = \sum_x \sum_y (x-\bar{x})^3 (y-\bar{y})^0 f(x,y)$$

$$= m_{30} - 3\bar{x} m_{20} + 2\bar{x}^2 m_{10}$$

$$\mu_{12} = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^2 f(x,y)$$

$$= m_{12} - 2\bar{y} m_{11} - \bar{x} m_{02} + 2\bar{y}^2 m_{10}$$

$$\mu_{21} = \sum_x \sum_y (x-\bar{x})^2 (y-\bar{y})^1 f(x,y)$$

$$= m_{21} - 2\bar{x} m_{11} - \bar{y} m_{20} + 2\bar{x}^2 m_{01}$$

$$\mu_{03} = \sum_x \sum_y (x-\bar{x})^0 (y-\bar{y})^3 f(x,y)$$

$$= m_{03} - 3\bar{y} m_{02} + 2\bar{y}^2 m_{01}$$

En resumen, simplificando a su mínima expresión los **momentos centrales de hasta orden 3** quedan de la manera siguiente:

$$\mu_{00} = m_{00} \tag{8}$$

$$\mu_{10} = 0 \tag{9}$$

$$\mu_{01} = 0 \tag{10}$$

$$\mu_{20} = m_{20} - \bar{x}m_{10} \tag{11}$$

$$\mu_{02} = m_{02} - \bar{y}m_{01} \tag{12}$$

$$\mu_{11} = m_{11} - \bar{y}m_{10} \tag{13}$$

$$\mu_{30} = m_{30} - 3\bar{x}m_{20} + 2m_{10}\bar{x}^2 \tag{14}$$

$$\mu_{12} = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2m_{10} \tag{15}$$

$$\mu_{21} = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2m_{01} \tag{16}$$

$$\mu_{03} = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2m_{01} \tag{17}$$

El siguiente paso es obtener los momentos invariantes a la rotación y a la escala. Para ello se definen primero los **momentos centrales normalizados** (Ec. 18 y Ec. 19).

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma}} \tag{18}$$

Donde, η_{pq} representa el momento central normalizado independiente del tamaño del objeto.

Siendo

$$\gamma = \frac{p+q}{2} + 1, \tag{19}$$

para $(p + q) = 2, 3, \ldots, n$.

Donde, γ representa el factor de normalización.

Al dividir cada momento por el área (momento de orden cero) se consigue que sean valores independientes de la escala. De los momentos segundo y tercero se pueden derivar un conjunto de siete momentos invariantes (Ec. 20-26).

$$\varphi_1 = n_{20} + n_{02} \tag{20}$$

$$\varphi_2 = (n_{20} + n_{02})^2 + 4n_{11}^2 \tag{21}$$

$$\varphi_3 = (n_{30} + 3n_{12})^2 + (3n_{21} - n_{03})^2 \tag{22}$$

$$\varphi_4 = (n_{30} + 3n_{02})^2 + (n_{21} + n_{03})^2 \tag{23}$$

$$\begin{aligned}\varphi_5 = {} & (n_{30} - 3n_{12})(n_{30} + n_{12})\{(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2\} \\ & + (3n_{21} + n_{03})(n_{21} + n_{03})\{3(n_{03} + n_{12})^2 - (n_{21} + n_{03})^2\}\end{aligned} \tag{24}$$

$$\begin{aligned}\varphi_6 = {} & (n_{20} + n_{02})\{(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2\} \\ & + 4n_{11}(2n_{30} + n_{12})(n_{21} + n_{03})\end{aligned} \tag{25}$$

$$\begin{aligned}\varphi_7 = {} & (3n_{21} - n_{03})(n_{30} + n_{12})\{(n_{30} + n_{12})^2 - 3(n_{21} + n_{03})^2\} \\ & - (3n_{21} + n_{03})(n_{21} + n_{03})\{3(n_{30} + n_{12})^2 - (n_{21} + n_{03})^2\}\end{aligned} \tag{26}$$

Donde φ_i representa el momento invariante a escala, rotación y traslación.

3.7. Clasificador *K-means*

MacQueen (1967), presentó un trabajo en el que se describía un proceso para repartir una población de N-dimensiones en conjuntos de *K* en la base de una muestra. Al proceso lo llamó *K-means*. En este método se asigna la clase mayoritaria entre los *k* vecinos más próximos (Hernández, Ramírez & Ferri 2004).

"Este algoritmo se puede programar fácilmente y es computacionalmente económico, por lo que es factible de procesar muestras muy grandes en un equipo digital" (Maqueen 1967).

El algoritmo parte del conocimiento a priori del número de clases pero no de sus características. Según De la Escalera (2001) el algoritmo mencionado consta de cinco etapas principales:

1. De entre la serie a clasificar, O_1, O_2, O_3, ..., O_n se escogen de forma arbitraria tantos elementos como números de clases y se considera que constituyen los centroides (valores prototipo) de cada clase.
2. El resto de los elementos se asigna a cada clase siguiendo el criterio de mínima distancia a los centroides antes elegidos.
3. Se calculan los centroides de cada clase. Para ello se toman la media de todos los valores dentro de cada clase.
4. Se vuelven a asignar, ahora todos los elementos, a cada clase con el criterio de mínima distancia.
5. Se vuelven a calcular los centroides, si no varían se considera que el algoritmo ha terminado, no ocurriendo este resultado se vuelve a repetir el paso anterior.

Este algoritmo, para encontrar el objeto más cercano a un centroide se apoya en obtener la menor distancia, para ello se utilizará la distancia euclidiana.

Distancia euclidiana. Es la distancia de la recta que une dos puntos en la n-dimensión (Lehmann 2004).

La distancia euclidiana entre los puntos $X = (x_1, x_2, x_3, \dots, x_n)$ y $Y = (y_1, y_2\ y_3, \dots, y_n)$ se define como:

$$d(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + \cdots + (x_n - y_n)^2}$$

Entonces, la fórmula para calcular dicha distancia se da en la ecuación 17.

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{27}$$

3.8. Java

Java inició como un lenguaje de programación pensado para electrodomésticos. En 1991 Sun Microsystems patrocinó un proyecto interno de investigación al que se le denominó *Green*, en el que se desarrolló un lenguaje de programación basado en C++ y se le llamó Oak. *Sun*, al enterarse de que ya había un lenguaje con el mismo nombre lo cambió a Java, este último se inspiró en una variedad de café cuando la gente de *Sun* visitó una cafetería (Deitel & Deitel 2004).

El proyecto estuvo a punto de cancelarse debido a que los dispositivos electrodomésticos inteligentes no se desarrollaban tan rápido como *Sun* había pensado, pero en 1993 estalló la popularidad de la Web y la gente de *Sun* se dio cuenta inmediatamente del potencial de Java para agregar contenido dinámico y animaciones a las páginas. En 1995 se anunció formalmente Java. En la actualidad se usa para desarrollar aplicaciones a gran escala, para mejorar las aplicaciones de servidores de la Web y aplicaciones para móviles (Deitel & Deitel 2004).

Entorno de desarrollo: NetBeans

NetBeans es un entorno de desarrollo integrado para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación, por ejemplo C++. Existe además un número importante de módulos para extender el NetBeans. Además de ser un producto libre y gratuito sin restricciones de uso (Oracle 2011).

En este capítulo se describieron los conocimientos que se fueron necesarios para resolver el problema mencionado en la introducción. A continuación se presenta el siguiente capítulo que trata sobre el desarrollo de la resolución del problema.

CAPÍTULO 4. DESARROLLO DEL TEMA

Este capítulo tiene como propósito validar el funcionamiento del sistema desarrollado. Tal objetivo se verifica por medio de cinco pruebas con diferentes objetos. Las pruebas consistirán en la clasificación de llaves y monedas, tornillos, letras, números, así como artículos varios. Además de determinar la robustez del extractor de características empleando los momentos invariantes de *Hu*, como son: rotación, traslación y tamaño de los objetos.

4.1. Análisis

Para el desarrollo de este software, antes se definió cuántos y cuáles serían los módulos a implementar. También se analizó cómo estaría distribuida la interfaz gráfica, los menús correspondientes y cada uno de los controles.

Los distintos módulos que se implementaron quedaron de la manera siguiente:

- Archivo
    - Abrir
    - Nuevo
    - Guardar
    - Cerrar
    - Exportar
    - Salir
- Edición
    - Copiar
    - Cortar
    - Pegar
- Procesamiento
    - Escala de grises
    - Binarización
    - Negativo
- Segmentación
    - Etiquetado de regiones
    - Momentos invariantes de *Hu*
- Clasificación de objetos
    - Manual
    - Automática

También se determinó cómo sería introducido el valor *k*, el cual es una parte esencial en la clasificación de los objetos y parte importante en este trabajo de tesis.

En lo que respecta a la parte gráfica, se estableció la implementación de una barra de herramientas, la que facilitaría el uso de las funciones descritas con anterioridad.

4.2. Diseño

En este apartado se muestra el proceso que se siguió para desarrollar el clasificador de objetos.

En el proceso descrito en la figura 4.1, se muestran los pasos generales del procesamiento digital de imágenes y visión por computador, por lo tanto es la metodología que se utilizó.

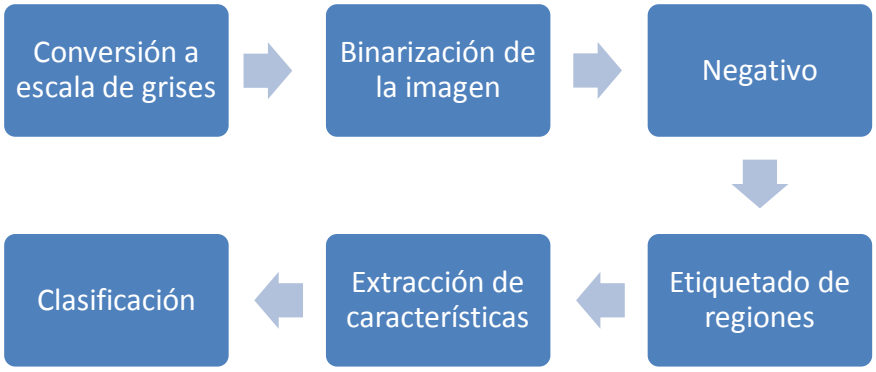


Figura 4.1. Pasos generales para la metodología de solución. Se muestra el proceso que se siguió para el desarrollo del clasificador.

Escala de grises

En este subproceso se convirtió una imagen en sus tres componentes RGB a un solo componente para visualizarla como escala de grises. En el algoritmo 4.1 se muestran los pasos que se siguieron para llevar a cabo este módulo.

```
Inicio
      CoversionEscalaGrises (Imagen)
      1. Extraer los componentes RGB de la matriz
      2. Obtener el promedio de los componentes extraídos
      3. Crear una nueva imagen con el promedio de los
         componentes extraídos
Fin
```

Algoritmo 4.1. Escala de grises. Algoritmo para convertir una imagen a color en una a escala de grises.

Binarización

El proceso de binarizar una imagen es hacer que ésta sólo tenga dos niveles de colores, blanco y negro. Esto se logra mediante la utilización de un umbral. El valor de umbral

juega un papel importante puesto que no todas las imágenes necesitan el mismo valor de umbral, éste dependerá de qué imagen se trate y qué calidad de color presente.

En el algoritmo 4.2 se puede apreciar cómo se logró el proceso de binarización de imágenes.

```
Inicio
      Binarizar (Imagen, Umbral)
      Si Imagen (i, j) es mayor que Umbral
            Convertir en 0
      Si_no
            Convertir en 255
      Fin-si
Fin
```

Algoritmo 4.2. Binarizar imagen. Se muestra el proceso para binarizar una imagen en escala de grises.

Negativo de la imagen

Esta etapa invierte los valores de la imagen binaria, es decir, aquellos valores que son blancos los convierte en negros y los negros en blancos. Al finalizar el etiquetado de regiones los objetos a clasificar podrían haber quedado de color blanco, y una condición del sistema desarrollado es que deben ser negros. Lo anterior se soluciona obteniendo el negativo de dicha imagen con el algoritmo 4.3.

```
Inicio
      Negativo (Imagen)
      Si Imagen (i, j) es igual que 255
            Convertir en 0
      Si_no
            Convertir en 255
      Fin_si
Fin
```

Algoritmo 4.3. Negativo de una imagen. Se muestra el proceso para obtener el negativo de la imagen.

Etiquetado de regiones

En esta etapa se colocó una etiqueta a cada objeto de la imagen y así diferenciarlos de los demás; es decir, se logró separar cada uno de los objetos que conforman la imagen. Esto con el fin de aplicar los momentos invariantes de *Hu* a cada imagen por separado. Para alcanzar esto se utilizó el algoritmo 4.4.

```
Inicio
      EtiquetadoDeRegiones ()
      1. Se inicia por el pixel superior izquierdo que este a
      nivel de gris alto y se le asocia la etiqueta 1.
      2. Si los vecinos están a nivel alto se le asigna la
      etiqueta 1.
      3. Cuando un pixel no sea vecino y este a nivel alto se
      le asociará la etiqueta 2 y así sucesivamente.
Fin
```

Algoritmo 4.4. Etiquetado de regiones.

Una ventaja de este algoritmo es que es sencillo obtener el número de regiones, porque sólo hay que obtener el histograma de la imagen y ver el nivel de gris más alto.

Momentos invariantes de *Hu*

En este punto se implementó el proceso matemático de obtención de los momentos de una imagen, los cuales fueron invariantes a rotación, traslación y escala del objeto (Ec. 10-16).

El algoritmo 4.5 muestra el proceso de obtención de los siete momentos invariantes de *Hu*.

```
Inicio
      MomentosInvariantesHu ()
      1. Obtener los momentos centrales de la imagen
      2. Obtener los momentos de tercer orden
      3. Obtener los momentos centrales normalizados
Fin
```

Algoritmo 4.5. Momentos invariantes de *Hu*.

Clasificador de los objetos: *K-means*

En esta etapa se implementó el clasificador de los objetos extraídos en el etiquetado de regiones (Algoritmo 4.4) a partir de las características extraídas con los momentos invariantes de *Hu* (Algoritmo 4.5). Esta clasificación se conoce como no supervisada, puesto que a partir de un valor *k* y *k* centroides iniciales realizará la clasificación de acuerdo a los datos obtenidos en el proceso de extracción de características.

Para llevar a cabo el proceso descrito en el párrafo anterior hubo que apoyarse en el algoritmo 4.6. Hay que recordar que el valor *k* debe ser introducido dependiendo de cuántas clases haya.

```
Inicio
      ClasificadorKMeans ()
      1. De entre la serie de objetos a clasificar, O1, O2,…, On, se
         escogen tantos elementos como números de clases.
      2. El resto de los elementos se asigna a cada clase siguiendo
         el criterio de mínima distancia a los centroides antes
         elegidos.
      3. Se calculan los centroides de cada clase. Para ello se toma
         la media de todos los valores dentro de cada clase.
      4. Se vuelven a asignar, ahora todos los elementos, a cada
         clase con el criterio de mínima distancia.
      5. Se vuelven a calcular los centroides, si no varían, se
         considera que el algoritmo ha terminado, sino, se vuelve a
         repetir el paso 3.
Fin
```

Algoritmo 4.6. Clasificador de objetos, *K-means*.

4.3. Implementación

Para llevar a cabo la etapa de implementación de este trabajo de tesis hubo que apoyarse del siguiente equipo de cómputo con las características que se mencionan a continuación:

- Dell Optiplex 745.
- Procesador Pentium D 3.0 GHz.
- 1 GB de RAM.
- Sistema Operativo Windows XP SP3.

El lenguaje de programación en que se desarrolló esta aplicación fue Java, específicamente la versión del JDK fue la 6 con la actualización 18. Para una mayor elegancia del código fuente se utilizó el entorno de desarrollo integrado NetBeans 6.9.1 desarrollado por Oracle, Inc.

4.4. Pruebas y resultados

Descripción de los conjuntos de datos utilizados en las pruebas

Para validar y verificar el funcionamiento del software desarrollado se utilizaron cinco imágenes como prueba, las cuales, presentan propiedades de rotación, traslación y escala de los momentos invariantes de *Hu*.

A continuación se hace un resumen de las pruebas realizadas:

1. Prueba 1. La imagen que se utilizó (Fig. A.1) tiene 10 objetos, de los cuales cinco son llaves y cinco monedas.
2. Prueba 2. En esta prueba la imagen (Fig. A.2) constó de nueve objetos, cinco pernos y cuatro tuercas.
3. Prueba 3. Se utilizó una imagen (Fig. A.3) con herramientas de trabajo con ocho objetos, dos pinzas, dos martillos, dos cucharas y dos serruchos.
4. Prueba 4. La imagen (Fig. A.4) utilizada muestra nueve imágenes, en las que contenían números. Los números contenidos son: uno, tres y cinco, con una cantidad de apariciones de tres por cada número.
5. Prueba 5. Esta imagen (Fig. A.5) constó de quince objetos, tres estrellas, cuatro círculos, tres rectángulos, cinco triángulos.

Prueba 1

En esta prueba se tiene como entrada una imagen (Fig. 4.2) compuesta por dos tipos de objetos (llaves y monedas), se hará la clasificación de éstos con base en las características extraídas con los momentos invariantes de *Hu*. Lo que se espera como resultado es que la clasificación de los objetos se realice al 100 %, es decir, que cada objeto se asocie a su respectiva clase. La imagen a probar contiene un conjunto de diez objetos en total, cinco llaves y cinco monedas (Fig. 4.2).

Figura 4.2. Llaves y monedas. Se presentan diez objetos, cinco llaves y cinco monedas.

En esta primera prueba se indicarán todos los pasos necesarios hasta llegar a la clasificación de los objetos. A partir de la prueba 2 sólo se describirá la prueba correspondiente y se indicará la referencia de donde encontrar los pasos en extenso.

1. Conversión a escala de grises. En esta etapa se persigue el objetivo de convertir el formato RGB en una imagen monocromática para la aplicación de los algoritmos posteriores, y como se puede observar, la figura 4.2 se encuentra en escala de grises y por lo tanto esta fase no será necesaria, si se intentará convertir a escala de grises una imagen que ya se encuentra en ese modo se recibirá un mensaje de notificación por parte de la aplicación diciendo que la imagen ya está en modo monocromático. En el caso de que la imagen estuviera a color, en la salida de este módulo se obtendría una imagen en escala de grises.
2. Binarizar la imagen. Este módulo tiene como entrada una imagen en escala de grises, en este caso es la figura 4.2. En la figura 4.3 se muestra el resultado de binarizar la imagen de la figura 4.2 con un umbral de 183.

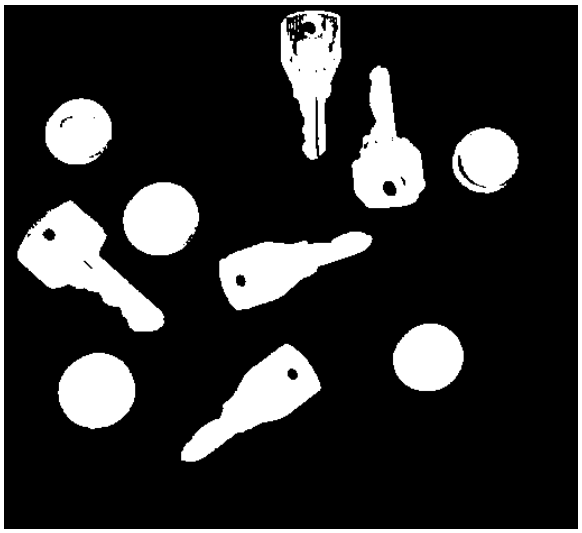


Figura 4.3. Llaves y monedas en binario. El umbral que se utilizó fue de 183.

3. Negativo de la imagen. Puesto que el software se desarrolló con la característica de que los objetos a clasificar sean de color negro, es necesario obtener el negativo de la imagen (Fig. 4.4), ya que se puede apreciar en la figura 4.3 que los objetos son blancos.

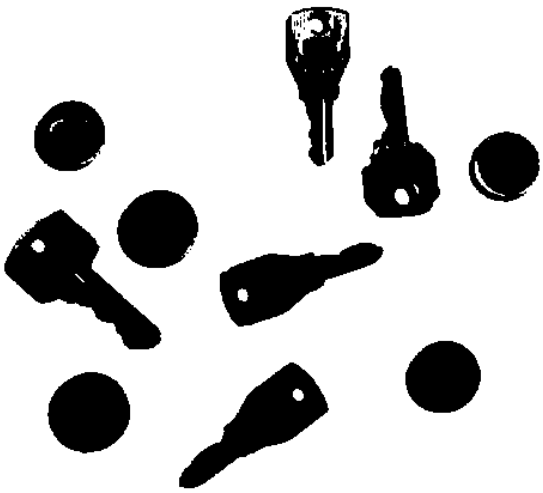


Figura 4.4. Negativo de llaves y monedas. Se obtiene el negativo para hacer que los objetos se vuelvan negros y el fondo blanco.

4. Etiquetado de regiones. El objetivo de esta técnica es separar los objetos que componen la imagen original en sus partes individuales. Este módulo tiene como entrada una imagen binaria compuesta por distintos objetos de color negro (Fig. 4.4), la salida que se obtiene es la separación de los objetos en archivos de imagen independientes (Fig. 4.5).

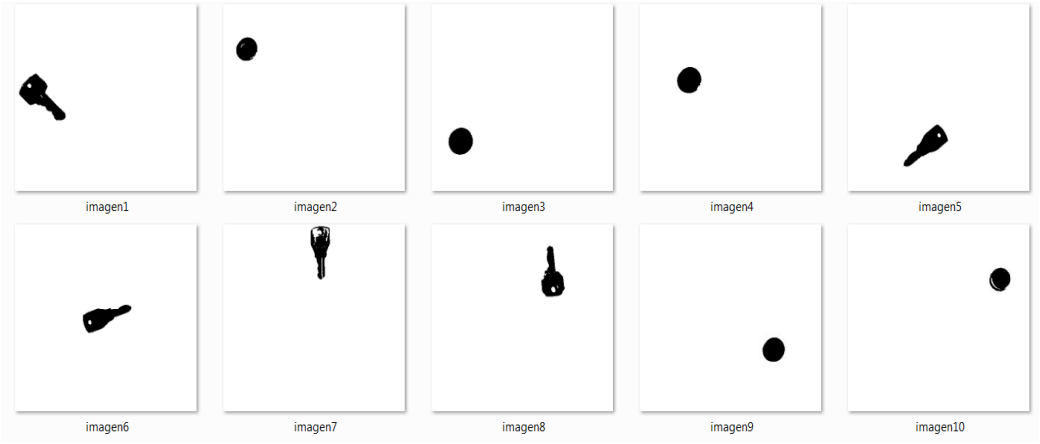


Figura 4.5. Objetos separados. Colección de todos los objetos contenidos en la figura 4.2.

5. Extracción de características. Una vez que se obtienen las imágenes por separado, las cuales son la entrada de esta fase, se calculan los momentos invariantes de *Hu*

(Ec. 10-16). El resultado de esta operación se muestra en la figura 4.6, en donde la columna uno pertenece al momento uno, la columna dos al momento dos y así hasta el momento siete que le corresponde la columna siete. La fila uno corresponde a la imagen uno, la fila dos a la imagen dos, y así sucesivamente hasta la última imagen.

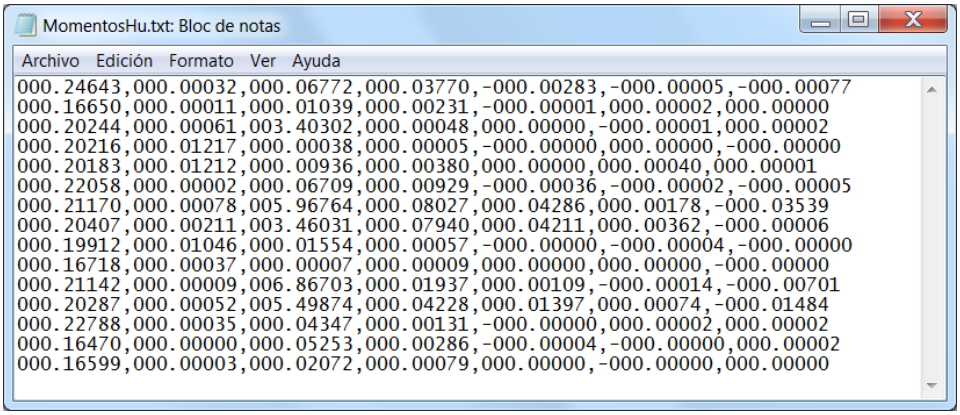


Figura 4.6. Momentos invariantes de *Hu*, prueba 1. Se visualizan los siete momentos invariantes de *Hu* de los objetos localizados. El umbral utilizado fue de 185.

6. Clasificación de objetos. Este módulo tiene como entrada los momentos invariantes de *Hu* extraídos en el paso 5 y mostrados en la figura 4.6. En los resultados mostrados se esperó que las llaves y las monedas se agruparan en clases diferentes, tales resultados fueron satisfactorios al visualizarse que cada objeto se asoció a su clase de manera inequívoca (Tabla I). Para lograr que la clasificación fuera satisfactoria, se ejecutó el algoritmo *K-means* con un valor para *k* de 2.

Tabla I. Objetos clasificados, prueba 1. Se muestran los diez objetos asociados a su respectiva clase.

Clase 1	Clase 2
	
	

Tabla I. Objetos clasificados, prueba 1. Se muestran los diez objetos asociados a su respectiva clase (continuación).

Clase 1	Clase 2
	
	
	

Por lo anterior, se concluye que esta prueba reporta un porcentaje de certeza de 100 % y un porcentaje de error de 0 %.

Prueba 2

Esta prueba consiste en clasificar pernos y tuercas contenidas en una imagen (Fig. 4.7). Como se ha mencionado la prueba realizará la clasificación de dos clases de objetos. La imagen cuenta con nueve objetos, cinco pernos y cinco tuercas.

En esta prueba se espera que cada perno y cada tuerca se asocien a una clase deferente.

Figura 4.7. Tuercas y pernos. Se muestra el conjunto de nueve objetos, cinco tuercas y cuatro pernos.

Los resultados obtenidos al aplicar los procesos de etiquetado de regiones, extracción de características (Fig. 4.8) y clasificación de objetos (utilizando un valor *k* de 2), se ilustran en la Tabla II.

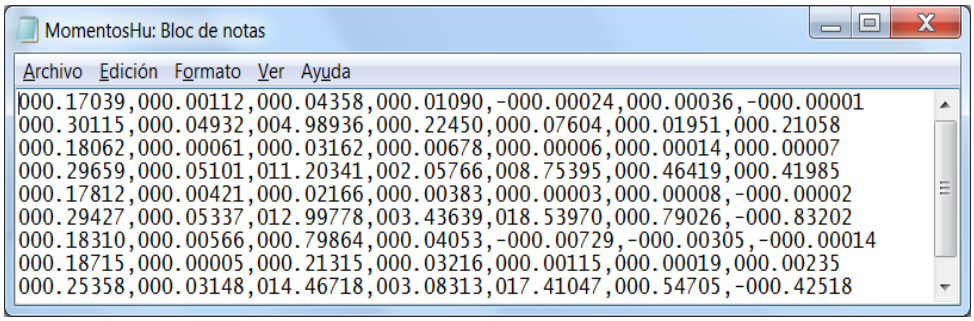


Figura 4.8. Momentos invariantes de *Hu*, prueba 2. Características de cada uno de los objetos etiquetados de la figura 5.6. Se utilizó un umbral de 225 para una mejor binarización.

Tabla II. Objetos clasificados, prueba 2. Cinco tuercas que pertenecen a la clase 1 y cuatro pernos que pertenece a la clase 2.

Clase 1	Clase 2
	
	
	
	
	

Con base en los resultados mostrados de la tabla II se puede observar que los objetos se asociaron a su clase correspondiente. Con lo anterior se concluye que en esta prueba, siendo que cada objeto se asoció a la clase esperada, se tiene un porcentaje de certeza de 100 %.

Prueba 3

Esta prueba se basó en la clasificación de herramientas de trabajo, como son: cucharas de albañil, serruchos, martillos y pinzas. La imagen cuenta con ocho objetos, dos de cada tipo de herramienta de los mencionados anteriormente (Fig. 4.9).

Figura 4.9. Herramientas de trabajo. Se muestran dos objetos de cada herramienta de trabajo, dos martillos, dos serruchos, dos pinzas y dos cucharas de albañil.

En este proceso se espera que el software haga la clasificación de tal manera que en una clase coloque los martillos, en otra las pinzas, en otra las cucharas y en una última clase los serruchos.

Después de haber aplicado los procesos necesarios para llevar a cabo la clasificación de los objetos como es conversión a escala de grises, binarización (es necesario aclarar que se utilizó un umbral de 230 para una mayor delineación de los objetos), etiquetados de regiones, extracción de características (Fig. 4.10) y clasificación con un valor para *k* de 4, se obtuvo el resultado que se muestra en la tabla III.

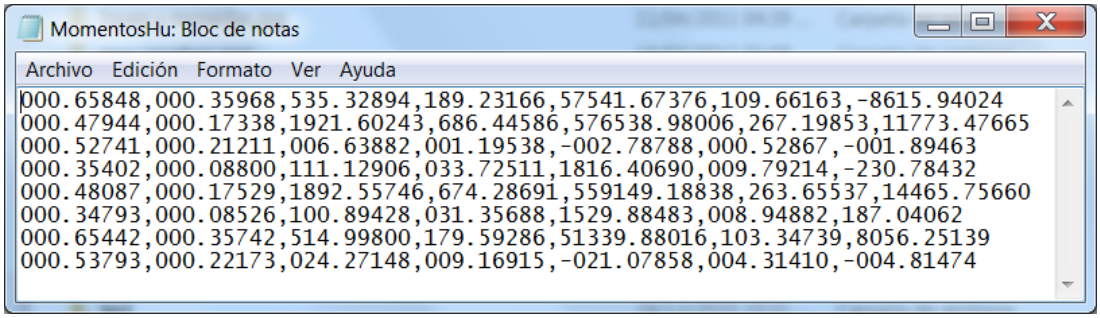
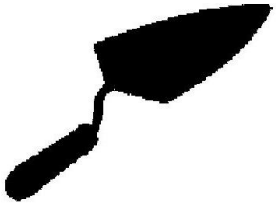
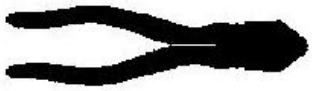
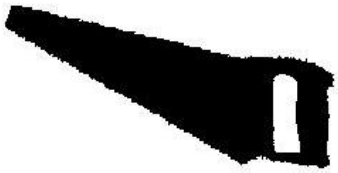
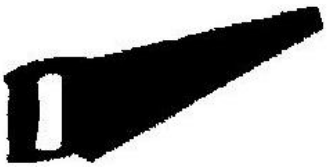


Figura 4.10. Momentos invariantes de *Hu*, Prueba 3. Con base en estas características se realizó la clasificación de los objetos.

Tabla III. Objetos clasificados, prueba 3. Se muestran los ocho objetos asociados a su respectiva clase.

Clase 1	Clase 2
	
	
Clase 3	**Clase 4**
	
	

De acuerdo a los resultados mostrados en la tabla III se concluye que esta prueba reporta un porcentaje de certeza de 100 %.

Prueba 4

En esta prueba se clasifican números contenidos en una imagen (Fig. 4.11). Los números contenidos son: uno, tres y cinco, con una cantidad de apariciones de tres por cada imagen. El resultado esperado es que cada número se asocie a una clase, es decir, que una clase contenga todos los unos, otra clase contenga todos los tres y una última clase contenga los números cinco.

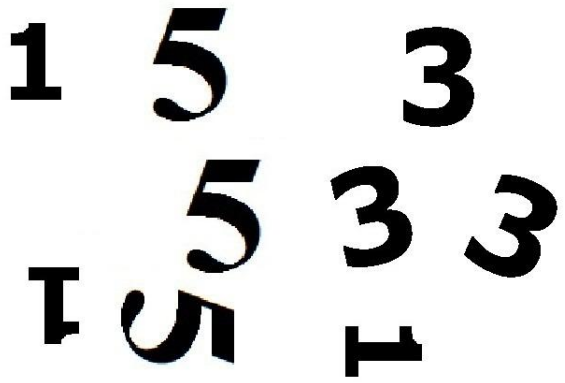


Figura 4.11. Imagen con números. Imagen que se utilizará en esta prueba de clasificación.

Cuando se aplicaron los procesos necesarios para llevar a cabo la clasificación, la etapa de extracción de características quedó como se ilustra en la figura 4.12.

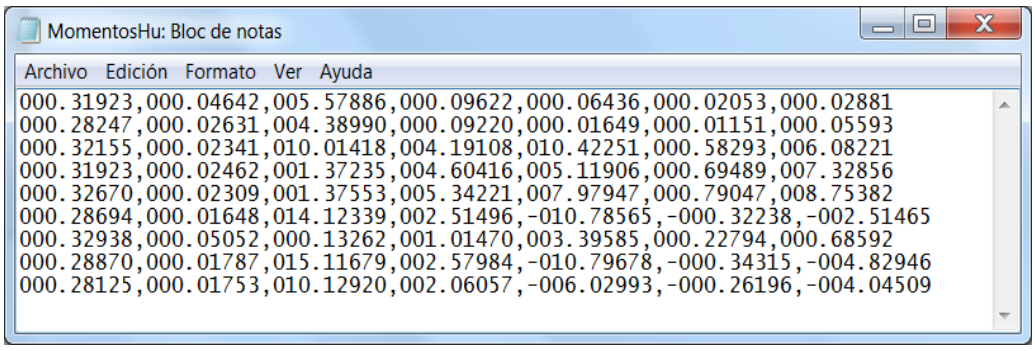


Figura 4.12. Momentos invariantes de *Hu*, Prueba 4. Características de la imagen 4.11. El umbral utilizado para la binarización de la imagen y la obtención de etas características fue de 127.

En la tabla IV puede observarse que hay un número uno asociado a la clase 1 cuando debería pertenecer a la clase 2, esto se debe a las formas de golfos y grietas que presentan los objetos de la imagen, por lo cual, el extractor de características no presenta un desempeño óptimo en objetos con este tipo de formas. Los demás objetos de la imagen, en este caso números, se asociaron correctamente a la clase que le correspondía.

La figura 4.11 presenta 9 objetos o números, de los cuales 8 fueron clasificados correctamente. De lo anterior se concluye que la clasificación de los objetos de la figura 4.11 reporta un porcentaje de certeza de 88.8 %.

Tabla IV. Objetos clasificados, prueba 4. La clasificación se llevó a cabo utilizando un valor *k* de 3.

Clase 1	Clase 2	Clase 3
		
		
		
		

Prueba 5

Esta prueba se basó en la clasificación de figuras geométricas, como son estrellas, círculos, triángulos y rectángulos. Se clasificaron un total de quince objetos; tres estrellas, cuatro círculos, tres rectángulos, cinco triángulos. En la figura 4.13 se ilustra la imagen que se utilizó en esta prueba.

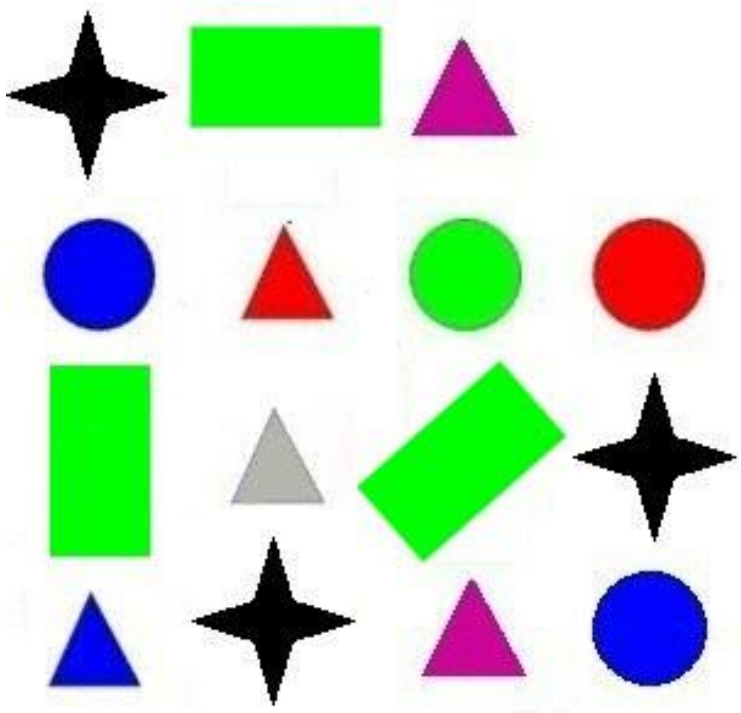


Figura 4.13. Figuras geométricas. Imagen con figuras geométricas: estrellas, círculos, rectángulos y triángulos.

Una vez aplicados los procesos correspondientes a la figura 4.13 para llevar a cabo la clasificación, el resultado de las características extraídas de los objetos se muestra en la figura 4.14.

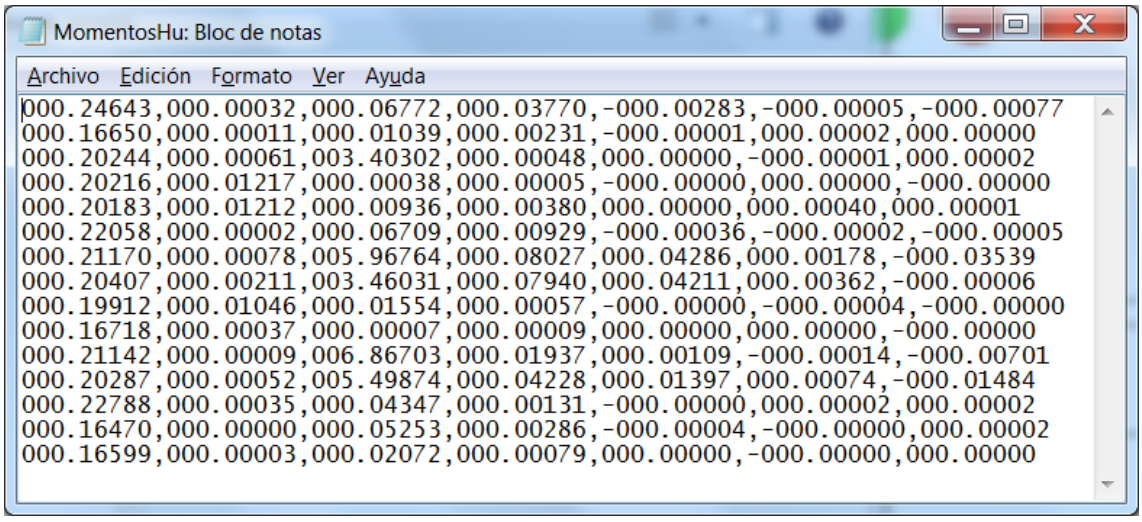
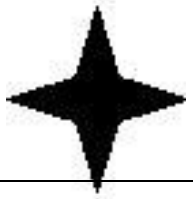
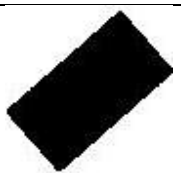
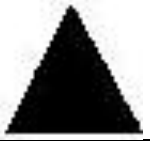


Figura 4.14. Momentos invariantes de *Hu*, Prueba 5.Características extraídas de los objetos de la figura 4.13.

En la tabla V se muestran los resultados de la clasificación. Puede observarse que cada objeto se asocia a la clase que le pertenece, es decir, todos los objetos se clasificaron correctamente. Dicho esto se concluye que esta prueba reporta un porcentaje de certeza de 100 %.

Tabla V. Objetos clasificados, prueba 5. Se muestran los objetos de la figura 4.13 asociados a sus diferentes clases.

Clase 1	Clase 2	Clase 3	Clase 4
			
			
			
			
			

De las cinco pruebas realizadas y el porcentaje de certeza de cada una de ellas, se reporta que el software desarrollado tiene un porcentaje promedio de certeza de 97.7 %. La mayoría de las figuras presentaron un porcentaje de certeza de 100 % a excepción de la figura 4.11 que presentó un porcentaje de 88.8 %. En la tabla VI se presenta un resumen de las pruebas realizadas, indicando el número de objetos qué se clasificó, así como su porcentaje de certeza y error.

Tabla VI. Resultados de las pruebas realizadas.

Prueba	Total de objetos	Número de clases	Objetos clasificados	Porcentaje de certeza
1	10	2	Llaves y monedas	100 %
2	9	2	Pernos y tuercas	100 %
3	8	4	Herramientas de trabajo	100 %
4	9	3	Números	88.8 %
5	15	4	Figuras geométricas	100 %

En el presente capítulo se presentaron las pruebas con las cuales se verificó tanto el sistema, como la fiabilidad de la extracción de características utilizando los momentos invariantes de *Hu*, éste se verificó en el módulo de clasificación. En el siguiente capítulo se darán las conclusiones generadas a partir de las pruebas, así como el planteamiento de trabajos futuros, que podrían desarrollarse para mejorar el sistema implementado en este trabajo de tesis o bien para desarrollar nuevos sistemas que partan de éste.

CAPÍTULO 5. CONCLUSIONES Y TRABAJOS FUTUROS

Durante el desarrollo de este trabajo de tesis se logró la implementación de un software clasificador de objetos a partir de imágenes digitales, en las que las características extraídas se obtuvieron utilizando los momentos invariantes de *Hu* y tomando como clasificador el *K-means*.

Al finalizar el desarrollo del software clasificador y después de la aplicación de las 5 pruebas realizadas, tomando en cuenta las características evaluadas y extraídas por los momentos invariantes de *Hu*, se reporta que este software tiene un porcentaje promedio de certeza de 97.76 %y un porcentaje de error de 2.24 %.

De lo anterior se concluye que el algoritmo de los momentos invariantes de *Hu* para extraer características de los objetos tiene buenos resultados, mismo que se ven minimizados en imágenes en donde los objetos son muy deformes o presentan grietas y entradas, como es el caso de la imagen de la figura A.4, en donde el porcentaje de certeza fue de 88.8 %, en cambio en las imágenes en donde no se presentan tales características se

tuvo un porcentaje de certeza de 100 %. Es por ello que para que el clasificador tenga una clasificación óptima es recomendable que las características de los objetos no presenten demasiadas grietas como es el caso de la figura A.4.

Trabajos futuros

Al finalizar el desarrollo de este trabajo de tesis se hacen las siguientes sugerencias:

Proyectos y trabajos futuros

1. Aplicación industrial en la clasificación de herramientas.
2. Aplicación en control de calidad para la separación de productos buenos y defectuosos.
3. Clasificación de objetos en tiempo real.
4. Separación de objetos (geométricos, numéricos y alfabéticos) para juegos didácticos elaborados con procesos industriales.
5. Detección de células cancerígenas y sanas
6. Clasificación de objetos a partir de un video.

Mejoras

1. Implementar algoritmos para calcular de manera automática, el umbral más adecuado para realizar la binarización de una imagen.
2. Implementar un algoritmo que obtenga de manera automática el número de clases presentes en una imagen.
3. Implementación de otros extractores de características como la transformada de Fourier o la transformada de Wavelet.
4. Implementación de otros clasificadores como: redes neuronales, algoritmos genéticos o clasificadores estadísticos.
5. Algoritmo de mejoramientos de la imagen, como son filtros o encadenamiento de bordes.

Este tipo de sistemas ayudará a la disminución de errores si se compara que un humano realizara el mismo trabajo. También los beneficios se verían en la rapidez de la relación del trabajo y en la exactitud del mismo. Por lo tanto, los beneficios directos se

verían reflejados en la economía de la organización que haga uso de este tipo de herramientas.

ANEXO A. IMÁGENES ORIGINALES

Este capítulo tiene como objetivo mostrar cada una de las imágenes que se utilizaron en la fase de pruebas y resultados. También se hará una descripción de cuál es la parte evaluada de cada una de ellas.

A.1. Caso de prueba 1 con diez objetos

La figura A.1 se utilizó en el caso de prueba 1. La evaluación que se realizó en esta prueba fue la rotación, traslación y escala de los objetos para probar la eficacia del extractor de características momentos invariantes de *Hu*. La imagen describe dos tipos de clases, donde cada objeto de cada clase presenta las características que utilizan lo momentos invariantes de *Hu*.

Figura A.1. Imagen usada en el caso de prueba 1.

A.2. Caso de prueba 2 con nueve objetos

En la figura A.2 se ilustra la entrada que tuvo lugar en el caso de prueba 2. La evaluación que se realizó en esta prueba fue la rotación, traslación y escala de los objetos, es decir, todas las tuercas están en una rotación, traslación y escala diferente; y los pernos en una traslación y escala diferentes. La imagen tiene un total de nueve objetos, cuatro pernos y cinco tuercas.

Figura A.2. Imagen usada en el caso de prueba 2.

A.3. Caso de prueba 3 con ocho objetos

La figura A.3 forma un conjunto de ocho herramientas de trabajo como son: dos martillos, dos pinzas, dos cucharas y dos serruchos. Las características de los momentos invariantes de *Hu* que se pusieron a prueba fueron las de rotación y traslación.

Figura A.3. Imagen usada en el caso de prueba 3.

A.4. Caso de prueba 4 con nueve objetos

La figura A.4 se utilizó para el caso de prueba 4, donde se validó la rotación, traslación y escala de los objetos presentes en la imagen, con el fin de verificar la eficacia del extractor de características momentos invariantes de *Hu*. La imagen forma un conjunto de 9 números u objetos, tres números uno, tres números cinco y tres números tres.

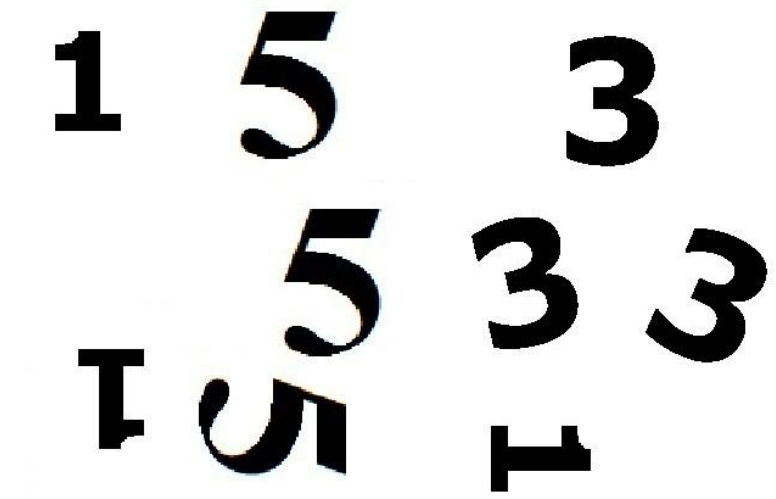


Figura A.4. Imagen usada en el caso de prueba 4.

A.5. Caso de prueba 5 con quince objetos

La figura forma un conjunto de quince figuras geométricas, tres estrellas, tres rectángulos, cinco triángulos y cuatro círculos. La parte de los momentos invariantes de *Hu* probados fue la rotación y traslación de los objetos.

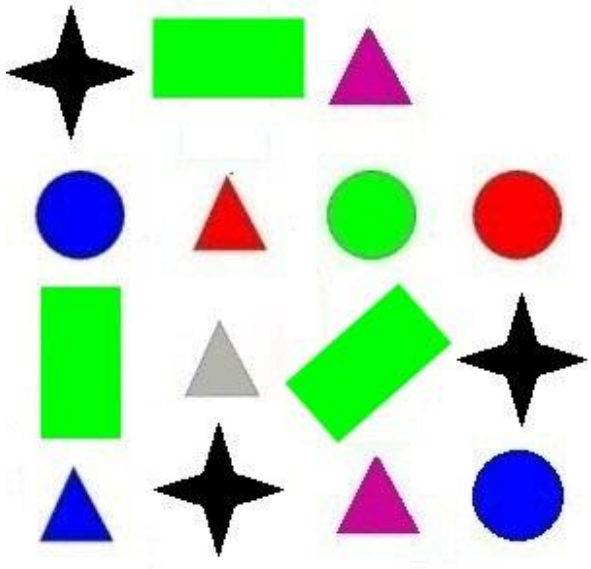


Figura A.5. Imagen usada en el caso de prueba 5.

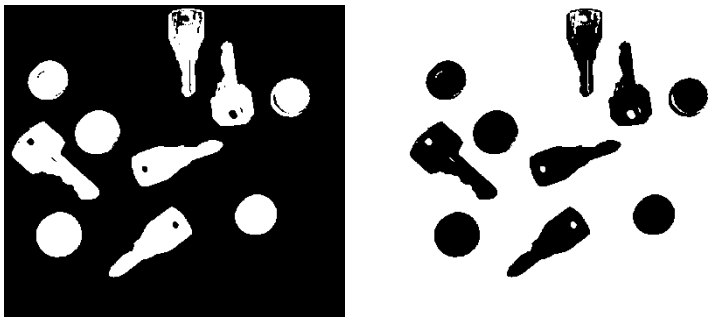
ANEXO B. IMÁGENES PROCESADAS

Este capítulo tiene como objetivo presentar todas las imágenes que se utilizaron en el caso de pruebas tal y como se ingresaron a los algoritmos de etiquetado, extracción de características y clasificación; esto es, las imágenes serán mostradas en escala de grises, binarización y negativo. Este último sólo será mostrado si la imagen lo necesitó.

B.1. Secuencia de procesamiento de la figura A.1

En la figura B.1 se presenta la secuencia de procesamiento de la figura A.1 para ser utilizada por los algoritmos de procesamiento.

La secuencia que se necesitó para que la imagen pudiera ser procesada fue la obtención de su binarización y como los objetos tienen que ser negros fue necesario obtener su negativo. Como la imagen ya está en escala de grises no es necesario repetir este paso.

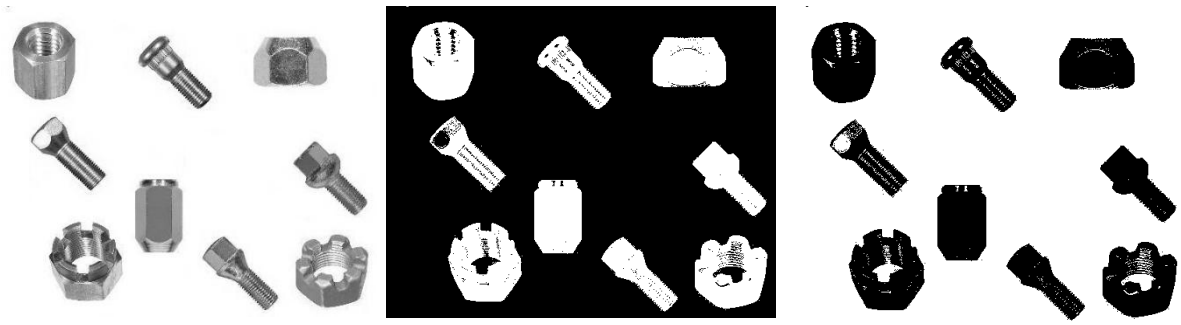


(a) (b)

Figura B.1. Procesamiento de la figura A.1. (a) Binarización. (b) Negativo.

B.2. Secuencia de procesamiento de la figura A.2

La figura B.2 muestra los tres procesos que se le hicieron a la figura A.2 para poder aplicarle los algoritmos de procesamiento principales.

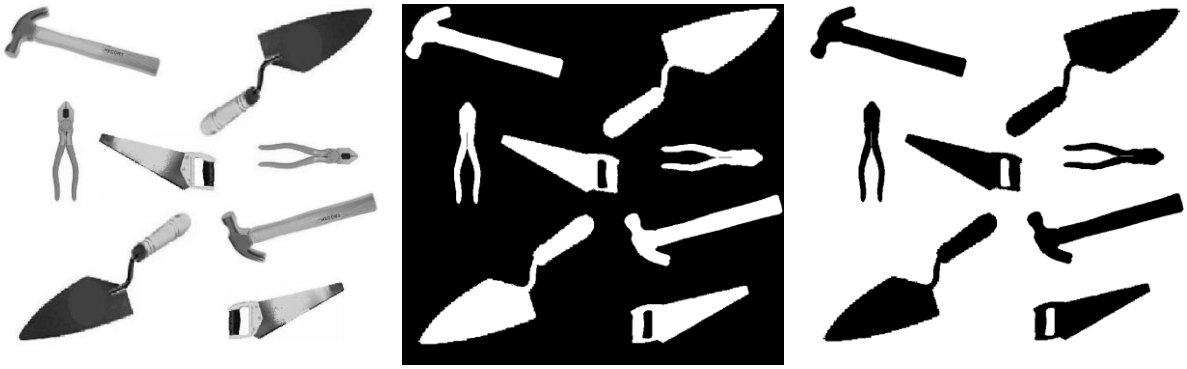


(a) (b) (c)

Figura B.2. Procesamiento de la figura A.2. (a) Escala de grises. (b) Binarización. (c) Negativo.

B.3. Secuencia de procesamiento de la figura A.3

En la figura B.3 se ilustra el proceso que se hizo para preparar la imagen y aplicar los algoritmos de procesamiento. Como los objetos son blancos fue necesario obtener el negativo.



(a) (b) (c)

Figura B.3. Procesamiento de la figura A.3. (a) Escala de grises. (b) Binarización. (c) Negativo.

B.4. Secuencia de procesamiento de la figura A.4

En la figura B.4 se ilustra la imagen utilizada como entrada para los algoritmos principales. Como se puede observar la figura B.4 es igual a la A.4, esto quiere decir, que la imagen no necesitó un procesado puesto que es suficiente con su estado actual.

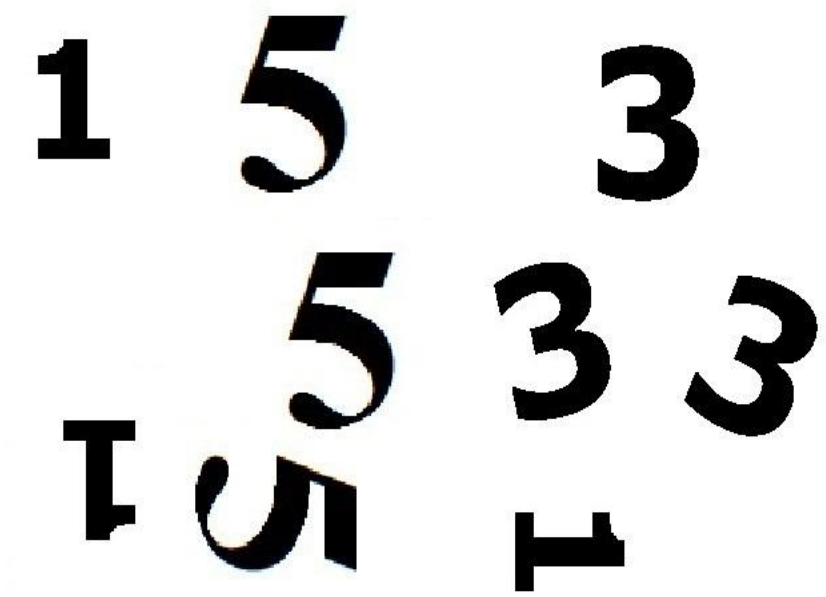
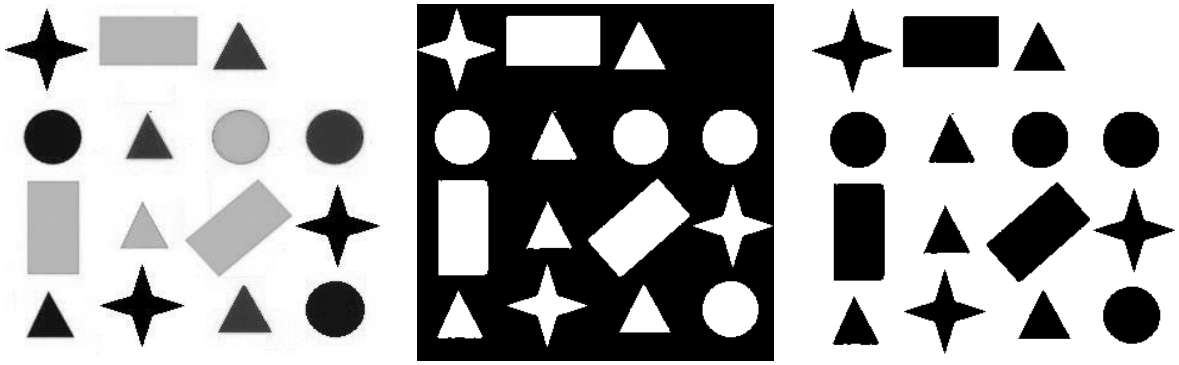


Figura B.4. Imagen usada para el caso de prueba 4. Esta imagen es la misma que su original, es decir, no fue necesario aplicarle alguna transformación como escala de grises, binarización o negativo.

B.5. Secuencia de procesamiento de la figura A.5

En la figura B.5 se ilustra la secuencia de imágenes que se obtuvo al prepararlas para aplicarles los algoritmos principales.



(a) (b) (c)

Figura B.5. Procesamiento de la figura A.5. (a) Escala de grises, (b) Binarización y (c) Negativo.

ANEXO C. CÓDIGO FUENTE

A continuación se describen las principales funciones y clases que se utilizaron en este proyecto de tesis además del adjunto de su código fuente. Las funciones son: Escala de grises, Binarización, Negativo, Momentos invariantes de *Hu* y el clasificador *K-means*.

C.1. Escala de grises

Esta función tiene por objetivo transformar una imagen en formato RGB a una imagen en formato en escala de grises (Capítulo 3.1). La función recibe un solo parámetro de tipo RGB, que es el objeto de la imagen que se desea convertir a escala de grises y retorna una matriz de enteros, que es la imagen en escala de grises con los valores de 0 a 255 (Código1, Algoritmo 1).

```
public int [][] toEscalaDeGrisesPromedio(RGB rgb){
int [][] retValue = new int[(rgb.getMatrizR()).length]
[(rgb.getMatrizR())[0].length];
for(int i=0; i<(rgb.getMatrizR()).length; i++){
      for(int j=0; j<(rgb.getMatrizR())[0].length;
j++){
                  int R=(rgb.getMatrizR())[i][j];
                  int G=(rgb.getMatrizG())[i][j];
                  int B=(rgb.getMatrizB())[i][j];
                  int Gris=(R+G+B)/3;
                  retValue[i][j]=Gris;
            }
      }
      return retValue;
}
```

Código 1. Conversión a escala de grises. Convierte una imagen a color en escala de grises obteniendo el promedio de sus componentes R, G y B

C.2. Binarización

La función siguiente convierte una imagen en escala de grises en una imagen binaria o imagen en blanco y negro (Capítulo 3.2). La función recibe una matriz de dos dimensiones que es la imagen en escala de grises y un valor de umbral (Código 1, Algoritmo 2). La salida es otra matriz de dos dimensiones que contiene la matriz binaria.

```
public int [][] toBinarizacion(int [][] matriz, intumbral){
      int [][] retValue = new int[matriz.length]
                  [matriz[0].length];
      for(int i=0; i<matriz.length; i++)
            for(int j=0; j<matriz[0].length; j++){
                  if(matriz[i][j] >= umbral)
      retValue[i][j]=0;
      else
      retValue[i][j]=255;
            }
      returnretValue;
}
```

Código 2. Binarización de la imagen.

C.3. Negativo

La función negativo invierte los colores de la imagen, es decir, una imagen que esta binarizada, el color blanco lo convierte en negro y el color negro en blanco (Capítulo 3.3). Esta función recibe como parámetro una matriz de dos dimensiones que es una imagen en

escala de grises o una imagen binaria y como salida otra matriz de dos dimensiones que es la imagen pero aplicando el negativo (Código 3, Algoritmo 3).

```
public int [][] toNegativo(int [][] matriz){
      int [][] retValue = new int[matriz.length]
[matriz[0].length];
      for(int i=0; i<matriz.length; i++){
       for(int j=0; j<matriz[0].length; j++){
            retValue[i][j] = 255 - matriz[i][j];
      }
      }
      return retValue;
 }
```

Código 3. Negativo de una imagen.

C.4. Etiquetado de regiones

Esta función tiene por objeto separar a cada uno de los objetos que componen la imagen en imágenes independientes (Código 4, Capítulo 3.4). Esto para facilitar la extracción de las características y poder aplicar el algoritmo de momentos invariantes de *Hu* por separado. La forma de obtener el etiquetado es la siguiente:

1. Primero se obtiene la imagen con la aplicación previa de los algoritmos de preprocesado y mostrados en los códigos 1, 2 y 3.
2. Después se empieza por un barrido hacia abajo y hacia arriba para encontrar cada uno de los objetos (Algoritmo 4).
3. Un vez encontrados lo objetos en la imagen se inicia el proceso de guardado de los objetos en archivos de imágenes separados.

```
public class Etiquetado {
    private Vector etiquetas;

    int alto;
    int ancho;
    int [][] matriz;

    File fichero = null;
    BufferedWriter bw = null;
    File directorio;

    private String nombre;

    public Etiquetado() {
```

Código 4. Etiquetado de regiones.

```
}

public String getNombre() {
    return nombre;
}

boolean etiquetado = false;


public boolean terminado = false;

public Etiquetado(int alto, int ancho, int [][] m, String n ){
    this.alto = alto;
    this.ancho = ancho;
    this.matriz = m;
    this.nombre = n;
    try {
        this.crearDirectorio();
    } catch (IOException ex) {
    }    }

public boolean isEtiquetado() {
    return etiquetado;
}

public void hacerEtiquetadoRegiones(){

    JOptionPane.showMessageDialog(null, "Se iniciará el proceso
        de etiquetado.\nSe indicará con un mensaje cuando el
        proceso termine.");

    int[][] img=new int[alto][ancho];
    etiquetas=new Vector();

    for(int i=0; i<alto; i++){
        for(int j=0; j<ancho; j++){
            if(matriz[i][j]==255)
                img[i][j]=0;
            else
                img[i][j]=1;
        }
    }

    for(int i=0; i<alto; i++){
        for(int j=0; j<ancho; j++){
            if(img[i][j] != 0){
                if(img[i][j] == 1){
                    img[i][j]=generaEtiquetas();
                    if((j+1)<(ancho-1) && img[i][j+1]==1)
                        img[i][j+1]=img[i][j];
                    if((i+1)<(alto-1) && img[i+1][j]==1)
                        img[i+1][j]=img[i][j];
                }
                else{
                    if((j+1)<(ancho-1) && img[i][j+1]==1)
                        img[i][j+1]=img[i][j];
```

Código 4. Etiquetado de regiones (continuación).

```
                if((i+1)<(alto-1) && img[i+1][j]==1)
                    img[i+1][j]=img[i][j];
            }
        }
    }
}


for(int i=alto-1; i>=0; i--){
    for(int j=ancho-1; j>=0; j--){
        if(img[i][j]!=0){
            if((j-1)>=0){
                if(img[i][j-1]!=0 && (img[i][j-
                1]!=img[i][j])){
                    img=norm(img,img[i][j-1],img[i][j]);

                }
            }
            if((i-1)>=0){
                if((img[i-1][j]!=0) && (img[i-
                1][j]!=img[i][j])){
                    img=norm(img,img[i-1][j],img[i][j]);
                }
            }
            etiquetas.remove(Integer.toString(img[i][j-
              1]));
        }
    }
}

//  Extraer Imagenes
int indiceimagen = 0;
for(int i=0; i<etiquetas.size(); i++){
    int[][] regionNuevaMatriz=new int[alto][ancho];
    boolean siGuardar=false;
    int
    dato=Integer.parseInt((etiquetas.get(i).toString()));
    for(int j=0; j<alto; j++){
        for(int k=0; k<ancho; k++){
            if(dato==img[j][k]){
                siGuardar=true;
                regionNuevaMatriz[j][k]=0;
            }
            else{
                regionNuevaMatriz[j][k]=255;
            }
        }
    }

    //Para guardar las imagenes generadas
    int area = this.getArea(regionNuevaMatriz) ;

    if(siGuardar &&  area > 100){
```

Código 4. Etiquetado de regiones (continuación).

```
                String formato = "jpg";
                String s = nombre +
                  "\\imagen"+(indiceimagen+1)+"."+formato;
                File archivo = new File(s);

                BufferedImage nuevaImagen = new BufferedImage(
                        alto,
                        ancho,
                        BufferedImage.TYPE_INT_RGB);
                Graphics gg = nuevaImagen.getGraphics();

                for(int j=0; j<alto; j++){
                    for(int k=0; k<ancho; k++){
                        gg.setColor(new Color(
                                regionNuevaMatriz[j][k],
                                regionNuevaMatriz[j][k],
                                regionNuevaMatriz[j][k]));
                        gg.drawLine(j,k,j,k);
                    }
                }
                try {ImageIO.write(nuevaImagen, formato, archivo);}
                catch (IOException e) {}

                indiceimagen ++;
            }

        }
        etiquetado = true;
        terminado = true;
    }


    private int getArea(int matriz [][]){
        int contador = 0;

        for(int j=0; j<alto; j++){
            for(int k=0; k<ancho; k++){
                if (matriz[j][k] == 0){
                    contador ++;
                }
            }
        }

        return contador;
    }

    public int generaEtiquetas(){
        int numeracion=0;
        if(etiquetas.isEmpty()){
            numeracion = 2;
        }
        else{
            String label=etiquetas.get(etiquetas.size() -
                  1).toString();
            int dato=Integer.parseInt(label);
            numeracion = dato + 1;
```

Código 4. Etiquetado de regiones (continuación).

```
            etiquetas.add(numeracion);

            return numeracion;
        }

        public int[][] norm(int [][] matriz,int datoAnterior,int
           datoNuevo){
            for(int i=0; i<alto; i++){
                for(int j=0; j<ancho; j++){
                    if(matriz[i][j]==datoAnterior){
                        matriz[i][j]=datoNuevo;
                    }
                }
            }
            return matriz;
         }
         private void crearDirectorio() throws IOException{
            directorio = new File(nombre);
            directorio.mkdir();
         }

    }
```

Código 4. Etiquetado de regiones (continuación).

C.5. Momentos invariantes de *Hu*

Esta función extrae las características que hacen que se diferencien unos de otros objetos en la imagen (Algoritmo 5). La clase `MomentosHu.java` se auxilia de sus tres métodos principales: `aplicarMomentosHu(String)`, `formulaMomentosHU()` y `momento(int, int)`, la clase mencionada puede verse en el código 5; donde la primera función recibe como parámetro la ruta de las imágenes separadas con el código 3. La segunda función aplica las fórmulas para obtener los momentos invariantes de *Hu* (Ec. 10-17). La tercera función aplica los **momentos centrales normalizados** descritos en la ecuación 8 y 9.

```
public class MomentosHu {
    private Image imagenGUI;
    private int alto;
    private int ancho;
    private int[][] mM;

    private File fichero;

    private BufferedWriter bw;
```

Código 5. Momentos invariantes de *Hu*.

```
 private DecimalFormat formato=new DecimalFormat("000.00000");

 /**
  * Funcion de llamada de la interfaz, desde se abren
  * las imagenes del directorio
  * @param rutaImagenes
  * @throws IOException
  */
 public void aplicarMomentosHu(String rutaImagenes) throws
   IOException{
     JOptionPane.showMessageDialog(null, "Se aplicarán los
         momentos de Hu.Se indicará con un mensaje cuando el
         proceso termine.");
     Image imagenHu = null;

     this.crearArchivo();

     int totalImg = new File(rutaImagenes).list().length;

     for(int i=0; i<totalImg; i++){
         String path = rutaImagenes + "\\imagen"+(i+1)+".jpg";
         imagenHu=Toolkit.getDefaultToolkit().getImage(path);
         imagenHu=new ImageIcon(imagenHu).getImage();
         imagenGUI=imagenHu;
         BufferedImage bufferedImage =
                 new BufferedImage(
                 imagenGUI.getWidth(null),
                 imagenGUI.getHeight(null),
                 BufferedImage.TYPE_INT_RGB );

         Graphics g = bufferedImage.createGraphics();
         g.drawImage(imagenGUI,0,0,null);
         g.dispose();

         this.getDatos(bufferedImage);

         this.normalizarMatriz();

         this.agregarMomentoArchivo(getMomentosDeHu() + "\n");

     }

     this.cerrarArchivo();
 }
 /**
  * Obtiene los datos que se necesitan en el proceso
  * @param bufferedImage
  */
private void getDatos(BufferedImage bufferedImage) {
     alto=bufferedImage.getWidth();
     ancho=bufferedImage.getHeight();
     mM=new int[alto][ancho];
     for(int j=0; j<bufferedImage.getWidth(); j++){
         for(int k=0; k<bufferedImage.getHeight(); k++){
             int R=bufferedImage.getColorModel().
```

Código 5. Momentos invariantes de *Hu* (continuación).

```
                    getRed(bufferedImage.getRGB(j,k));
            mM[j][k]=R;
        }
    }
 }

/**
 * Para fines prácticos, el 255 se transforma en 0
 */
private void normalizarMatriz(){
    for(int j=0; j<alto; j++){
        for(int k=0; k<ancho; k++){
            if(mM[j][k]==255)
                mM[j][k]=0;
            else
                mM[j][k]=1;
        }
    }
}

 /**
 * Esta función hace todo el proceso de obtener todos lo
   momentos de una imagen
 * @return momentos de Hu
 */
 public String getMomentosDeHu(){
    /*
        * MOMENTOS ORDINARIOS
        */
    // momento ordinario de orden cero
    double m00=getMomento(0,0);
    // momentos ordinarios de orden 1
    double m10=getMomento(1,0);
    double m01=getMomento(0,1);
    //momentos ordinarios de orden 2
    double m11=getMomento(1,1);
    double m02=getMomento(0,2);
    double m20=getMomento(2,0);
    // momentos ordinarios de orden 3
    double m21=getMomento(2,1);
    double m12=getMomento(1,2);
    double m30=getMomento(3,0);
    double m03=getMomento(0,3);
    // centro de masas o de gravedad
    double _x=m10/m00;
    double _y=m01/m00;

    /*
        * MOMENTOS CENTRALES
        */
    // momentos centrales de orden 2
    double u11=m11-(_y*m10);
    double u20=m20-(_x*m10);
    double u02=m02-(_y*m01);
```

Código 5. Momentos invariantes de *Hu* (continuación).

```
        double u12=m12-(2*_y*m11)-
            (_x*m02)+(2*(Math.pow(_y,2))*m10);
        double u21=m21-(2*_x*m11)-
            (_y*m20)+(2*(Math.pow(_x,2))*m01);
        double u30=m30-(3*_x*m20)+(2*(Math.pow(_x,2))*m10);
        double u03=m03-(3*_y*m02)+(2*(Math.pow(_y,2))*m01);

        /*
            * MOMENTOS CENTRALES NORMALIZADOS
            */
        // Factores de normalizacion
        double y11=(((1+1)/2)+1);
        double y12=(((1+2)/2)+1);
        double y21=(((2+1)/2)+1);
        double y20=(((2+0)/2)+1);
        double y02=(((0+2)/2)+1);
        double y30=(((3+0)/2)+1);
        double y03=(((0+3)/2)+1);
        // Obtención de los momentos centrales normalizados
        double n11=u11/(Math.pow(m00,y11));
        double n12=u12/(Math.pow(m00,y12));
        double n21=u21/(Math.pow(m00,y21));
        double n20=u20/(Math.pow(m00,y20));
        double n02=u02/(Math.pow(m00,y02));
        double n30=u30/(Math.pow(m00,y30));
        double n03=u03/(Math.pow(m00,y03));

        /*
            * 7 MOMENTOS INVARIANTES DE HU
            */
        double O1=n20+n02;
        double O2=Math.pow((n20-n02),2)+(4*(Math.pow(n11,2)));
        double O3=(Math.pow((n30+(3*n12)),2))+(Math.pow((3*n21-
            n03),2));

        double O4=(Math.pow((n30+n12),2))+(Math.pow((n21+n03),2));
        double O5=(n30-(3*n12))*(n30+n12)*((Math.pow((n30+n12),2))
                -(3*(Math.pow((n21+n03),2)))) + (((3*n21)-
            n03)*(n21+n03)
                *((3*(Math.pow((n30+n12),2)))-
            (Math.pow((n21+n03),2))));

        double O6=(n20-n02)*((Math.pow((n30+n12),2))-
                  (Math.pow((n21+n03),2)))
                + ((4*n11)*(n30+n12)*(n21+n03));

        double O7=((3*n21-n03)*(n30+n12))*((Math.pow((n30+n12),2))
                -(3*(Math.pow((n21+n03),2))))+(3*n12-
                  n30)*(n21+n03)*
                 (3* Math.pow((n30+n12),2)-Math.pow((n21+n03),2));

        return
                f(O1)+","+f(O2)+","+f(O3)+","+
                f(O4)+","+f(O5)+","+f(O6)+","+f(O7);
    }
```

Código 5. Momentos invariantes de *Hu* (continuación).

```
     * Convierte el double a string agregándole solos tres
      decimales
     * @param d
     * @return
     */
    private String f(double d){
        return formato.format(d);
    }

    /**
     * Obtiene el momento ordinario de la imagen, p, q
     * @param p
     * @param q
     * @return
     */
    public double getMomento(int p,int q){
        double momento=0;
        for(int y=1; y<=alto; y++){
            for(int x=1; x<=ancho; x++){
                momento += Math.pow(y,p)*Math.pow(x,q)*mM[y-1][x-
                  1];
            }
        }
        return momento;
    }

    private void crearArchivo() throws IOException{
        fichero = new File("MomentosHu.txt");
        bw = new BufferedWriter(new FileWriter(fichero));
    }

    private void agregarMomentoArchivo(String linea) throws
            IOException{
        bw.write(linea);
    }

    private void cerrarArchivo() throws IOException{
         bw.close();
    }
}
```

Código 5. Momentos invariantes de *Hu* (continuación).

C.6. Clasificador *K-means*

Esta clase implementa el algoritmo *K-means*, en el que se basó este trabajo de tesis para corroborar visualmente la funcionalidad el extractor de características (Código 6). Este algoritmo se basa en encontrar la distancia más cercana de los objetos, teniendo como base la cantidad de objetos como tantas clases se requieran.

La clase recibe como parámetros el archivo de momentos invariantes de *Hu* obtenido con el código 3. Después de obtener estos datos se procede a elegir la cantidad k

de centroides aleatoriamente entre el conjunto de datos. A partir de ahí se inicia el proceso del algoritmo 5. Este proceso finaliza cuando los datos se terminaron de clasificar dando un mensaje de información.

```
public class K_Means {
    int _K;
    Vector cen = new Vector();
    Vector cc = new Vector();
    Vector obj = new Vector();
    private double[][] matriz;
    private boolean bandera = true;

    public K_Means(int K){
        this._K = K;
    }

       public void cargarArchivo(String ru){

        // si desea probar cn negativos
        JFileChooser archivo = new JFileChooser();
        int fil = 0 , col = 0;
        if (ru == null){
            archivo.setDialogTitle("Seleccione archivo con
          Momento de Hu");
            archivo.setSelectedFile(new
          File("").getAbsoluteFile());
            if(archivo.showOpenDialog(null) ==
                JFileChooser.APPROVE_OPTION){
                ru = archivo.getSelectedFile().toString();
            }
        }

        try {
            BufferedReader leerArchivo=new
                BufferedReader(new FileReader(ru));
            String datosPorLinea="";
            int i=0, j=0;
            double temp[][] = new double[100][100];

            while((datosPorLinea=leerArchivo.readLine())
                != null){

                    StringTokenizer tokens=new
                      StringTokenizer(datosPorLinea, ",");

                while(tokens.hasMoreElements()){
                    try{
                        temp[i][j] =

                Double.parseDouble(tokens.nextToken());
                        j++;
                    }
                    catch(NumberFormatException e){
```

Código 6. Clasificador *K-means*.

```
                    JOptionPane.showMessageDialog(
                            null,
                            "Error al leer los datos
                  en llenar objetos 1.");
                    return;
                }
            }
            col = j;
            j=0;
            i++;
            fil = i;
        }

        matriz = new double[fil][col];
        for (i=0;i<matriz.length;i++){
            for (j=0;j<matriz[0].length;j++){
                matriz[i][j] = temp[i][j];
            }
        }
    }
    catch(IOException ex){
        JOptionPane.showMessageDialog(
                null,
                "Error al leer los datos en llenar
      objetos 2.");
        return;
    }
}


public void llenarObjetos(){
    for (int i=0;i<matriz.length;i++){
        obj.add(new Objeto(matriz[i]));
    }
}

private void getCentroides(){
    for (int i=0;i<_K;i++){
        int indice = (int)(Math.random() *
            obj.size());
        cen.add(((Objeto)(obj.get(i))).clone());
    }
}

private double getD(Objeto centroide, Objeto objeto){
    double p, q;
    double sumatoria = 0;

    for(int i=0;i<centroide.getC().length;i++){
        p = centroide.getC()[i];
        q = objeto.getC()[i];
        sumatoria += Math.pow((p - q), 2);
    }

    return Math.sqrt(sumatoria);
```

Código 6. Clasificador *K-means* (continuación).

```
}
public void iniciar(int tipoDistancia){
    DecimalFormat f = new DecimalFormat("####.####");
    this.getCentroides();
    double m [] = new double[cen.size()];
    int iteracion  = 1;



    while(bandera){
        System.out.println("Iteracion : " +
            iteracion++);
        for (int i=0;i<obj.size();i++){
            for (int j=0;j<_K;j++){
                if (tipoDistancia == 0)
                    m[j] = getD(
                            (Objeto)cen.get(j), (
                            Objeto)obj.get(i));
                else
                    m[j] = getD((
                            Objeto)cen.get(j),
                            (Objeto)obj.get(i));
            }

            ((Objeto)obj.get(i)).setClase(getMenor(m)
            + 1);

        }

        cc.removeAllElements();
        for (int i=0;i<cen.size();i++){
            double [] aux = new
      double[((Objeto)cen.get(0)).getC().length];
            for (int
      j=0;j<((Objeto)cen.get(0)).getC().length;j++){
                aux[j] =
      ((Objeto)cen.get(i)).getC()[j];
            }
            cc.add(   new Objeto(aux)   );
        }

        modificarCentroides();

        bandera = false;

        for (int i=0;i<cen.size();i++){
            for (int
      j=0;j<((Objeto)cen.get(0)).getC().length;j++){
                if (((Objeto)cen.get(i)).getC()[j] !=

            ((Objeto)cc.get(i)).getC()[j]){
                    bandera = true;
                    break;
                }
            }
```

Código 6. Clasificador *K-means* (continuación).

```
            if (bandera == true) break;
        }

    }
}

private void modificarCentroides() {
    int total = 0;
    double s [] =
            new double[

      ((Objeto)cen.get(0)).getC().length];


    for (int i=0;i<cen.size();i++){
        total = 0;
        for (int ii=0;ii<s.length;ii++)
            s[ii] = 0

        for (int j=0;j<obj.size();j++){
            if (((Objeto)obj.get(j)).getClase() ==
            (i+1)){
               total ++;
               for (int k=0;k<y;k++){
                    s[k] +=
                  ((Objeto)obj.get(j)).getC()[k];
               }
            }
        }
        int y = ((Objeto)cen.get(0)).getC().length;
        for (int k=0;k<y;k++){
            ((Objeto)cen.get(i)).getC()[k] = s[k] /
            total;
        }
    }
}

private int getMenor(double [] medidas) {
   double menor = medidas[0];
   int p = 0;

   for (int i=1;i<medidas.length;i++){
       if(menor > medidas[i]){
           menor = medidas[i];
           p = i;
       }
   }

   return p;
}

public void crearCarpetas(String r) throws IOException
{
    GregorianCalendar gc = (GregorianCalendar)
            Calendar.getInstance();
```

Código 6. Clasificador *K-means* (continuación).

```
    String c = "Imagenes Clasificadas - " +
            gc.get(Calendar.DAY_OF_MONTH) + "-" +
            gc.get(Calendar.MONTH) + "-" +
            gc.get(Calendar.YEAR) + " - " +
            gc.get(Calendar.HOUR) + "." +
            gc.get(Calendar.MINUTE);

    File a=new File("");
String ruta=a.getAbsolutePath();

    File directorio = new File(c);
    directorio.mkdir();

    for (int i=0;i<cen.size();i++){
        String car = ruta + "\\" + c + "\\clase" +
            (i+1);
        File dir = new File(car);
        dir.mkdir();
    }

    Copy copiar = new Copy();

    for (int i=0;i<obj.size();i++){
        copiar.copyFile( new File(r + "\\imagen" +
            (i+1) + ".jpg"),
            new File(ruta + "\\" + c + "\\clase" +
            (((Objeto)obj.get(i)).getClase())
            +"\\imagen" + (i+1) + ".jpg"));
    }

    File informacion=new File(ruta + "\\" + c +
      "\\informacion.txt");

    BufferedWriter bw = new BufferedWriter(new
      FileWriter(informacion));
    String linea = null;

    bw.write("nombre de archivo, clase\n"); // Se
      ponen los encabezados
    for (int i=0;i<obj.size();i++){
        linea = "imagen" + (i+1) + "," +
            (((Objeto)obj.get(i)).getClase());

        bw.write(linea + "\n");
    }

    bw.close();

    try{
        for (int i=0;i<_K;i++){

            File i2=new File(ruta + "\\" +
                    c + "\\clase" + (i+1) +
            "\\caracteristicas.txt");
```

Código 6. Clasificador *K-means* (continuación).

```
            BufferedWriter bw2 = new
            BufferedWriter(new FileWriter(i2));
            bw2.write("nombre de
            archivo,clase,momentos\n");

            for (int j=0;j<obj.size();j++){
                if (((Objeto)obj.get(j)).getClase() ==
                  i + 1){
                linea = "imagen" + (j+1) + "," +
                  (i+1);
                linea += ",";
                int y =
                  ((Objeto)obj.get(0)).getC().length;
                for (int k=0;k<y;k++){
                    linea +=
            ((Objeto)obj.get(j)).getC()[k] + ", ";
                }

                bw2.write(linea + "\n");
                }
            }
            bw2.close();
        }

    }

    catch(Exception e){e.printStackTrace();}

    bw.close();

    JOptionPane.showMessageDialog(null, "Se ha
      terminado la clasificacion "
            + "con éxito.\nSe generó un directorio con
            las "
            + "diferentes clases generadas.");

}
                        }
```

Código 6. Clasificador *K-means* (continuación).

ANEXO D. FUNCIONAMIENTO DEL SISTEMA

En este apartado se explica el funcionamiento del sistema, producto de este trabajo de tesis.

La estructura del documento se basa en una revisión por cada uno de los menús y sus respectivos submenús.

La lista de los menús que se explicarán forma la siguiente estructura, según el orden en que aparecen en el sistema.

1. Archivo
2. Edición
3. Procesamiento
4. Segmentación
5. Clasificación

Se da una explicación de cada módulo por menú, qué se puede hacer y el momento que se debe utilizar.

La interfaz del programa cuenta con una barra de herramientas (misma que se encuentra al lado izquierdo del área de trabajo) de accesos rápidos a las opciones del menú `Archivo`, `Edición` y `Procesamiento`. Los accesos rápidos de la barra de herramientas estarán clasificados respecto al orden en que aparecen en los propios menús.

La estructura básica del programa se basa en pestañas, donde cada pestaña muestra la actividad u operación que se hará, se le está haciendo o se le hizo a una imagen. Cabe aclarar que cada cambio que se le aplique, producto de un algoritmo, se mostrará en una nueva pestaña. Todos los cambios de las operaciones que se hagan se efectuarán a partir de la imagen de la pestaña actual.

En la figura D.1 se muestra la pantalla principal del sistema que se desarrolló en este proyecto de tesis.

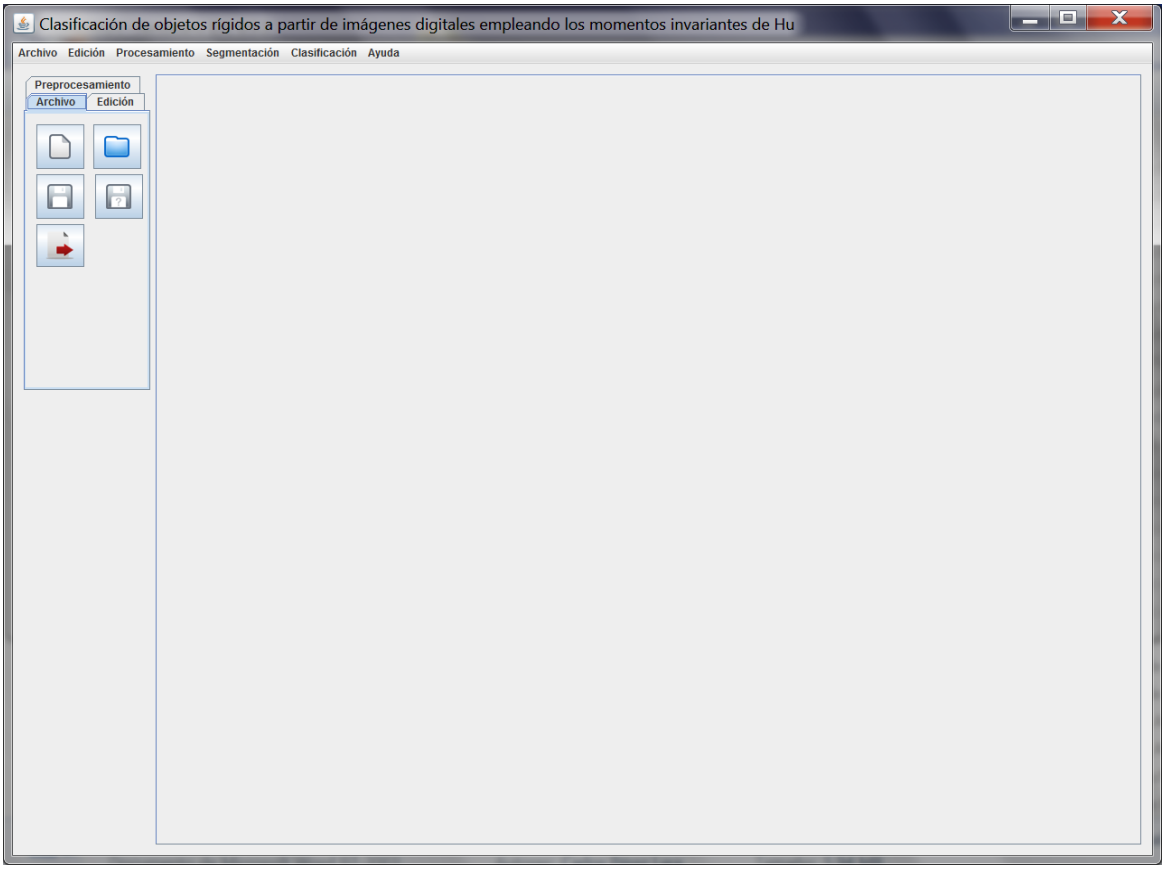


Figura D.1. Pantalla principal. Ventana principal con los menús y barra de herramientas.

D.1. Menú Archivo

Es el menú con las operaciones estándar de cualquier sistema de cómputo (Fig. D.2). Estas operaciones van desde la apertura de una imagen, creación de una nueva imagen, guardar la ya existente, cerrar la imagen actual o guardar la imagen actual en otro formato.

Figura D.2. Menú `Archivo`. Se muestra el menú archivo con todas las opciones que facilita.

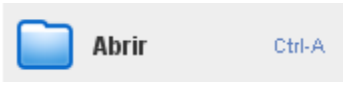
Nuevo

Se utilizará esta opción del menú `Archivo` cuando se tenga la necesidad de abrir una nueva imagen en una nueva pestaña (Fig. D.3).

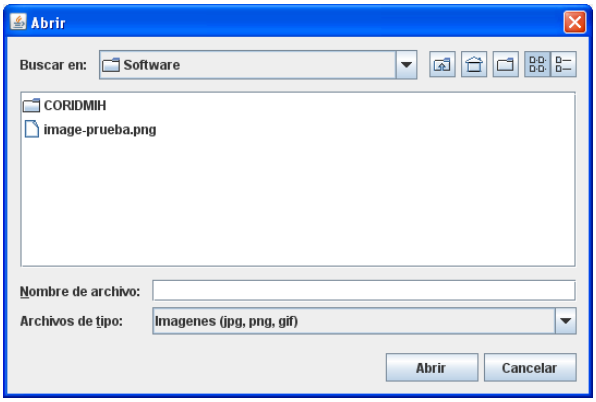
Figura D.3. Nuevo documento. Opción para abrir una nueva pestaña y mostrar una nueva imagen.

Abrir

Esta opción es útil para ingresar una nueva imagen a la aplicación y hacerle las operaciones correspondientes (Fig. D.4 (a)). La imagen se buscará en un cuadro de diálogo (Fig. D.4 (b)), dónde se buscará la ruta y posteriormente la imagen. Los archivos permitidos por el sistema son PNG, JPG y GIF.



(a)

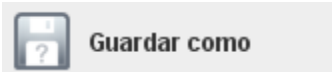


(b)

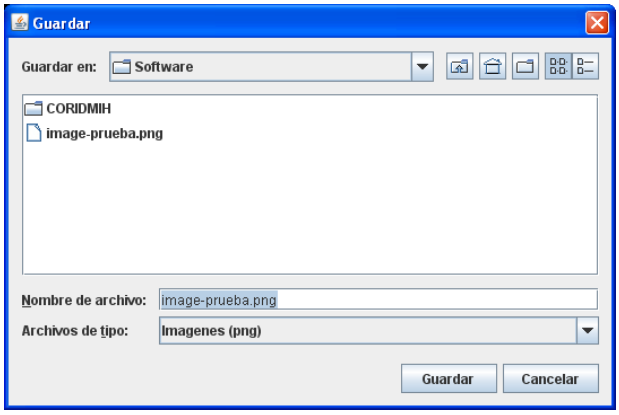
Figura D.4. Abrir imagen. (a) Opción para abrir una imagen desde un navegador de archivos y (b) Busca la ruta de la imagen para realizar el proceso de clasificación.

Guardar como

Esta opción permite guardar la imagen con un nuevo nombre y en una ruta diferente (Fig. D.5).



(a)



(b)

Figura D.5. Guardar imagen como. (a) Abre el cuadro de diálogo `Guardar y` (b) Escoge la ruta y guarda la imagen actual en un archivo de imagen.

Cerrar

Si se desea cerrar una pestaña que ya no sea útil se aconseja esta opción. Se perderán los cambios hechos a la imagen si no se ha guardado anteriormente (Fig. D.6).

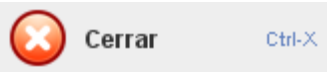


Figura D.6. Cerrar imagen. Cierra la pestaña actual.

Exportar

Cuando se tenga de necesidad de cambiar el formato de la imagen a otro podrá hacer uso de este módulo (Fig. D.7). Entre los formatos a los que se puede exportar las imágenes están: PNG, JPG y GIF.

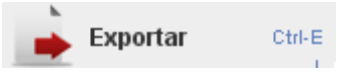
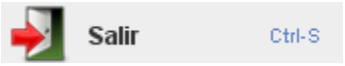


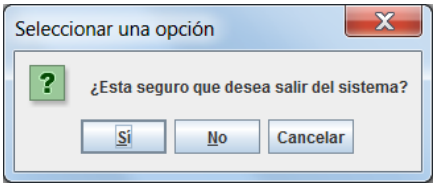
Figura D.7. Exportar una imagen. Guarda la imagen actual en otro formato de imagen.

Salir

Sale completamente del sistema (Fig. D.8 (a)), recibiendo antes la notificación que se muestra en la figura D.8 (b) por si activó esta opción erróneamente.



(a)



(b)

Figura D.8. Salir del sistema. (a) Aborta completamente el sistema y (b) Mensaje de confirmación del cierre del sistema.

D.2. Menú Edición

El menú `Edición` forma parte de las operaciones básicas de un sistema para formato de imágenes (Fig. D.9). Con este menú podrán ingresarse imágenes al sistema mediante el portapapeles del SO.

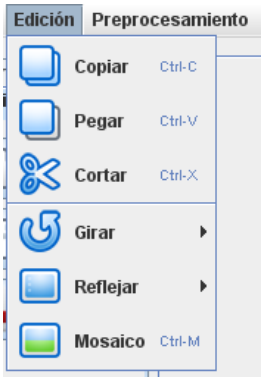


Figura D.9. Menú `Edición`. Se muestra el menú edición con las opciones básicas.

La figura D.10 está en formato de escala de grises y es utilizada en todos los filtros y procesamientos siguientes.

Figura D.10. Imagen de prueba. Imagen que fue usada para las pruebas de este menú.

Copiar

Es la forma más fácil de crear una copia de una imagen, ya sea, exportarla o copiarla en una nueva pestaña (Fig. D.11). También se puede hacer uso de esta función con la combinación [Ctrl+C].

Figura D.11. Copiar imagen. Crea una copia de la imagen.

Pegar

Se puede usar esta función para abrir una imagen sin necesidad de usar el menú `Abrir` (Fig. D.12). Dicha imagen deberá estar contenida en el portapapeles, o que antes se hubiera utilizado la función copiar.

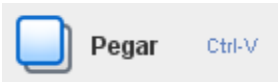


Figura D.12. Pegar imagen. Pega la imagen guardada en el portapapeles.

Cortar

Guarda una copia en de la imagen contenida en la pestaña actual en el portapapeles del sistema operativo, eliminando dicha imagen (Fig. D.13).

Figura D.13. Cortar imagen. Transfiere la imagen actual pero sin conservar una copia de la original

Girar

Se usa cuando se desea girar o rotar la imagen para una mejor visibilidad (Fig. D.14). Los giros serán de 90, 180 y 270, siendo la medición utilizada en grados.

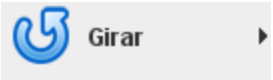


Figura D.14. Girar imagen. Hace rotar la imagen actual.

En la figura D.15 se muestra un giro de 90 grados.

Figura D.15. Rotación de una imagen. Imagen rotada 90 grados.

Reflejar

Si se quiere crear un reflejo de la imagen actual basta con usar esta opción Reflejar (Fig. D.16). El reflejo será Vertical u Horizontal.

Figura D.16. Reflejar imagen. Crea un reflejo de la imagen

En la figura D.17 se muestra el resultado de aplicarle un reflejo horizontal la figura D.10.

Figura D.17. Imagen reflejada. Ejemplo de la aplicación de un reflejo horizontal.

Mosaico

Cuando se le están aplicando cambios a una imagen estos se muestran en una nueva pestaña. Mientras más algoritmos y filtros se le apliquen a las imágenes más pestañas se habrán generado. A veces se tiene la necesidad de ver todas esas imágenes en una sola

exhibición. Para lograr lo anterior basta con usar la función `Mosaico` (Fig. D.18), o bien, presionar [Ctrl+M] y se abrirá una nueva pestaña con todas las imágenes del documento.

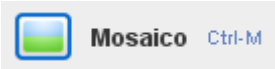


Figura D.18. Mosaico. Crea un mosaico de todas las imágenes abiertas y creadas.

En la figuraD.19 se visualiza un ejemplo del uso de la opción `Mosaico` cuando se tienen ocho imágenes abiertas.

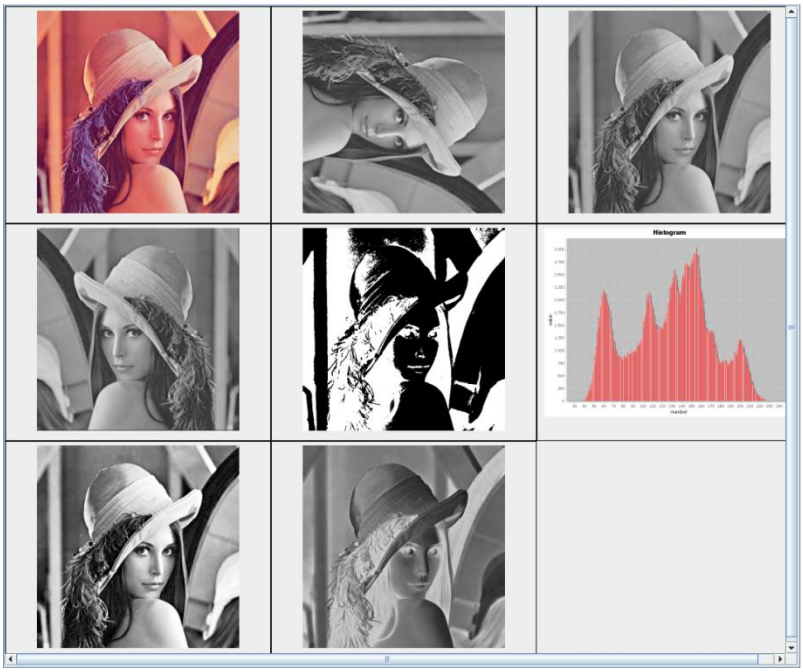


Figura D.19. Ejemplo de mosaico. Se muestra el uso del proceso de mosaico.

D.3. Menú Procesamiento

Para aplicar un filtro o algoritmo es necesario preparar la imagen. Dentro de las condiciones que se necesitan para aplicar cualquier filtro, es necesario que la imagen esté en escala de grises y, en algunos casos, que este binarizada. También habrá necesidad de obtener el negativo de la imagen.

En este menú existen opciones que ayudarán a preparar la imagen para efectuar cada una de las operaciones que se deseen aplicar (Fig. D.20).

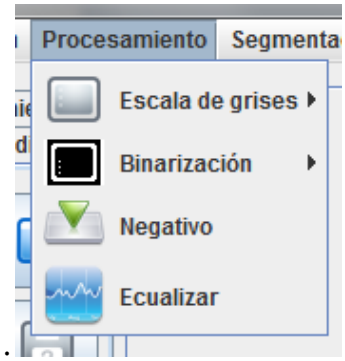


Figura D.20. Menú `Procesamiento`. Se muestra todas las opciones del menú procesamiento.

Escala de Grises

La mayoría de los algoritmos de procesado de imágenes exigen que la imagen actual se encuentre representada como una matriz, esto es, en escala de grises (Fig. E.21). Existen dos métodos de lograr que la imagen quede en forma de grises que son: el promedio y el ponderado.

Figura D.21. Escala de grises. Opción que convierte una imagen a color en escala de grises.

La figura D.22 se tomó como prueba para aplicar los dos tipos de conversión de RGB a escala de grises.

Figura D.22. Imagen original de pruebas. Esta imagen fue utilizada para mostrar las diferentes opciones de este menú.

Promedio

Es el método más utilizado para transformar una imagen que está en RGB a escala de grises. Consiste en sacar el promedio de los valores de rojo, verde y azul de la imagen a color. Se muestra un ejemplo de la conversión en la figura D.23.

Figura D.23. Escala de grises por promedio. Visualización de la escala de grises obtenida por promedio.

Ponderado

Es otro modo de calcular el color gris en una imagen a color. Se basa en la utilización de factores sobre cada color de la imagen.

La figura D.24 es un ejemplo de la conversión de la imagen por el método ponderado.

Figura D.24. Escala de grises ponderado. Visualización de la escala de grises obtenida por el ponderado de colores.

Binarización

Como en la escala de grises, también la binarización es necesaria para aplicar diversos algoritmos de reconocimiento. Para ello se utilizará la opción Binarización (Fig. D.25) para transformar la imagen en dos colores: blanco y negro.

Figura D.25. Binarización. Opción que crea una imagen binaria a partir de una imagen en escala de grises.

Personalizado

Esta opción calcula el binarizado pero desde un umbral definido (Fig. D.26). Esta imagen es una aplicación de Binarización con un umbral de 90.

Figura D.26. Binarización personalizado. Binarización de una imagen con un umbral de 90.

Automática

La opción automática permite calcular el binarizado a partir de un umbral de 127.Se muestra un ejemplo de Binarización automática en la figura D.27.

Figura D.27. Binarización automática. Binarización de una imagen con la opción automático.

Negativo

Cuando se tiene la necesidad de invertir los colores en una imagen representada en escala de grises o binarizada se utilizará esta opción (Fig. D.28). Por ejemplo, en una imagen binarizada, al aplicar esta opción, los negros se transformaran en blancos y los blancos en negros. En la figura D.29 se muestra la aplicación de esta opción.

Figura D.28. Negativo. Obtiene el negativo de la imagen actual.

(a) (b)

Figura D.29. Negativo de una imagen. (a) Imagen binarizada y (b) Negativo de la imagen.

D.4. Menú Segmentación

Este menú se compone de dos submenús, `Etiquetado de regiones` y `Momentos invariantes de Hu` (Fig. D.30). El objetivo del primer submenú es tratar de localizar cada una de los objetos que componen una imagen y el segundo obtiene la matriz de los momentos invariantes de *Hu* para un posterior análisis.

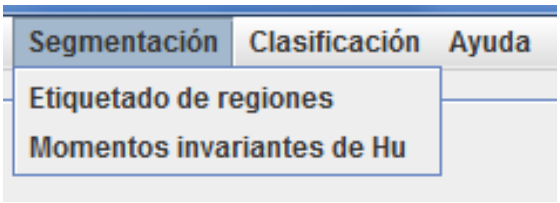


Figura D.30. Menú `Segmentación`. Se muestran todas las opciones del menú segmentación.

Etiquetado de Regiones

El procedimiento de etiquetado de regiones asigna una etiqueta a todos los pixeles que se encuentren en una vecindad y pertenezcan a la misma región (Fig. D.31). Este proceso se aplicará a todas las imágenes razonadamente separables.

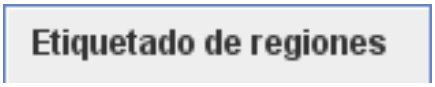


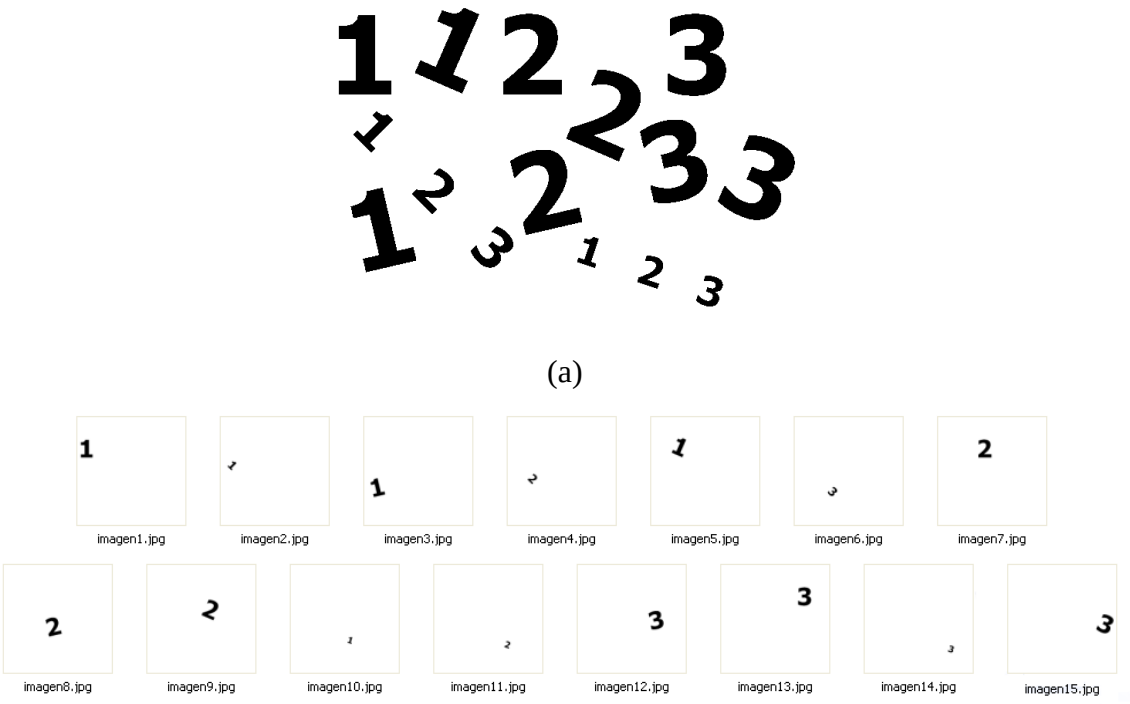
Figura D.31. Submenú `Etiquetado de regiones`. Opción que etiqueta cada objeto de la imagen.

Para aplicar este método se seguirán los siguientes pasos:

1. Convertir la imagen a escala de grises.
2. Binarizar la imagen.
3. Obtener el negativo de la imagen binarizada (este paso es opcional, puesto que será necesario que los objetos no sean negros).

Este proceso se iniciará con un mensaje, que el proceso se ha iniciado y se notificará cuando este haya terminado.

Al terminar se creará un directorio en la ruta de ejecución del programa con el nombre completo de la imagen original, y dentro del mismo estarán todos los objetos encontrados en forma de archivos de imágenes. En la figura D.32 se muestra un ejemplo de la ejecución de una prueba.



(a)

(b)

Figura D.32. Etiquetado de regiones. (a) Imagen original y (b) Objetos encontrados en la imagen original.

Momentos invariantes de *Hu*

Es el proceso opcional siguiente al etiquetado de regiones (Fig. D.33). Este proceso extraerá los siete momentos invariantes de cada imagen etiquetada, esta información será almacenada en un archivo de texto de nombre **MomentosHu.txt** (tomando en cuenta que la ruta utilizada será relativa) en el que se tendrán los siete momentos invariantes de cada una de las imágenes.

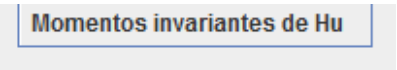


Figura D.33. Submenú Momentos invariantes de *Hu*. Opción que extrae las características de las imágenes empleando los momentos invariantes de *Hu*.

En la figura D.34 se presenta los momentos invariantes de *Hu* extraídos de la figura D.33 (a).

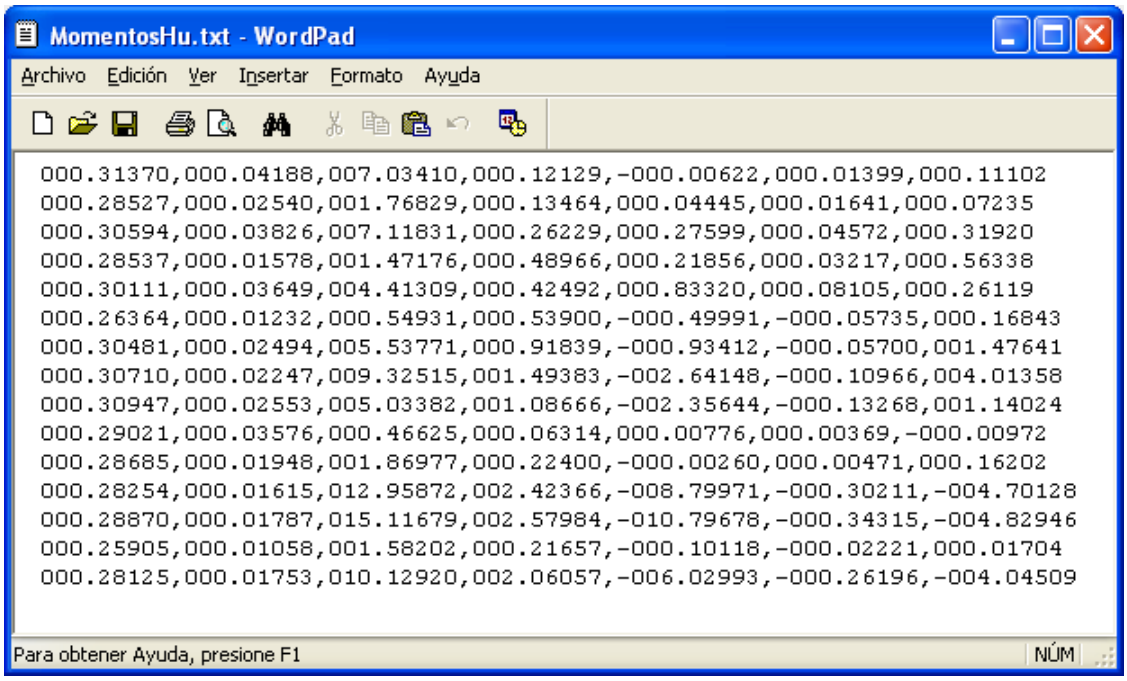


Figura D.34. Momentos invariantes de *Hu*. Archivo de momentos invariantes de *Hu* extraídos de la figura D.32 (b).

D.5. Menú Clasificación

En este menú se realizará la clasificación a partir de las características extraídas empleando los momentos invariantes de *Hu* (proceso hecho en el menú `Procesamiento`).

La clasificación será de dos formas (Fig. E.35). La primera se llama Manual, en la que se tendrá que aplicar cada uno de los pasos anteriores, estos son: Etiquetado de regiones y Momentos invariantes de *Hu*. La segunda se llama Automático, aquí no será necesario haber llevado los procesos anteriores como en el caso de la opción Manual. Tanto para la opción `manual` como la opción `automático` se ha implementado la distancia euclídea (también llamada euclideana (Fig. D.36).

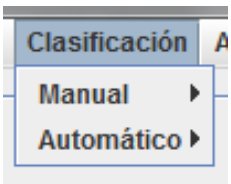


Figura D.35. Menú `Clasificación`. Se muestran las opciones para realizar la clasificación de los objetos de la imagen.

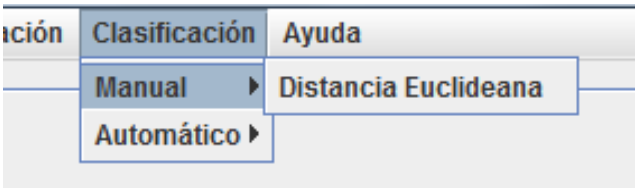
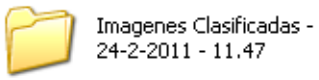


Figura D.36. Distancias del clasificador. Tipos de distancia que fueron implementadas en el clasificador.

Al elegir cualquier de los tipos de clasificación el sistema enviará un mensaje de notificación diciendo que el proceso se ha iniciado y este avisará cuando termine.

El sistema, antes de iniciar la clasificación pedirá el valor de *k* (número de clases que se desean obtener). Este valor deberá ser de tipo entero, de lo contrario se avisará con un mensaje de error.

Se informará cuando este proceso haya terminado. También el sistema creará un directorio con el nombre de **Imágenes Clasificadas** seguido de **la fecha y la hora** (Fig. D.37 (a))**,** y dentro de éste se creará el archivo **informacion.txt**, el cual contendrá que imágenes pertenecen a que clase, también se crearán tantos directorios como clases se hayan creado, en cada directorio de cada clase se tendrá un archivo, este mismo almacenará los momentos invariantes de *Hu* de las imágenes pertenecientes a dicha clase, así como las imágenes asociadas a dicha clase. En la figura D.37 se muestra la salida de una clasificación de las imágenes de la figura D.32 (b).



(a)

(b)

Figura D.37. Imágenes clasificadas. (a) Directorio en el que están todas las imágenes clasificadas en diferentes carpetas y (b) Directorios que está en el interior de la carpeta de la figura (a). Hay tantas carpetas como clases, además del archivo **informacion.txt**.

ANEXO E. CONTENIDO DEL CD

En este capítulo se presenta la estructura del CD-ROM que se anexa este documento de tesis. El contenido del disco se muestra ordenado en directorios, los cuales son: Anexos, Aplicación y Documento de tesis (Fig. E.1).

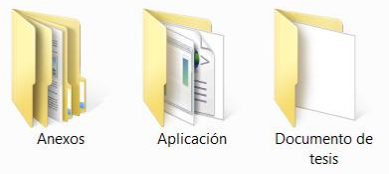


Figura E.1. Directorios principales del contenido del CD.

E.1. Anexos

El directorio de anexos se visualiza en el mismo orden que en este documento, como es: Imágenes originales, Imágenes procesadas, Código fuente y Funcionamiento del sistema (Fig. E.2).

Figura E.2. Distribución de los directorios de los anexos.

E.2. Aplicación

En esta parte se puede apreciar el ejecutable del sistema desarrollado. En la figura E.3 el archivo CORIDMIH.exe es el que se tiene que ejecutar para poder acceder al sistema (Fig. E.3).

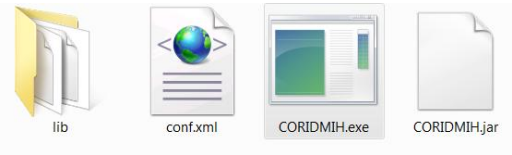


Figura E.3. Organización del directorio Aplicación.

E.3. Documento de tesis

Este mismo archivo también esta anexado al contenido del CD en formato PDF. El archivo se visualiza en la figura E.4 y se llama **Clasificación de objetos rígidos a partir de imágenes digitales, empleando los momentos invariantes de *Hu*.pdf**.

Figura E.4. Documento en formato PDF del trabajo de tesis realizado.

REFERENCIAS

Aapo H S.F.‘What is Independent Component Analysis?’, University of Helsinki, vista el 18 de junio de 2011, < http://www.cs.helsinki.fi/u/ahyvarin/whatisica.shtml>, Finlandia.

Agilent Technologies 2005, ‘Principal Components Analysis’, Agilent Technologies, Inc., vista el 18 de junio de 2011, <http://www.chem.agilent.com/cag/bsp/products/gsgx/Downloads/pdf/pca.pdf>, Estados Unidos.

Balakrishnama, S y Ganapathiraj, A S.F.‘Linear discriminant analysis - a brief tutorial’, Institute for Signal and Information Processing, vista el 18 de junio de 2011, < http://www.isip.piconepress.com/publications/reports/isip_internal/1998/linear_discrim_analysis/lda_theory.pdf>, Estados Unidos.

Castrillon, WA, Álvarez, DA & López, AF 2008, 'Técnicas de extracción de características en imágenes para el reconocimiento de expresiones faciales', *Scientia Et Technica*, junio 2008, p. 7, Colombia.

Cervantes, J 2009, 'Clasificación de grandes conjuntos de datos vía Máquinas de Vectores Soporte y aplicaciones en sistemas biológicos', Tesis de Doctorado, Instituto Politécnico Nacional, México.

Chacón, M 2007, Procesamiento digital de imágenes, 1ra edición, Trillas, México

Dasgupta, S, Papadimitriou, CH & Vazirani, UV 2006, Algorithms, McGraw Hill, 1ra edición, Estados Unidos.

De la Escalera, A 2001, Visión por Computador. Fundamentos y métodos, 1ra edición, Pearson Educación, S.A., España.

Deitel, H & Deitel, P 2004, Como programar en JAVA, 5ta edición, Pearson Educación, México.

Del Brio BM & Sanz A 2007, Redes neuronales artificiales y sistemas borrosos, Alfaomega, 3ra edición, México.

Fernández, R 2001, Glosario básico inglés-español para usuarios de Internet, Asociación de Técnicos de Informática, 4ta edición, España.

García, E 2008, 'Detección y clasificación de objetos dentro de un salón de clases empleando técnicas de procesamiento digital de imágenes', Tesis de Maestría, Universidad Autónoma Metropolitana, México.

Gonzalez, RC & Woods, RE 1992, Digital Image Processing, Addison-Wesley, 3ra edición, Reading, Estados Unidos.

Guerequeta, R & Vallecillo, A 1998, Técnicas de diseño de algoritmos, Servicio de Publicaciones de la Universidad de Málaga, 2da edición, España.

Hernández, J, Ramírez, J & Ferri, C 2004, Minería de datos. Análisis de Datos, Prentice Hall, 1ra edición, España.

Joyanes, L, Zahonero, I, Fernández, M & Sánchez, L 1999, Estructura de datos, 1ra edición, McGraw Hill, España.

Lehmann, C 2004, Geometría Analítica, Limusa, 1ra edición, México.

MathWorks 2011, 'Overview of the MATLAB Environment', MathWorksInc, vista el 18 de junio de 2011, http://www.mathworks.com/help/techdoc/learn_matlab/f0-14059.html, Estados Unidos.

Montes, EV, Guarín, GA, & Castellanos, G 2005, 'Extracción de características de ECG basadas en transformaciones no lineales y Wavelets', Ingeniería e Investigación, diciembre 2005, p. 39-48, Colombia.

Nature Publishing Group 2006, 'What is a support vector machine?' NatureBiotechnology, vista el 18 de junio de 2011, <http://www.fml.tuebingen.mpg.de/raetsch/lectures/ismb09tutorial/images/WhatIsASVM.pdf> , Inglaterra.

Oracle 2011a, 'Learning Java', Oracle, Inc, vista el 18 de junio de 2011, <http://netbeans.org/kb/articles/learn-java.html>, Estados Unidos.

Oracle 2011b, 'Getting Started with an Integrated Development Environment (IDE)', Oracle, Inc, vista el 18 de junio de 2011, < http://java.sun.com/developer/technicalArticles/tools/introhtml>, Estados Unidos.

Pajares, G & De la Cruz, J 2008, Visión por computador, Alfaomega Grupo Editor, 2da edición, España.

Pereyra, CG 2011, 'Reconocimiento de rostros en un ambiente de iluminación controlado no invariantes a expresiones faciales', Tesis de Licenciatura, Universidad del Mar campus Puerto Escondido, Oaxaca, México.

Russel, S & Norving, P 2004, Inteligencia artificial. Un enfoque moderno, Person Educacion, 2da edición, España.

Sahin, F 1997, 'A Radial Basis Function Approach to a Color Image Classification Problem in a Real Time Industrial Application', vista 18 de junio de 2011, < http://scholar.lib.vt.edu/theses/available/etd-6197-223641/unrestricted/Ch3.pdf>, Estados Unidos.

Toscano, JH 2009, 'Detección y seguimiento de objetos invariantes en color en una secuencia de imágenes', Tesis de Licenciatura, Universidad del Mar campus Puerto Escondido, Oaxaca, México.

Vargas, HD 2009, 'Clasificación de cicatrices cutánea en conejos empleando algoritmos estadísticos', Tesis de Licenciatura, Universidad del Quindío, Armenia, Colombia.

Vélez, NJ, Erazo, JH & Loaiza, H 2009, 'Sistema de clasificación de imágenes basado en técnicas de reconocimiento de patrones aplicado en termografía y robótica', *El hombre y la máquina*, julio 2009, p. 45, Colombia.